Agentic AI - Complete Study Notes

1. Transformer Architecture

The Transformer is the foundational architecture behind modern LLMs like GPT, Claude, LLaMA, and Qwen. It processes input tokens in parallel instead of sequentially, using **self-attention**. Architecture flow: Input Tokens → Embedding → Positional Encoding → Encoder/Decoder Layers → Attention → Output Tokens. Each layer has Multi-Head Attention, Add & Norm, and Feed-Forward layers.

2. Attention, Q, K, V Vectors, SoftMax

Each token produces Query (Q), Key (K), and Value (V) vectors using learned matrices. Attention = Softmax((Q × K■) / √dk) × V. Softmax normalizes attention scores into probabilities that highlight important tokens.

3. Pre-training Objective - Next Token Prediction

LLMs are pre-trained using next-token prediction: given tokens [t1, t2, …, tn], predict tn+1. It builds contextual understanding through billions of token predictions.

4. Decoding Methods - Temperature, Top P, Top K, Greedy

• Temperature: controls randomness (low = deterministic, high = creative). • Top-K: samples from top K most probable tokens. • Top-P (nucleus): samples tokens whose cumulative probability ≤ P. • Greedy Decoding: always selects token with highest probability.

5. Tokenization

Text → Subword Units → IDs (tokens). Tokenizer ensures consistent mapping between text and model input. Example: “playing” → [‘play’, ‘ing’].

6. Supervised Finetuning & Instruction Tuning

Fine-tuning aligns LLMs with human instructions using supervised datasets (question → answer). It helps transform base LLM into instruction-following assistants like ChatGPT.

7. Prompting Techniques

• Zero-shot: no examples, only prompt. • Few-shot: includes few examples for context. • Chain-of-Thought (CoT): guides model step-by-step reasoning. Prompt Templates: System Prompt (sets behavior), User Prompt (task input).

8. RAG (Retrieval-Augmented Generation)

RAG enhances LLMs by combining external knowledge. Steps: Parsing → Chunking (Overlap/Semantic) → Embedding → Store in Vector DB → Query Embedding → Retrieval → Reranking → LLM Response Generation.

9. LangChain

LangChain is a Python framework that connects LLMs with tools, memory, and data sources. It simplifies building conversational agents, RAG pipelines, and chains of logic.

10. Gradio / Streamlit

Frameworks for creating user interfaces for AI models. • Gradio: Quick web demos with few lines of Python. • Streamlit: Interactive dashboards for ML apps.

11. FastAPI & Pydantic

FastAPI is a high-performance web framework for serving ML/LLM apps. Pydantic validates input/output using Python models. Used for API deployment, logging, and error handling.

12. LLaMA, DeepSeek, Claude, Qwen

• LLaMA (Meta): Open-source transformer models optimized for efficiency. • DeepSeek: Chinese LLM focused on reasoning and coding. • Claude (Anthropic): Family - Opus, Sonnet, Haiku (different model sizes for reasoning vs speed). • Qwen (Alibaba): Multilingual and code-capable transformer model family.

13. Agents & Tools

Agent = LLM + Tools (e.g., Web search, Python exec, APIs). They can act autonomously, plan tasks, and use memory.

14. Multi-Agent Architectures

Multiple agents collaborate. • Supervisor Agent: assigns tasks. • Worker/Swarm Agents: perform subtasks. • Handoffs: communication between agents. State management ensures context sharing.

15. Memory Types

• Episodic Memory: Stores recent conversations. • Semantic Memory: Stores structured knowledge. • Long-term memory ensures context continuity across sessions.

16. LangGraph

LangGraph provides visual graphs for defining agent workflows. Useful for building complex, stateful agent interactions.

17. MCP (Model Context Protocol)

MCP is a framework for connecting AI models and external tools (APIs, databases). Building an MCP server: define tool schema → register endpoints → connect to model (e.g., Claude Desktop).

18. Developer Tools

• Cursor: AI code editor for LLM pair programming. • Claude Desktop: Anthropic’s local chat & integration tool. • GitHub Copilot: AI code assistant for IDEs. • Lovable: UI-based app builder for AI workflows.

19. Memories

Persistent storage of user context across sessions. Used to personalize LLM interactions and improve long-term understanding.