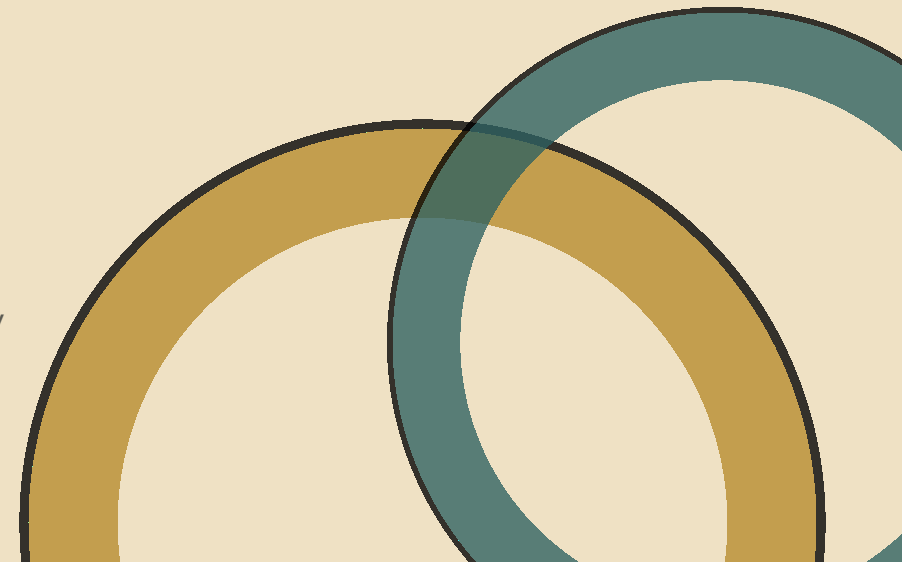


What an agent is, and the engine underneath

Chatbot, workflow, agent • the compass •
tokens, the desk and the dice



The moment this course is about

“I didn’t see every step. How much of this can I trust?”

Preface

Thesis: an agent is only as trustworthy as the signal you can use to verify it.

By the end of today you can

Four objectives

1. Classify a system as chatbot, workflow or agent, by **who owns the control flow**
2. State the four compass bearings and a design fork each one decides
3. Explain tokens, the context window as a finite desk, and why two identical runs differ
4. Compute p^n : whole-run success from per-step reliability

Part 1 • Chapter 1 (free)

What is an agent?

The working definition

An agent runs tools in a loop to achieve a goal.

Simon Willison's one-sentence definition, adopted in Chapter 1

Tools

connect the model to the world

Loop

lets it react to what it finds

Goal

ends the loop: a stopping condition

What is genuinely new

“In an agent, the sequence is decided by the model, at runtime, in response to what it observes.”

In every program you have written so far, you decided the sequence, in advance, in code.

Chapter 1, “The Agent Idea”

Why it became practical: three ingredients

Fluency was never the missing piece

1 • Post-training

Models take instructions

An agent is a delegation;
it needs a worker that
accepts the job

2 • Function calling

A bridge to action

“The model proposes;
your code disposes.”

3 • Context window

Working memory grew

Instructions, history and
the latest result fit
together

Chapter 1 • deliberately without dates: ask which ingredient just got better

Decide before the next slide

Which one is the agent?

A

Q&A assistant: paste a function, get an explanation; then it waits

B

Ticket pipeline: classify
→ fetch policy → draft
reply → check against
rubric → human queue.
Same four steps, every
time

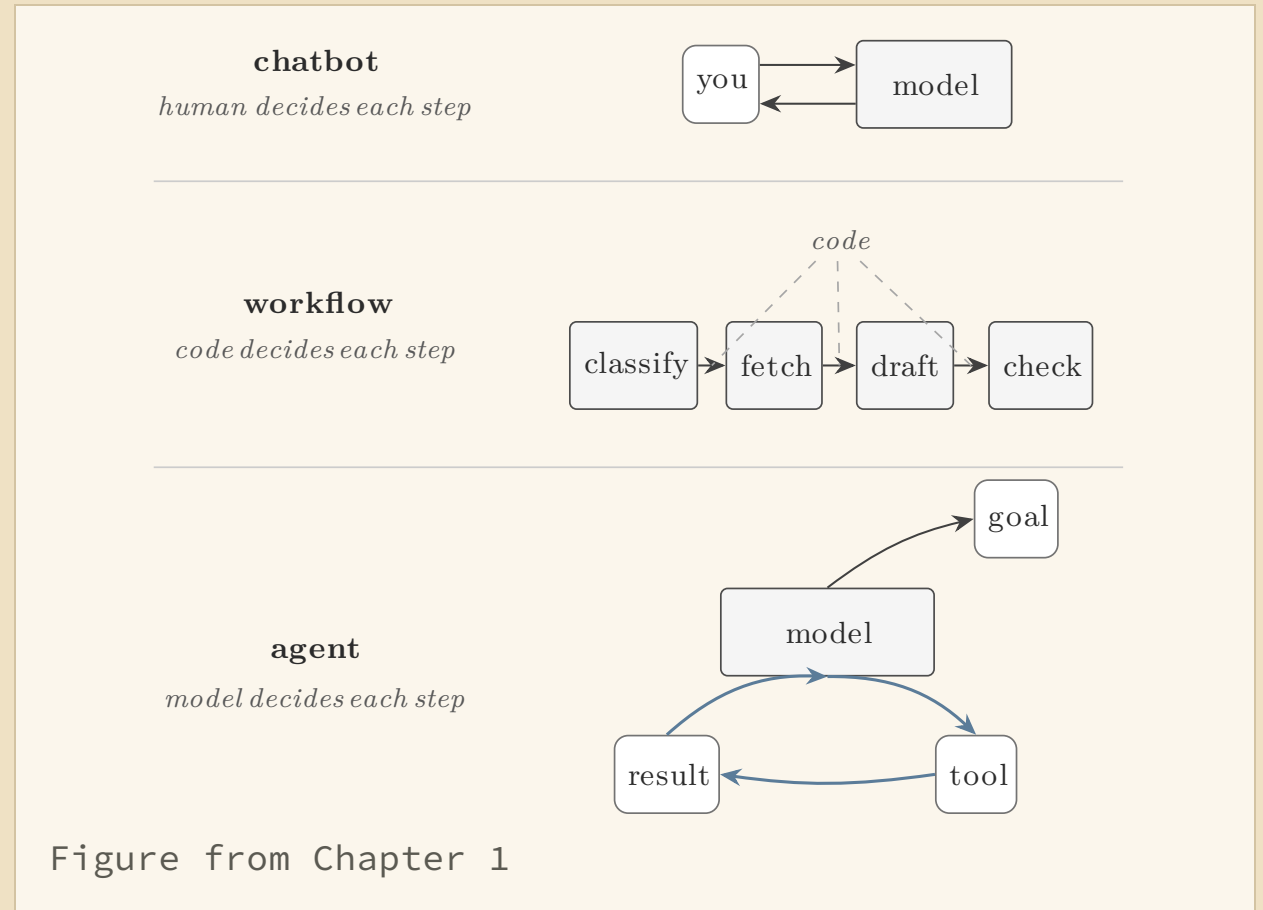
C

“The checkout tests are failing. Fix them.” It searches, reads, edits, reruns, and stops when green

All three may run on exactly the same model.

Who decides what happens next?

- **Chatbot:** you do, one message at a time
- **Workflow:** your code does, on paths drawn in advance
- **Agent:** the model does, as it goes



The litmus test for the rest of the course

“If you can confidently draw the control-flow diagram before the request arrives, you are looking at a workflow. If the diagram can only be drawn in hindsight, once the model has reacted to what it found, you are looking at an agent.”

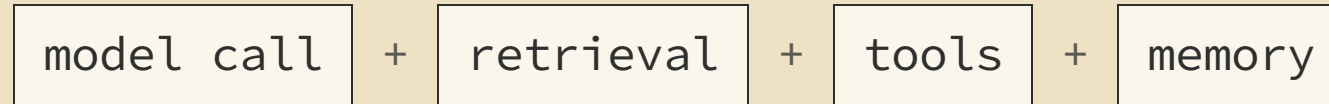
Chapter 1, “Agents, Workflows, and Chatbots”

Use it on any system, whatever its label says.

Relabeling a pipeline as an “agent” is **agent-washing**.

Same atom, two arrangements

The augmented LLM



- **Workflow:** several atoms, wired together by your code along a fixed path
- **Agent:** one atom inside a loop, deciding what to do on each pass
- **Chatbot:** the bare model call, augmentations mostly unused

Not three boxes: a dial

How much of the control flow does the model own?

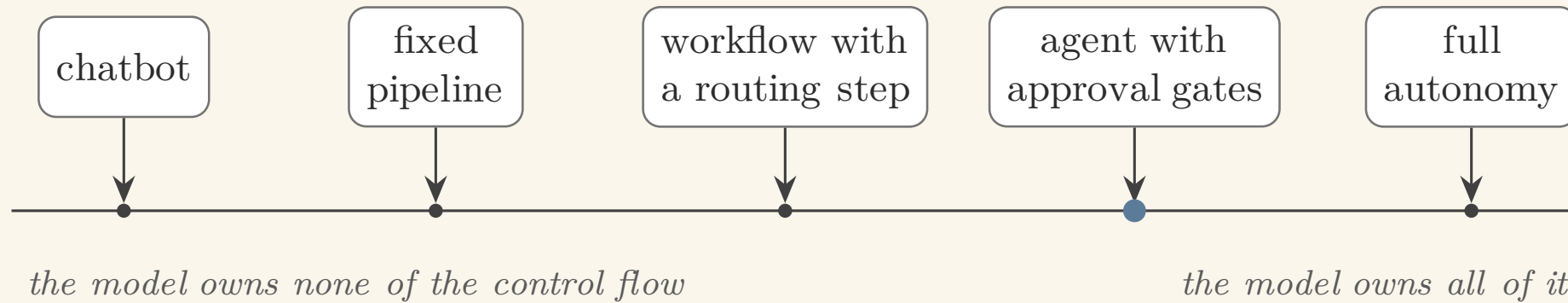


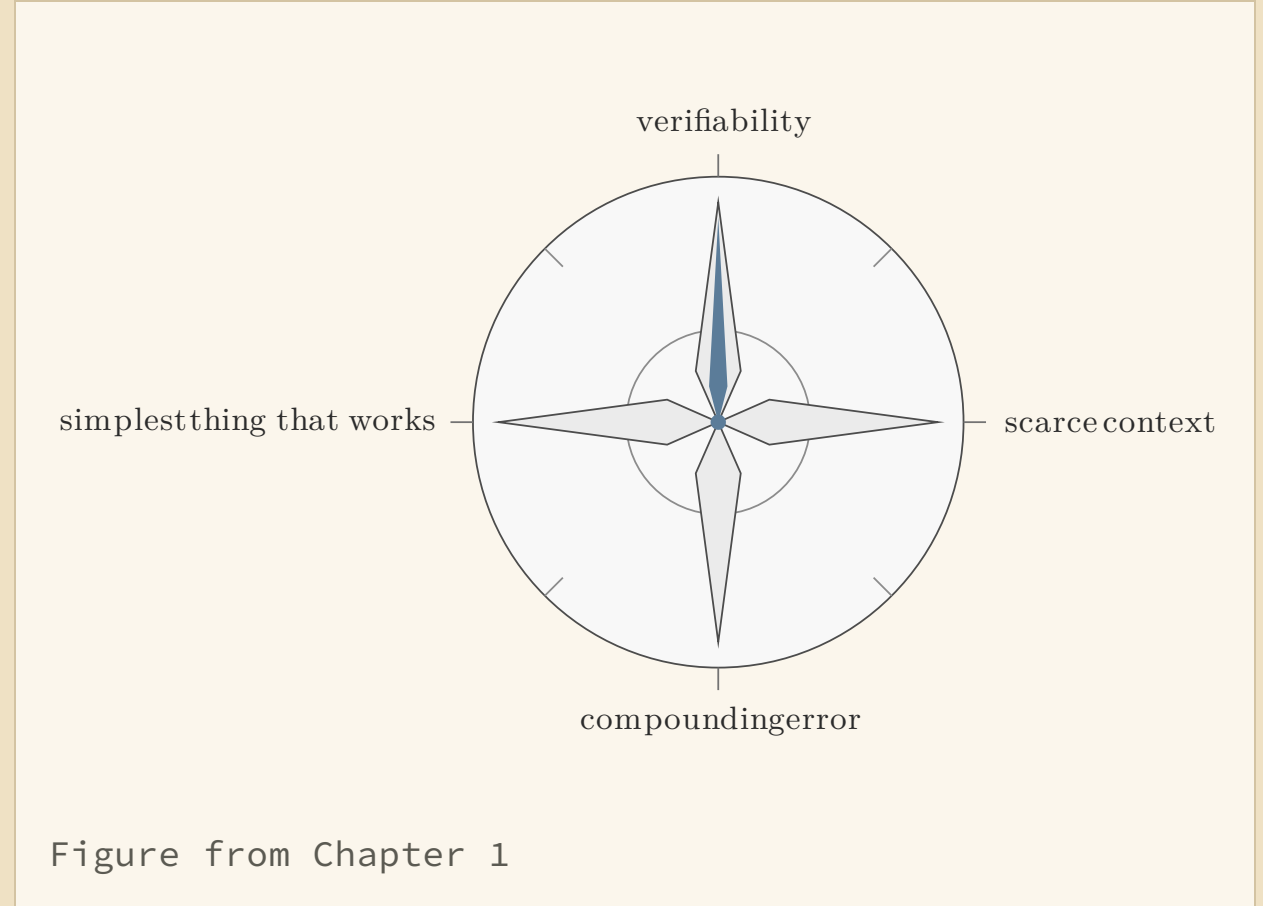
Figure from Chapter 1

- Much of production: **an agent that pauses for human approval** before anything consequential
- The far end buys capability and pays in **blast radius**
- The common shape: “a workflow shell with one or two genuinely agentic steps inside it”

The compass the course steers by

Four bearings

1. Verifiability
2. Scarcity of context
3. Compounding error
4. The simplest thing that works



A compass decides forks

One design fork per bearing

Bearing	The fork it decides	The question to ask
Verifiability	More autonomy or less	What signal tells you it worked?
Scarcity of context	Richer context or leaner	Does this deserve the desk?
Compounding error	A long free run or checkpoints	What is p^n for this chain?
Simplest thing	An agent or a plain script	Which rung solves it?

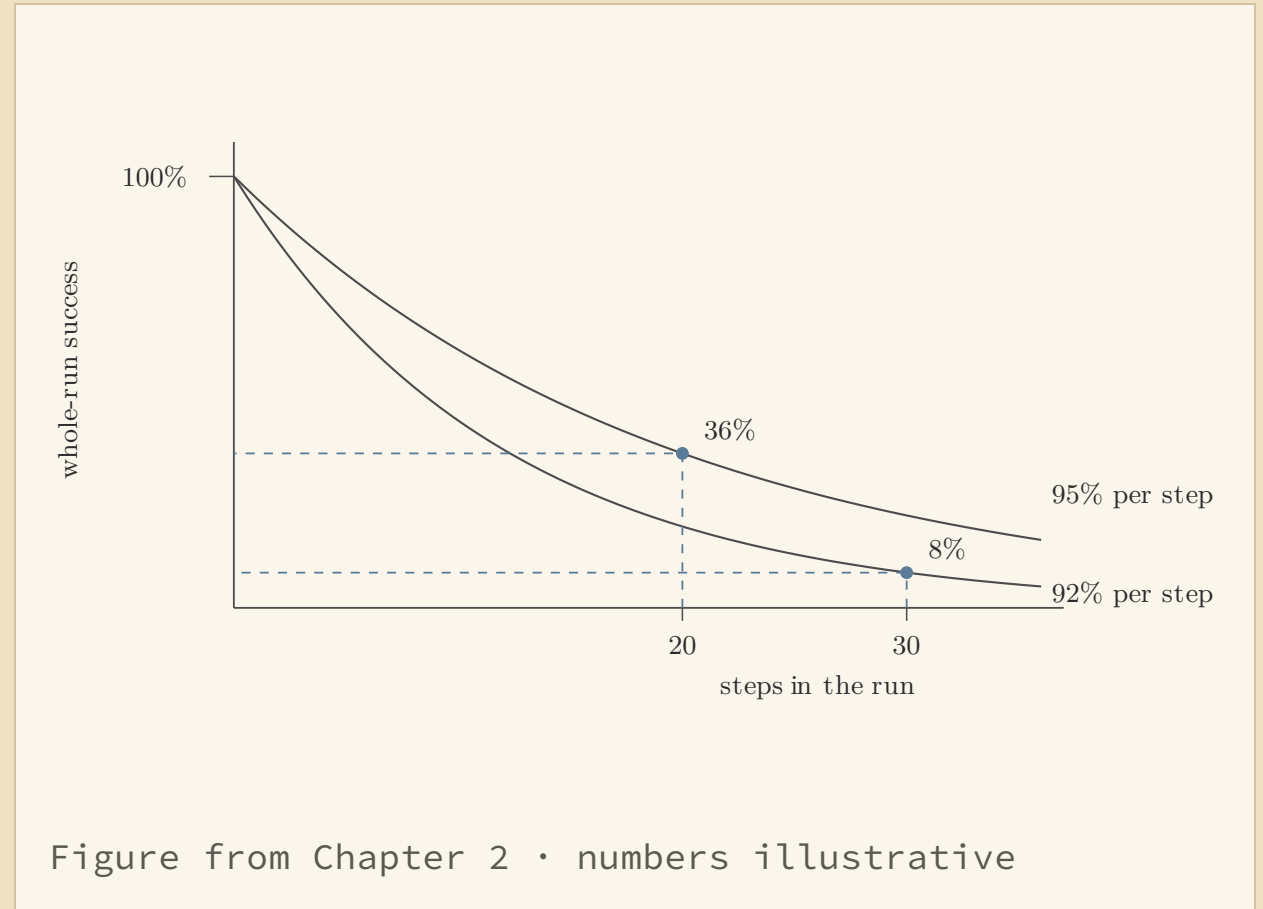
Three forks quoted from Chapter 1: “more autonomy or less, another agent or a plain script, richer context or leaner”; the checkpoint fork follows from its compounding-error bearing

Worked example • compounding error

Reliability multiplies

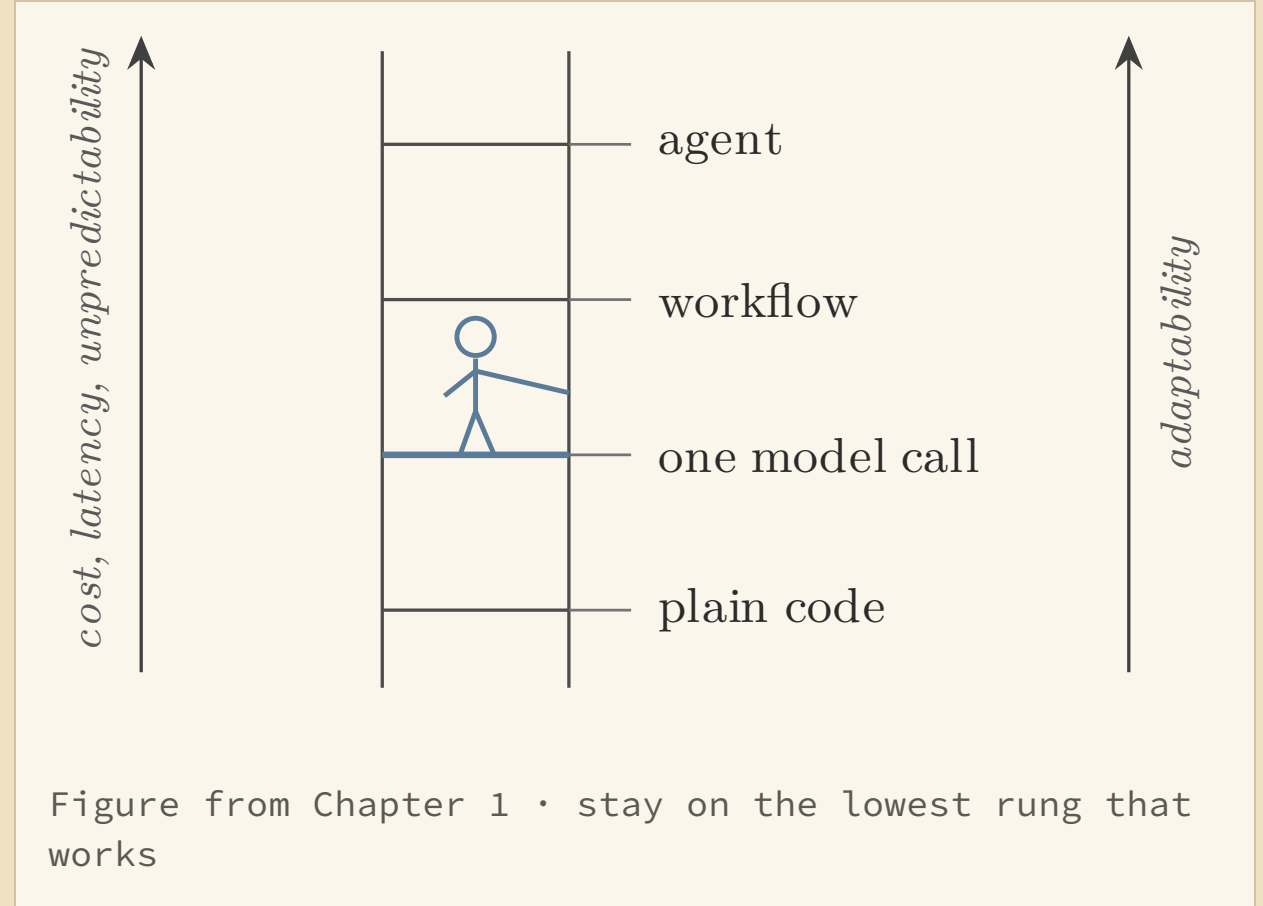
$$0.95^{20} \approx 36\%$$

p per step	n	whole run
99%	10	90%
95%	10	60%
90%	10	35%



Do you even need an agent?

- More calls, each re-sending a growing history: **money**
- Sequential calls: **latency**
- Different path each run: **nondeterminism**
- A wrong *action*, then ten more built on it
- Wider attack surface; a whole **harness** to build



Calibrate on the book's pair

Workflow

"Route each incoming email to billing, technical, or sales and draft a first reply"

Knowable steps, high volume

Agent

"Make this failing test pass in a codebase you have never seen"

Unknowable steps, built-in verifier

"An agent, then, is a cost you pay for adaptability you can name. If you cannot name the adaptability, keep your money."

Chapter 1, "Do You Even Need an Agent?"

In class • pairs • 15 minutes

Should this be an agent?

1. Each pair writes down three requests from their own work or studies
2. Run each through the browser tool:
aiagentsengineered.com/tools/should-this-be-an-agent/
3. For each verdict, name who owns the control flow and the verifying signal
4. Copy the result as Markdown and submit all three

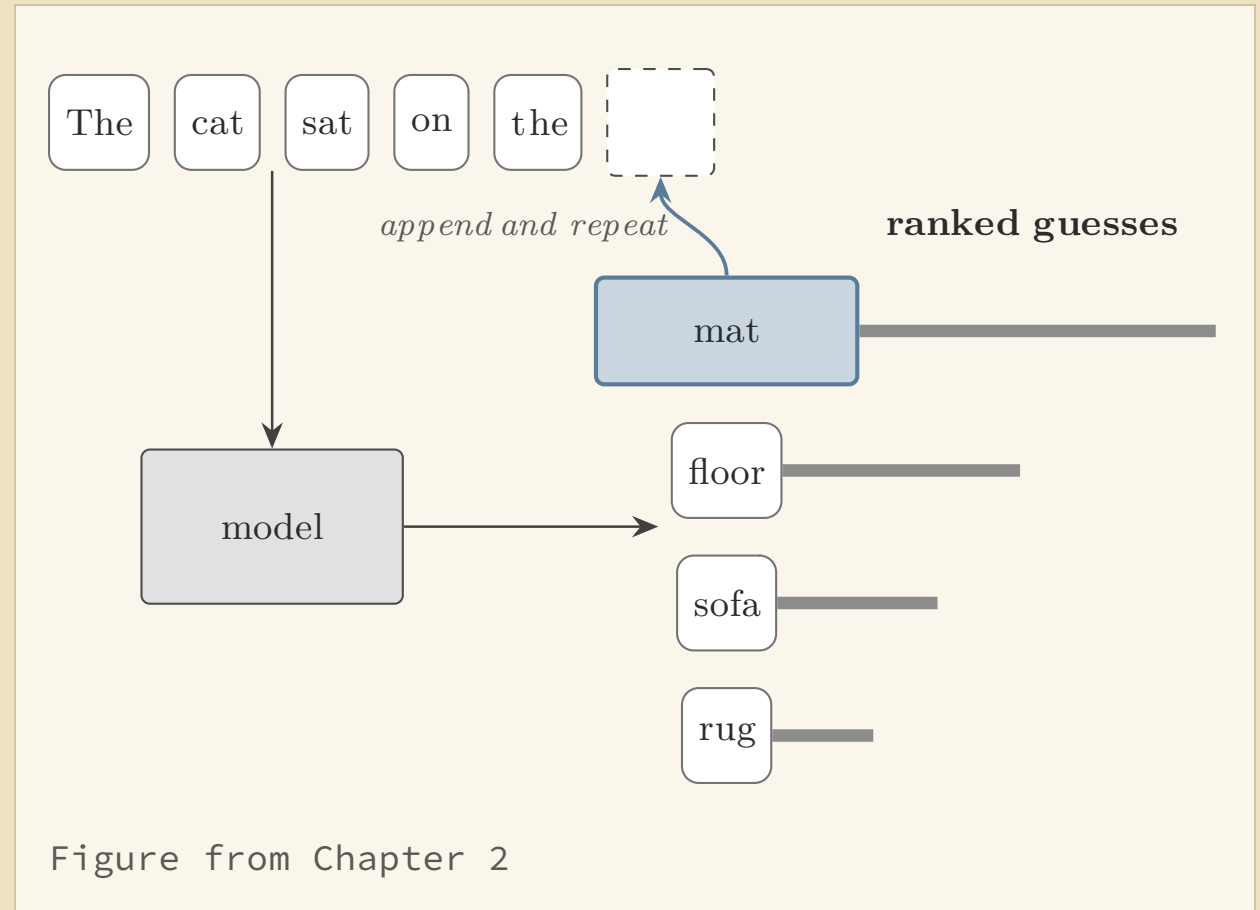
Runs in the browser • no account, no API key

Part 2 · Chapter 2, sections 1-3 (free)

The engine underneath

"The capital of France is ____"

- In: text so far. Out: a probability for every possible next **token**
- Pick one, append it, run again: **autoregressive generation**
- "the model never drafts its answer in advance"



Keep these two apart

Training vs inference

Training

- Once, by the model's makers
- Sets the **weights**
- Ends at a **training cutoff**

Inference

- Every use: every chat, every agent step
- Weights frozen
- “the model does not learn from your conversations”

Parametric knowledge (in the weights) vs **context** knowledge (on the desk now): the model is more reliable with the second.

The model's world has a grain

annoying + ly · un + happi + ness

- **Subword tokenization:** common words whole, rare ones in pieces
- A fixed lookup table: “the network never sees your characters”
- So counting letters or spelling backwards is a memory task
- Code, JSON and many non-English languages cost more tokens

Chapter 2 • example splits as in the book; actual splits differ per tokenizer

The context window is a desk

"There is no drawer, no shelf, no second desk."

One call, tokens on the desk

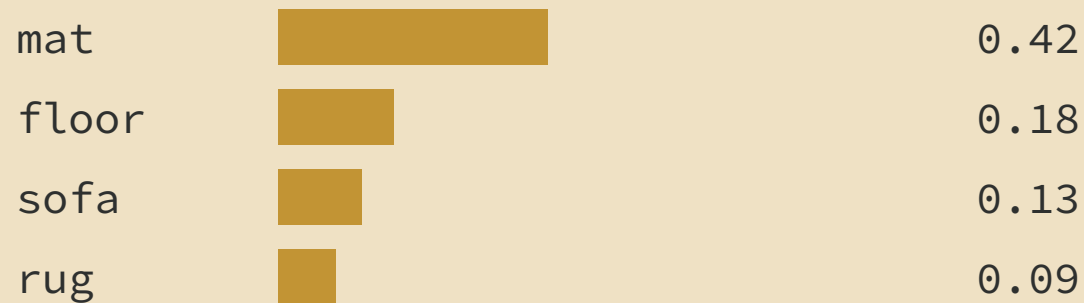


- Over 50,000 tokens read, and paid for, before the first word
- Stateless: the whole transcript goes back on the desk every call
- Too big: an error, or **silent truncation**
- Mid-pile facts get **lost in the middle**

Chapter 2 • token counts are the book's made-up but realistic sketch

"By default, the machinery rolls dice."

The cat sat on the ...



Probabilities invented, shape typical (Chapter 2)

- **Greedy:** always the top token; deterministic, flat, loops
- **Sampling:** a weighted lottery, one draw per token
- **Temperature:** low buys consistency, high buys variety; correctness at neither end
- **Top- p :** the pool sizes itself to the model's confidence

Why two identical runs differ

80

distinct answers from 1,000 runs of one prompt at **temperature zero**; identical for the first 102 tokens

He et al., Thinking Machines Lab (2025), as reported in Chapter 2 • illustrates the effect, not a constant

- Source 1: sampling, any temperature above zero
- Source 2: hosted endpoints batch many users; numerics shift with batch size (**batch invariance** missing)
- Near-ties at the top flip; the text forks for good
- Habit: test **properties** of the output, never exact strings

Five things to keep

- An agent runs tools in a loop to achieve a goal;
the model owns the control flow
- Draw the flowchart before the request arrives?
Then it is a workflow
- Compass: verifiability, scarce context,
compounding error, simplest thing
- The model predicts tokens on a finite desk it re-
reads every call
- It writes by lottery, so check outcomes, not
strings

Before next week

Homework and reading

Tokenizer experiment

- Ask any free chat interface or local model to count a letter in a word and to spell a word backwards
- Explain the outcome with subword tokenization, in one page
- Reading quiz on Chapters 1 and 2 (§1–3)

Reading for week 2

- Ch. 2, The Engine, §4–6 free
- Ch. 3, The Agent Loop full book
- App. A, A Minimal Agent, Annotated full book

This week's reading, all free: Preface · Ch. 1 · Ch. 2 §1–3

Engineering AI Agents • Week 1

What signal tells you it worked?

Slides from the teaching kit of *AI Agents, Engineered* by Enrique Gutiérrez, CC BY 4.0 • aiagentsengineered.com/teach/

Reuse, adapt and translate with attribution • creativecommons.org/licenses/by/4.0/

