Large Language Models - Exam Questions

Objective Questions (MCQ/MSQ) - 20 Questions

1. Which technique helps prevent hallucinations in LLM responses?

A) Temperature adjustment
B) Retrieval Augmented Generation (RAG)
C) Fine-tuning
D) All of the above

Answer: D

2. What does “few-shot prompting” mean?

A) Using very short prompts
B) Providing examples in the prompt
C) Using multiple models
D) Reducing model parameters

Answer: B

3. Which embedding model is commonly used for semantic search?

A) Word2Vec
B) BERT
C) Sentence-BERT
D) All of the above

Answer: D

4. In function calling, what format is typically used to define functions?

A) XML
B) JSON Schema
C) YAML
D) Plain text

Answer: B

5. Which technique combines dense and sparse retrieval methods?

A) Hybrid search
B) Vector search
C) Keyword search
D) Semantic search

Answer: A

6. What does RLHF stand for in LLM training?

A) Reinforcement Learning from Human Feedback
B) Recursive Learning from Human Features
C) Rapid Learning from Historical Facts
D) Random Learning from Hybrid Functions

Answer: A

7. Which parameter controls randomness in LLM outputs?

A) Top-k
B) Top-p
C) Temperature
D) All of the above

Answer: D

8. What is the primary purpose of embeddings in RAG systems?

A) Text generation
B) Semantic similarity matching
C) Model fine-tuning
D) Data preprocessing

Answer: B

9. Which evaluation metric measures factual accuracy in LLM outputs?

A) BLEU
B) ROUGE
C) Faithfulness
D) Perplexity

Answer: C

10. In prompt engineering, what does "chain-of-thought" prompting encourage?

A) Faster responses
B) Step-by-step reasoning
C) Shorter outputs
D) Multiple perspectives

Answer: B

11. Which vector database is commonly used for RAG implementations?

A) Pinecone
B) Weaviate
C) Chroma
D) All of the above

Answer: D

12. What does “grounding” mean in LLM context?

A) Connecting responses to factual sources

B) Reducing model size

C) Improving speed

D) Increasing creativity

Answer: A

13. Which technique helps LLMs handle longer contexts?

A) Attention mechanisms

B) Context windowing

C) Memory augmentation

D) All of the above

Answer: D

14. In multimodal LLMs, what types of input can be processed?

A) Text only

B) Text and images

C) Text, images, and audio

D) Any data type

Answer: C

15. What is the purpose of system prompts?

A) Define model behavior and role

B) Provide examples

C) Set temperature

D) Control output length

Answer: A

16. Which technique helps reduce bias in LLM outputs?

A) Diverse training data

B) Bias detection tools

C) Careful prompt design

D) All of the above

Answer: D

17. What does “fine-tuning” accomplish?

A) Adapts model to specific tasks

B) Reduces model size

C) Increases speed

D) Improves general knowledge

Answer: A

18. In RAG systems, what is chunking?

A) Dividing documents into smaller pieces

B) Combining multiple documents

C) Removing irrelevant content

D) Compressing text

Answer: A

19. Which approach helps LLMs provide citations?

A) Retrieval with source tracking

B) Fine-tuning on academic papers

C) Increasing model size

D) Using higher temperature

Answer: A

20. What is the main advantage of local LLM deployment?

A) Better performance
B) Data privacy and control
C) Lower cost
D) Easier setup

Answer: B

Subjective/Scenario Questions - 20 Questions

1. RAG System Architecture

Design a complete RAG system for a company's internal knowledge base that includes document ingestion, embedding generation, retrieval, and response generation. What components and considerations would you include?

Answer: Components: **Document processor** (PDF, Word, HTML parsing), **chunking strategy** (semantic/fixed-size), **embedding model** (Sentence-BERT), **vector database** (Pinecone/Chroma), **retrieval system** (hybrid search), **LLM integration** (OpenAI/local), **response synthesis**, **source attribution**. Considerations: chunk overlap, metadata preservation, update mechanisms, evaluation metrics, and user feedback loops.

2. Prompt Engineering Strategy

Develop a comprehensive prompt engineering framework for a customer service chatbot. Include techniques for handling different query types, maintaining consistency, and ensuring appropriate responses.

Answer: Framework: **System prompts** for role definition, **few-shot examples** for common scenarios, **chain-of-thought** for complex queries, **output formatting** instructions, **safety guidelines**, **escalation triggers**. Techniques: persona consistency, context preservation, error handling, response validation, A/B testing of prompts, and continuous refinement based on user interactions.

3. LLM Evaluation Framework

Create an evaluation framework for comparing different LLMs for a specific business use case. What metrics, test datasets, and evaluation methods would you use?

Answer: Metrics: **Accuracy** (factual correctness), **relevance** (query alignment), **coherence** (logical flow), **safety** (harmful content detection), **latency** (response time), **cost** (per token/request). Methods: **Human evaluation**, **automated metrics** (BLEU, ROUGE), **adversarial testing**, **bias assessment**, **benchmark datasets**, **A/B testing** with real users, and **longitudinal performance** monitoring.

4. Function Calling Implementation

Design a function calling system that allows an LLM to interact with external APIs and databases. How would you handle function definition, parameter validation, and error handling?

Answer: Implementation: **JSON Schema** for function definitions, **parameter validation** with type checking, **authentication** management, **rate limiting**, **error handling** with graceful fallbacks, **logging** for debugging, **security** measures (input sanitization), **function registry**, **dynamic function loading**, and **response formatting** for LLM consumption.

5. Multimodal LLM Application

Create a multimodal application that processes text, images, and audio for content analysis. What architecture and processing pipeline would you design?

Answer: Architecture: **Input preprocessing** (image/audio normalization), **multimodal embedding** generation, **cross-modal attention** mechanisms, **unified representation** space, **task-specific heads**, **output synthesis**. Pipeline: format detection, quality assessment, modality-specific processing, feature fusion, context integration, and response generation with confidence scoring.

6. LLM Fine-tuning Strategy

Design a fine-tuning approach for adapting a general-purpose LLM to a specific domain (e.g., medical, legal, financial). What data, techniques, and evaluation methods would you use?

Answer: Strategy: **Domain-specific dataset** curation, **data quality** assessment, **instruction tuning** format, **LoRA/QLoRA** for efficient training, **hyperparameter** optimization, **catastrophic forgetting** prevention, **evaluation** on domain tasks, **safety** validation, **deployment** pipeline, and **continuous learning** from user feedback.

7. Bias Mitigation Framework

Implement a comprehensive bias detection and mitigation framework for LLM applications. What techniques and monitoring systems would you establish?

Answer: Framework: **Bias detection** tools (demographic parity, equalized odds), **diverse training** data, **adversarial** debiasing, **prompt** engineering for fairness, **output** filtering, **human** oversight, **continuous monitoring**, **bias** metrics tracking, **stakeholder** feedback, **regular audits**, and **mitigation** strategy updates.

8. Scalable LLM Infrastructure

Design a scalable infrastructure for serving LLMs to thousands of concurrent users. What architecture, caching, and optimization strategies would you implement?

Answer: Infrastructure: **Load balancers**, **model serving** frameworks (vLLM, TensorRT), **GPU clusters**, **auto-scaling**, **caching** layers (Redis), **request** batching, **model** quantization, **distributed** inference, **monitoring** systems, **cost** optimization, **failover** mechanisms, and **performance** tuning.

9. LLM Security Framework

Establish security measures for LLM applications including prompt injection prevention, data privacy, and output safety. What controls would you implement?

Answer: Security: **Input validation** and sanitization, **prompt injection** detection, **output filtering**, **data encryption**, **access controls**, **audit logging**, **rate limiting**, **content** moderation, **privacy** preservation (differential privacy), **secure** model serving, **vulnerability** scanning, and **incident** response procedures.

10. Hybrid Search Implementation

Implement a hybrid search system that combines keyword search, semantic search, and knowledge graphs. How would you balance and optimize these different approaches?

Answer: Implementation: **BM25** for keyword search, **dense embeddings** for semantic search, **knowledge graph** traversal, **score fusion** algorithms, **query** understanding, **result** ranking, **relevance** feedback, **performance** optimization, **index** management, **query** expansion, and **evaluation** metrics for each component.

11. LLM Agent Architecture

Design an LLM agent system that can plan, execute actions, and learn from feedback. What components and decision-making processes would you include?

Answer: Architecture: **Planning** module (goal decomposition), **action** execution engine, **memory** system (short/long-term), **tool** integration, **feedback** processing, **learning** mechanisms, **safety** constraints, **monitoring** systems, **human** oversight, **error** recovery, **knowledge** updating, and **performance** evaluation.

12. Content Moderation System

Create a content moderation system using LLMs that can detect harmful content while minimizing false positives. What approach would you take?

Answer: System: **Multi-stage** filtering (rule-based + ML), **ensemble** models, **human-in-the-loop** validation, **context-aware** analysis, **severity** scoring, **appeal** processes, **bias** monitoring, **performance** metrics, **continuous** training, **cultural** sensitivity, **transparency** reporting, and **stakeholder** feedback integration.

13. LLM Cost Optimization

Optimize costs for a high-volume LLM application while maintaining quality. What strategies would you implement?

Answer: Strategies: **Model selection** (size vs. performance), **caching** frequent queries, **request** batching, **prompt** optimization, **output** length control, **model**

quantization, **inference** optimization, **usage** monitoring, **cost** alerts, **alternative** models for simple tasks, **user** education, and **ROI** tracking.

14. Multilingual LLM System

Design a multilingual LLM system that handles translation, cross-lingual understanding, and cultural adaptation. What challenges would you address?

Answer: System: **Language detection**, **translation** quality assessment, **cultural** context adaptation, **multilingual** embeddings, **cross-lingual** transfer learning, **evaluation** in multiple languages, **bias** across cultures, **localization**, **performance** parity, **resource** allocation, **human** evaluation, and **continuous** improvement.

15. LLM Monitoring and Observability

Implement comprehensive monitoring for LLM applications including performance, quality, and business metrics. What would you track and how?

Answer: Monitoring: **Response quality** metrics, **latency** and throughput, **error rates**, **user satisfaction**, **cost** per request, **model drift**, **bias** indicators, **safety** violations, **usage** patterns, **business** KPIs, **real-time** dashboards, **alerting** systems, and **automated** reporting.

16. Ethical AI Governance

Establish governance frameworks for responsible LLM deployment. What policies, processes, and oversight mechanisms would you implement?

Answer: Governance: **Ethics** committee, **risk** assessment frameworks, **approval** processes, **impact** evaluations, **stakeholder** engagement, **transparency** requirements, **accountability** measures, **regular** audits, **incident** response, **policy** updates, **training** programs, and **compliance** monitoring.

17. LLM Testing Strategy

Create a comprehensive testing strategy for LLM applications including unit tests, integration tests, and user acceptance tests. What would you test and how?

Answer: Testing: **Unit tests** for components, **integration tests** for workflows, **performance** testing, **safety** testing, **bias** testing, **adversarial** testing, **user** acceptance testing, **A/B** testing, **regression** testing, **load** testing, **security** testing, and **continuous** testing in production.

18. Knowledge Management Integration

Integrate LLMs with existing knowledge management systems and workflows. How would you ensure consistency and maintain data quality?

Answer: Integration: **API** connections, **data** synchronization, **version** control, **access** controls, **workflow** automation, **quality** assurance, **change** management, **user** training, **feedback** loops, **performance** monitoring, **migration** strategies, and **legacy** system support.

19. LLM Personalization

Implement personalization features for LLM applications that adapt to individual user preferences and contexts. What approaches would you use?

Answer: Personalization: **User** profiling, **preference** learning, **context** awareness, **adaptive** prompting, **memory** systems, **feedback** incorporation, **privacy** preservation, **cold start** handling, **explanation** generation, **user** control, **performance** tracking, and **ethical** considerations.

20. Future-Proofing LLM Systems

Design LLM systems that can adapt to rapidly evolving technology and requirements. What architectural decisions and practices would ensure longevity?

Answer: Future-proofing: **Modular** architecture, **API** abstraction layers, **model** agnostic design, **continuous** learning capabilities, **version** management, **backward** compatibility, **monitoring** for drift, **upgrade** pathways, **technology** evaluation processes, **skill** development, **vendor** diversification, and **innovation** adoption frameworks.