# EVERYTHING IN THE CONTEXT, THEN RAG

**Youth AI Lab · Workshop 2 of 6**

Build your own chatbot



Co-funded by the European Union

# ❓ The question of the series

A chatbot only answers from **what it reads** before it answers. Every workshop of the series is **one answer** to the same question.

> **What enters the model's context, when, and who decides?**

📥 **What?**
rules, documents, search results, live data

⏱️ **When?**
always, at startup, or for each question

🤦 **Who decides?**
you, a search engine, or the model itself

# 🧭 Today: who picks the pieces?

| 1. Instructions | 2. Knowledge, RAG<br>today | 3. Search, skills | 4. Tools | 5. Wiki | 6. Fine-tuning |
|---|---|---|---|---|---|
| 🎭 | 📚 | 🔍 | 📡 | 🗺️ | 🧪 |

**Mode** `B`: everything enters the context, every time. **Nobody** picks.

**RAG**: a **search engine** picks a few pieces for each question, before the model.

# 🗓️ The plan: 2 hours

| 👋 Welcome | 📚 Everything | ✂️ Unplugged | 🧩 Pieces | 🎛️ Settings | ⚖️ RAG vs B | 🏁 Closing |
|---|---|---|---|---|---|---|
| 25 min | mode B<br>15 min | paper<br>15 min | query<br>20 min | size, top-k<br>20 min | columns<br>10 min | 10 min |

📄 Follow on your screen: the `README.md` of your Codespace, or the one-page sheet.

# 💻 Your workspace: one Codespace per pair

A **Codespace** is VS Code in your browser: **nothing to install**. Everything you need is already inside.

> ⏱️ **Before you open it**, once per GitHub account: **github.com/settings/codespaces → Default idle timeout → 120 → Save**. It only applies to the Codespaces you create after.

1. Open the **Codespace link** of the workshop, at the top of its README
2. In the terminal, type `workshop-login` and paste the key of your pair
3. Wait for **Key OK**, then type `workshop bare`. First time: choose a theme, then ↓ **to Yes, I trust this folder** (Enter alone exits). To leave: **Ctrl+C twice**
4. Then type `workshop-start` and the number of the workshop. With a save from last time, add `--from my-work.zip`

> ⚠️ **The key is not shown** while you paste it: that is normal. If the Codespace stops after a long pause, reopen it: **nothing is lost**.

💻 No Codespace? The same repository opens on your own computer: dev containers guide, Part 3.

# 🐳 Open WebUI starts by itself

1. Drag `my-work.zip` into the file list.
2. `workshop-start 2 --from my-work.zip`: your work is back, and Open WebUI starts in the background.
3. Second terminal: `tail -f .workshop/openwebui-start.log`. No need to wait: it must say **READY** before Part 4 (about 4 to 5 minutes the first time). **Ctrl+C** stops following, not Open WebUI. Other commands: first terminal.

🚫 **Never make port 3000 public.** This Open WebUI has no accounts: anyone with the link could use your key.

# 📍 Question 12: your territory

1. Read the section **Question 12: your territory** of `test-questions.md`.
2. Write the question, the answer and **the address of the official page** in row 12.
3. `replay bare 12`, then `replay A 12`.
4. `score bare 12 --blind`, then `score A 12 --blind`: your score first, then the judge's.

> ⏱️ 5 minutes. With row 12, a complete column is out of **24**. In workshop 3, its page enters your chatbot.

# 📚 Everything in the context

## What does reading everything cost?

15 min · mode `B`

# 🌡️ The gauge: how full is the context?

The **context** is everything the model reads before it answers. Its maximum size is the **context window**. A line above the input box shows **how full** it is.

🪟

**200,000 tokens**

the size of the window (a token is a piece of a word)

💶

**Every message pays**

for everything in the window

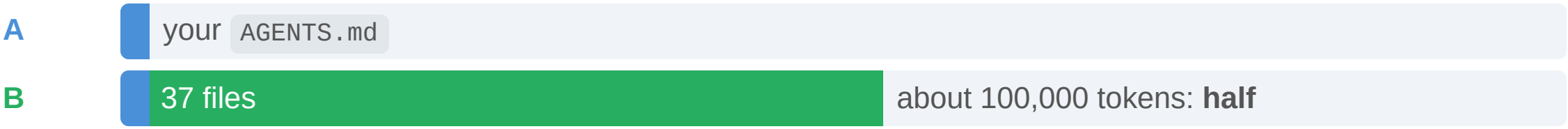
🤝

**Same model for all**

so that the pairs can compare

👁️ **Watch the gauge** at every change: it shows what entered the context.

# 📚 Mode B: nobody picks




✅ The model can find any fact, and see two documents that disagree.

💶 Every message pays for all of it, again.

# 📏 How to score an answer

Each answer gets **0, 1 or 2**. The score rewards **honesty** as much as knowledge.

## 2

**Right**, and it says **where it comes from**. Or "I don't know" when nobody knows.

## 1

**Honest but not enough**: no source, only partly right, or "I don't know" although the answer exists.

## 0

**Wrong** or **invented**.

💡 An **invented answer** is worse than **"I don't know"**.

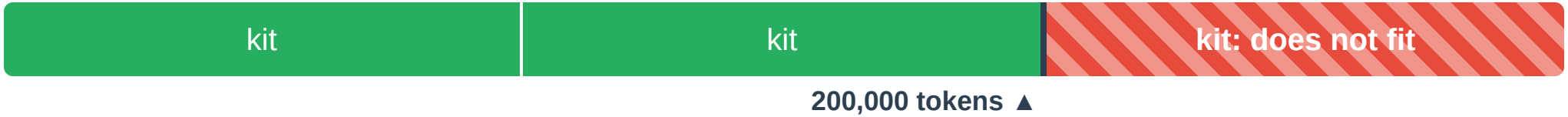
# 🎬 Your turn: mode B

1. `workshop B`: ask question 10, then look at the gauge. **Ctrl+C twice**.
2. **Predict**: which questions go up from A to B?
3. `replay B`, then `score B 3 7 10`: the judge proposes, you accept (Enter) or correct. Time left? `score B 1 2 4 5 6 8 9 11 12`.

❓ The kit fills half of the window. **And with ten times more documents?**

# 🪟 Three times the kit?

The window: 2 m of paper, **1 cm = 1,000 tokens**




```
❯ How many hectares burnt outside the EU in 2025?
  ⎿  Prompt is too long · the request is ~237206 tokens (limit 200000) …
     A single-exchange conversation cannot be compacted …
```

🚫 No answer at all: the model read nothing.

❓ So who chooses what enters?

# ✂️ Unplugged: be the search engine

**Pick 3 pieces. Answer with them only.**

15 min · paper strips · two pairs together

# ✂️ How it works

### 🔎 Pair 1: the search engine

Reads the question card, looks through the strips, and passes **3 strips** (round 1) or **2 strips** (round 2). It must always pass strips.

### 🤖 Pair 2: the model

Answers the question with the strips it received, and **nothing else**: not its memory, not the other strips.

**Round 1**: one sentence per strip, question 10, then question 7. **Round 2**: one paragraph per strip, question 10, roles swapped.

❓ How did the search engine choose: by the **words** of the question, or by their **meaning**?

# 🧩 Pieces and embeddings

## What does the RAG give the model?

20 min · `openwebui query` · no model, no cost

# 🧩 A RAG in three words

### ✂️ Chunk

a piece of a document, cut before any question (here, at most 1,000 characters)

### 🔢 Embedding

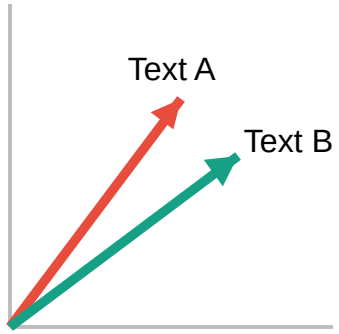
a list of numbers for a text: close meanings, close numbers

### 🔝 Top-k

the number of pieces the model receives (here, 5)

💡 The search works **before** the model. The model never sees the rest.

# 📐 How two texts are compared




Each text is an arrow. The **angle** between two arrows says how close they are.

$$\cos(\text{angle}) = a_1 b_1 + a_2 b_2 + \cdots + a_{384} b_{384} \qquad \text{score} = \frac{1 + \cos(\text{angle})}{2}$$

Text A (0.6, 0.8), text B (0.8, 0.6): 0.48 + 0.48 = **0.96**

| The two arrows | Angle | Cosine (the sum) | Score |
|---|---|---|---|
| Same direction | 0° | 1 | **1** |
| Text A and text B (made up) | 16° | 0.96 | **0.98** |
| *wildfire* and *fire in the woods* (measured) | 51° | 0.625 | **0.813** |
| Right angle: no link | 90° | 0 | **0.5** |
| Opposite | 180° | -1 | **0** |

# 🔢 Predict: which pairs are close?

Each text gets a list of **384 numbers**: its place on a map of meanings. Near **1**: same direction. Around **0.5**: no link for the small model.

- *wildfire* and *forest fire*
- *wildfire* and *campfire*
- *wildfire* and *birthday cake*
- *wildfire* and *incendie de forêt*
- *burnt outside the EU* and *burnt beyond the EU*
- *Access to the forest is forbidden.* and *Access to the forest is allowed.*

# 🔢 What the small model says

| Text 1 | Text 2 | Score |
|---|---|---|
| Access to the forest is forbidden. | Access to the forest is allowed. | **0.964** |
| burnt outside the EU | burnt beyond the EU | **0.944** |
| wildfire | campfire | **0.865** |
| wildfire | forest fire | **0.844** |
| wildfire | birthday cake | **0.618** |
| wildfire | incendie de forêt | **0.559** |

💡 The search finds **what a text is about**, not whether it is true or whether it answers you.

# 🔎 One question, from start to end

**Before:** 37 files, 326 pieces, 384 numbers for each piece. Done once.

**1.** Question 10 becomes 384 numbers.

**2.** Each of the 326 pieces gets a score: from 0.861 down to 0.490.

**3.** Sort, cut after 5. Summary 90 (2.2 million) is 5th (0.847). The 1 092 095 hectares are **7th** (0.832): out.

**4.** Open WebUI pastes the 5 excerpts into the message, in front of your question.

**5.** The model reads this message and writes. It searched nothing.

💡 **R**etrieval (1 to 3), **A**ugmented (4), **G**eneration (5).

# 🎬 Your turn: read the pieces

1. `openwebui query "How many hectares burnt outside the EU in 2025?"`
2. Do summary 10 (**1 092 095** hectares) and summary 90 (**2.2 million**) come back? Look at the score of each excerpt.
3. The same for question 7 (Campania 2025) and question 3 (Bouches-du-Rhône).

| We measured (pieces of 1000, top-k 5) | |
|---|---|
| Q10 | the 2.2 million, **not** the 1 092 095 |
| Q7 | **5 excerpts**, although nothing answers |
| Q3 | the first page of document 43 and a summary about Charente-Maritime, never the articles that answer |

# 🎛️ Change the settings

## Bigger pieces, or more pieces?

20 min · `--size` · `--top-k` · Open WebUI

# 🎛️ What the pieces contain

**Bigger pieces**

```
openwebui query --size 2000 "How many hectares burnt outside the EU in 2025?"
```

Question 10: **both figures** come back.

**More pieces**

```
openwebui query --size 1000 --top-k 10 "How many hectares burnt outside the EU in 2025?"
```

Question 10: **both figures** come back too.

Both send **more text** to the model at every question. Question 3 is still missing.

💡 What matters is **what the pieces contain**.

# ⚖️ Question 10, two sizes, one model

**Wildfire RAG 1000 · judge: 1**

"2 242 195 − 1 079 538 = about 1 162 657 hectares" · "my calculation, not a figure from the documents" · "I can't tell which reading is correct from the excerpts alone."

**Wildfire RAG 2000 · judge: 2**

"Non-EU European countries: 1,092,095 hectares" · "Middle East and North Africa: 70,562 hectares" · "The figures add up only if the 2.2 million includes the EU."

🎬 In Open WebUI: question 10 to each model, then click the **sources**. Optional: `openwebui top-k 10`, question 10 again to **Wildfire RAG 1000**, then `openwebui top-k 5` : `replay RAG` needs 5.

# ⚖️ RAG against everything

## Same questions, another way to read

10 min · `replay RAG` · `score RAG 3 7 10`

# 🎬 Your turn: column RAG

1. `replay RAG`: each question to "Wildfire RAG (workshop kit)", the same knowledge base as Wildfire RAG 1000 (pieces of 1000), top-k 5.
2. `score RAG 3 7 10`: a column RAG appears after B in `scores.md`. Time left? `score RAG 1 2 4 5 6 8 9 11 12`.
3. Compare rows 3, 7 and 10 of columns B and RAG, and the costs. Totals only if both columns are complete.

| We measured | Cost of 11 questions | Judge, 11 questions (3 runs) |
|---|---|---|
| Mode B | $0.47 | TO BE COMPLETED |
| RAG, pieces of 1000 | about $0.14 (estimate) | 18 to 19/22 |
| RAG, pieces of 2000 | about $0.16 (estimate) | 16 to 19/22 |

# 🏁 When to choose a RAG

| | Everything in the context (B) | RAG |
|---|---|---|
| Who picks | nobody: everything enters | a search engine, before the model |
| Size of the knowledge | limited by the window | much bigger |
| Cost of a question | everything, every time | only the pieces |
| Its failure | stops when the knowledge grows | **always gives something**: the failure is invisible |

💡 The test questions show the failures that the chat does not show.

# 🔀 Some chat tools switch on their own

**Claude Projects** (paid plans)

When the knowledge "approaches the context window limit, Claude will automatically enable RAG mode", with "a visual indicator". You cannot choose.

**Gemini**

Going beyond the window "could lead to responses that don't take into account all the content provided". No switch to a search is documented.

📦 The same *Knowledge* box can read everything, search for pieces, or miss part of it.

# 💾 Your chatbot as a RAG, then save

🎬 Second terminal: `openwebui chatbot --name "<your chatbot>" --rag`, then **<your chatbot> chatbot (RAG)** in Open WebUI: `Who are you?`, then your question 12. Your instructions, the same search as "Wildfire RAG".

### 💾 Save

`workshop-save`
right-click on `my-work.zip` → **Download...**
Bring the newest one (maybe `my-work (1).zip`).

Next time: `workshop-start 3 --from my-work.zip`
(or its exact name between quotes, e.g. `"my-work (1).zip"`)

### 📝 **Exit card**, alone

1. Who picks the pieces in a RAG, and when?
2. One question where the size or the number of pieces changed the answer.
3. Why is a RAG's failure harder to see than one of mode B?

# 🆘 If you are stuck

Every command explains itself. From workshop 3, a few **slash commands** inside the chatbot show what it is using.

### ❓ In the terminal

`workshop help`

`replay --help`

`score --help`

### 🔍 In the chatbot

`/context` what fills the window

`/status` the key in use

`/mcp` the tools (from workshop 4)

Workshops 1 and 2: they answer "Unknown command". Watch the gauge above the input box.

### 🤦 Something breaks?

Ask the facilitator. Your files **stay in the Codespace**.

💡 To leave a conversation: **Ctrl+C twice**. In an open `.md` file, to read it as a page: **Ctrl+Shift+V** (**Cmd+Shift+V** on a Mac). In the terminal, copy / paste: **Ctrl+Shift+C / Ctrl+Shift+V** (**Cmd+C / Cmd+V** on a Mac). To change mode: `workshop <mode>`.

# 🧯 Backup: what a RAG hands to the model

If Open WebUI does not start: what we measured for question 10, *"How many hectares burnt outside the EU in 2025?"*, top-k 5, same model.

**1.** The documents are cut into **pieces** (chunks), before any question.

**2.** Each piece becomes a list of numbers: its **embedding**.

**3.** The **5 pieces** closest to the question enter the context.

**4.** The model answers from them only. It never sees the rest.

### Pieces of 1000 · context: 2,291 tokens · judge: 1

Excerpts from `summaries/10`, `corpus/text/10`, `summaries/90`
✅ "2.2 million" (summary 90)
❌ 1 092 095 (document 10): other pieces of it came
➡️ "I can't tell which reading is correct from the excerpts alone."

### Pieces of 2000 · context: 3,145 tokens · judge: 2

Excerpts from `corpus/text/10`, `summaries/10`, `summaries/90`, `corpus/text/90`
✅ "2.2 million" and ✅ 1 092 095
➡️ "The figures add up only if the 2.2 million includes the EU."

💡 Who decides what enters the context? **A search engine**, before the model. Question 7: **5 excerpts**, although nothing answers.