Transformer Basics

A transformer is a neural network that uses self-attention so every token can look at every other token at once.

Key ideas: self-attention (Q, K, V), multi-head attention, feed-forward layers, residual connections, layer normalization,

positional encodings. Benefits: parallel training (faster than RNNs), long-range dependence modeling, great scaling.

Encoder models (BERT) are strong for classification and retrieval. Decoder models (GPT) are strong for generation.

Encoder-decoder models (T5) are strong for translation and text-to-text tasks.

Embeddings (Meaning as Vectors)

An embedding maps text (or images/audio) to a dense vector where semantic closeness ≈ geometric closeness.

Cosine similarity or dot-product measure closeness. Sentence/paragraph embeddings (e.g., MiniLM, mpnet) are used for

semantic search, clustering, deduplication, and retrieval. Normalize embeddings for cosine. Good chunking and clean text

improve retrieval quality. Embeddings are the backbone of retrieval-augmented generation (RAG).

Chunking & Preprocessing

Long documents are split into chunks so they fit in context windows and support precise retrieval.

Typical chunk size is 200–500 tokens with 10–20% overlap to preserve context across boundaries.

Clean steps: remove boilerplate, fix encodings, keep useful headings as metadata. Each chunk stores (text, embedding, metadata).

Quality chunking + meaningful metadata (title, section, page, date) make retrieval much better.

Vector Databases

A vector database stores embeddings and supports fast nearest-neighbor search (top-K).

Popular engines: FAISS (local), ChromaDB (local/dev), Pinecone/Milvus/Weaviate (managed/scale).

Index types: Flat (exact), HNSW/IVF/PQ (approximate) for speed/memory tradeoffs.

Metadata filters narrow results (e.g., source='handbook', section='policy'). Retrieval returns (ids, scores, texts).

RAG (Retrieval-Augmented Generation)

Pipeline: (1) Retrieve relevant chunks for a question, (2) Augment the prompt with those chunks, (3) Generate an answer.

Goal: ground the model in real sources to reduce hallucinations and keep knowledge fresh. Good practices:

use a strong embedder, tune chunk size/overlap, maybe re-rank retrieved chunks, and include citations (chunk ids).

Evaluate with faithfulness (does answer stick to retrieved text?) and answer relevance.

Prompting for RAG

Prompt template should clearly instruct the model to answer only from provided context and to say 'I don't know'

if the answer is missing. Keep prompts concise; include role, task, constraints, and citation format.

Consider delimiters around context. Control length with max tokens. Use few-shot examples only if they improve reliability.

Agentic AI (Tool Use & Planning)

Agents iterate: think → act (call a tool) → observe → think again, with step limits and safeguards.

Useful tools: search_docs (vector DB), web_search, run_code, call_API. Policies: cap steps, timeouts, and refuse unsafe actions.

A tiny agent for RAG: if retrieval score is low, ask a clarifying question or try another collection, then answer or gracefully fail.

LLMOps & Evaluation

Track: inputs, retrieved chunk ids, outputs, latency, and cost. Keep test questions for regression checks.

Metrics: faithfulness, answer relevancy, context precision/recall, latency, tokens. Set alerting for sudden quality drops or cost spikes.

Version prompts, models, and indices. Add PII masking/toxicity filters as guardrails. Canary/shadow deploys reduce risk when updating.

Ethics, Privacy, and Safety

Respect user privacy: collect minimum data, anonymize logs, mask PII, and follow data retention policies.

Bias & fairness: check performance across groups; monitor for harmful outputs. Safety filters for toxicity and unsafe instructions.

Be transparent about limitations and cite sources in RAG answers to build trust.

Glossary (Quick Reference)

Self-attention: weighting tokens by relevance to each other. Embedding: dense vector representing meaning.

Cosine similarity: angle-based similarity between vectors. Vector DB: database for embeddings + fast nearest neighbors.

RAG: retrieval + generation with grounding in documents. Agent: multi-step LLM that uses tools. Faithfulness: answer sticks to sources.

Metadata: extra info attached to chunks (title, section, date).