Beyond the API

Open and local models for digital humanities and cultural heritage

Daniel van Strien

Hugging Face — Machine Learning Librarian

2026-05-05

Hello

- **Daniel van Strien** — Machine Learning Librarian @ Hugging Face
- Background in libraries, archives, computational humanities
- Communities: BigLAM, Small Models for GLAM

Why “Beyond the API”?

The default answer

When someone asks: *“What can we do with AI for our collection?”*

The default answer in 2026 is still:

“Let’s build a chatbot.”

A useful analogy

> A frontier model like GPT-4 is like a Ferrari. Obvious triumph of engineering, designed to win races. But it takes a special pit crew just to change the tires.
>
> A smaller specialised model is like a Honda Civic. Engineered to be affordable, reliable, extremely useful. And that's why they're absolutely everywhere.

— Adapted from "Finally, a Replacement for BERT"

In 2026, our community has the tools to build its own.

A question for the next year

What infrastructure does our community want built — for ML and AI **for and with** *DH and cultural heritage collections?*

What I want you to leave with

1. **What "open" means** — and why it matters for that question
2. **Three shapes of independence** — local, portable, scalable
3. **A demo** of what this looks like end-to-end
4. **Three places** your work matters most — datasets, evaluations, models

Part 1 — What is “open” AI?

Open vs closed: who controls the artifact?

Open weights

- The file is yours
- Run on your laptop, on your server, in the cloud
- Inspect, modify, fine-tune
- The vendor can disappear; the model can’t

Closed weights

- API access only
- Black box
- Vendor controls availability, pricing, deprecation
- Your data leaves your infrastructure

There’s a stricter sense — “fully open” includes training code, data, and evaluation. Important, but a separate conversation. (Appendix.)

Why this matters for DH and cultural heritage

Privacy — unpublished and sensitive material stays on your infrastructure

Longevity — vendors deprecate; archives don't. The artifact has to outlast the company.

Reproducibility — research depends on the model still being available, unchanged, in five years.

Part 2 — How do you run an open model?

First: not everything is an LLM

Many DH/CH tasks don't need a chat model:

- **Classification** — categorise documents, detect language, find a topic
- **Embeddings** — semantic search, similarity, clustering
- **Vision** — OCR, image classification, object detection in archive images
- **Structured extraction** — names, dates, places from historical text

Three shapes of independence

Shape	What it looks like	Example tools
Local	Model file on your machine	`llama.cpp`, `transformers`, MLX
Portable	OpenAI-compatible API, many providers	HF Inference Providers, Together, Fireworks
Rent on-demand	Cloud GPU for one job	`hf jobs`, `vLLM` for batch inference

Open ≠ local. Three shapes — different ways to *not* depend on one provider.

A pattern: Big AI → Little AI

1. Use a **big general-purpose model** to label a small sample
2. Train a **tiny, fast classifier** on those labels
3. Run that classifier on **millions** of documents

This is what web-scale data curation looks like (FineWeb-Edu, WebOrganizer).
It's also what *your* archive looks like — scaled the same way.

Part 3 — A walkthrough of the Hub

What you can find on Hugging Face

- **Models** — over 2.8M, filterable by task, language, size, licence
- **Datasets** — nearly 1M, including DH/CH collections from BigLAM, Europeana, national libraries
- **Spaces** — interactive demos
- **Buckets** — storage built for AI teams: Xet deduplication, included CDN, no git overhead (huggingface.co/storage)
- **Papers** — with code, models, datasets attached

*The Hub is **the platform where the open AI community builds together** — and where the durable artifacts get parked.*

Live walkthrough — switch to browser.

Part 4 — Demo: training a classifier with one prompt

The shape of the problem

Today's example: classify historical legal text as discriminatory or not.

Same shape, applied differently:

- Topic classification across a newspaper corpus
- Semantic search and similarity
- Document type detection
- Language identification
- OCR triage

All variants of: *assign a label to text*.

Why not just call a frontier API?

You could send each document to a closed API. Works for prototyping.

For *research infrastructure*:

- **Cost** — millions of records becomes real money
- **API churn** — the model behind the API changes; results stop reproducing
- **Reproducibility** — your results need to re-run in 5 years
- **Scale** — rate limits + per-request costs collapse at corpus size

Train a small classifier instead. Used to be hard. With data and agents, much less so.

Today's example: biglam/on_the_books

- **UNC Chapel Hill Libraries** — *On the Books: Jim Crow and Algorithms of Resistance*
- Labelled NC session laws, 1866–1967
- 1,785 chapter/section pairs, binary `jim_crow` / `no_jim_crow`
- Sources: Pauli Murray's *States' Laws on Race and Color* + project experts

The prompt

The core ask is one sentence:

> "Fine-tune a model on `biglam/on_the_books` to identify Jim Crow laws.
> Train via `hf jobs` and push to my namespace."

No notebook. No training script. Replace the dataset and the label, and it's your task.

The full prompt I'll paste

```
Fine-tune a model on biglam/on_the_books to identify Jim Crow laws.
Train via hf jobs and push the trained model to my namespace.

Run `hf --help` to understand the Hub CLI and `hf jobs uv run --help`
to understand how to submit uv scripts. You can use `uv run --with`
to run small scripts for exploring the dataset.

Start by exploring the dataset structure, then proceed to choose
and fine-tune an appropriate model.

Push the final model to davanstrien/dhd-demo.
```

The extra lines point the agent at the right docs and naming. The core task is still the first line.

Live demo

Trained from the same one-line prompt, different agents:

davanstrien/jim-crow-laws-claude-code
Claude Code + Opus 4.7 (closed)

davanstrien/jim-crow-laws-pi-kimi
Pi + Kimi K2.6 (open-weight)

Full writeup: danielvanstrien.xyz/posts/2026/agent-race

What just happened

- A **closed tool** (Claude Code) made an **open artifact**
- Training ran on rented compute — gone in 13 minutes
- The model is now in **my namespace**, on my disk, on the Hub
- Tomorrow the agent goes away — the model stays
- Anyone can reproduce, fine-tune, or build on it

Plurality, not monoculture. Use the big general-purpose tools to craft the small, durable ones.

What if you don't have labelled data?

A different bottleneck — same kind of solution.

NLS index card detector

1. SAM3 zero-shot on `hf jobs` (~31% accurate)
2. Correct errors (with agent-built tooling)
3. Train YOLO
4. Repeat

→ **99% mAP in three rounds, one afternoon.**

"The bottleneck was always collecting training data. This is becoming much less of a barrier."

→ `uv-scripts/sam3`

Part 5 — How DH and cultural heritage can contribute

Three places your work matters

Datasets

Curated, well-documented collections.

Domain expertise.

Multilingual / under-represented languages.

Evaluations

Move past “vibe checks”.

Domain experts know what *good* output is.

Edge cases models routinely miss.

Models

Fine-tuned for your domain.

OCR for historical scripts.

Domain-specific embeddings.

Datasets as infrastructure

Back to the question we opened with.

We don’t need fancy AI infrastructure projects.

We need datasets.

Examples: domain-specific models

NLS index card detector

YOLO26n, 99.1% mAP

Built with the same `Claude Code + hf jobs` stack you just saw

— *the agent-built path*

Yiddish OCR

150k training lines
→ ~99% accuracy uplift
→ Mass digitisation now feasible

— *the community-curated path*

Examples: open datasets

Europeana Newspapers — 4M+ documents, multiple languages

→ Powers projects like German Commons (154B tokens of training data)

FineWeb-C — 500+ contributors building multilingual training infrastructure together

Beyond Words / Newspaper Navigator — 16M pages with visual annotations from Library of Congress

Evaluations: the under-explored opportunity

Most LLM benchmarks are:

- English-centric
- Web-data-flavoured
- Narrow (multiple choice, math)

DH/CH benchmarks would test things models **actually struggle with**:

- Multilingual historical text
- OCR-noisy input
- Domain-specific entity recognition
- Period-appropriate language understanding

Try this after the talk

- **Play with an agent on the Hub** → `smolagents/ml-intern`
- **BigLAM community** → `huggingface.co/biglam`
- **Small Models for GLAM** → `huggingface.co/small-models-for-glam`
- **Re-run today's demo** → agent-race writeup + traces
- **Free Hub courses** → `huggingface.co/learn`

If you want to put a dataset on the Hub, get in touch — that's part of what I'm here for.

Thank you

Daniel van Strien

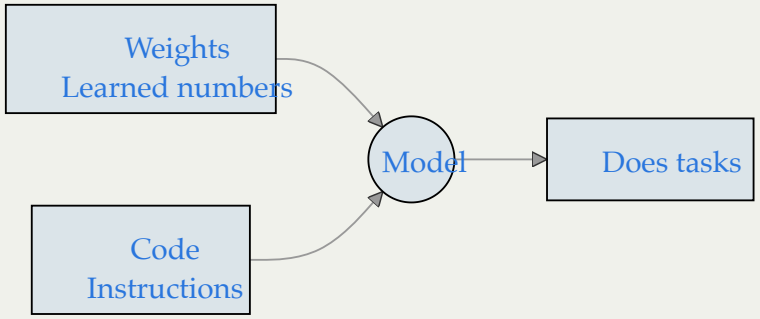
davanstrien on Hugging Face
linktr.ee/danielvanstrien
daniel@huggingface.co

Questions?

Appendix

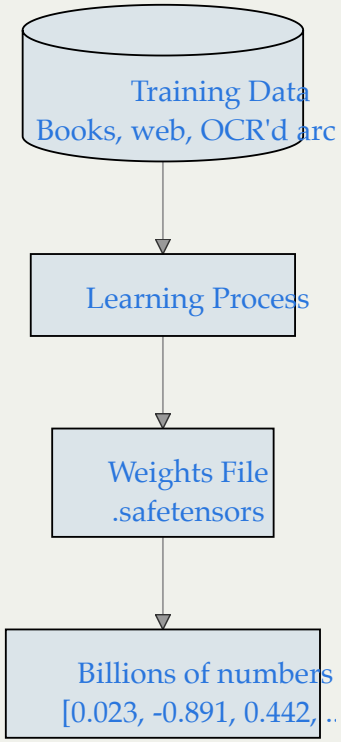
Reference slides — not part of the linear talk. Reachable for Q&A.

What's in a model?



The **weights** are the "brain" — patterns learned from training data.

What are weights, concretely?



These numbers encode everything the model “knows”.

“Open weights” ≠ “open source”

Truly open source AI also includes:

- **Training code** — how the model was built
- **Training data** — what it learned from
- **Evaluation** — how performance was measured

Examples of “fully” open models:

- OLMo (Allen AI) — fully open LLM
- SmolLM — small, efficient, fully open

(a) Pay per token / serverless

Pros

- Minimal setup
- No hardware
- Easy model switching
- Often OpenAI-compatible API

Cons

- Vendor dependency (lighter than closed APIs, but still)
- Results can vary across providers
- Ongoing costs
- Data leaves your infra

(b) Local hardware

Pros

- Full control
- Data stays on your machine
- No ongoing API costs
- Works offline

Cons

- Hardware limits model size
- You manage updates
- Setup learning curve
- Slower for very large models

(c) Rent hardware (cloud)

Pros

- Run any model size
- Scalable
- Production-ready
- No local setup

Cons

- Costs scale with usage
- Cloud-skill required
- Cold starts

The local-models ecosystem