AI Engineering — Deep Question Bank

All chapter questions consolidated, plus additional harder, mixed-topic, sample-exam-style questions. Organized by chapter, then by difficulty.

Difficulty levels: B Basic | I Intermediate | H Hard | VH Very hard | DT Deep thinking.

How to use this question bank

1. Cover the answer; attempt the question.
2. Compare with the answer.
3. If you got it wrong, revisit the chapter section.
4. Tag uncertain questions; come back to them.

Chapter 1 — Introduction

Conceptual

Q1.1 (B). What does the acronym GPT stand for, and which of the four words refers to the architecture? > A. Generative, Pretrained, Transformer; "Transformer" is the architecture. (The "Chat" in "ChatGPT" is the conversational interface, not part of GPT.)

Q1.2 (B). Define a language model in one sentence. > A. A language model assigns probabilities to sequences of tokens, $P(w_1, \ldots, w_T)$, by predicting each next token from preceding tokens.

Q1.3 (I). Why is next-word prediction sufficient training to produce a multi-task assistant? > A. To minimise next-word loss across web-scale text, the model must internalise grammar, world knowledge, arithmetic, and reasoning patterns. The objective is implicitly multi-task.

Q1.4 (H). Compare a 5-gram model and a transformer LLM along (a) memory, (b) data efficiency, (c) generalization. > A. (a) 5-gram: $O(V^5)$ worst case; transformer: fixed model size. (b) Transformer wins via embeddings and attention sharing. (c) Transformer dramatically generalises better; n-gram fails on unseen tuples.

Q1.5 (DT). A friend says: "ChatGPT remembers me across sessions." Refute or support. > A. Refute. ChatGPT is stateless; the chat application re-sends the conversation history each turn. Cross-session memory exists only if the application explicitly stores and restores it.

Mathematical

Q1.6 (B). Compute $P(\text{cat} \mid \text{the})$ from corpus "the cat sat on the mat". > A. "the" appears 2× and is followed by "cat" once $\rightarrow 1/2 = 0.5$.

Q1.7 (I). Given $P(\text{the}) = 0.05, P(\text{cat} \mid \text{the}) = 0.5, P(\text{sat} \mid \text{cat}) = 1.0$, compute $P(\text{the cat sat})$. > A. $0.05 \times 0.5 \times 1.0 = 0.025$.

Q1.8 (H). Derive the chain rule of probability for a length-T sequence. > A. Iterate the definition $P(A, B) = P(A)P(B \mid A)$: $P(w_1, w_2) = P(w_1)P(w_2 \mid w_1)$. Apply repeatedly to obtain $P(w_{1..T}) = \prod_t P(w_t \mid w_{<t})$.

Q1.9 (H). State and explain the role of $P(w)$ in ASR's posterior $\arg\max_w P(x \mid w)P(w)$. > A. $P(x \mid w)$ is the acoustic model (how likely audio is given words); $P(w)$ is the language model (how plausible a word sequence is). The LM penalises ungrammatical hypotheses; without it, ASR commits embarrassing word-level errors like "wreck a nice beach" for "recognize speech".

Coding

Q1.10 (B). Write Python that returns the bigram MLE probability $\hat{P}(w \mid w')$. > A. > python > from collections import Counter > def bigram_prob(corpus, prev, curr): > toks = corpus.split() > unigram = Counter(toks) > bigram = Counter(zip(toks, toks[1:])) > return bigram[(prev, curr)] / unigram[prev] if unigram[prev] else 0.0 >

Q1.11 (I). Add Laplace add-1 smoothing to the previous function. > A. > python > def bigram_lap(corpus, prev, curr): > toks = corpus.split(); V = len(set(toks)) > unigram = Counter(toks); bigram = Counter(zip(toks, toks[1:])) > return (bigram[(prev,curr)] + 1) / (unigram[prev] + V) >

Chapter 2 — Foundation Models / LLMs

Conceptual

Q2.1 (I). Why is dot-product attention divided by $\sqrt{d_k}$? > A. Without scaling, $(QK^T)_{ij}$ has variance $\propto d_k$, pushing softmax into a saturated regime where gradients vanish. Dividing by $\sqrt{d_k}$ keeps variance ~1, preserving usable gradients.

Q2.2 (I). What is the role of the causal mask in a decoder-only transformer? > A. It prevents each position from attending to future tokens during training (no leakage of the answer) and during inference (preserves autoregressive validity).

Q2.3 (H). Compare SFT, RLHF, and DPO on (a) signal type, (b) cost, (c) stability. > A. SFT uses positive demonstrations only (cheap, stable, but no signal about which alternative is worse). RLHF uses preference pairs through a reward model + PPO (rich signal, expensive, less stable). DPO uses preference pairs with a closed-form supervised loss (rich signal, cheap, stable).

Q2.4 (H). Why does multi-head attention often outperform a single head with the same total d? > A. Heads can specialise (e.g., one for local syntax, another for coreference), enriching the representation. Empirically, h heads of width d/h beat a single head of width d.

Q2.5 (DT). Why do real labs over-train models well past the "compute-optimal" Chinchilla ratio? > A. Chinchilla balances training compute. In production, inference dominates lifetime cost, so a smaller-but-overtrained model (cheap to deploy) is preferred even if training compute is non-optimal.

Q2.6 (DT). Explain why benchmark contamination is a training-data problem rather than a model problem. > A. If the benchmark text leaked into the training corpus, the model has memorised answers; no architectural change can clean that. The fix is rigorous decontamination of pretraining data.

Mathematical

Q2.7 (B). Given logits $z = (2, 1, 0)$ and $T = 1$, compute softmax. > A. $e^2 = 7.389,\ e^1 = 2.718,\ e^0 = 1$. Sum = 11.107. Probabilities: $(0.665, 0.245, 0.090)$.

Q2.8 (I). Same logits with $T = 0.5$; compute and discuss. > A. $z/T = (4, 2, 0)$; $e = (54.6, 7.39, 1)$; sum=63.0; $P = (0.867, 0.117, 0.016)$. Sharper distribution → more deterministic output.

Q2.9 (H). Derive $\partial L/\partial z_i$ for cross-entropy with softmax. > A. With one-hot label y, $L = -\log P_c$ where $P_i = e^{z_i}/\sum e^{z_j}$. Compute $\partial \log P_c/\partial z_i = \mathbb{1}_{i=c} - P_i$. Therefore $\partial L/\partial z_i = P_i - y_i$.

Q2.10 (H). For LoRA on a 4096×4096 matrix with $r = 8$, count trainable parameters. > A. $r(d + k) = 8 \cdot (4096 + 4096) = 65{,}536$ — about 0.4% of the original 16.78M.

Q2.11 (VH). Write the DPO loss in closed form for a single (prompt, y_w, y_l) triple. > A. >

$$\mathcal{L} = -\log \sigma \left(\beta \left[\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right] \right)$$

Coding

Q2.12 (I). Implement scaled dot-product attention from scratch (single head). > A. > python > import numpy as np > def softmax(z): z -= z.max(-1, keepdims=True); e = np.exp(z); return e/e.sum(-1, keepdims=True) > def attention(Q, K, V, mask=None): > d_k = K.shape[-1] > scores = Q @ K.T / np.sqrt(d_k) > if mask is not None: scores += mask > return softmax(scores) @ V >

Q2.13 (H). Implement BPE merge step on a tokenized vocabulary. > A. See Chapter 2 file Section 2.2.

Q2.14 (VH). Implement causal masked multi-head attention with PyTorch. > A. See Chapter 2 task 1 (Block class).

Application / mixed

Q2.15 (H). Outline the data path of a single user prompt through GPT-style inference, naming every transformation. > A. Text → tokenizer (BPE byte-level) → IDs → embedding lookup → +positional encoding → N transformer blocks (each: residual+LN, multi-head causal attn, residual+LN, FFN with GELU/SwiGLU) → final LN → unembedding (often tied to input embedding) → softmax → sample under chosen strategy → next token. Loop.

Chapter 3 — Prompt Engineering

Conceptual

Q3.1 (B). Distinguish system, user, assistant roles. > A. System: meta-instructions that shape behavior (persona, rules, refusal policy). User: the request. Assistant: prior model replies (chat memory). System enjoys higher priority due to alignment training.

Q3.2 (I). Why does zero-shot prompting work better on aligned models than on base (pretrained-only) models? > A. Aligned models were post-trained to interpret instructions as task specifications. Base models continue prefixes naively, often missing the task framing.

Q3.3 (H). Explain why CoT can hurt on simple factual recall tasks. > A. CoT injects intermediate reasoning text. On tasks where the answer is one step, this introduces chances of incorrect intermediate steps that bias the final answer. CoT helps when reasoning is genuinely multi-step.

Q3.4 (H). What is "lost in the middle" and how should you design prompts to mitigate it? > A. Transformers attend more strongly to the start and end of the context, weakly to the middle (Liu et al. 2023). Place key information at the top and again at the bottom; compress middle history; prefer shorter prompts.

Q3.5 (DT). A slide claims "native tool calling uses 0 prompt tokens for schemas". Explain why and what trade-offs result. > A. Modern APIs supply tool schemas via separate metadata; the model is trained to attend to them outside the prompt and emit special tool-call tokens. Trade-off: requires native API support; vendor lock-in vs prompt-based portability.

Mathematical

Q3.6 (I). With K self-consistency samples, derive the rate at which majority-vote error decays in K when each sample is correct with probability $p > 1/2$. > A. Hoeffding gives $P(\text{wrong vote}) \leq \exp(-2K(p-1/2)^2)$ — exponentially decreasing in K.

Q3.7 (H). Show that constrained decoding to a JSON Schema is equivalent to projected sampling on the regular language defined by the schema. > A. A JSON Schema corresponds to a regular language L (over tokens). Constrained decoding zeroes out tokens that would break prefix-completability to L and renormalises — exactly the projection of P onto L.

Coding

Q3.8 (I). Implement an automatic few-shot selector that picks the top-k most similar examples to the query. > A. > python > import numpy as np > def select_few_shot(q_emb, examples, k=3): > sims = [(np.dot(q_emb, e_emb), x, y) for (x, y, e_emb) in examples] > return sorted(sims, reverse=True)[:k] >

Q3.9 (H). Implement an LLM-as-judge evaluator that scores answers 1–5. > A. See Chapter 3 file Section 11 — Coding 2.

Chapter 4 — RAG

Conceptual

Q4.1 (B). Define RAG in one sentence. > A. RAG = LLM generation conditioned at inference time on retrieved external context, formally $P(y \mid q, D)$ where D is the retrieved document set.

Q4.2 (I). Why does hybrid retrieval (BM25 + dense + RRF) beat dense alone in many production systems? > A. BM25 captures rare exact-match terms (IDs, names, jargon) that embeddings smooth away. Combining the ranked lists via RRF preserves both signals.

Q4.3 (H). Why is reranking with a cross-encoder good for precision but cannot improve recall? > A. Reranking only re-orders the candidate set returned by retrieval. Items missed by retrieval cannot be recovered, so recall is bounded by retrieval. Reranking can push the truly relevant items higher within that set.

Q4.4 (H). Compare HNSW and IVF for production-scale ANN. > A. HNSW: graph-based, $O(\log N)$ query, high recall without sharding, easy upserts. IVF: cluster-and-probe, requires high nprobe for high recall (eroding speed), better at extreme scale with PQ compression. Default modern choice: HNSW.

Q4.5 (DT). Distinguish between recall and faithfulness in RAG, and explain the failure modes. > A. Recall is a retrieval metric (does top-k contain the relevant chunk?). Faithfulness is a generation metric (is the answer entailed by the context?). High recall + low faithfulness = model ignored good context. Low recall + high faithfulness = model honestly fails. Both must be measured.

Mathematical

Q4.6 (B). Compute Recall@5 for relevant=[1,4], retrieved=[3,1,2,4,5]. > A. Both relevant items in top-5 → Recall@5 = 2/2 = 1.0.

Q4.7 (I). Compute MRR for retrieved=[3,1,2,4,5] with relevant=[1,4]. > A. First relevant rank = 2 → reciprocal = 0.5. With one query: MRR = 0.5.

Q4.8 (H). Show that RRF is invariant to monotone transformations of underlying scores. > A. RRF uses ranks, not raw scores. Any monotone transformation preserves order, hence preserves ranks, hence yields identical RRF scores.

Coding

Q4.9 (I). Implement BM25 from scratch. > A. See Chapter 4 file Section 4.4 code block.

Q4.10 (H). Implement an end-to-end mini-RAG (retrieve + rerank + generate). > A. See Chapter 4 file Section 12 Task 1.

Chapter 5 — Agents

Conceptual

Q5.1 (B). Define an agent in this course's terms. > A. Agent = Model + Control Plane. The Control Plane provides tools, memory, budget, and the loop that turns LLM outputs into actions and observations.

Q5.2 (I). Distinguish workflow vs agent. > A. Workflow: developer writes the loop (deterministic, fixed control flow). Agent: LLM writes the loop (decides next step from observations).

Q5.3 (I). Sketch one cycle of a ReAct loop. > A. LLM emits Thought → Action (tool call) → environment returns Observation → state updated → next iteration or DONE.

Q5.4 (H). Compare ReAct vs Planner-Executor on robustness and cost. > A. ReAct: cheaper (one LLM context, tight feedback). PE: more robust on multi-step tasks (explicit plan, replannable on failure), but doubles LLM cost (planner + executor).

Q5.5 (DT). Why is prompt injection from tool outputs an open security problem? > A. Tool outputs can include arbitrary text the LLM treats as input. Defenses (allow-lists, regex, classifiers) are bypassable; outputs are needed by the LLM, so we cannot strip them. There is no cryptographic-style boundary yet.

Mathematical

Q5.6 (I). Show that without a stopping condition, the agent's expected step count is unbounded. > A. With per-step stop probability $p > 0$, expected steps = $1/p$ (geometric). With $p = 0$, no termination guarantee → unbounded. Hence budgets are required.

Q5.7 (H). Compare expected cost of ReAct (K steps) vs PE (planner + K executors) under per-call costs c_p and c_e. > A. ReAct $\approx Kc_e$. PE $\approx c_p + Kc_e$. PE is cheaper only if planning shortens the executor sequence by more than c_p/c_e steps.

Coding

Q5.8 (I). Implement a budget-bounded ReAct loop. > A. See Chapter 5 file Section 5.1 code block.

Q5.9 (H). Implement a JSON-Schema-validating tool wrapper. > A. > python > import jsonschema > def safe_call(tool, schema, args): > try: jsonschema.validate(args, schema) > except jsonschema.ValidationError as e: return {"error": f"schema:{e.message}"} > try: return {"result": tool(**args)} > except Exception as e: return {"error": f"runtime:{e}"} >

Chapter 6 — Fine-tuning

Conceptual

Q6.1 (B). What does PEFT stand for and why is it preferred to full fine-tuning for most teams? > A. Parameter-Efficient Fine-Tuning. Reasons: vastly reduced memory, faster training, smaller checkpoints, multiple task-specific adapters per shared base, lower risk of catastrophic forgetting.

Q6.2 (I). Explain LoRA's hypothesis in one sentence. > A. The fine-tuning weight update ΔW is approximately low-rank; learn it as BA with $r \ll \min(d, k)$.

Q6.3 (H). Why can LoRA be merged into the base weights at inference, but adapters cannot? > A. LoRA's $\Delta W = (\alpha/r)BA$ is additive in the same matrix shape; you can replace W with $W+\Delta W$ — zero extra latency. Adapters insert new modules in the residual path, requiring extra forward computation.

Q6.4 (DT). When is fine-tuning the wrong answer to a deployment problem? > A. Volatile knowledge → use RAG. Tiny dataset (<100 ex) → use few-shot. No evaluation harness → cannot detect regressions. Compliance forbids weight changes → not feasible.

Mathematical

Q6.5 (B). For a 4096×4096 matrix and LoRA $r = 4, 8, 16, 64$, compute trainable params. > A. $r(d+k) = r \cdot 8192$: 32 768 / 65 536 / 131 072 / 524 288.

Q6.6 (I). Show that initialising $B = 0$ recovers the pretrained model exactly at training start. > A. $\Delta W = (\alpha/r)BA = 0$ when $B = 0$, so $W' = W$. Outputs match the frozen base before any training step.

Q6.7 (VH). Derive $\partial L/\partial A$ and $\partial L/\partial B$ for $L = \frac{1}{2}\|y - Wx - (\alpha/r)BAx\|^2$. > A. Let $r = y - Wx - (\alpha/r)BAx$. Then $\partial L/\partial B = -(\alpha/r)r(Ax)^T$; $\partial L/\partial A = -(\alpha/r)B^T r x^T$.

Coding

Q6.8 (I). Implement merge_lora(W, A, B, alpha, r). > A. > python > def merge_lora(W, A, B, alpha, r): > return W + (alpha / r) * (B @ A) >

Q6.9 (H). Wrap a frozen nn.Linear with a LoRA layer in PyTorch. > A. See Chapter 6 file Section 6.3.E LoRALinear class.

Chapter 7 — Legal & Ethical

Conceptual

Q7.1 (B). Name the four risk tiers of the EU AI Act. > A. Unacceptable, High, Limited, Minimal.

Q7.2 (I). State the 80% rule and one of its limitations. > A. Rule: $DI = r_{\min}/r_{\max} \geq 0.8$ to avoid disparate-impact suspicion. Limitation: it ignores intersectional bias, severity of harm, and can be gamed via base-rate manipulation.

Q7.3 (H). Why is AI "not just software"? Give three concrete differences. > A. Probabilistic outputs (same input → different outputs); data dependence (model behavior changes with retraining); drift (performance degrades as world changes); adversarial fragility; emergence at scale. Hence statistical QA, monitoring, and human oversight are necessary.

Q7.4 (DT). Could RLHF create bias even when the training data is unbiased? > A. Yes. Preference annotators may share systematic cultural / political biases; RLHF then encodes them into the model. Mitigations: diverse annotators, audits, value-pluralistic frameworks.

Mathematical

Q7.5 (I). Compute disparate impact ratio: hire rate group A = 0.18, group B = 0.30. > A. $DI = 0.18/0.30 = 0.6 < 0.8$ → likely disparate impact.

Q7.6 (H). Show that perfect calibration + equalised TPR + equalised FPR is impossible when group base rates differ (Chouldechova / Kleinberg result). > A. Sketch: from $\text{TPR}_A = \text{TPR}_B$, $\text{FPR}_A = \text{FPR}_B$, and $P(Y = 1 \mid \hat{Y} = 1)$ equal across groups, accounting identities force $P(Y = 1 \mid A = a)$ to be equal across a. Contradiction with differing base rates. Hence at most two of the three can hold.

Coding

Q7.7 (I). Implement a fairness audit returning DI and per-group rates. > A. See Chapter 7 file Section 11 — Coding 1.

Q7.8 (H). Implement PSI drift detection across two categorical distributions. > A. See Chapter 7 file Section 11 — Coding 2.

Mixed-topic / integration questions

M1 (DT). RAG vs Fine-tune vs Prompting — for which scenarios is each best? > A. Prompting: cheapest; volatile, exploratory, low-stakes tasks. RAG: when knowledge is external and changes over time (docs, FAQs, code). Fine-tune: when style/behavior must be persistent and prompts plateau. Often combined: SFT for behavior + RAG for facts + prompting for runtime steering.

M2 (DT). Sketch the full inference-time pipeline of a RAG agent: which tools come from Ch3, which from Ch4, which from Ch5? > A. Ch3: prompt template (system + structured output schema), tool calling. Ch4: chunking → embedding → vector store → BM25 + dense → RRF → cross-encoder rerank → context assembly. Ch5: agent loop with retrieve / read / cite tools, budget enforcement, faithfulness check before DONE.

M3 (VH). Argue why LoRA plus DPO is a popular training stack and how each component contributes. > A. LoRA reduces trainable parameters by ~99% — affordable per-task fine-tuning. DPO replaces RLHF's PPO with a closed-form supervised loss — stable convergence on preference pairs. Combined: efficient compute, rich preference signal, low risk of catastrophic forgetting (frozen base). Industry-standard for chat / safety alignment of medium models.

M4 (DT). A user asks an LLM about a private medical document. Design an end-to-end safe deployment. > A. (Ch7) Compliance: HIPAA/GDPR-aware — explicit consent, data minimisation, retention limits. (Ch4) RAG with PII redaction at ingest, source-licence tracking, citations mandatory. (Ch5) agent loop with retrieve + disclaimer_check + escalate_to_human tools; budget; logging. (Ch3) prompt: persona = medical assistant, refusal policy in system, structured output. (Ch2 / 6) prefer SFT'd + DPO'd model with safety RLHF; LoRA only for institution-specific tone.

M5 (DT). Identify the failure modes of each of the eight major topics in this course and the recommended mitigation. | Topic | Failure | Mitigation | |—|—|—| | Bigram LM | zero unseen bigrams | Laplace smoothing | | BPE | poor multilingual coverage | byte-level BPE | | Self-attention | quadratic memory | Flash-Attention, sliding window | | RLHF | reward hacking, sycophancy | DPO, value pluralism, audits | | Prompt | injection | input sanitization, role hierarchy | | RAG | irrelevant retrieval | hybrid + reranker | | Agents | infinite loops, drift | budgets, allow-listed tools | | Fine-tune | catastrophic forgetting | LoRA, KL constraint, eval suite |

Sample-Exam-Style Questions (matched to released sample format)

> The released Example_AI_Engineering_WS20252026_Exam.pdf shows that the real exam has three exercise sections — Fundamentals (MC, 1 P), Analysis (free text, 3 P), Application (mini-case, 5 P) — with English wording. The sample contained 3 example questions; below are 12 new questions in identical style for practice. Full official-style answers follow each.

Fundamentals (MC, 1 P each)

SE-F1. Which statement best describes the role of the causal mask in a decoder-only transformer? > (a) It scales the attention scores by $\sqrt{d_k}$. > (b) It zeros out attention to future positions during training and inference. > (c) It removes padding tokens from the loss. > (d) It clips gradients during back-propagation. > > Answer: (b).

SE-F2. Which retrieval strategy is most appropriate when queries contain rare exact-match terms such as product codes? > (a) Dense vector retrieval only. > (b) BM25 (sparse) retrieval, possibly fused with dense via Reciprocal Rank Fusion. > (c) Cross-encoder reranking with no retrieval. > (d) HyDE (hypothetical document embedding). > > Answer: (b).

SE-F3. Which property is guaranteed at LoRA initialisation when $B = 0$? > (a) Loss is exactly zero. > (b) The model output equals the frozen base model's output. > (c) The optimizer is in steady state. > (d) The trainable parameter count equals the base model count. > > Answer: (b).

SE-F4. Under the EU AI Act, which of the following is classified as unacceptable risk (banned)? > (a) Spam filter using AI. > (b) Social scoring of citizens by public authorities. > (c) Customer-support chatbot. > (d) AI in video-game NPCs. > > Answer: (b).

SE-F5. Which sampling configuration gives deterministic generation? > (a) Temperature = 1.0, top-p = 0.9. > (b) Temperature = 0.0 (greedy decoding). > (c) Temperature = 2.0, top-k = 40. > (d) Random sampling without temperature. > > Answer: (b).

Analysis (free text, 3 P each)

SE-A1. Explain why multi-head attention generally outperforms single-head attention with the same total dimensionality d. > Three-bullet answer: > 1. Heads can specialise (one captures syntax, another

coreference, another positional patterns), enriching the representation. > 2. Specialisation works in parallel, so multiple linguistic relations are encoded in one layer rather than competing in a single head. > 3. Empirical benchmarks (e.g., MMLU, HellaSwag, perplexity) consistently confirm the gain at constant total dimensionality. > Trade-off: extra projection / concatenation overhead; rarely material in practice.

SE-A2. Explain why Direct Preference Optimization (DPO) is more stable than PPO-based RLHF, despite using the same human-preference data. > 1. DPO replaces the RL loop with a closed-form supervised loss on $(prompt, y_w, y_l)$. > 2. No separate reward model or critic is trained, eliminating two error sources and a moving target. > 3. The loss is convex in the log-policy for fixed reference, so optimisation behaves predictably and converges reliably. > Trade-off: less expressive than full RL when long-horizon credit assignment is required.

SE-A3. Explain the lost-in-the-middle phenomenon and propose two prompt-design countermeasures. > Transformer attention is empirically bowl-shaped: tokens at the start and the end of the context are weighted more strongly than tokens in the middle (Liu et al. 2023). Therefore key information placed mid-prompt may effectively be ignored. > Countermeasures: (1) place critical instructions at the top of the prompt and repeat the user question at the bottom; (2) summarize / compress older conversation history so the prompt stays short and the relevant material lands in the high-attention zones.

SE-A4. Explain why Retrieval-Augmented Generation (RAG) mitigates but does not fully eliminate hallucinations. > 1. RAG injects authoritative external context, raising the probability the model cites correct facts. > 2. The model can still ignore the context (faithfulness failure) or follow the context when the context itself is wrong. > 3. Retrieval recall is bounded by the corpus and the embedding model; out-of-corpus questions remain ungrounded. > Trade-off: mitigations such as cross-encoder reranking and faithfulness checks add latency and cost.

Application (mini-case, 5 P each)

SE-AP1. A start-up wants to deploy a customer-support chatbot. The product team observes that the assistant gives outdated answers about company policies that change weekly. Propose one concrete architectural change and justify it. Mention one trade-off. > Solution. Add a Retrieval-Augmented Generation (RAG) layer using the company's policy database as the retrieval corpus. The LLM is conditioned at inference time on freshly retrieved policy chunks; updates are reflected as soon as the index is re-built (e.g., daily ingest). The base model never has to be retrained. > Trade-off: retrieval and reranking add 100–500 ms latency and increase prompt-token cost; chunk-quality issues can re-introduce noise.

SE-AP2. A medical Q&A LLM frequently outputs fluent but incorrect dosage information. Identify the failure mode, name one specific evaluation metric that would catch it, and propose one mitigation. > Solution. The failure mode is hallucination. The metric faithfulness (e.g., RAGAS faithfulness, NLI-based entailment of the answer against the retrieved context) flags claims not supported by the evidence. Mitigation: ground the model with a curated medical knowledge base via RAG, set a faithfulness threshold below which answers are blocked or routed to a human pharmacist for review. > Trade-off: added latency, operational cost, and a small fraction of legitimate answers blocked.

SE-AP3. Your team has 50 000 training examples in a niche legal domain and a 70 B-parameter base model. Compare full fine-tuning vs LoRA in this scenario. Recommend one and justify. > Solution. Full fine-tuning would update ~70 B parameters → roughly 140 GB optimizer state plus model weights, requiring multi-GPU clusters and risking catastrophic forgetting of general competence on a relatively small dataset. LoRA with $r = 16$ on the Q/K/V/O matrices yields a few hundred million trainable parameters, fits on a single 80 GB GPU, keeps the base frozen so general capability is preserved, and

ships the adaptation as a tens-of-MB file. > Recommendation: LoRA. Trade-off: small accuracy gap on extremely large datasets relative to full fine-tuning, but unlikely to matter at 50 000 examples.

SE-AP4. A team integrates a tool-using LLM agent into their product. After deployment, the agent occasionally executes destructive actions (e.g. "delete user account") triggered by a malicious instruction inside a fetched web page. Identify the threat, name two mitigations, and discuss residual risk. > Solution. The threat is prompt injection through tool outputs. Mitigations: (1) apply the principle of least privilege to tools — read-only tools by default; destructive tools require explicit human approval before each call; (2) sanitise / classify tool outputs (strip or redact instruction-like patterns; run an injection classifier) before re-feeding them to the LLM. > Residual risk: sanitisation is a defence-in-depth layer, not a guarantee — sophisticated attacks can evade regex/classifier filters. Hence the durable defence is the privilege boundary; sanitisation reduces the rate of attempts.

Sample-Exam-Style Hard Questions

S1. Erläutern Sie den Self-Attention-Mechanismus mathematisch und geben Sie ein numerisches Beispiel mit n=2 Tokens und $d_k = 2$. > Stub answer: define $Q, K, V \in \mathbb{R}^{2\times 2}$; compute QK^T; divide by $\sqrt{2}$; softmax row-wise; multiply by V.

S2. Berechnen Sie für eine LoRA mit r=8 und einer 4096×4096-Matrix die Anzahl trainierbarer Parameter und vergleichen Sie sie mit dem vollen Feintuning. > $r(d+k) = 65{,}536$ vs $4096^2 = 16{,}777{,}216$ — ratio ≈ 0.39 %.

S3. Vergleichen Sie SFT, RLHF und DPO bezüglich Signaltyp, Kosten und Stabilität. > See Q2.3 above (SFT positive demos only, RLHF rich pairs but unstable + expensive, DPO rich pairs + closed-form + stable + cheap).

S4. Skizzieren Sie die End-to-End-Pipeline eines RAG-Systems und nennen Sie für jeden Schritt mindestens einen Algorithmus oder ein Verfahren. > Ingest (chunker), embed (bi-encoder), store (HNSW), retrieve (BM25 + dense + RRF), rerank (cross-encoder), assemble (token-budget packer), generate (autoregressive LLM with structured output).

S5. Erklären Sie das Lost-in-the-Middle-Phänomen und geben Sie zwei prompttechnische Maßnahmen dagegen. > Bowl-shaped attention; mitigation: place key info at top and bottom; compress middle history via summarization.

S6. Definieren Sie ein "Hochrisiko-KI-System" gemäß EU AI Act und nennen Sie zwei Pflichten. > Systems used in critical infra, education, employment, essential services, law enforcement, migration, justice. Obligations: risk-management process, technical documentation, logging, transparency to user, human oversight, accuracy/robustness/cybersecurity standards.

S7. Erläutern Sie das Prinzip "Garbage in, Garbage out" am Beispiel von LLM-Trainingsdaten. > Models learn statistical regularities of their input. Biased / low-quality / underrepresented data produces biased / low-quality / underperforming models for those domains. No architectural fix replaces good data.

S8. Geben Sie die Formel der Reciprocal Rank Fusion an und begründen Sie ihre Skaleninvarianz. > $\text{score}(d) = \sum_l 1/(k + \text{rank}_l(d))$; uses ranks not scores → invariant under monotone transformations of underlying scores.

Index of difficulty across the bank

Chapter	B	I	H	VH/DT
1	4	2	1	2
2	1	4	6	4
3	1	3	4	1
4	2	3	4	1
5	1	4	3	1
6	2	3	3	1
7	2	3	2	1
Mixed	—	—	—	5
Sample-format (SE-)	5	4	—	4
Total	18	26	23	20

Total: 87 distinct questions (74 chapter + 13 sample-format) with full answers, plus 8 sample-exam-style questions.

Final advice

- After completing this bank once, shuffle and redo only the H/VH/DT questions under timed conditions.
- Practise writing answers in English (the real exam is in English) for every chapter.
- For coding questions, type code from memory (don't copy-paste); muscle memory matters under exam pressure.
- Drill the 3-sentence structure for Analysis questions and the mechanism + justification + trade-off structure for Application questions — confirmed by the sample exam.

End of Deep Question Bank.