Contents

Chapter 4 — Intuitive MCQ Bank (Retrieval-Augmented Generation)

68 multiple-choice questions, intuitive/conceptual level. One correct option each. Cover the answer (blockquote) while testing yourself. Bilingual explanations (English + বাংলা).

Exam style: "Which statement best describes…", exactly one correct option; use full technical terms, no abbreviations.

বাংলা ব্যাখ্যা: এই ব্যাংকে অধ্যায় ৪-এর ৫০+ ধারণামূলক MCQ আছে প্রতিটি প্রশ্নের নিচে উত্তর ও সংক্ষিপ্ত ব্যাখ্যা (ইংরেজি + বাংলা)

Topic: Motivation — Why Retrieval-Augmented Generation

Q1. Which statement best describes how Retrieval-Augmented Generation addresses outdated knowledge, hallucination, and lack of private data?

- A. It fixes each problem with a separate dedicated module bolted onto the model.
- B. It retrains the model weights whenever any of the three problems appears.
- C. It uses one mechanism — injecting retrieved evidence into the prompt at inference time — to address all three.
- D. It increases the model's parameter count so that more facts fit in the weights.

> **Answer: C.** All three weaknesses share one cure: condition the unchanged generator on retrieved context at inference time. B describes finetuning, which the chapter says is explicitly *not* needed; D confuses raw capacity with the real issues of freshness and provenance. **বাংলা:** তিনটা সমস্যারই একটাই ওষুধ — উত্তরের আগে প্রমাণ খুঁজে প্রম্পটে বসানো; ওজন বদলায় না

Q2. Why is updating knowledge in a Retrieval-Augmented Generation system described as a "data-engineering operation" rather than a "machine-learning operation"?

- A. Because new knowledge is added by re-indexing documents, leaving the model weights untouched.
- B. Because the documents must be relabelled by human annotators before each update.
- C. Because the model is fine-tuned on the new documents overnight.
- D. Because embeddings are recomputed inside the transformer during generation.

> **Answer: A.** Knowledge lives in the external store, so refreshing it means re-chunking and re-indexing — no gradient step. C is finetuning (a machine-learning operation); B and D invent steps the chapter does not require. **বাংলা:** জ্ঞান বাইরের স্টোরে থাকে, তাই আপডেট মানে শুধু ডকুমেন্ট আবার ইনডেক্স করা — মডেল রিট্রেইন নয়

Q3. A student claims "Retrieval-Augmented Generation eliminates hallucination." Which correction best matches the chapter?

- A. Correct — once evidence is in the prompt, the model can no longer fabricate.
- B. Wrong — retrieval only matters for stale knowledge, not hallucination.
- C. Wrong — retrieval mitigates hallucination, but the generator can still ignore or contradict the provided context.
- D. Correct — retrieval guarantees every claim is entailed by the context.

> **Answer: C.** The chapter stresses retrieval *mitigates*, not *eliminates* — a faithfulness failure (ignoring the context) is still possible. A and D overstate the guarantee; B denies retrieval's main motivation. **বাংলা:** প্রমাণ সামনে থাকলেও মডেল তা উপেক্ষা করতে পারে, তাই হ্যালুসিনেশন কমে কিন্তু শূন্য হয় না

Q4. Using the open-book exam analogy from the chapter, what role does the retriever play?

- A. The textbook itself, containing all the facts.

- B. The librarian who finds the right page in the textbook.
- C. The examiner who grades the citation.
- D. The student who writes the answer.

Answer: B. In the analogy the language model is the student, the document store is the textbook, and the retriever is the librarian who locates the relevant page. D is the generator; A is the document store. **বাংলা:** মডেল = ছাত্র, ডকুমেন্ট স্টোর = বই, রিট্রিভার = লাইব্রেরিয়ান যে সঠিক পৃষ্ঠা খুঁজে দেয়

Topic: Architecture — The Six-Step Pipeline

Q5. Which statement best describes what Retrieval-Augmented Generation changes *inside* the transformer?

- A. It replaces autoregressive decoding with a nearest-neighbour answer lookup.
- B. It adds a retrieval layer between the attention blocks.
- C. It modifies the attention mechanism to attend over the vector store.
- D. It changes nothing inside the transformer; only the conditioning set is extended from $P(y \mid q)$ to $P(y \mid q, D)$.

Answer: D. Decoding stays autoregressive, one token at a time; the only formal change is the extra conditioning set D. A, B, C all invent internal architectural changes the chapter explicitly rules out. **বাংলা:** ট্রান্সফরমারের ভেতরে কিছুই বদলায় না — শুধু কন্ডিশনিং-এ D যোগ হয়, ডিকোডিং আগের মতোই

Q6. Why does the chapter split the pipeline into an "offline" phase and an "online" phase for cost analysis?

- A. Because offline work happens on a different server than online work.
- B. Because offline cost is paid once per corpus while online cost is paid on every query, so moving work offline amortizes it.
- C. Because online steps are always faster than offline steps.
- D. Because only the offline phase uses the language model.

Answer: B. The economic point is amortization: chunk embedding and index construction are paid once (and on updates), whereas query embedding, search, reranking, and generation recur per query. A is incidental; C and D are false. **বাংলা:** অফলাইন খরচ একবার, অনলাইন খরচ প্রতি প্রশ্নে — তাই যা পারো অফলাইনে সরাও, খরচ ভাগ হয়ে যায়

Q7. In the six-step pipeline, which step is performed *online* and on *every* query?

- A. Chunking documents into 200–800 token units.
- B. Building the vector index over all chunk embeddings.
- C. Embedding the query and searching for the top-k.
- D. Embedding each chunk with the bi-encoder.

Answer: C. Chunking, chunk embedding, and indexing are offline (steps 2–4); only query embedding plus search (step 5a) recurs per query. A, B, D are all one-time offline operations. **বাংলা:** চাঙ্কিং-এমবেডিং-ইনডেক্সিং অফলাইন; প্রতি প্রশ্নে শুধু কোয়েরি এমবেড করে খোঁজা হয়

Q8. Why must the query be embedded with the *same* model that embedded the chunks?

- A. To save memory by reusing the model weights.
- B. Because the query is always shorter than a chunk.
- C. Because different models place text in incompatible vector spaces, so the similarity comparison would be meaningless.
- D. Because the reranker requires identical tokenizers.

Answer: C. Two different encoders define two different geometries; cosine between vectors from incompatible spaces is meaningless and retrieval fails. A is a side benefit at best; B and D are unrelated to the failure. **বাংলা:** ভিন্ন মডেল মানে ভিন্ন ভেক্টর-জগৎ; দুই জগতের ভেক্টরে কোসাইন তুলনা অর্থহীন, তাই খোঁজ ব্যর্থ

Q9. In the chapter's pipeline trace, the model answers "not found in the provided documents" when the fact is absent. How should this be interpreted?

- A. As a valid grounded answer — refusal is the instructed behaviour when evidence is missing.
- B. As a hallucination, because the model refused to answer.
- C. As a system failure that should trigger a retry.
- D. As proof that the retriever returned too few chunks.

Answer: A. The chapter explicitly states that refusal is a valid grounded answer when the context lacks the fact. C and D assume something broke; B inverts the meaning — refusing to fabricate is the opposite of hallucinating. **বাংলা:** প্রসঙ্গে তথ্য না থাকলে "পাইনি" বলাই সঠিক গ্রাউন্ডেড উত্তর — এটা ব্যর্থতা নয়

Q10. The chapter says knowledge in Retrieval-Augmented Generation is "explicit" rather than "implicit." What does this buy you?

- A. Faster token generation.
- B. A smaller model that needs less memory.
- C. The ability to cite and audit the visible chunks the answer is based on.
- D. Automatic correctness of every answer.

Answer: C. Because the evidence is visible chunks (not compressed in the weights), answers can be traced, cited, and audited. A is unrelated; B confuses where knowledge lives with model size; D overstates — visibility does not guarantee correctness. **বাংলা:** জ্ঞান দৃশ্যমান চাঙ্কে থাকে বলে উত্তর সাইট করা ও যাচাই করা যায় — এটাই মূল সুবিধা

Topic: Embedding Models and Chunking

Q11. Which statement best describes the key difference between a bi-encoder and a cross-encoder?

- A. The bi-encoder reads query and chunk together; the cross-encoder reads them separately.
- B. The bi-encoder encodes query and chunk independently so chunk vectors can be precomputed; the cross-encoder reads the pair jointly, so nothing can be precomputed.
- C. Both precompute chunk vectors; the cross-encoder is just larger.
- D. The cross-encoder is used for indexing and the bi-encoder for reranking.

Answer: B. Independence of the two towers enables offline precomputation; the cross-encoder's joint attention over the pair makes it accurate but non-precomputable. A swaps the definitions; C is false for the cross-encoder; D reverses their roles. **বাংলা:** bi-encoder আলাদা এনকোড করে বলে আগে হিসাব করা যায়; cross-encoder জোড়া একসাথে পড়ে, তাই আগে হিসাব করা যায় না

Q12. Why does a better embedding model directly raise Recall@k, even with a fixed index and reranker?

- A. Because a better embedding model returns more chunks per query.
- B. Because better embeddings make the reranker unnecessary.
- C. Because if the relevant chunk is embedded far from the query, no downstream index or reranker can recover it.
- D. Because larger embeddings always fit more in the context window.

Answer: C. Recall is set at the embedding stage: a relevant chunk placed far from the query in vector space will never be surfaced, and nothing downstream can fix that miss. A confuses quality with k; B and D are unrelated. **বাংলা:** প্রাসঙ্গিক চাঙ্ক যদি কোয়েরি থেকে দূরে এমবেড হয়, পরের কোনো ধাপ তাকে ফেরাতে পারে না — তাই এমবেডিং-ই recall ঠিক করে

Q13. Cosine similarity is preferred over the raw dot product for ranking embeddings because it —

- A. is faster to compute than a dot product.
- B. measures the angle between vectors and ignores their length, so longer texts cannot win merely by magnitude.
- C. always returns values between 0 and 1 regardless of the vectors.
- D. counts the number of shared tokens between two texts.

Answer: B. Cosine normalizes out magnitude and compares direction; the worked example shows d_3 has the larger dot product yet d_1 wins on cosine. A is false (cosine adds a normalization step); C

is only true for non-negative vectors; D describes lexical overlap, not cosine. **বাংলা:** কোসাইন দৈর্ঘ্য বাদ দিয়ে কোণ মাপে, তাই লম্বা টেক্সট শুধু ম্যাগনিটিউড দিয়ে জিততে পারে না

Q14. Query embedding $q = (2, 1, 0)$ ranks $d_3 = (2, 0, 1)$ *below* $d_1 = (1, 1, 0)$ even though d_3 has the larger dot product. What is the lesson?

- A. Dot product and cosine always agree, so the example must be a mistake.
- B. d_3 should be discarded because it has an extra dimension active.
- C. The query should be re-embedded with a different model.
- D. Normalization by vector length can change the ranking, so embeddings must be normalized before cosine retrieval.

Answer: D. Dividing by the norms reorders the candidates — exactly the trap the chapter highlights. A is contradicted by the worked numbers; B and C misread a normalization effect as a data problem. **বাংলা:** নর্ম দিয়ে ভাগ করলে র‍্যাঙ্ক বদলে যায়, তাই কোসাইনের আগে এমবেডিং নর্মালাইজ করতে হবে

Q15. Why is a chunk that is "too long" bad for retrieval precision?

- A. Long chunks always exceed the context window.
- B. Long chunks cannot be embedded by a bi-encoder.
- C. Long chunks mix multiple topics, so the embedding becomes a blurry average that matches less precisely.
- D. Long chunks are automatically truncated to one sentence.

Answer: C. A multi-topic chunk averages into a vague vector that no single query matches sharply, lowering precision and wasting the token budget. A overgeneralizes; B and D are false. **বাংলা:** বড় চাঙ্কে অনেক বিষয় মেশে, এমবেডিং ঝাপসা গড় হয়ে যায়, ফলে নির্ভুলতা কমে

Q16. What problem does a chunk that is "too short" cause?

- A. It loses surrounding context and becomes ambiguous, e.g. "It increased by 12%" with no referent.
- B. It overflows the vector store.
- C. It cannot be assigned an embedding vector.
- D. It forces the reranker to run more forward passes.

Answer: A. Without enough context a short chunk is ambiguous — the antecedent of "it" is gone. B, C, D describe mechanisms that short length does not cause. **বাংলা:** ছোট চাঙ্ক চারপাশের প্রসঙ্গ হারায়, তাই "এটা ১২% বাড়ল" — কী বাড়ল বোঝা যায় না

Q17. Why is 10–20% overlap added between adjacent chunks?

- A. To double the storage on purpose for redundancy.
- B. So that a fact straddling a chunk boundary survives intact in at least one chunk.
- C. To make every chunk exactly the same length.
- D. To let the reranker skip duplicate chunks.

Answer: B. Overlap ensures a sentence sitting on a boundary is fully contained in one of the neighbouring chunks. A is a side effect, not the goal; C and D are unrelated. **বাংলা:** ওভারল্যাপ থাকলে সীমানায় বসা তথ্য অন্তত একটা চাঙ্কে পুরোটা থাকে, কাটা পড়ে না

Q18. A 2000-token document is chunked with size 400 and overlap 100. Using $n = \lceil (L - C)/(C - O) \rceil + 1$, how many chunks result, and why the "+1"?

- A. 5 chunks; the +1 corrects for zero-based indexing.
- B. 6 chunks; the +1 accounts for overlap storage.
- C. 7 chunks; the +1 counts the first window, since the formula counts how many strides fit *after* it.
- D. 8 chunks; the +1 rounds the stride up.

Answer: C. Stride = 300; $\lceil 1600/300 \rceil + 1 = 6 + 1 = 7$, with start positions 0, 300, …, 1800. The "+1" is the initial window the ceiling term measures past. A, B, D miscount the arithmetic. **বাংলা:** স্ট্রাইড ৩০০, $\lceil 1600/300 \rceil + 1 = 7$; "+1" হলো প্রথম জানালা, সিলিং তার পরের ধাপ গোনে

Topic: Vector Stores and Index Structures

Q19. Which index structure always returns the exact nearest neighbours?

- A. Inverted file index with nprobe < nlist.
- B. Hierarchical Navigable Small World graph.
- C. Product quantization codes.
- D. Flat (brute-force) index.

Answer: D. Flat search compares the query against every vector, so Recall = 1.00 by construction. The inverted file can miss neighbours in unprobed cells; the graph is greedy and approximate; product quantization compares compressed approximations. **বাংলা:** Flat সব ভেক্টর দেখে, তাই নির্ভুল; বাকিগুলো আনুমানিক, কিছু মিস করতে পারে

Q20. Why does an inverted file index sometimes silently miss the true nearest neighbour?

- A. Because the k-means centroids drift during search.
- B. Because the true neighbour can lie in a cell that was not among the `nprobe` probed cells.
- C. Because product quantization corrupts the vector.
- D. Because the graph layers are too sparse.

Answer: B. Search only scans the nprobe closest cells; a neighbour in an unprobed cell is never compared, so recall drops below 1.00. A misattributes the cause; C and D describe different index families. **বাংলা:** খোঁজ শুধু কাছের কয়েকটা ক্লাস্টার দেখে; সঠিক উত্তর অন্য ক্লাস্টারে থাকলে চুপচাপ মিস হয়

Q21. In an inverted file index, what happens as you raise `nprobe` toward `nlist`?

- A. Recall and latency both fall.
- B. Memory usage doubles but recall is unchanged.
- C. Recall rises and latency rises, until at `nprobe = nlist` the search degenerates into flat (exact) search.
- D. The index must be rebuilt from scratch.

Answer: C. Probing more cells finds more true neighbours (recall up) but costs more comparisons (latency up); probing all cells equals brute force. A reverses the direction; B and D are false. **বাংলা:** nprobe বাড়লে recall ও latency দুটোই বাড়ে; nprobe = nlist হলে তা flat সার্চে পরিণত হয়

Q22. Why is the Hierarchical Navigable Small World graph described as the production default beyond roughly 10^5 vectors?

- A. Because it guarantees exact results like flat search.
- B. Because it uses the least memory of all index types.
- C. Because it requires no build step.
- D. Because it offers $O(\log N)$ search with high recall and supports incremental inserts.

Answer: D. Logarithmic greedy descent gives high recall at low latency, and the graph accepts new vectors incrementally. A is false (it is approximate); B is false (graph edges add memory); C is false (build is $O(N \log N)$). **বাংলা:** গ্রাফ-ইনডেক্স লগারিদমিক খোঁজে, recall বেশি, আর নতুন ভেক্টর যোগ করা যায় — তাই বড় কর্পাসের ডিফল্ট

Q23. With $d = 128$, $m = 8$ subvectors, and 8 bits per subvector, product quantization stores each vector in 8 bytes versus 512 bytes uncompressed. What is the cost of this 64× compression?

- A. Search becomes exact but slower.
- B. Distances are computed between approximations, so recall degrades slightly.
- C. The index can no longer be searched at all.
- D. The vectors must be re-embedded after every query.

Answer: B. Storing centroid indices means distances use approximated vectors, so ranking quality drops a little (often mitigated by IVF-PQ plus exact re-scoring). A contradicts the approximation; C and D are false. **বাংলা:** আনুমানিক ভেক্টরে দূরত্ব মাপা হয়, তাই recall একটু কমে — এটাই সংকোচনের মূল্য

Q24. A team has only 50,000 chunks and needs guaranteed exact retrieval. Which index is the best fit, and why?

- A. Product quantization, because it saves memory.
- B. Hierarchical Navigable Small World, because it is fastest.
- C. Flat index, because it is exact (Recall = 1.00) with zero build cost at this small scale.
- D. Inverted file, because nprobe can be tuned.

Answer: C. Below ~10^5 vectors, brute force is fast enough and exact with no build overhead. A and D trade away exactness needlessly; B adds build cost and approximation for a corpus that does not need it. **বাংলা:** ~10^5-এর নিচে flat যথেষ্ট দ্রুত, নির্ভুল আর বিল্ড খরচ শূন্য — তাই সেরা পছন্দ

Q25. The chapter says every index tuning knob "trades the same three currencies." What are they?

- A. Accuracy, recall, and precision.
- B. Speed, recall, and memory.
- C. Latency, throughput, and bandwidth.
- D. Build time, query time, and disk space.

Answer: B. Whether it is nprobe, M, efSearch, or the quantization codes, each knob spends some combination of speed, recall, and memory. A and C list overlapping or off-list terms; D is a narrower subset. **বাংলা:** প্রতিটি নব আসলে গতি–recall–মেমরি, এই তিন মুদ্রার মধ্যেই দর-কষাকষি করে

Q26. Raising the graph index parameter `efSearch` at query time has which primary effect?

- A. Recall rises and latency rises, because a larger candidate list is explored per query.
- B. Build time rises while query latency falls.
- C. Memory per node rises permanently.
- D. The number of edges per node increases.

Answer: A. `efSearch` is the query-time candidate-list size: larger means a more thorough search (higher recall) at higher latency. B describes `efConstruction`; C and D describe `M`. **বাংলা:** efSearch হলো কোয়েরি-সময়ের প্রার্থী-তালিকার আকার; বাড়ালে recall ও latency দুটোই বাড়ে

Topic: Retrieval Mechanisms — Sparse, Dense, Hybrid

Q27. Which statement best describes how the k_1 saturation term in Best Match 25 behaves?

- A. Each additional occurrence of a term adds the same weight, so 10 occurrences count exactly 10 times.
- B. It penalizes rare terms by shrinking their weight.
- C. The term-frequency factor grows with frequency but is bounded by $k_1 + 1$, so the 10th occurrence adds far less than the 1st.
- D. It normalizes for document length instead of term frequency.

Answer: C. Saturation means diminishing returns capped at $k_1 + 1$; in the worked toy corpus, doubling the frequency raised the score by only 1.45×, not 2×. A describes linear term frequency; B is the inverse-document-frequency role; D is the b parameter's role. **বাংলা:** k_1 স্যাচুরেশন মানে একই শব্দ বারবার এলে লাভ কমতে থাকে, সর্বোচ্চ $k_1 + 1$-এ আটকে যায়

Q28. What does the +0.5 smoothing in the Best Match 25 inverse-document-frequency term accomplish?

- A. It boosts very common terms.
- B. It prevents division by zero and unbounded weights for terms in zero or all documents.
- C. It converts the score into a probability.
- D. It normalizes the document length.

Answer: B. The continuity correction keeps the estimate finite when $n(t) = 0$ or $n(t) = N$. A is the opposite (rare terms are boosted); C and D are unrelated to the smoothing. **বাংলা:** +0.5 শূন্য-ভাগ আর অসীম ওজন ঠেকায়, যখন কোনো শব্দ শূন্য বা সব ডকুমেন্টে থাকে

Q29. With $b = 0.75$, how does length normalization treat a document longer than the corpus average?

- A. It boosts its term frequency because it has more words.
- B. It ignores the document entirely.

- C. It discounts its effective term frequency, so it cannot win merely by being longer.
- D. It sets its score to zero.

Answer: C. Length normalization penalizes longer-than-average documents so they do not dominate just by containing more words; $b = 1$ is full normalization, $b = 0$ is none. A inverts the effect; B and D are false. **বাংলা:** গড়ের চেয়ে লম্বা ডকুমেন্টের কার্যকর term frequency কমিয়ে দেয়, যাতে শুধু লম্বা হওয়ায় না জেতে

Q30. For the query "error code XJ-4711," why does sparse retrieval (Best Match 25) typically beat dense retrieval?

- A. Dense retrieval cannot read alphanumeric strings.
- B. Sparse retrieval has a larger context window.
- C. Dense retrieval only works on cross-lingual queries.
- D. The exact rare token gets a large inverse-document-frequency weight in sparse scoring, while the dense embedding smooths it into the surrounding semantics.

Answer: D. A code occurring in few documents earns a huge sparse weight, but the embedding has no dedicated dimension for "XJ-4711," so dense retrieval misses it. A and B are false mechanisms; C is too narrow. **বাংলা:** বিরল কোড sparse-এ বড় IDF ওজন পায়, কিন্তু dense এমবেডিং তাকে আশপাশের অর্থে মিশিয়ে ফেলে — তাই sparse জেতে

Q31. For the query "how do I make my model stop inventing facts," why does dense retrieval beat sparse retrieval against a corpus that uses the word "hallucination"?

- A. Dense retrieval matches the semantic meaning even though there is no shared vocabulary with the documents.
- B. Sparse retrieval has a larger embedding dimension.
- C. The query is too long for Best Match 25 to process.
- D. Dense retrieval only counts exact token overlap.

Answer: A. "Inventing facts" and "hallucination" share no tokens, so sparse term overlap fails, but the embeddings are close in meaning, so dense wins. B and D misstate how the methods work; C is false. **বাংলা:** "inventing facts" আর "hallucination" শব্দে মেলে না, তাই sparse ব্যর্থ; অর্থে কাছাকাছি বলে dense জেতে

Q32. Why does hybrid retrieval combine sparse and dense rankings instead of relying on either alone?

- A. Because combining always halves latency.
- B. Because sparse and dense fail on *different* kinds of queries, so together they cover more cases.
- C. Because dense retrieval is always wrong on its own.
- D. Because sparse retrieval cannot be indexed.

Answer: B. Sparse wins on rare exact tokens; dense wins on paraphrases and cross-lingual matches; fusing them raises recall across both query types. A is false; C and D overstate. **বাংলা:** sparse আর dense আলাদা ধরনের প্রশ্নে ব্যর্থ হয়, তাই একসাথে ব্যবহারে বেশি ক্ষেত্র ঢাকা পড়ে

Q33. Why does Reciprocal Rank Fusion combine *ranks* rather than the raw scores from the sparse and dense systems?

- A. Because ranks are integers and easier to store.
- B. Because scores are not available from a dense retriever.
- C. Because Best Match 25 scores and cosine similarities live on incomparable scales, so fusing ranks avoids any score calibration.
- D. Because ranks ignore the query entirely.

Answer: C. A Best Match 25 score (say 0–25) and a cosine (-1 to 1) cannot be added directly; using ranks makes fusion invariant to scale and removes the calibration burden. A is incidental; B and D are false. **বাংলা:** BM25 স্কোর আর কোসাইন আলাদা এককে, সরাসরি যোগ করা যায় না; র‍্যাঙ্ক ব্যবহারে ক্যালিব্রেশন লাগে না

Q34. In the worked fusion ($k = 60$) with sparse list A, B, C, D and dense list B, D, E, A, document B ends up ranked first. Why?

- A. Because B tops both lists.
- B. Because B is good in both lists (rank 2 sparse, rank 1 dense), beating documents that top only one list.
- C. Because B has the highest raw cosine score.
- D. Because B appears the most times overall, regardless of rank.

Answer: B. B accumulates reciprocal-rank contributions from both lists ($1/62+1/61$), so consistent goodness across lists beats a single-list champion. A is false (A tops the sparse list, B tops only the dense list); C and D misread the mechanism. **বাংলা:** B দুই তালিকাতেই ভালো, তাই এক তালিকায় চ্যাম্পিয়নকেও হারায় — এটাই হাইব্রিডের শক্তি

Q35. What is the role of the constant $k = 60$ in Reciprocal Rank Fusion?

- A. It is the number of documents fused.
- B. It sets the minimum cosine threshold for inclusion.
- C. It is the number of lists being combined.
- D. It damps the influence of top ranks so that rank 1 does not dominate every other contribution.

Answer: D. Without k, the $1/\text{rank}$ contribution of rank 1 would overwhelm everything; $k = 60$ flattens the curve. A, B, C misidentify the constant entirely. **বাংলা:** $k = 60$ উপরের র‍্যাঙ্কগুলোর প্রভাব কমায়, যাতে র‍্যাঙ্ক ১ সবকিছুকে চাপা না দেয়

Q36. When would you switch from Reciprocal Rank Fusion to weighted score fusion, according to the chapter's engineering advice?

- A. Always, because weighted fusion is strictly better.
- B. Never, because rank fusion cannot be improved.
- C. Only when you have evaluation data to tune the weight α and want to encode "trust one system more."
- D. Only when the corpus is smaller than 10^5 documents.

Answer: C. Weighted fusion is more expressive but needs score normalization plus a tuned α, so the lecture says start with rank fusion and switch only with evaluation data. A and B are too absolute; D is unrelated to the choice. **বাংলা:** weighted fusion বেশি প্রকাশক্ষম কিন্তু α টিউন করতে eval ডেটা লাগে — তাই আগে rank fusion, পরে দরকারে weighted

Q37. Why is Reciprocal Rank Fusion invariant to any strictly increasing transformation of the underlying scores?

- A. Because it averages the scores before ranking.
- B. Because monotone transformations preserve all pairwise comparisons, hence the ranks, hence every $1/(k + \text{rank})$ term.
- C. Because it discards the dense list entirely.
- D. Because it normalizes scores to the range $[0, 1]$ first.

Answer: B. Ranks are defined by comparisons; a strictly increasing f preserves every comparison, so the ranks and the fused ordering are unchanged. A and D involve scores (which fusion ignores); C is false. **বাংলা:** মনোটোন রূপান্তর সব তুলনা অক্ষুণ্ণ রাখে, তাই র‍্যাঙ্ক ও ফলাফল একই থাকে — স্কোর লাগে না

Q38. Which statement best describes how Best Match 25 improves on plain term-frequency–inverse-document-frequency weighting?

- A. It removes the inverse-document-frequency factor entirely.
- B. It makes term frequency grow linearly without any bound.
- C. It uses neither length normalization nor tunable parameters.
- D. It saturates term frequency (bounded by $k_1 + 1$) and adds tunable length normalization via b.

Answer: D. Best Match 25 is the tunable, probabilistically motivated refinement: saturating term frequency plus explicit length normalization, with k_1 and b knobs. A, B, C describe the opposite of what it does. **বাংলা:** BM25 হলো TF-IDF-এর টিউনযোগ্য রূপ — term frequency স্যাচুরেট করে ($k_1 + 1$ সীমা) আর b দিয়ে দৈর্ঘ্য-নর্মালাইজেশন যোগ করে

Topic: Reranking and Context Assembly

Q39. Which statement best describes why the retrieval stage targets recall while the reranking stage targets precision?

- A. Recall is cheaper to compute than precision.
- B. A chunk missed by retrieval is unrecoverable, but false positives from retrieval can still be filtered by the expensive reranker — so cast a wide net first, then filter.
- C. The reranker scans millions of chunks while retrieval sees only dozens.
- D. Precision must be maximized first to keep latency low.

Answer: B. Errors of omission are fatal in stage one (recall is bounded there); errors of inclusion are correctable in stage two. C reverses the candidate-set sizes; A and D misstate the reasoning. **বাংলা:** রিট্রিভাল যা মিস করে তা আর ফেরে না, কিন্তু বাড়তি নয়েজ পরে ছাঁটা যায় — তাই আগে জাল বড়, পরে ফিল্টার

Q40. Why can a reranker never raise recall, only precision?

- A. Because it uses a weaker model than retrieval.
- B. Because it outputs a permutation of the candidate set, so the set of relevant chunks it contains is unchanged; it can only move them to the front.
- C. Because recall and precision are the same metric.
- D. Because it discards half the candidates at random.

Answer: B. A reranker reorders the candidate set C; $|R \cap C|$ (recall's numerator) is fixed, but moving the relevant members forward improves precision at small k. A, C, D are false. **বাংলা:** রির্যাঙ্কার শুধু একই সেট সাজায়, প্রাসঙ্গিক চাঙ্কের সংখ্যা বদলায় না — তাই recall স্থির, precision বাড়ে

Q41. Running a cross-encoder over an entire 1,000,000-chunk corpus at 10 ms per pass is described as infeasible (~2.78 hours per query). What is the standard fix?

- A. Use a smaller cross-encoder for the whole corpus.
- B. Precompute the cross-encoder scores offline.
- C. Retrieve a small candidate set (e.g. top 20–100) with a cheap stage, then cross-encode only those.
- D. Skip reranking and trust the raw cosine ranking.

Answer: C. Two-stage design caps cross-encoder work at k passes (top $20 \approx 0.22$ s), making it feasible. A still scans millions; B is impossible (scores depend on the query); D abandons the precision gain. **বাংলা:** আগে সস্তা ধাপে অল্প প্রার্থী বাছো, তারপর শুধু সেগুলোতে cross-encoder চালাও — তবেই সম্ভব

Q42. Why can a bi-encoder's chunk embeddings be precomputed offline while a cross-encoder's scores cannot?

- A. Because the bi-encoder is always larger.
- B. Because the cross-encoder's score is a function of the (query, chunk) *pair*, which does not exist until the query arrives.
- C. Because chunk embeddings change on every query.
- D. Because the cross-encoder ignores the query.

Answer: B. The bi-encoder's chunk tower runs without the query, so vectors are stored ahead of time; the cross-encoder needs the pair, so each candidate is a fresh forward pass at query time. A, C, D are false. **বাংলা:** cross-encoder-এর স্কোর প্রশ্ন-চাঙ্ক জোড়ার ফাংশন; প্রশ্ন না এলে জোড়াই নেই, তাই আগে হিসাব করা যায় না

Q43. During context assembly, why does the chapter advise placing the best chunk first or last rather than in the middle?

- A. Because the middle of the prompt is truncated by the tokenizer.
- B. Because models attend most to the beginning and end of the prompt ("lost in the middle"), so buried evidence is under-used.
- C. Because the reranker only scores the first and last chunks.
- D. Because citations can only be attached to the first chunk.

Answer: B. The bowl-shaped attention pattern means middle content is attended to least, so the strongest evidence should sit at an edge. A, C, D are not supported by the chapter. **বাংলা:** মডেল প্রম্পটের শুরু ও শেষে বেশি মনোযোগ দেয় ("lost in the middle"), তাই সেরা চাঙ্ক মাঝে নয়, কিনারায় রাখো

Q44. How does an LLM-as-reranker compare to a cross-encoder reranker?

- A. It is faster and less accurate.
- B. It is identical in cost and accuracy.
- C. It is slower and even more accurate — the same trade-off taken one step further.
- D. It can recover chunks the retrieval stage missed.

Answer: C. The chapter places the LLM-as-reranker further along the same speed-versus-accuracy curve: more accurate, more expensive. A reverses it; B is false; D violates the recall bound. **বাংলা:** LLM-রির্যাঙ্কার আরও ধীর কিন্তু আরও নির্ভুল — একই trade-off আরও এক ধাপ এগিয়ে

Topic: Query Decomposition

Q45. Why does a single-shot similarity search fail on "Who supervised the doctoral thesis of the founder of company X?"

- A. The query is too short to embed.
- B. The founder's name is always in the corpus.
- C. The evidence spans two facts that no single chunk contains, so the query's embedding lands between them and matches neither well.
- D. Similarity search cannot handle questions about people.

Answer: C. Multi-hop questions need evidence from separate chunks; the combined query embeds in no single passage's neighbourhood. A, B, D are false generalizations. **বাংলা:** উত্তরটা দুটি আলাদা তথ্যে ছড়ানো, কোনো এক চাঙ্কে নেই — তাই কোয়েরির এমবেডিং মাঝামাঝি পড়ে, কোনোটাই ভালো মেলে না

Q46. In a "retrieve–decompose–retrieve" loop, why must retrieval be interleaved with generation?

- A. Because the generator is faster than the retriever.
- B. Because the answer to sub-question 1 parameterizes sub-question 2, so the second query cannot be formed until the first is answered.
- C. Because the index must be rebuilt between steps.
- D. Because each sub-question uses a different embedding model.

Answer: B. The intermediate answer (e.g. the founder's name) is needed to phrase the next retrieval, forcing an interleaved loop — the seed of agentic behaviour. A, C, D are unrelated. **বাংলা:** প্রথম উপ-প্রশ্নের উত্তর দ্বিতীয় উপ-প্রশ্ন তৈরি করে, তাই খোঁজা ও জেনারেশন পালাক্রমে চালাতে হয়

Q47. What does "query rewriting" accomplish in the example "what about its latency?" → "what is the latency of the Hierarchical Navigable Small World index?"

- A. It turns a conversational follow-up into a standalone query by resolving the pronoun reference.
- B. It adds synonyms to broaden the search.
- C. It splits the question into multiple sub-questions.
- D. It compresses the query to fewer tokens.

Answer: A. Query rewriting resolves "its" to make the follow-up self-contained for retrieval. B describes query expansion; C describes decomposition; D is unrelated. **বাংলা:** কোয়েরি রিরাইটিং সর্বনাম মিটিয়ে অনুসরণ-প্রশ্নকে স্বয়ংসম্পূর্ণ করে তোলে

Topic: Evaluating Retrieval-Augmented Generation

Q48. Why does the chapter insist on evaluating retrieval and generation *separately*?

- A. Because they run on different hardware.

- B. Because the generator cannot be measured at all.
- C. Because retrieval metrics are always more important.
- D. Because a bad answer can come from bad retrieval *or* bad generation, and the fix differs for each.

Answer: D. Separating the stages isolates the failure (a missed chunk vs. unfaithful generation) so you can apply the right fix. A is incidental; C and B are false. **বাংলা:** খারাপ উত্তর retrieval-এর দোষেও হতে পারে, generation-এর দোষেও — সমাধান আলাদা, তাই আলাদা মাপতে হয়

Q49. Ground truth relevant set is $\{d_2, d_5, d_9\}$ and the system returns $[d_7, d_2, d_9, d_1, d_3]$. Which statement about Recall@5 versus Recall@3 is correct?

- A. Recall@5 equals Recall@3 (both 0.67), because d_5 is missed at both cutoffs while d_2 and d_9 are within rank 3.
- B. Recall@5 is higher than Recall@3 because more results are returned.
- C. Recall@5 is 1.00 because five results were returned.
- D. Recall@3 is 0.00 because the top result is irrelevant.

Answer: A. Hits d_2 (rank 2) and d_9 (rank 3) fall inside both cutoffs and d_5 is absent from the returned list, so both recalls are $2/3 = 0.67$. B assumes new hits appear at ranks 4–5 (they do not); C and D miscompute. **বাংলা:** d_2, d_9 র‍্যাঙ্ক ৩-এর ভেতরে, d_5 একেবারেই নেই — তাই Recall@3 = Recall@5 = 0.67

Q50. Which statement best captures the difference between Recall@k and Precision@k?

- A. Recall@k divides by k; Precision@k divides by the number of relevant chunks.
- B. Recall@k only counts the first relevant result.
- C. They are always equal.
- D. Recall@k divides hits by the number of relevant chunks; Precision@k divides hits by k.

Answer: D. Recall's denominator is the relevant-set size; precision's denominator is k. A swaps the denominators; C is false; B describes reciprocal rank, not recall. **বাংলা:** Recall = hits / প্রাসঙ্গিক সংখ্যা; Precision = hits / k — ডিনমিনেটরই পার্থক্য

Q51. Why does Mean Reciprocal Rank care only about the *first* relevant result?

- A. Because it is defined as the average of 1/(rank of the first relevant result), ignoring later hits.
- B. Because it counts every relevant chunk equally.
- C. Because it averages precision over all ranks.
- D. Because it is identical to Recall@1.

Answer: A. By definition, Mean Reciprocal Rank uses only the rank of the first hit per query, then averages. C and B describe other metrics; D is false (Recall@1 is binary at cutoff 1, not a reciprocal). **বাংলা:** MRR প্রতিটি প্রশ্নে শুধু প্রথম সঠিক ফলের র‍্যাঙ্কের বিপরীত নেয়, পরেরগুলো অগ্রাহ্য করে

Q52. A run finds two of three relevant chunks but wastes rank 1 on an irrelevant chunk, giving nDCG@3 ≈ 0.53. What does normalized discounted cumulative gain reward that plain Recall@k does not?

- A. Returning fewer total chunks.
- B. Reducing the size of the relevant set.
- C. Matching exact tokens between query and chunk.
- D. Placing relevant items *higher* in the ranking, via a position discount.

Answer: D. The $1/\log_2(i+1)$ discount means earlier relevant items score more, so wasting rank 1 roughly halves the score — recall ignores position. A, C, B are unrelated. **বাংলা:** nDCG প্রাসঙ্গিক চাঙ্ক উপরে রাখলে বেশি নম্বর দেয় (পজিশন ডিসকাউন্ট), যা Recall@k করে না

Q53. Which statement best describes the difference between faithfulness and answer relevance?

- A. Faithfulness checks whether the answer is entailed by the context; answer relevance checks whether the answer addresses the question.
- B. Faithfulness checks the answer against the question; answer relevance checks it against the context.
- C. They are the same metric measured twice.
- D. Faithfulness measures latency; answer relevance measures cost.

Answer: A. Faithfulness compares answer ↔ context (groundedness); answer relevance compares answer ↔ question (pertinence). B swaps the comparison targets; C and D are false. **বাংলা:** Faithfulness মাপে উত্তর প্রসঙ্গ থেকে এসেছে কি না; answer relevance মাপে উত্তর প্রশ্নের জবাব দেয় কি না

Q54. Why is "faithfulness" not the same as "factual correctness"?

- A. Because faithfulness is measured by humans and correctness by machines.
- B. Because faithfulness only applies to numerical answers.
- C. Because correctness ignores the question.
- D. Because faithfulness compares the answer to the *context*, so if the retrieved context is itself wrong, a perfectly faithful answer is still factually wrong.

Answer: D. Faithfulness is answer ↔ context, not answer ↔ world; garbage context yields faithful-but-wrong answers, so factuality needs trusted corpora. A, C, B are false. **বাংলা:** Faithfulness উত্তরকে প্রসঙ্গের সাথে মেলায়, বাস্তবের সাথে নয়; প্রসঙ্গ ভুল হলে বিশ্বস্ত উত্তরও ভুল হতে পারে

Topic: Advanced Retrieval-Augmented Generation Architectures

Q55. In Hypothetical Document Embeddings, why does it work to embed a generated hypothetical answer even if that answer is factually wrong?

- A. Because the hypothetical answer is shaped — stylistically and semantically — like the target passage, so it embeds near the real answer chunks even if its facts are wrong.
- B. Because the hypothetical answer is always factually correct.
- C. Because it replaces the need for any document store.
- D. Because it is shorter than the query.

Answer: A. B short question and a long factual passage sit in different "registers"; a hypothetical answer matches the passage's shape, closing that gap. B contradicts the chapter ("shape matters, truth does not"); C and D are false. **বাংলা:** কাল্পনিক উত্তর লক্ষ্য-প্যাসেজের মতো আকৃতির হয়, তাই ভুল হলেও আসল উত্তর-চাঙ্কের কাছে এমবেড হয়

Q56. Why does graph-based retrieval (Graph-RAG) succeed on multi-hop questions where similarity search cannot "jump" between documents?

- A. Because it embeds documents with a larger model.
- B. Because it stores every document twice.
- C. Because it ignores the query entirely.
- D. Because it retrieves connected subgraphs by traversing entity relations, linking documents that share entities but not vocabulary.

Answer: D. When evidence is spread over documents that share entities but no common words, cosine cannot bridge them, yet graph edges (acquired, founded by, supervised by) can be traversed. A, B, C do not describe the mechanism. **বাংলা:** ডকুমেন্টগুলো এক এনটিটি ভাগ করে কিন্তু শব্দ নয়; কোসাইন সেতু গড়তে পারে না, কিন্তু গ্রাফের প্রান্ত ধরে হাঁটা যায়

Q57. What is the recurring cost the chapter attaches to increasingly autonomous retrieval (agentic / multi-step) architectures?

- A. They require multiple model calls per question, adding cost and unbounded latency.
- B. They make the index impossible to update.
- C. They reduce accuracy on every question.
- D. They remove the need for evaluation.

Answer: A. Letting the model decide what, whether, and when to retrieve means several model calls and latency that is not fixed in advance. C overstates; B and D are false. **বাংলা:** এজেন্টিক রিট্রিভাল মানে প্রশ্নপ্রতি একাধিক মডেল কল — খরচ ও অনির্ধারিত লেটেন্সি বাড়ে

Q58. When is hierarchical retrieval (retrieve summaries first, then drill into chunks) most useful?

- A. For tiny corpora that fit in one prompt.

- B. For cross-lingual retrieval only.
- C. For queries that contain rare error codes.
- D. For very large or very structured corpora such as legal codes, where summary-level retrieval narrows the search before drilling down.

Answer: D. Indexing section/document summaries first, then drilling into the winning summaries' chunks, helps large or structured corpora. A defeats the purpose; C is a sparse-retrieval strength; B is unrelated. **বাংলা:** খুব বড় বা সুগঠিত কর্পাসে (যেমন আইনগ্রন্থ) আগে সারাংশ স্তরে খুঁজে পরে চাঙ্কে নামা কার্যকর

Topic: Retrieval Latency, Cost, and System Considerations

Q59. In the worked latency budget (total $\approx$ 6,215 ms, of which generation is $\approx$ 96.5%), which optimization changes perceived latency the most?

- A. Trimming the context (faster prefill) or streaming tokens to the user, since generation dominates.
- B. Switching the query embedder to a faster model.
- C. Speeding up the index search from 5 ms to 1 ms.
- D. Increasing nprobe in the index.

Answer: A. When generation is 96.5% of the budget, shaving 4 ms off a 5 ms index step is invisible; reducing prefill or streaming output is what users feel. C and B optimize negligible stages; D increases latency. **বাংলা:** জেনারেশনই বেশিরভাগ সময় খায়; ইনডেক্স ৫→১ ms করলে কিছু বদলায় না, কনটেক্সট ছোট করা বা স্ট্রিমিং অনুভূত লেটেন্সি বদলায়

Q60. Why does better retrieval quality (fewer, better chunks) directly buy faster generation?

- A. Because fewer chunks mean a smaller embedding model.
- B. Because the index search becomes exact.
- C. Because the reranker is skipped when chunks are good.
- D. Because stuffing more chunks into the prompt increases prefill time and cost, so trimming the context speeds generation.

Answer: D. Prefill grows with context length; fewer, higher-quality chunks shorten the prompt and thus the prefill the generator must process. A, C, B are unrelated mechanisms. **বাংলা:** বেশি চাঙ্ক মানে লম্বা প্রম্পট, বেশি prefill; কম কিন্তু ভালো চাঙ্ক প্রম্পট ছোট করে জেনারেশন দ্রুত করে

Q61. For a 250,000-token manual at \$3.00 per million input tokens, retrieval ($\approx$2,500 tokens/query) costs about 100$\times$ less than pasting the whole manual. When does long context still win?

- A. When the corpus is small, every part matters (e.g. whole-document summarization), or building an index is not worth it.
- B. When the corpus is large and changes constantly.
- C. When query volume is extremely high.
- D. When exact error-code lookups dominate.

Answer: A. Long context wins on small corpora, global tasks needing the whole document, or low-volume use where index engineering is unjustified. C and B favour retrieval; D favours sparse retrieval. **বাংলা:** কর্পাস ছোট, পুরোটা দরকার (যেমন পূর্ণ সারাংশ), বা ইনডেক্স বানানো অর্থহীন হলে long context জেতে

Q62. Why does deleting documents from a graph index often use "tombstones" rather than immediate physical removal?

- A. Because deletion is forbidden in graph indexes.
- B. Because tombstones reduce memory immediately.
- C. Because tombstones make queries exact.
- D. Because in-place deletion degrades the graph structure, so deletions are marked as tombstones and cleaned up in a periodic rebuild.

Answer: D. Removing nodes damages the navigable graph, so deletions are flagged (tombstoned) and physically purged on a periodic rebuild. A is false; C and B are the opposite of what tombstones

do. **বাংলা:** গ্রাফ থেকে সরাসরি নোড মুছলে গঠন নষ্ট হয়, তাই tombstone দিয়ে চিহ্নিত করে পরে rebuild-এ পরিষ্কার করা হয়

Q63. What is the "consistency trade-off" the chapter warns about between an update and a full re-index?

- A. Between update and re-index, queries may retrieve stale or deleted content, citing a document version that no longer exists.
- B. The index uses more disk during the update.
- C. The embedding model must be retrained.
- D. The reranker stops working during updates.

Answer: A. During the gap, retrieval can surface stale or removed content unless filters or versioning guard it — the classic freshness hazard. B, C, D are not the stated trade-off. **বাংলা:** আপডেট ও re-index-এর ফাঁকে পুরনো বা মুছে-ফেলা তথ্য উঠে আসতে পারে — এটাই consistency-এর ঝুঁকি

Q64. What is the "dual cost" of any document change in a Retrieval-Augmented Generation system?

- A. The cost of two separate servers.
- B. The query embedding and the chunk embedding.
- C. The reranker pass and the generation pass.
- D. The embedding compute *and* the index maintenance.

Answer: D. Changing a document means re-embedding it (compute) *and* updating the index structure (maintenance) — two costs, not one. A, C, B mislabel the pair. **বাংলা:** ডকুমেন্ট বদলালে এমবেডিং হিসাব আর ইনডেক্স রক্ষণাবেক্ষণ — দুটো খরচই দিতে হয়

Topic: Deployment Patterns and Choosing the Right Tool

Q65. A coding assistant must answer queries containing exact function names and identifiers. Which design choices best fit, per the chapter?

- A. Code-aware chunking by functions/classes, a small fast embedder, and sparse retrieval for exact-identifier queries.
- B. Whole-file chunks, the largest possible embedder, and dense-only retrieval.
- C. Hierarchical retrieval with no chunking.
- D. Long context only, never an index.

Answer: A. The chapter recommends function/class chunking, a small fast embedding model, and sparse retrieval for exact identifiers in coding assistants. B, C, D contradict those recommendations. **বাংলা:** কোডিং অ্যাসিস্ট্যান্টে ফাংশন/ক্লাস ভিত্তিক চাঙ্কিং, ছোট দ্রুত এমবেডার, আর সঠিক আইডেন্টিফায়ারের জন্য sparse retrieval ভালো

Q66. Why is prompt injection through retrieved documents a real security risk in Retrieval-Augmented Generation?

- A. Because a retrieved chunk may contain adversarial instructions ("ignore your previous instructions…") that the generator reads with near-instruction authority.
- B. Because retrieved text is encrypted and cannot be inspected.
- C. Because the index leaks user queries to attackers.
- D. Because embeddings can be reversed into the original text.

Answer: A. Retrieved content enters the prompt and the model can treat embedded instructions as commands; the mitigation is to treat retrieved text as data. B, C, D describe different (and unsupported) concerns. **বাংলা:** রিট্রিভ করা চাঙ্কে শত্রুতাপূর্ণ নির্দেশ থাকতে পারে, যা মডেল প্রায় কমান্ডের মতো পড়ে — তাই রিট্রিভ টেক্সটকে ডেটা হিসেবে দেখতে হয়

Q67. The chapter's rule of thumb is "knowledge → retrieval; behaviour → finetuning; small static corpus + global tasks → long context." A team needs the model to adopt a strict new output *format and tone*. Which approach fits best?

- A. Retrieval, because formats live in documents.
- B. Long context, because tone fits in the prompt.

- C. Product quantization, to compress the style.
- D. Finetuning, because changing style/format/behaviour is exactly what finetuning excels at.

Answer: D. The decision table marks style/format/behaviour change as "excellent" for finetuning and "weak" for retrieval and long context. A and B misuse knowledge tools for a behaviour change; C is an index technique. **বাংলা:** ভঙ্গি/ফরম্যাট/আচরণ বদলানো finetuning-এর শক্তি; জ্ঞান নয় বলে retrieval বা long context দুর্বল

Q68. Why does the chapter say Retrieval-Augmented Generation "relocates the knowledge problem" rather than solving coverage?

- A. Because it cannot answer what is not in the corpus — missing documents mean missing answers.
- B. Because it always needs finetuning afterward.
- C. Because it moves knowledge into the model weights.
- D. Because it deletes documents during indexing.

Answer: A. Retrieval can only surface what the corpus contains; gaps in the corpus become gaps in answers, so coverage is still a curation problem. C, B, D misstate the limitation. **বাংলা:** কর্পাসে যা নেই, retrieval তা বলতে পারে না — তাই কভারেজ সমস্যা থেকেই যায়, শুধু জায়গা বদলায়

Answer Key

Q	Ans	Q	Ans	Q	Ans	Q	Ans
Q1	C	Q18	C	Q35	D	Q52	D
Q2	A	Q19	D	Q36	C	Q53	A
Q3	C	Q20	B	Q37	B	Q54	D
Q4	B	Q21	C	Q38	D	Q55	A
Q5	D	Q22	D	Q39	B	Q56	D
Q6	B	Q23	B	Q40	B	Q57	A
Q7	C	Q24	C	Q41	C	Q58	D
Q8	C	Q25	B	Q42	B	Q59	A
Q9	A	Q26	A	Q43	B	Q60	D
Q10	C	Q27	C	Q44	C	Q61	A
Q11	B	Q28	B	Q45	C	Q62	D
Q12	C	Q29	C	Q46	B	Q63	A
Q13	B	Q30	D	Q47	A	Q64	D
Q14	D	Q31	A	Q48	D	Q65	A
Q15	C	Q32	B	Q49	A	Q66	A
Q16	A	Q33	C	Q50	D	Q67	D
Q17	B	Q34	B	Q51	A	Q68	A

Distribution count: A = 17, B = 17, C = 17, D = 17 — total 68 questions.

Self-check: 68 questions (Q1–Q68), exceeding the 52 minimum; each has exactly four options and one correct answer. Distribution is balanced across A/B/C/D (≈25% each).