TEACH A SMALL MODEL TO SORT MESSAGES

Youth AI Lab · Workshop 6: going further

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: how can a small model that you teach help a big one?

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

A model that writes, fine-tuned: graphics cards, thousands of examples, and it can still invent.

A model that chooses (a classifier): a few dozen examples, one or two minutes, no graphics card. It **cannot invent**.

🔀 The task: sort each message before the chatbot

Class	What it means	Example	What should happen
`topic`	a question about the chatbot's topic	How can I protect my garden from wildfires?	the chatbot answers, with its instructions and knowledge
`today`	a question that needs the facts of the moment, tonight or this weekend included	What is the fire danger level today?	the chatbot calls the live tool (workshop 4) and reads the result
`urgent`	a fire happening now, or a person in danger now, with or without fire (violence, threats, a wish to die)	There is smoke coming from the woods behind my house!	a fixed safety answer, at once: 112, an adult they trust
`needs-care`	someone sad, scared, lonely or worried, with no danger right now	nobody talks to me at school, i feel so alone	a fixed kind answer, at once: an adult they trust, a support contact, 112 if in danger
`off-topic`	anything else	Can you help me with my maths homework?	a polite refusal: 112, an adult they trust, no tokens spent

💻 Your workspace: one Codespace per pair

1. Open **codespaces.new/Youth-AI-Lab/yal-workshop-finetune** → **Create codespace**. Wait for the terminal.
2. Nothing to bring from the earlier workshops: this one stands on its own.

> ⏱️ **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.

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

🧰 Part 1: install the sorter

1. `workshop-login`, paste the key → `Key OK`. It fails? Tell the facilitator and go on: Parts 1 to 4 do not need it.
2. `workshop-start 6`, then `./sorter setup` → about 2 min 30 → `Model ready`.

> **The model**: `paraphrase-multilingual-MiniLM-L12-v2`, about 118 million parameters, about 50 languages. It turns a sentence into 384 numbers, then compares sentences by **closeness**, whatever the language. It does not write.

🧠 What is a model, and what does training do?

A model is a long list of numbers

Today's small model has 117,653,760 of them. Nobody chose them: training found them.

Training is one loop

An example → the model answers → how wrong is it? → change the numbers, a tiny bit → next example.

Instructions, documents and tools change what a model **reads**. Training changes what a model **is**.

🔍 Part 2: a model that has not learned your classes yet

`./sorter baseline`: each test message gets the class of its **closest example**. Nothing is learned here: it only compares.

```
30 of 40 messages in the right class.
                topic      today     urgent needs-care  off-topic
      topic       …  (rows: right class, columns: chosen class)
```

⚠️ **The mistake that matters most**: an emergency sorted as something else. Read the line **Emergencies missed** first, every time, then **Needs-care messages missed**.

✍️ Part 3: write your examples, and a test for another pair

Good examples

• **examples and test: invented, nothing about a real person**

• sound real: slang, mistakes, "help!!"

• vary: words, languages, 15 words at most

• include the hard cases: "there is smoke, what do I do?", a person in danger with no fire, fear with or without a danger now

What to do

1. ≥ 10 messages per class in `messages/examples.md`, in your language
2. 3 per class (15) on paper or in the chat, the class above each group, for another pair: swap
3. Type theirs into `messages/test.md`, under the classes they gave you

⚠️ **Golden rule**: a test message never goes into the examples. Otherwise the test measures memory, not understanding.

🧪 Part 4: fine-tune, test, improve

SetFit, in two steps

1. The body changes a little, with pairs of examples
2. The head learns the borders between the five classes

What to do

1. Read two sections of the page (4 min), predict,
`./sorter baseline`
2. `./sorter train`. When it ends, `./sorter test`: first score. The talk goes on
3. 2 lines of `Mistakes:` or `Right but below 0.5` at most, emergencies first → 2 examples like each → train, test: improved score, not honest
4. `./sorter try`: the hard messages, the only honest test left, then short cries in your language

🧩 What do you train? Two parts

	The body	The head
What it does	sentence → 384 numbers	384 numbers → class
Size	117,653,760 numbers (about 470 MB)	1,925 numbers (about 9 KB)
Before today	trained by its authors	does not exist yet
Today	changed a little	built from nothing

First, the body: 1,080 pairs from 54 examples, target 1 for the same class, 0 for different classes; 16 pairs at a time, so 68 rounds.

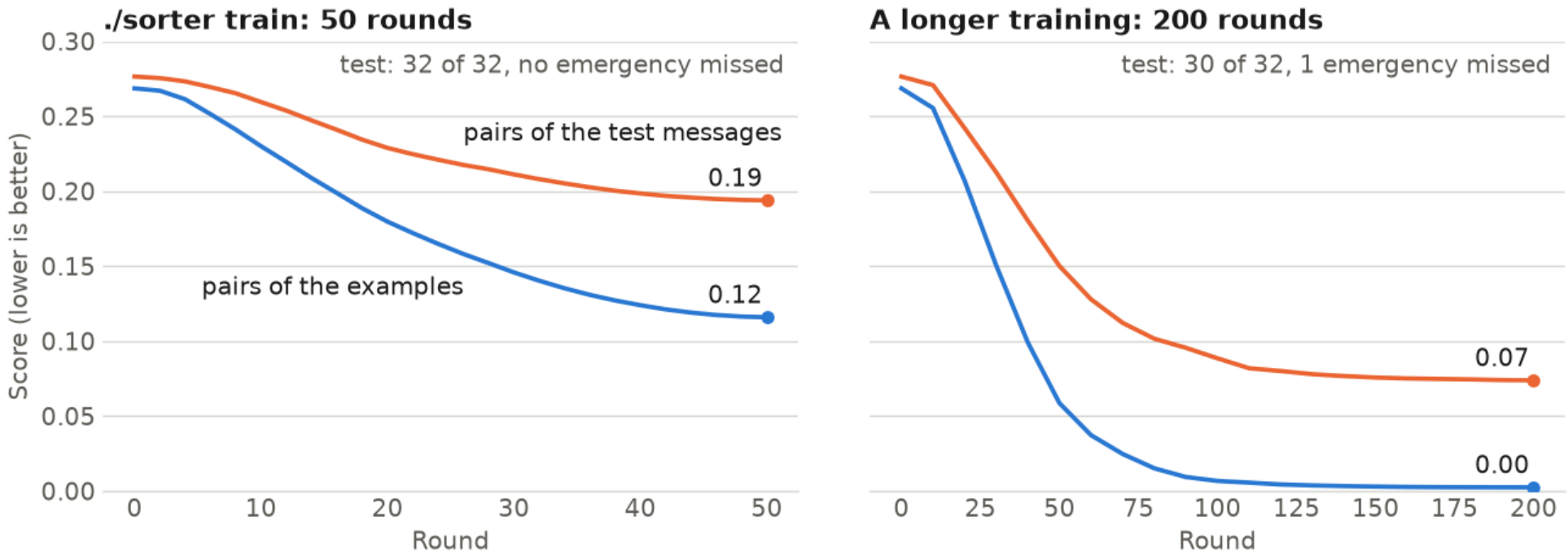
Then, the head: it learns the borders between the five classes, on the new addresses of your examples.

📉 How does the training know it is improving?

The training score (loss function): one number, lower is better.
For a pair: (closeness - target) × (closeness - target).
Target 1, closeness 0.207: score 0.63.
After training, closeness 0.636: score 0.13.

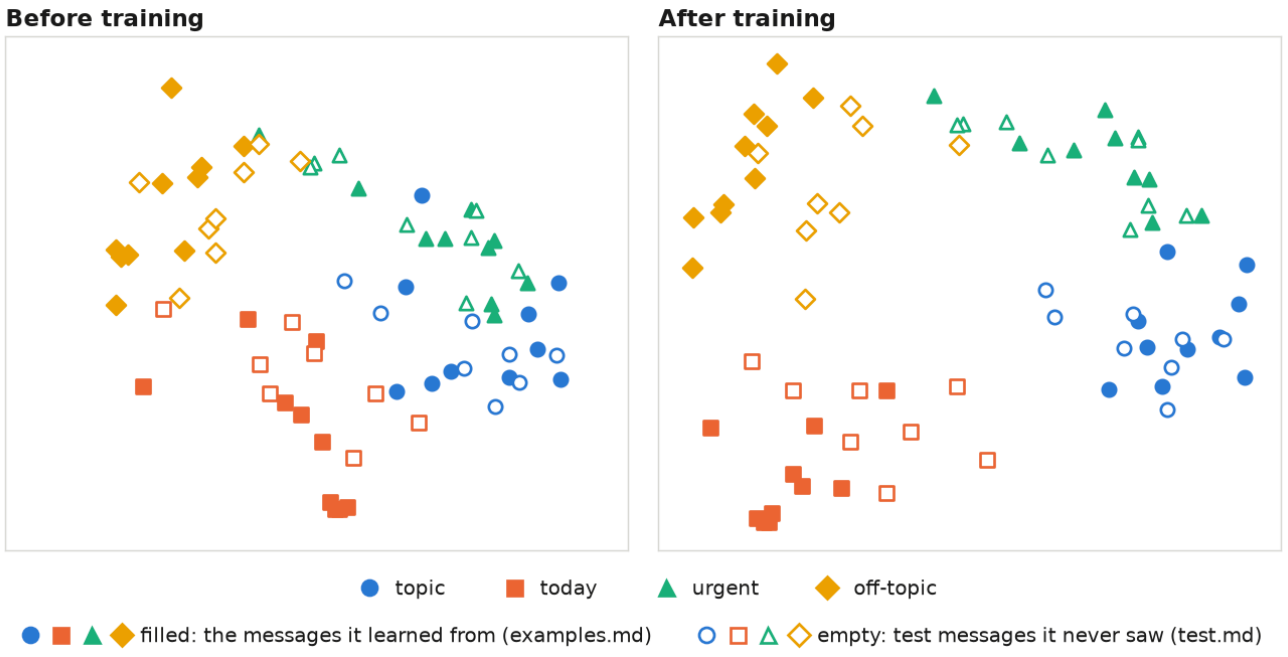
The gradient: for each number, the slope of the score. Each number with a slope takes a very small step downhill, like walking down a hill in fog, feeling the slope under your feet. This is gradient descent.

⏱ When does the training stop?



Here, the number of rounds is fixed in advance. Blue, the lower line: what the model learns from. Orange, the upper line: what it never saw. **A better training score does not guarantee a better sorter.** (First version: four classes, 40 examples.)

🗺️ What the training changes



The 21 million numbers of the 12 layers moved by 0.30% to 0.53%.

Same class: closeness 0.36 → 0.65.
Different classes: 0.20 → 0.20.

"My friend … wants to hurt himself" and "Help, the field next to us is on fire", both urgent: 0.21 → 0.64.
"Je me sens très mal...", now needs-care, and the field on fire: 0.14 → 0.28.

Map: first version, four classes.

⚖️ Part 5: the big model, without training

`./sorter compare`: Claude Haiku, or a chat model with an OpenRouter key, sorts the same test from a **description** of the classes, with no examples.

	Your fine-tuned model	The big model, without training
Right answers on the test (first score)		
Emergencies missed		
Needs-care messages missed		
Time per message	5 to 6 ms in a batch	
Cost per 1,000 messages	none per message	
Works without the network	yes	no
Where sorting happens	in the Codespace	at Anthropic, or at a company reached through OpenRouter
	your examples, one or two	

🏁 Closing: where the pieces could go

🔀 **A sorter could sit before a chatbot**: `urgent`, `needs-care` and `off-topic` would get a fixed answer (112, an adult they trust; for `needs-care`, also a support contact) and never reach the big model. `topic` and `today` would, `today` with the live tool.

🔌 **Yours is connected to nothing**: `./sorter try` prints a class and what would happen. A real service needs a separately tested emergency path, and a person who takes over.

❓ **Who decides what enters the model's context?** Here, a small model could decide whether a message reaches the big model at all.

📝 **Online mistakes**: a person reviews them and writes new invented examples that show the same pattern. Real messages from young people never become examples, even with names removed.

🌍 **The method**: any short list of classes, in about 50 languages. This tool has five, fixed in its code.

🌱 **Why route at all?** Well-being, private data, the energy of data centres: SDGs 3, 12 and 13. A prototype to think with, not a tool.

> 📝 **Exit card**, alone: (1) Why can a classifier not invent? (2) Why never a test message in the examples? (3) One case for fine-tuning a small model, one for asking a big one.

📚 Local AI guide (a model on a machine of your organisation) · how an LLM works · Open WebUI guide (the chatbot a sorter would sit before)

🆘 If you are stuck

❓

In the terminal

`./sorter` lists the commands

📄

On the page

Troubleshooting, at the end of the README: the messages you may see, and what to do

🙋

Something breaks?

Ask the facilitator. Your files stay in the Codespace.

💡 Copy / paste in the terminal: **Ctrl+Shift+C** / **Ctrl+Shift+V** (Cmd+C / Cmd+V on a Mac). An open `.md` file as a page: **Ctrl+Shift+V**.