

Study Delight Publications

*From
Models
to
Meaningful
Applications*

GENERATIVE AI & LLMs

*Learn
Build
Experiment
Create*

Simplified

PART 4

Transformers, Prompting & Real-World Use



LLMs &
Transformers



Prompt
Engineering



Write a short story about a robot
and a child.



Once upon a time, a curious child
and a friendly robot explored
the world together...



RAG &
Vector Search



AI Safety
& Ethics



Concepts
Made Simple



Prompt
Examples



Use Cases
Across Industries



Build Useful
Projects



Future Ready
Skills

pythonforaiml.com

PART 4 • GENERATIVE AI & LLMs

From next-token prediction to RAG, evaluation and practical GenAI systems.

30-PAGE ROADMAP



01 Generative AI: Models That Create

16 Large Language Models (LLMs)

02 Transformers — The Big Shift

17 Prompt Engineering

03 Attention — Which Words Matter?

18 Prompt Patterns

04 Self-Attention

19 Few-Shot Prompting

05 Query, Key & Value

20 Structured Outputs

06 Transformer Blocks

21 LLM + Application

07 Encoder-Decoder

22 Hallucinations

08 Tokenization

23 Context Windows

09 Embeddings

24 Retrieval-Augmented Generation (RAG)

10 Positional Information

25 Vector Databases

11 Language Models

26 RAG Pipeline

12 Pretraining

27 LLM Evaluation

13 Fine-Tuning

28 AI Safety & Guardrails

14 Instruction Tuning

29 Build a Document Q&A Assistant

15 Preference Optimization

30 Generative AI Master Map

LEARN • PRACTICE • BUILD • GROW

Use this index as your quick navigation map while studying.

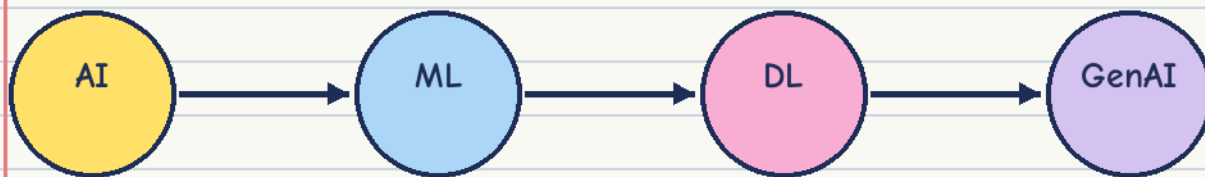
PART 4 • GENERATIVE AI & LLMs

From models that predict to systems that create

THEORY

Generative AI learns patterns in data and uses them to create new text, images, audio, code and more. The key is understanding the pipeline behind the magic.

☒ core idea



predict ☒ learn patterns ☒ generate ☒ build systems

☒ understand, then memorise

BIG IDEA

Generative AI is a pipeline: data ☒ representation ☒ model ☒ generation ☒ evaluation.

WHAT IS GENERATIVE AI?

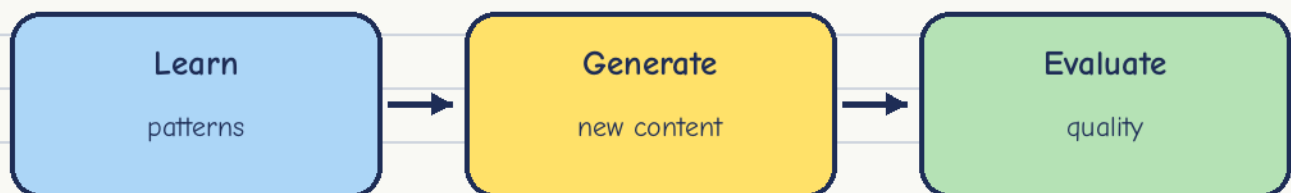
Prediction vs generation

THEORY

Generative AI produces new content instead of only assigning labels. A language model can generate the next token; image models can generate pixels matching learned patterns.

WHY?

What problem does this solve?



🔗 connect this to the previous page

REMEMBER

Prediction asks "which class?" Generation asks "what should come next?"

GENERATIVE MODELING

Learning the structure of data

THEORY

A generative model tries to capture useful structure in a data distribution. After training it can generate samples that are plausible under that learned structure.

Generation depends on learned patterns plus the



? ask: why does this work?

EXAM TIP

Generation depends on learned patterns plus the decoding or sampling strategy.

TRANSFORMERS • THE BIG SHIFT

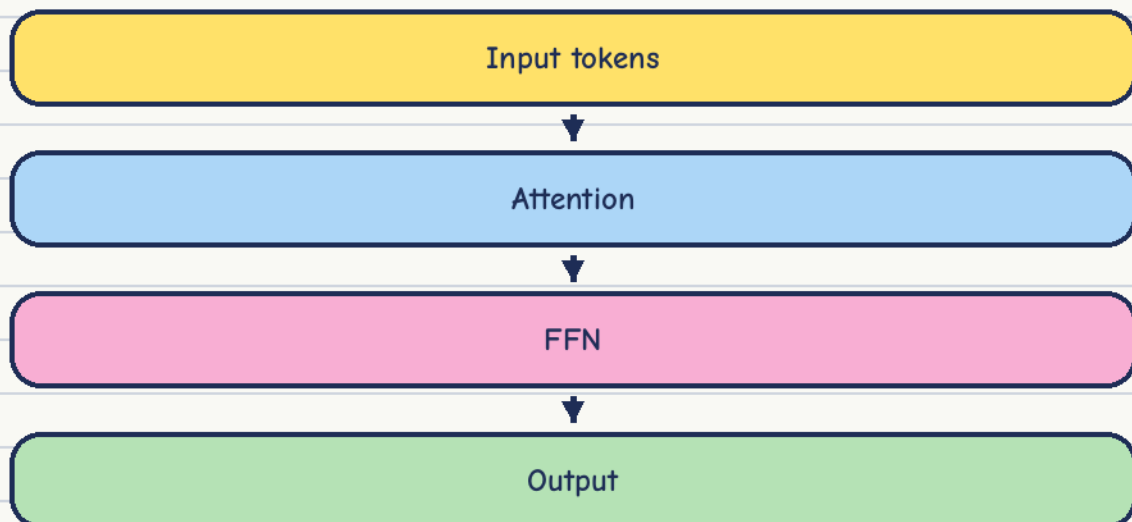
Attention changed sequence modeling

THEORY

Transformers model relationships between tokens using attention. Repeated blocks combine attention, feed-forward layers, normalization and residual connections.

COMMON MISTAKE

Fluent output is not automatically verified truth.



📌 useful for interviews

TRY THIS

Attention gives the network a flexible way to decide which context is important.

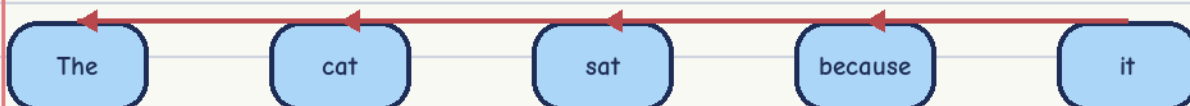
ATTENTION • INTUITION

Which words should matter?

THEORY

Attention answers: when processing this token, which other tokens should matter most? Context lets a token borrow useful information from other tokens.

☒ learn the intuition before memorising the formula



The token "it" can use surrounding context to build a better representation.

! common beginner trap

WATCH OUT

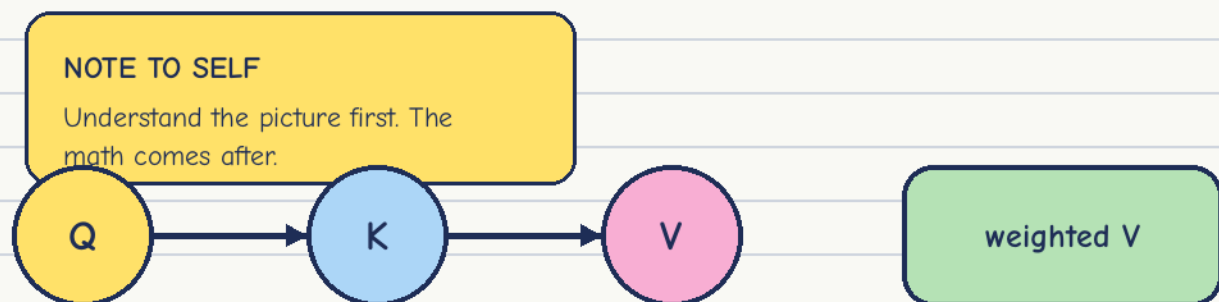
Each token becomes context-aware by mixing information from other tokens.

SELF-ATTENTION

Tokens attend to tokens

THEORY

In self-attention, queries, keys and values come from the same sequence. Similarity between a query and keys creates weights that mix the values.



📝 write your own example

KEY TAKEAWAY

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d_k}) V$$

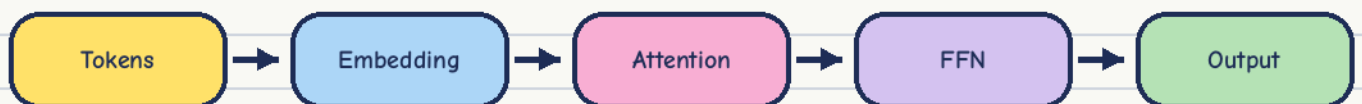
QUERY • KEY • VALUE

A simple search analogy

THEORY

Imagine a library. A query describes what you need, keys describe what each item contains, and values contain the information retrieved. Matching produces attention weights.

☒ core idea



☒ understand, then memorise

BIG IDEA

Strong query-key match ☒ larger weight ☒ more of that value contributes.

TRANSFORMER BLOCK

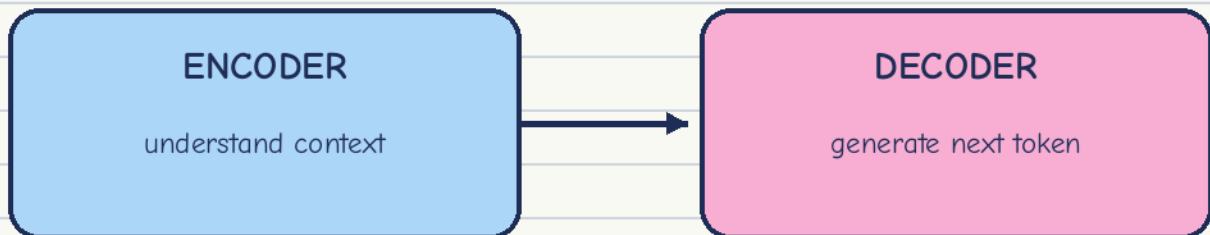
One block, repeated many times

THEORY

A transformer block commonly combines multi-head self-attention, a feed-forward network, normalization and residual paths. Stacking blocks gradually refines representations.

WHY?

What problem does this solve?



☒ connect this to the previous page

REMEMBER

Tokens ☒ embeddings ☒ attention + FFN ☒ repeat ☒ output probabilities

ENCODER vs DECODER

Understanding context vs generating

THEORY

Encoder-style models can use context from both directions. Decoder-style models generate autoregressively using earlier tokens. Many modern chat LLMs are decoder-only.

Encoder = representation. Decoder = sequential g

I love generative AI!



Tokens are pieces of text, not necessarily complete words.

? ask: why does this work?

EXAM TIP

Encoder = representation. Decoder = sequential generation. Both can be adapted.

TOKENIZATION

Text becomes tokens

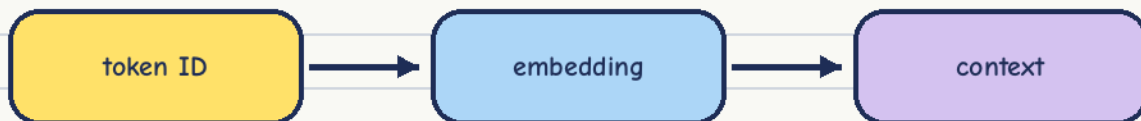
THEORY

Models do not read words exactly like humans. Text is split into tokens: whole words, subwords, punctuation or symbols. Tokenization affects cost and context length.

COMMON MISTAKE

Fluent output is not automatically verified truth.

TEXT → TOKENS → VECTORS



📌 useful for interviews

TRY THIS

Take a sentence and mark likely word pieces. Which unusual words may require several tokens?

EMBEDDINGS

Token IDs become vectors

THEORY

A token ID is only an integer. An embedding maps it into a dense vector learned during training. These vectors provide useful numerical representations to the neural network.

☒ learn the intuition before memorising the formula

dog

bites

man

☒

man

bites

dog

☒

Same tokens + different order = different meaning.

! common beginner trap

WATCH OUT

Token ID ☒ vector ☒ transformer layers ☒ contextual representation.

POSITIONAL INFORMATION

Order carries meaning

THEORY

Attention alone does not inherently encode sequence order. Positional methods give the network information about where tokens occur, so word order can change meaning.

NOTE TO SELF

Understand the picture first. The

math comes after

the		0.42
model		0.20
is		0.14
great		0.08
...		0.16

📝 write your own example

KEY TAKEAWAY

Meaning depends on both the tokens and their order.

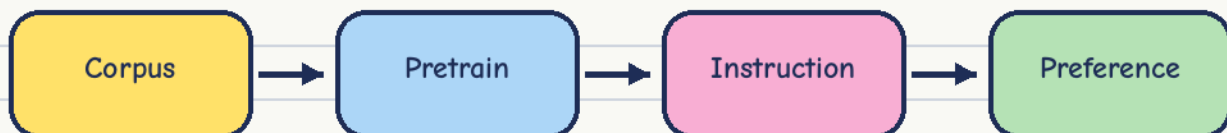
LANGUAGE MODELING

Predict the next token

THEORY

A language model learns probabilities over token sequences. A common objective is next-token prediction: given earlier tokens, predict the next one.

☒ core idea



Broad capability ☒ useful behavior ☒ better alignment

☒ understand, then memorise

BIG IDEA

$P(x_t | x_1 \dots x_{(t-1)})$ = probability of the next token given previous tokens.

PRETRAINING

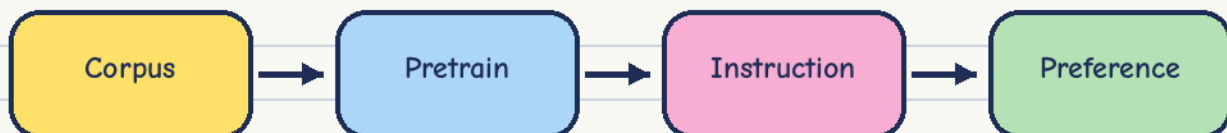
Broad learning stage

THEORY

Pretraining exposes a model to a huge corpus and updates parameters to reduce a training objective. The aim is to learn reusable representations and patterns.

WHY?

What problem does this solve?



Broad capability ☒ useful behavior ☒ better alignment

☒ connect this to the previous page

REMEMBER

Corpus ☒ tokenize ☒ batches ☒ forward pass ☒ loss ☒ backprop ☒ update

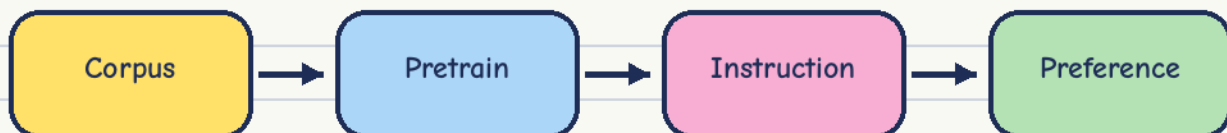
FINE-TUNING

Specializing a pretrained model

THEORY

Fine-tuning continues training on a narrower dataset. It nudges a pretrained model toward a task, style or domain. Full fine-tuning is not always necessary.

Pretraining builds broad capability; fine-tuning



Broad capability ☒ useful behavior ☒ better alignment

? ask: why does this work?

EXAM TIP

Pretraining builds broad capability; fine-tuning specializes behavior.

INSTRUCTION TUNING

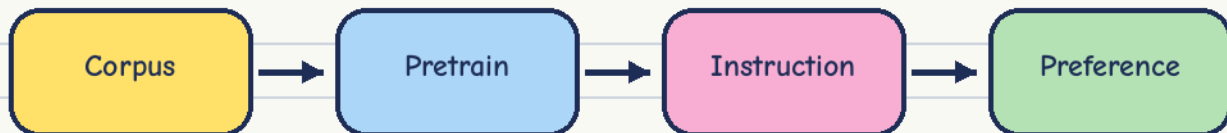
Teaching the model to follow tasks

THEORY

Instruction tuning uses examples formatted as instructions and desired responses. It improves the model's ability to respond to intent instead of merely continuing raw text.

COMMON MISTAKE

Fluent output is not automatically verified truth.



Broad capability ☒ useful behavior ☒ better alignment

☒ useful for interviews

TRY THIS

The model learns how to respond to a task, not just how text continues.

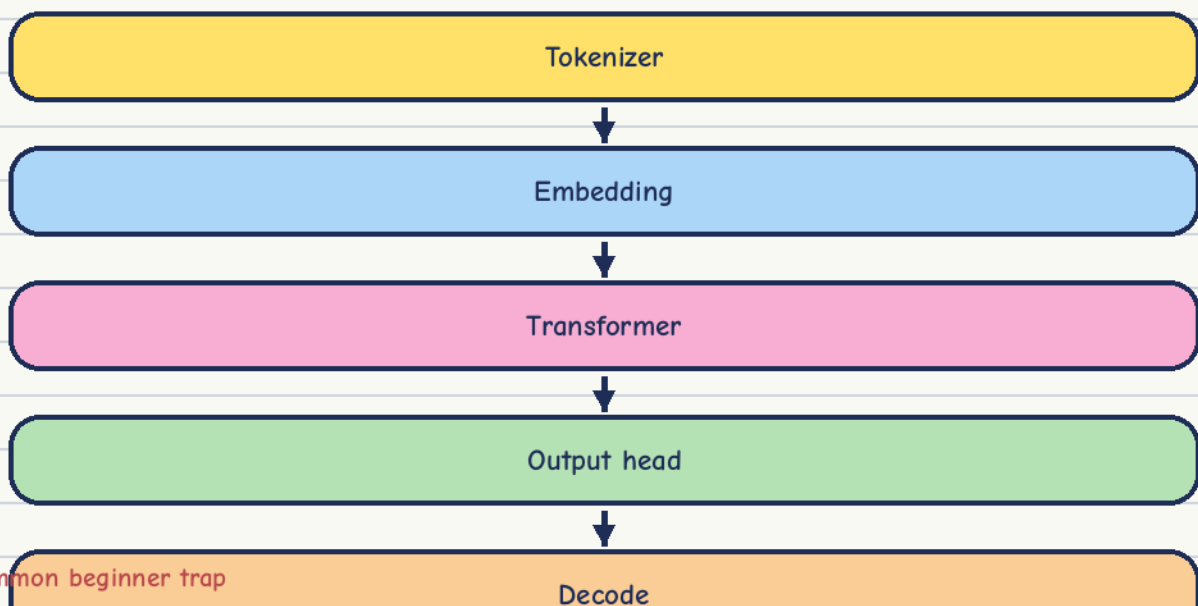
RLHF & PREFERENCE OPTIMIZATION

Aligning behavior

THEORY

Preference methods use comparisons or preference signals to make responses more useful and aligned with desired behavior. Modern systems use several variants of preference optimization.

☒ learn the intuition before memorising the formula



WATCH OUT

Preference training changes behavior; it does not guarantee factual correctness.

WHAT IS AN LLM?

Large Language Model

THEORY

An LLM is a neural language model trained at large scale. Scale can involve parameters, data, compute, context length and training diversity.

NOTE TO SELF

Understand the picture first. The

VAGUE Explain transformers.



BETTER Explain self-attention to a beginner using one analogy, one equation, and five short bullets.

✎ write your own example

KEY TAKEAWAY

LLM ✎ tokenizer + embeddings + transformer stack + output head + decoding.

PROMPTING BASICS

Give the model a clear job

THEORY

A prompt should reduce ambiguity by specifying goal, context, inputs, constraints and output format. A useful pattern is context ✕ task ✕ data ✕ constraints ✕ format.

✕ core idea

VAGUE Explain transformers.



BETTER Explain self-attention to a beginner using one analogy, one equation, and five short bullets.

✕ understand, then memorise

BIG IDEA

If a human teammate would ask for clarification, your prompt probably needs more context.

PROMPT PATTERNS

Simple patterns that work

THEORY

Useful patterns include zero-shot instructions, few-shot examples, decomposition, critique-and-revise, structured output and role/context framing.

WHY?

What problem does this solve?

VAGUE Explain transformers.



BETTER Explain self-attention to a beginner using one analogy, one equation, and five short bullets.

🔗 [connect this to the previous page](#)

REMEMBER

Rewrite a vague prompt using objective + context + constraints + output format.

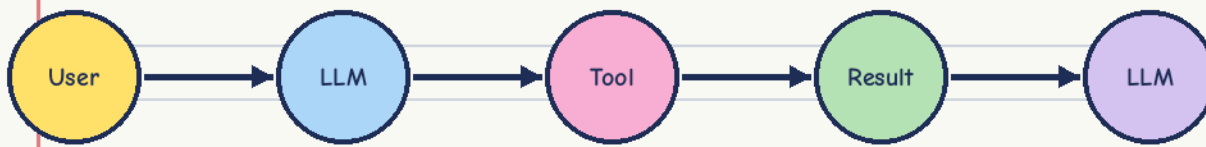
FEW-SHOT PROMPTING

Examples teach the pattern

THEORY

Few-shot prompting places examples inside the prompt without changing model parameters. Examples can demonstrate classification, extraction, formatting or transformation.

Examples teach the pattern; instructions define



The model requests an action; the application controls execution.

? ask: why does this work?

EXAM TIP

Examples teach the pattern; instructions define the goal.

STRUCTURED OUTPUT & TOOLS

LLM + application

THEORY

LLM applications often need reliable structure. Schemas can constrain fields and types. Tool calling lets a model request an external action such as search or database lookup.

COMMON MISTAKE

Fluent output is not automatically verified truth.

LLM says:

"I am certain this fact is correct."

ENGINEER ASKS:

"Where is the evidence?"

☑ useful for interviews

TRY THIS

User ☑ LLM ☑ tool request ☑ tool ☑ result ☑ LLM ☑ final response

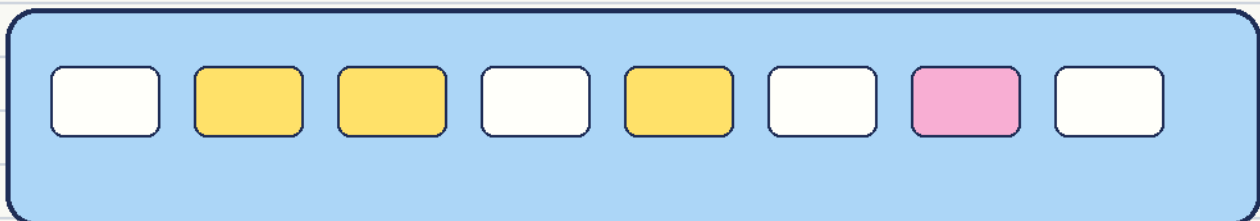
HALLUCINATIONS

Fluent does not mean factual

THEORY

A language model can produce convincing text that is unsupported or wrong. Generation follows learned patterns; it is not a universal truth database.

☒ learn the intuition before memorising the formula



Relevant

irrelevant text

Good context = the right information, in the right order.

! common beginner trap

WATCH OUT

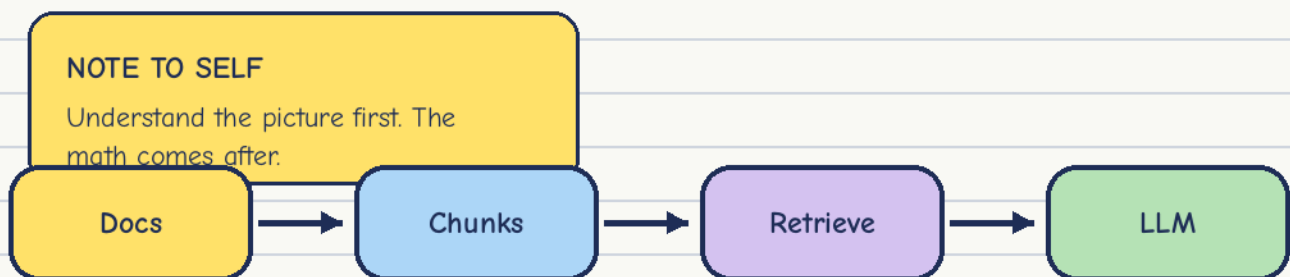
Confidence of wording is not proof of correctness. Use retrieval, tools, validation and review.

CONTEXT WINDOWS

Relevant context beats more context

THEORY

A context window is the amount of tokenized information a model can process. Long context is useful, but irrelevant text can dilute attention and increase cost.



RAG = retrieve evidence first, then generate.

✎ write your own example

KEY TAKEAWAY

More context ✎ better context. Prefer relevant, well-structured context.

RAG • RETRIEVAL AUGMENTED GENERATION

Connect the LLM to knowledge

THEORY

RAG retrieves relevant external documents and places them into the prompt. This reduces dependence on what the model may or may not have learned during pretraining.

📌 core idea



Nearest chunks become retrieval candidates.

📌 understand, then memorise

BIG IDEA

RAG separates knowledge retrieval from language generation.

VECTOR DATABASES

Semantic search with embeddings

THEORY

Embeddings map text into vectors. A vector database stores vectors and supports similarity search. Metadata such as source, date and permissions makes retrieval safer and more useful.

WHY?

What problem does this solve?



Debug retrieval before changing the prompt.

🔗 connect this to the previous page

REMEMBER

Store each chunk with source + section + embedding + access metadata.

RAG PIPELINE • STEP BY STEP

Build the retrieval layer

THEORY

1 Load documents. 2 Clean and chunk. 3 Create embeddings. 4 Index vectors. 5 Embed query. 6 Retrieve matches. 7 Build context. 8 Generate and cite.

When answers are wrong, inspect retrieved chunks

Factuality

Relevance

Format

Safety

Latency

Cost

Evaluate the application, not just the base model.

? ask: why does this work?

EXAM TIP

When answers are wrong, inspect retrieved chunks before blaming the LLM.

LLM EVALUATION

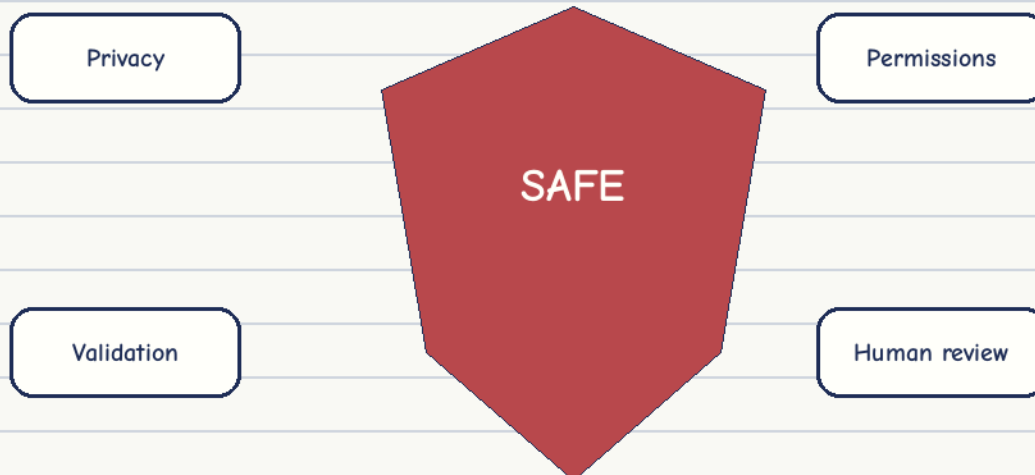
Measure the whole system

THEORY

Evaluate factuality, relevance, instruction following, format validity, latency, cost and safety. Use representative, edge and adversarial test cases.

COMMON MISTAKE

Fluent output is not automatically verified truth.



📌 useful for interviews

TRY THIS

Quality + reliability + safety + latency + cost = useful production system.

RESPONSIBLE GENERATIVE AI

Safety is part of engineering

THEORY

Protect user data, respect permissions, reduce harmful outputs and communicate uncertainty. Design for privacy, access control, prompt-injection resistance, logging and human oversight.

📌 learn the intuition before memorising the formula

Documents

Chunks

Embeddings

Vector DB

Retriever

Prompt

LLM

Sources

FINAL CHALLENGE

! common beginner trap

WATCH OUT

Build guardrails into the workflow, not only into the prompt.

GENAI MINI PROJECT

Build a document Q&A assistant

THEORY

Project: create a small RAG assistant for PDFs or notes. User asks a question; the system retrieves relevant chunks and generates a grounded answer with sources.

NOTE TO SELF

Understand the picture first. The math comes after.

✎ write your own example

FINAL CHECK

Documents ✎ chunks ✎ embeddings ✎ vector store ✎ retrieval ✎ prompt ✎ LLM ✎ cited answer