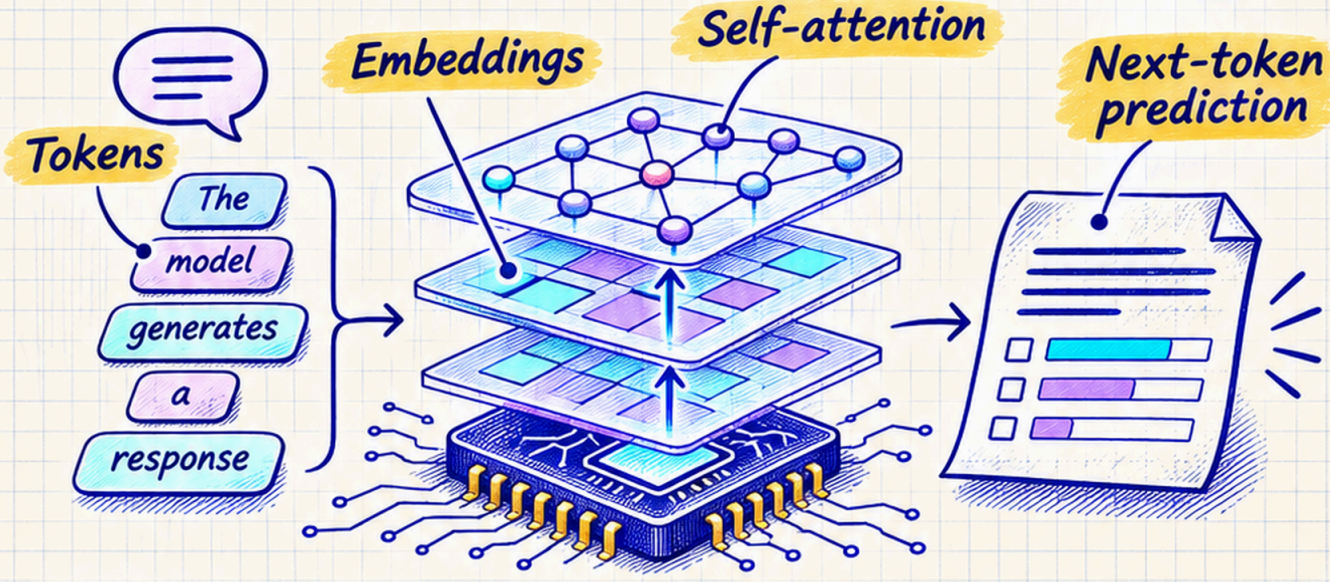
Large Language Models

LLMs Complete Handbook

30 pages

Beginners to Advanced



Understand the model

- NLP & Transformers
- GPT, Gemini, Llama & Claude
- Prompt engineering

Build an application

- RAG & vector databases
- Python & Hugging Face
- LangChain & AI agents

2 practical projects



- Meeting action extractor



- Policy assistant

Evaluation • Deployment • Interview questions

Created by Coursya.com

Inside the handbook

Start at page 3 for the fundamentals, or jump to the topic you need.

01 Foundations

03–05 Language, tokens & embeddings

06–08 Transformer architecture & training

09–10 Decoding & checking claims

02 Models, prompting & retrieval

11–12 GPT, Gemini, Llama & Claude

13–15 Prompt engineering & structured output

16–20 RAG, vector search & fine-tuning

03 The application notebook

21–24 Python, Hugging Face, LangChain & agents

25–26 Evaluation, latency & cost

27–30 Two practical projects & interview practice

Working through the code?

Run the three Python cells on pages 21–23 in one session. Then try the meeting-action extractor on page 27 with notes of your own.

What does an LLM learn?

A single meeting note can become a label, a list of names, or a summary.

Start with language



Natural language processing (NLP) includes translation, classification, and extraction. A large language model (LLM) is a neural network trained on large collections of data to learn patterns in language. A generative LLM uses those patterns to produce text.

One note, three NLP tasks

Meeting note	"Mina will send the budget on Friday."
Classify	Action item
Extract	Owner = Mina. Due = Friday.
Summarize	Mina is responsible for Friday's budget update.

How a reply grows

A causal language model scores possible next tokens using the text so far. After a token is chosen, it joins the context for the next prediction. Repeating this small operation produces a phrase, a paragraph, or a structured record.

Try it

Change "will send" to "might send." Which output should change? The owner still appears, but the sentence no longer commits Mina to the task.

READ MORE Hugging Face · Causal language modeling

Tokens and context budgets

A short-looking question can carry a long chat history and several pages of evidence.

Count what the model sees



A tokenizer maps text to integer IDs. A token can represent a word, part of a word, punctuation, or a sequence of bytes. Different tokenizers can assign different lengths to the same email, especially when it mixes languages or contains code.

A 4,096-token example

Shared window — **4,096 tokens**

640	2,400	800	256

- Instructions + chat formatting
- Selected evidence
- Reserved for the answer
- Space still available

Reserve space before sending

For a model with a shared input-and-output limit, count the formatted prompt, earlier messages, and attached evidence. Reserve room for the response and check any separate output cap. Use the tokenizer or counting method for the exact model.

What would you trim?

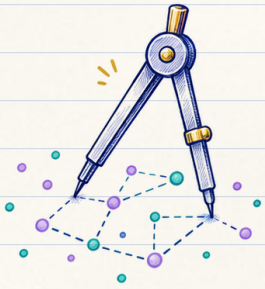
A 350-token appendix exceeds the spare space by 94 tokens. Keep the clause that answers the question and remove unrelated background.

READ MORE Hugging Face · Tokenizer

Embeddings and position

How does the same word come to mean something different across two sentences?

Turn a token ID into a vector



An embedding table gives each vocabulary entry a learned list of numbers. Whenever the tokenizer produces the same ID, it retrieves the same starting vector. Later layers combine that vector with information from nearby and distant tokens.

Follow the final token

River bank	"They rested on the river bank."
Savings bank	"They deposited cash at the bank."
At the lookup table	If both instances of " bank" have the same ID, they start alike.
After several layers	Their vectors can encode different clues from the preceding words.

Give the model word order

"Mina thanked Omar" and "Omar thanked Mina" share their words, but reverse the roles. The original Transformer added position signals to embeddings. Another method, RoPE, rotates attention queries and keys by position so their comparison reflects relative distance.

Read the distinction

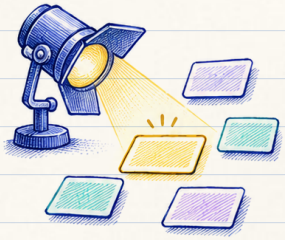
An input embedding identifies a token. A contextual representation describes that occurrence of the token in its surroundings.

READ MORE Vaswani et al. · Transformer Su et al. · RoFormer

Inside self-attention

Each position learns which earlier signals to combine when updating its representation.

Three learned projections



For each token position, an attention head produces a query (Q), a key (K), and a value (V). A query scores the available keys by scaled dot product. Softmax turns those scores into weights. The head then takes a weighted sum of the values.

A toy head reading "Team ships Friday"

Causal mask

	Team	ships	Fri.
Team	•	—	—
ships	•	•	—
Friday	•	•	•

• visible — hidden

At the "ships" position

Weight	Team	0.25	ships	0.75
Value*	Team	2	ships	6

0.25 × 2 + 0.75 × 6 = 5

* One coordinate of each value vector.

What the causal mask hides

In a causal decoder, a position can attend to itself and preceding positions. Future positions receive zero attention weight. That restriction keeps the training input from revealing the token the model is asked to predict.

Scale up the example

A real value has many coordinates. The same attention weights combine every coordinate in that head. Other heads can learn different patterns.

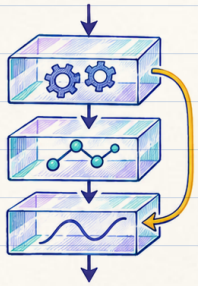
READ MORE Vaswani et al. · Attention Is All You Need

Build a Transformer block

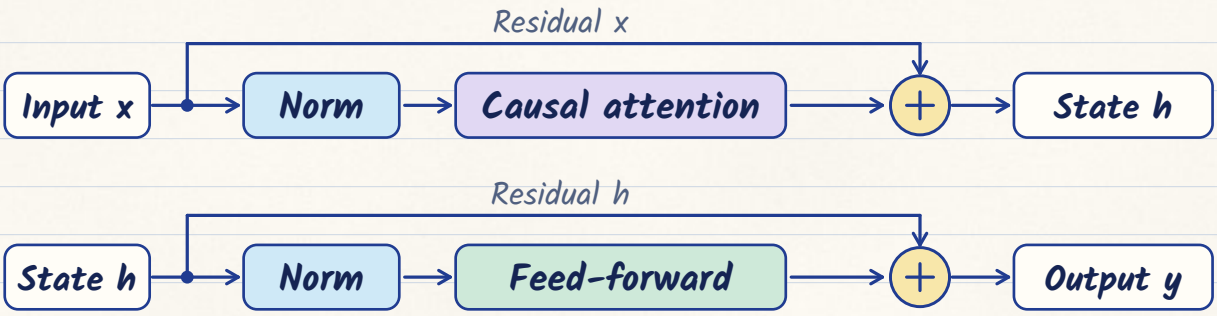
Attention mixes information across positions. A feed-forward network reshapes each one.

Follow one token through a block

Attention can bring a person's name and a deadline into the same representation. A feed-forward network then applies a learned nonlinear transformation at each position, using the same network weights for every position.



One common pre-normalized decoder block



From one block to a prediction

A residual connection adds each update back to its input. This gives information and gradients a direct path across layers. After many blocks, the output layer maps the final position's state to vocabulary scores, called logits.

Architecture detail

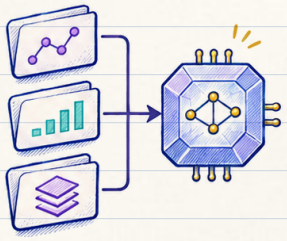
Normalization placement varies across designs. LLaMA uses pre-normalization. The original 2017 Transformer normalized after each residual addition.

READ MORE Touvron et al. · LLaMA Vaswani et al. · Transformer

How assistants are trained

Predicting text, imitating examples, and learning preferences supply different signals.

Pretraining supplies the foundation



For a causal model, the next token in a training passage supplies a target. Training adjusts weights to give observed continuations higher probability across many examples. The resulting base model can continue many kinds of text.

Three ways to learn from a meeting note

Pretraining signal	Given "Mina will send the", predict the token that follows in the text.
Demonstration	Given a note and an extraction request, imitate a labeled action item.
Preference pair	A reviewer favors a faithful answer over one that invents a due date.

Teach a response pattern

Supervised fine-tuning (SFT) uses prompt-and-answer demonstrations. Preference training uses comparisons between candidate answers. In the InstructGPT approach, human rankings trained a reward model that guided further updates to the assistant.

At inference time

A worked example in a prompt can change the next answer while the model weights stay fixed. Training changes the weights themselves.

READ MORE Ouyang et al. · InstructGPT Hugging Face · Causal language modeling

Decoding the next token

A model supplies scores. A decoding rule determines how those scores become a choice.

Temperature changes the spread



During sampling, temperature divides the logits before softmax converts them to probabilities. A positive temperature below 1 concentrates probability on higher-scoring options. A value above 1 spreads it more evenly. Greedy decoding simply picks a highest-scoring token.

Toy probabilities after "The task is..."

Candidate	Probability	Cumulative
ready	50%	50%
late	30%	**80%**
open	15%	95%
new	5%	100%

Top-p changes the candidate set

Sort tokens by probability and retain the smallest leading set whose cumulative probability reaches p. Rescale that set to sum to 1 and draw a token. The size of the set can change at every generation step.

Work it out

At p = 0.80, "ready" and "late" reach 80% together. Their sampling probabilities become 50/80 = 62.5% and 30/80 = 37.5%.

READ MORE Holtzman et al. · Nucleus sampling Hugging Face · Generation settings

Check the evidence

An accurate name in a model's answer can sit next to an invented deadline.

Separate the claims



A hallucination is a generated claim that is false or unsupported in the task. Break an answer into small claims and compare each with the relevant evidence. Names, numbers, dates, and words such as "approved" deserve their own checks.

Audit a release update

Meeting notes	"Mina proposed a Friday release. Finance has not reviewed the budget."
Generated draft	"Finance approved Mina's Friday release."
Check the verbs	Proposed is supported. Approved contradicts the note.
Revised update	"Mina proposed Friday. Finance's budget review is still pending."

Write an answer the source can support

If a detail is missing, identify it as unknown. If a decision is tentative, preserve that uncertainty. A citation helps a reader find the passage, but each cited passage still needs to support the specific claim attached to it.

A useful test case

Ask, "Who in Finance approved the budget?" A grounded answer points out the pending review. Giving a person's name would introduce an unsupported claim.

READ MORE Min et al. · FActScore Gao et al. · Citation evaluation

Four families of language models

GPT, Gemini, Llama and Claude each span many releases. Here are four documented examples of capabilities you might test.

OPENAI
GPT-4o
Image and text in. Text answer out.

GOOGLE
Gemini 2.5 Flash
Accepts audio and video as inputs.



META
Llama 3.2 3B Instruct
Download weights for local text chat.

ANTHROPIC
Claude 3.5 Sonnet
Reads charts supplied as images.

Which model could read a scanned invoice?

Start with image input. Then give each candidate the same blurry scan and check the total and currency. If the files must stay on your hardware, also check the release's license and memory requirements.

> **Keep the test reproducible**
> Record the exact model ID, hosting service and prompt. A family name alone leaves room for different results.

SOURCES OpenAI GPT-4o · Gemini 2.5 Flash · Meta Llama card · Claude 3.5 Sonnet

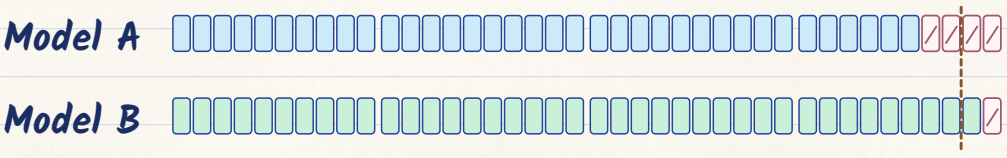
Which model passes the pilot?

An invoice extractor must return the correct total and currency. Set the acceptance thresholds before testing either model.

Accuracy

Pass with at least 38 of 40 correct

One square = one held-out invoice



36/40 90%

39/40 97.5%

Filled = correct *Hatched = incorrect*

Response time

Pass with p95 under 3 seconds



A · Misses accuracy

Two more correct invoices needed.

B · Passes both gates

Advance to a larger trial.

Read the two measures together

Model A answers faster but gets four invoices wrong. Model B makes one error and stays under the latency limit. Advance B to a larger trial with credit notes, blurred scans and vendors it has never seen.

What does p95 mean?

Roughly 95% of requests finished within that time. Measure both models under the same load.

SOURCES Evaluation design

Anatomy of a useful prompt

A project manager needs a blocker register from meeting notes. Four short instructions turn that request into a result you can check.

Meeting-note prompt

Extract each unresolved blocker from the meeting notes.

For each blocker, give the owner and the next decision.

Use only the notes. If a detail is missing, write "unknown".

Return one row per blocker.

Read the text inside <notes>.

Task — What to find

Fields — What to capture

Evidence rule — When to abstain

Format — How to return it

Example source

`<notes>` The vendor file is late. We will decide Friday whether to reschedule. `</notes>`

Read the resulting register

The blocker is the late vendor file. Its owner is unknown. The team will decide on Friday whether to reschedule. An answer that lists Friday as the file's due date has changed the meaning of the note.

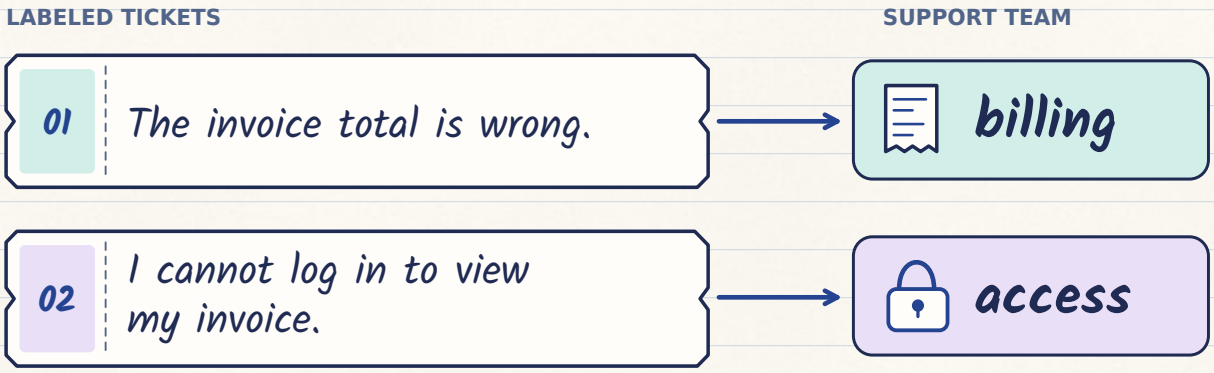
Change one line

Add "The vendor file arrived before the meeting ended." The blocker is resolved, so its row should disappear.

SOURCES Prompt design

Teach the boundary with examples

A billing ticket and a login ticket can both contain the word "invoice." Show the model what separates the two intents.



HELD-OUT TICKET

"My invoice includes the same fee twice."

Expected JSON

```
{"category": "billing"}
```

✓ Valid category

✓ Correct team

What a schema can enforce

Limit the category field to billing, access or other. Validate the object in your application. A reply can pass that check and still route a customer to the wrong team.

Test on new wording

"I was charged twice after resetting my password." Which detail determines the queue? The duplicate charge makes billing the expected label.

SOURCES Few-shot prompting Structured output

Does the source support the answer?

A citation earns its place when the passage supports the exact claim. Compare the answer with this fictional travel policy.

TRAVEL HANDBOOK / RECEIPTS — P3

"Submit receipts within **10 calendar days of return.**"

READ THE CLAUSE

- Quantity: **10**
- Unit: **calendar days**
- Trigger: **return from trip**

ANSWER THE POLICY SUPPORTS

Within 10 calendar days of your return [P3].

EXCEPTION TO CHECK

What if the office is closed?

P3 gives no extension rule. Find the exception policy.

A citation is useful when a reader can verify every part of the claim.

Keep the words that change the deadline

"Ten days" drops the unit. "Ten days after purchase" changes the starting event. Check each number, unit and condition against the passage a reader will actually open.

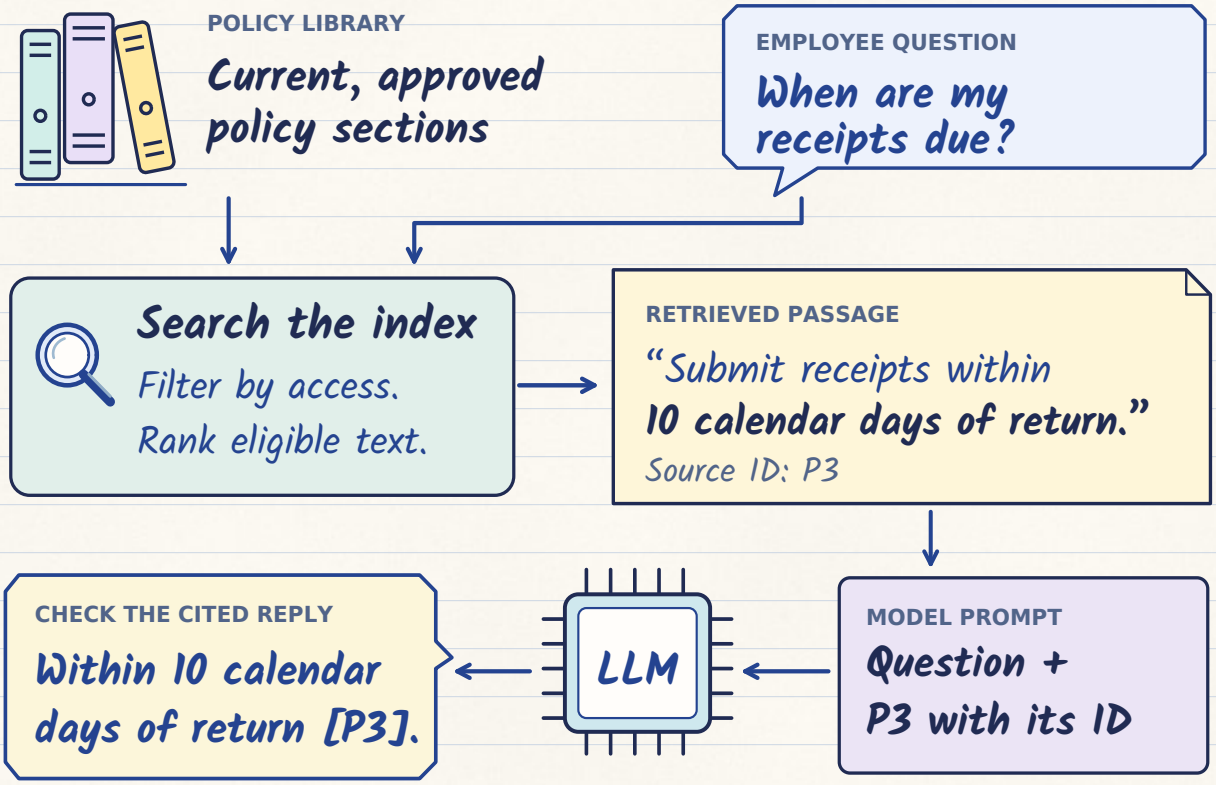
Try a date on the calendar

A traveler returns May 2. Does the return date count as day one? Find the counting rule before promising an exact deadline.

SOURCES Evaluating answers with citations

Follow one question through RAG

Retrieval-augmented generation (RAG) puts relevant source passages into the model's input when a question arrives.



Locate a wrong answer

Suppose the assistant says 30 days. Look at the passages it received. If P3 is missing, inspect indexing and retrieval. If P3 is present, check whether the answer used the wrong passage or changed its wording.

When the policy changes

Reindex the revised clause and retire the older active version. Rerun the same question and check its citation.

SOURCES Original RAG paper

Chunk the rule and its exception

Search returns passages, often called chunks. A split just before an exception can make a correct policy answer incomplete.

SOURCE DOCUMENT

Travel handbook

Revision 4

§3 Receipts

Submit receipts within 10 calendar days of return.

If the receipt is lost, attach a signed form.

§4 Accommodation

SEARCH INDEX

Chunk P3

SECTION HEADING
Receipts

PASSAGE
Deadline + exception

REVISION
4

ACCESS SCOPE
Covered employees

Rule + exception ⟶ one retrievable passage

Filter by access scope before the reader sees a result.

Compare two ways to split the same policy

Try sections first, then fixed-size passages with overlap. Use the same ten questions for both indexes. Count how often the first three results contain the full rule and every relevant exception.

Keep the source traceable

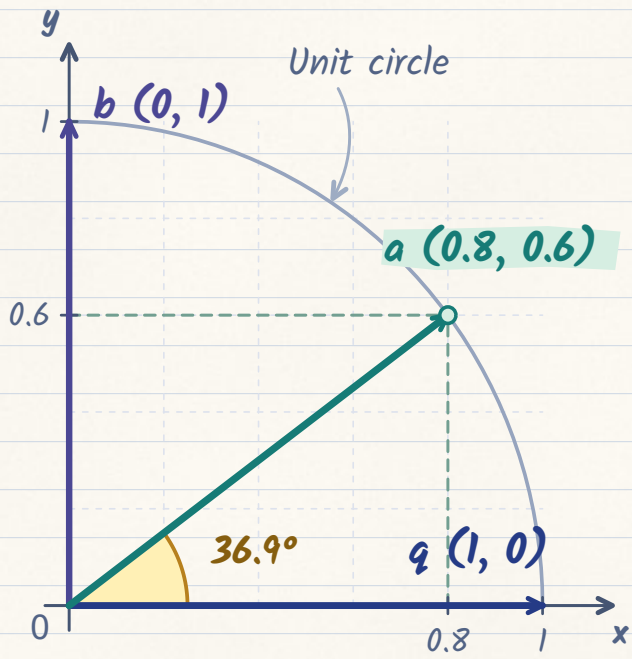
Store the document ID, section, revision and access scope with each chunk. Apply access checks during search.

SOURCES Chunking guidance

How vector search ranks a passage

An embedding expresses a query or passage as a vector. In this two-dimensional example, cosine similarity compares their directions.

01 / MAP THE DIRECTIONS



02 / COMPARE THE SCORES

All three lengths = 1

cosine(q, d) = q · d

Passage A

(1 × 0.8) + (0 × 0.6)

= 0.8

Passage B

(1 × 0) + (0 × 1)

= 0.0

A points closer to q, so A ranks above B.

From vectors to documents

Store each passage vector with a source ID and access scope. Encode the question with a compatible embedding model, filter eligible records and rank them by similarity. Return the passage text with its source ID.

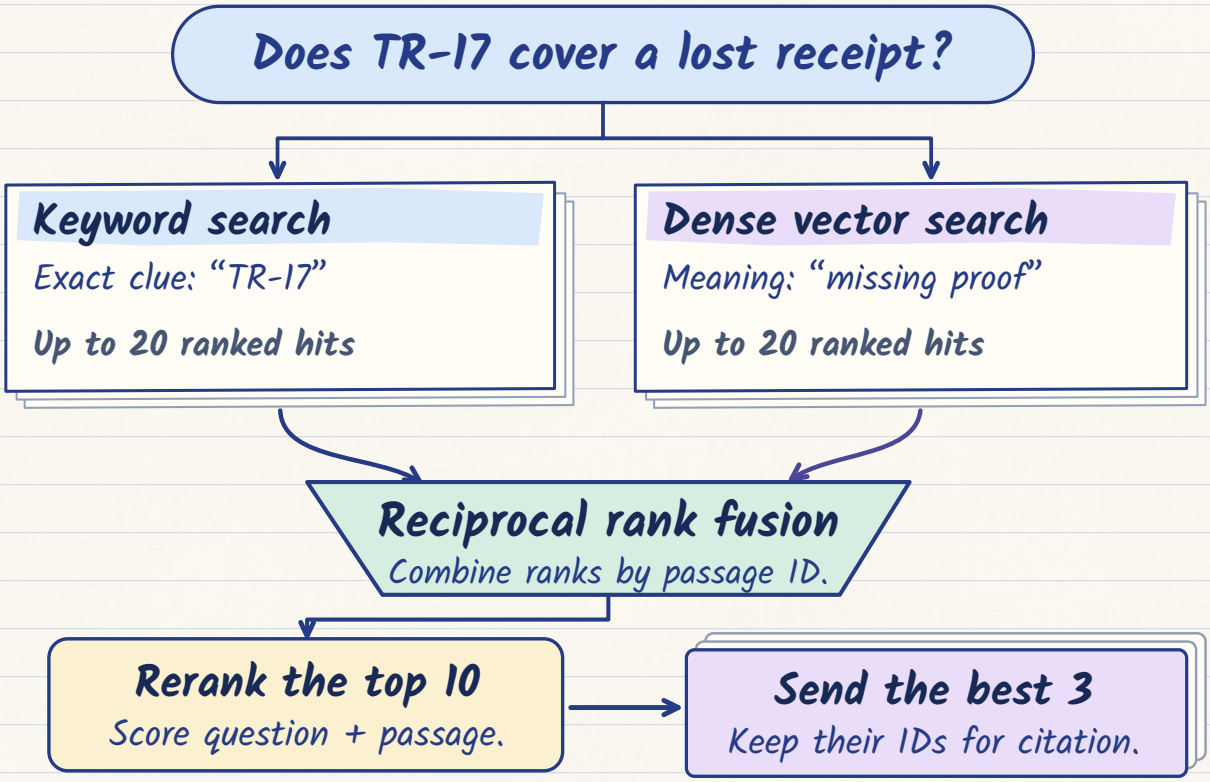
Choose a useful threshold

A score of 0.8 is a similarity measure. Test it against labeled questions before using it as a cutoff for relevant evidence.

SOURCES Vector search Metadata filters

Hybrid search and reranking

One question can contain an exact policy code and a paraphrase. Run two searches so either clue can bring the relevant passage into view.



Reranking can only promote a passage already retrieved.

How reciprocal rank fusion scores a passage

Suppose the same passage ranks 2nd in one list and 7th in the other. With a rank constant of 60, it scores 1/62 + 1/67 ≈ 0.0311. Sort the combined candidates by this score before reranking.

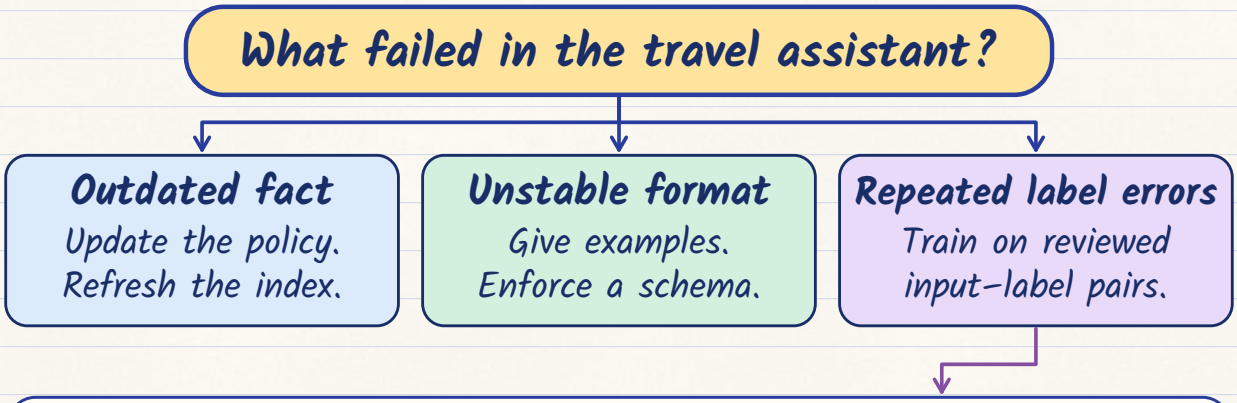
Find the bottleneck

If the answer-bearing passage misses the merged list, adjust retrieval. If it enters the shortlist but misses the final three, inspect reranking.

SOURCES Rank fusion Reranking

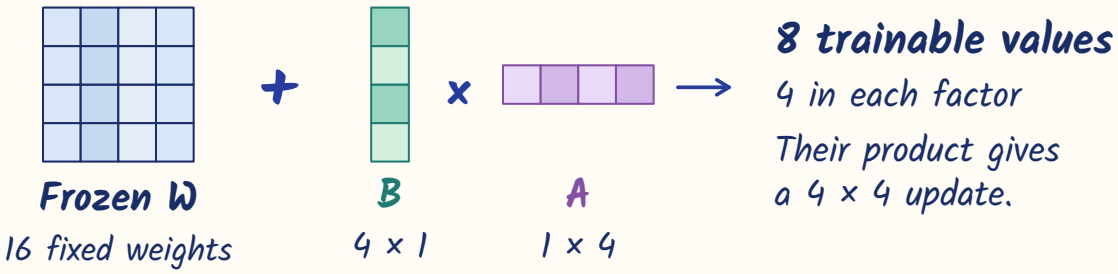
Fine-tune, prompt or retrieve?

A changed reimbursement limit and a consistently wrong expense label need different fixes. Trace one bad answer to its cause.



Inside a rank-one LoRA adapter

One simplified layer, with a 4 × 4 weight matrix.



What low-rank adaptation changes

LoRA keeps base weights fixed and learns a small update from reviewed examples. Use the two factors to adapt how the model labels expenses. Current limits can still come from retrieved policy text.

Check for a real improvement

Keep an unseen set of examples. Compare the adapter with your best prompt and check whether previously correct labels now fail.

SOURCES The LoRA paper The RAG paper

Load a language model in Python

Run a small instruction-tuned model on your CPU. Keep this Python session open for all three cells on pages 21–23.

Run this terminal command in a fresh Python 3.11 environment.

```
python -m pip install torch transformers langchain-core
```

PYTHON **Cell 1 / Load the checkpoint**

```
from transformers import AutoModelForCausalLM, AutoTokenizer

repo = "HuggingFaceTB/SmolLM2-360M-Instruct"
tok = AutoTokenizer.from_pretrained(repo)
model = AutoModelForCausalLM.from_pretrained(repo)
model = model.float().to("cpu").eval()
print(tok.encode("handover", add_special_tokens=False))
```

What did the checkpoint load?

Tokenizer

Maps "handover" to vocabulary IDs. The chat template adds speaker markers for this checkpoint.

Model weights

Learned parameters turn input IDs into next-token scores. Evaluation mode selects inference behavior.

Plan for the first download

The first run needs internet access to download the checkpoint. Later runs can reuse its local cache. At four bytes per parameter, 360 million FP32 weights alone use roughly 1.44 GB. Allow extra RAM for activations and generation.

- **Repeat a run** *Pin the same repository commit with revision= in both loaders.*

Quick check

Try "handover" and " handover". The space can change the IDs. Use the tokenizer from the same repository as the model.

SOURCES SmolLM2 model card Transformers quickstart

Generate and decode a reply

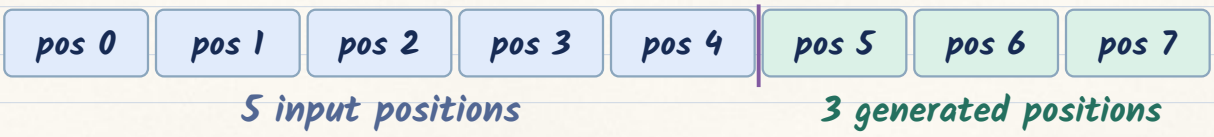
A chat model needs a token sequence with speaker markers. Use its own chat template, then decode only the tokens it adds.

PYTHON **Cell 2 / Create the reply function**

```
def ask(text):
    messages = [{"role": "user", "content": text}]
    batch = tok.apply_chat_template(
        messages, add_generation_prompt=True, tokenize=True,
        return_dict=True, return_tensors="pt")
    out = model.generate(**batch, max_new_tokens=40,
                         do_sample=False, num_beams=1)
    start = batch["input_ids"].shape[1]
    return tok.decode(out[0, start:], skip_special_tokens=True)

print(ask("Define a project milestone in one sentence."))
```

Find the boundary in a decoder-only output



Example: start = 5. The slice keeps positions 5, 6 and 7.

Control the length and selection rule

The added generation marker opens the assistant turn. Greedy decoding takes the highest-scoring token at each step. The 40-token cap applies only to the continuation. An end-of-sequence token can stop it earlier.

- **Compare sampling** *Set do_sample=True and temperature=0.7. Run the same prompt twice.*

Debug an echoed question

Inspect the decoded continuation first. If you decode the whole output, the original question and its role markers can appear before the answer.

SOURCES Chat templates Generation settings

Build a local LangChain workflow

Turn one local function into a repeatable meeting-note workflow. The same prompt can now run against a whole set of examples.

PYTHON **Cell 3 / Compose and invoke**

```
from langchain_core.prompts import PromptTemplate
from langchain_core.runnables import RunnableLambda

prompt = PromptTemplate.from_template(
    "Extract owner and task. Use unknown for missing fields."
    "\nNote: {note}")
def run_local(value):
    return ask(value.to_string())

chain = prompt | RunnableLambda(run_local)
case = {"note": "Noor will send the revised budget."}
print(chain.invoke(case))
```

Trace the values between stages



The pipe passes a PromptValue to run_local. Calling to_string() on that value gives ask the plain text it expects. Keep the input note beside the reply when reviewing a run.

Check the answer against the note

For this test, the owner is Noor and the task is sending the revised budget. Score both fields. A date in the answer would be unsupported.

- **Keep a test log** *Save the input note, formatted prompt, actual reply and expected fields.*

Run a harder case

Try "The revised budget must be sent." Expect an unknown owner. If a name appears, inspect prompt.format(note=...) before changing the model.

SOURCES PromptTemplate RunnableLambda

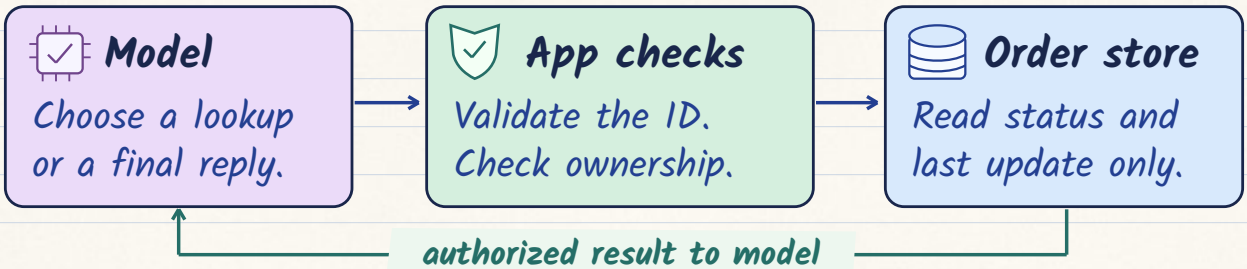
Give an agent one tool

An agent can request an order lookup. The application decides which calls can run and when the loop must stop.

The contract for one read-only tool

```
lookup_order(order_id: str)
```

Returns status and last update for an order the signed-in customer owns.



Trace one customer request

Ask	"Has order A17 shipped?"
Call	The model requests lookup_order("A17").
Read	The app confirms ownership. The store returns "packed, 09:20."
Answer	"Order A17 was packed at 09:20. I have no shipping date yet."

Test both exits

An unknown order and another customer's order receive the same generic reply. This avoids revealing which IDs exist. A missing ID prompts the customer to supply it. A valid status without a shipping date supports a partial answer.

Limit the loop in application code

Count every lookup attempt, including invalid ones. After two attempts or the time limit, end the run with the evidence available. Log the tool name, outcome and stop reason for debugging.

SOURCES Agent design LangChain agents

Score the whole system

Give a policy assistant a fixed answer key. Score what it retrieved, what it answered and when it correctly declined.

A fictional 20-question held-out test

16 answerable from the policies | **4 need more evidence**

MEASURE	COUNT	RATE
Retrieval hit rate @3 *Target passage in the first 3*	14 / 16	87.5%
Answer correctness *Matches the answer key*	12 / 16	75%
Correct abstention *Declines an unanswerable case*	3 / 4	75%
Citation support *Cited passage supports the claim*	18 / 20	90%

The first three rates count questions. The last counts individual cited claims. A single answer can contain several claims, so these percentages have different denominators.

Trace an error to its stage

P3 missing from top 3

Check whether the current P3 was indexed and eligible. Compare the query wording with the passage.

P3 retrieved, wrong date

Inspect the prompt. Did an older passage distract the model from the current 10-day rule?

Make the test reproducible

Record the expected answer, evidence ID and whether abstention is correct before running the test. Keep the questions out of prompt development. Review failures by case ID when comparing two runs.

SOURCES Evaluating a RAG application

Budget for a real deployment

A release needs a latency target, a cost estimate and an acceptance test. Track the cost of a usable result.

One request, two clocks



Time to first token is 0.6 s. The complete answer takes 2.4 s, including 1.8 s after the first token. Under realistic load, sort 100 end-to-end times. With the nearest-rank method, p95 is the 95th value.

Price one call at fictional rates

Input	2,000 tokens × $1 per million	**$0.002**
Output	500 tokens × $4 per million	**$0.002**
Total	Model charge per call	**$0.004**

Cost per accepted task

At this token mix, 10,000 calls cost $40 in model charges. If those calls, including retries, yield 9,500 accepted tasks, the model cost per accepted task is $40 / 9,500 ≈ $0.00421. Add retrieval, tools and hosting for a full estimate.

Set a release gate

For this pilot, require p95 below 3 s and at least 95% task acceptance. Start with 5% of traffic. Compare against the previous version at similar load. Roll back if either threshold fails.

SOURCES NVIDIA inference metrics

Project 1 · Extract meeting actions

Build an action register a project manager can verify. Every open commitment should point back to the words that created it.

SOURCE EXCERPT / MEETING TRANSCRIPT

L4	Noor: I'll send the revised budget.	KEEP
L5	Omar: Could we review hiring?	SUGGESTION
L6	Mira: I'll book the room.	SEE L7
L7	Mira: Cancel the room booking.	CANCELLED

One surviving commitment ⟶ expected JSON array

```
[{
  "owner": "Noor",
  "action": "send the revised budget",
  "due": null,
  "source_line": 4,
  "quote": "send the revised budget"
}]
```

Why these fields?

No date was spoken, so due is null. The quoted phrase must occur on the cited line. Mira's later cancellation removes her open task.

Build and test the extractor

1. **Parse** — *Keep speaker names and line numbers. Read the whole meeting before returning one object per open commitment.*
2. **Validate** — *Require all five keys. Check the line number, match its quote, and verify any due date against the transcript.*
3. **Review** — *Show the cited line beside each draft. Have a reviewer confirm ownership before an action enters the team register.*

Acceptance test

Expect exactly one record. Add "Noor: The budget update is cancelled." The result should become an empty array. Track invented actions and missed commitments separately on held-out meetings.

SOURCES LangChain · Structured output

Project 2 · Build a policy assistant

Build a small retrieval-augmented assistant that can answer for a specific date and cite the policy version it used.

A two-version policy library

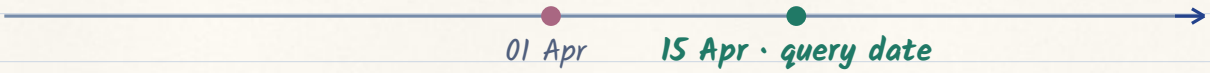
Fictional expense policy

EXP-1 Before 01 Apr 2026

File a claim within 30 days after the trip ends.

EXP-2 From 01 Apr 2026

File a claim within 14 days after the trip ends.



Trace the answer for 15 April 2026

Ask *"How long after a trip can I file my claim?"*

Retrieve **EXP-2** ⟶ *"Within 14 days of the trip ending [EXP-2]."*

Build the retrieval path

Store each clause with its ID, effective dates and access scope. Filter by user and date before retrieval. Pass both the clause and its ID to the model, then verify the cited deadline and its starting event.

Run three acceptance tests

CURRENT *On 15 Apr, expect 14 days and EXP-2. Log the retrieved ID to show which version reached the model.*

HISTORICAL *Repeat for 15 Mar 2026. The answer must switch to 30 days and cite EXP-1.*

UNANSWERED *Ask "Can a manager extend the deadline?" Neither clause authorizes an extension. The answer should identify the missing rule.*

Debug with the question date, allowed IDs, retrieved IDs and final citation.

SOURCES LangChain · Retrieval and generation

Five questions for an LLM interview

Give a short answer, then use the example to show how you would test it.

01 Why does fluent text sometimes invent a deadline?

A language model scores likely continuations. "Friday" can sound plausible even when the meeting never set a date. Check the claim against the transcript and return null for a missing date.

02 How does temperature affect the next token?

For positive temperature T, divide the logits by T before applying softmax. A lower T concentrates probability on higher-scoring tokens. Greedy decoding chooses a highest-scoring token directly.

03 When would you choose retrieval for a policy assistant?

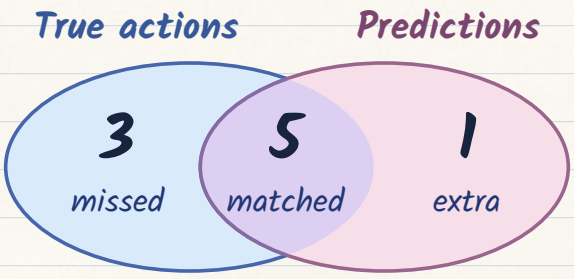
Use retrieval when policies change and readers need citations. Test with a superseded rule and a current rule. Inspect the passage ID and date that reached the model.

04 Can valid JSON still assign a task to the wrong person?

Yes. A schema can enforce an owner field and its type. Check the owner against the cited line and test meetings where a suggestion never becomes a commitment.

05 How would you score the meeting extractor?

Eight real actions. Six predictions. Five correct matches.



Precision	**5/6 = 83.3%**
Recall	**5/8 = 62.5%**

Count each matched action once.

Inspect the three missed actions before changing the prompt.

SOURCES Generation settings Precision & recall

Diagnose a production failure

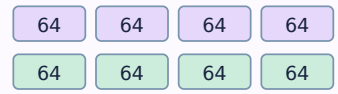
Three advanced interview scenarios. For each one, connect a symptom to a measurement and a decision.

01 Why do long chats reduce serving capacity?

During decoding, each attention layer can reuse cached keys and values from earlier tokens. More active tokens require more cache memory. Estimate the persistent cache for this hypothetical decoder.

A worked key-value cache budget

At one layer, for one token



4 KV heads × 64 values per head

One active sequence

16 layers · 2,048 tokens

2 arrays (K and V) · 2 bytes/value

2 × 16 × 2,048 × 4 × 64 × 2 bytes = 32 MiB

Four equally long sequences need roughly 128 MiB of KV cache. Model weights, activations and allocator overhead add to that total.

02 The policy answer is stale. Where do you look?

Log the question date, selected passage IDs, revisions and actual prompt. If the new clause never arrived, inspect the index and filters. If it arrived and the answer still cites an old rule, remove the competing passage and rerun the same question.

03 The first token is fast, but users still wait. Why?

Time the first token and the complete answer separately. Compare output lengths and generated tokens per second under the same load. If long replies dominate, test a tighter output budget and recheck answer quality.

Change one thing, rerun the same cases, and explain one remaining failure.

SOURCES KV caching RAG evaluation