C U R S 2

Ecosistemul de modele

Alegerea modelului pentru proiectul EchoChamber

Ingineria Ai – Aplicații cu agenți inteligenți
Analiza Datelor Complexe, Facultatea de Sociologie și Asistență Socială, Universitatea Babeș Bolyai

CURS 2.

C1 LLM-uri, API-uri – ce construim și cum pornim
C2 Ecosistemul de modele - alegem modelul de bază
C3 Date și corpus – colectare, curățare, metadate
C4 Prompting, adnotare, context engineering
C5 Embeddings, retrieval, RAG
C6 RAG, agenți, LangChain, LangGraph
C7 Agenți orchestrați – LangGraph,metrici,etică
C8 Integrare aplicație – Gradio
C9 Demo final – prezentări și feedback

Tematică

- Ecosistemul de modele: unde găsim și cum accesăm modele
- Cum comparăm modelele: benchmark-uri, limite și criterii relevante
- Evaluare în limba română, JSON / structured output și comportament la rol
- Failure modes: halucinații, refuzuri, instabilitate, trunchiere
- Alegerea modelului ca decizie metodologică pentru proiect

Activități practice

- Alegem 2 modele cu acces gratuit: Gemini și OpenRouter
- Rulăm aceleași prompturi pe fiecare model
- Testăm: română, adnotare, structured output / JSON, stabilitate la temperaturi diferite
- Fiecare student modifică prompturile și compară rezultatele
- Echipa discută rezultatele și alege model principal + fallback
- Pornim app.py cu un test minim de model

Livrabile

- Notebook individual C2 în notebooks/student_XX/
- Prompturi proprii + comparație Gemini / OpenRouter
- Decizie individuală: model, fallback, temperature
- Config final de echipă în core/config.py
- App inițial în app.py
- DONE în GitHub Issues

→ **La finalul C2 trebuie să știi nu doar să folosești un model, ci să justifici de ce l-ai ales pentru proiectul tău.**

Cum diferă modelele LLM și de ce contează

CE ESTE UN MODEL

- Un LLM nu stochează fapte - generează răspunsuri probabilistice din tipare. Două modele pe același prompt produc rezultate diferite: alt stil, precizie, prudență.
- Diferențele vin din: datele de antrenare (ce texte a văzut și în ce proporții), alignment (cum a fost instruit să se comporte) și fine-tuning (specializare pe domeniu sau tip de task).

PROPRIETARY VS. OPEN-WEIGHT MODELS VS. LOCAL

- **Proprietary** (GPT, Claude, Gemini) - fără acces la weights, rulează doar prin API.
- **Open-weight** (Llama, Mistral, Gemma, Qwen) - weights publice, pot fi rulate local sau prin API-uri terțe.
- **Cloud/API vs. local** sunt axe separate: Llama 3.3 rulează prin API pe Together AI sau OpenRouter; Gemma rulează local prin Ollama.
- În curs: folosim API cloud la început. Rularea locală și open-weights le discutăm ulterior.

SCOPUL ȘI DIMENSIUNEA MODELULUI

- **Reasoning / Pro** (o3, Gemini 2.5 Pro, Claude Opus 4) - sarcini complexe: analiză, cod, raționament multi-pas. Mai lent și mai scump.
- **Flash / Mini / Small** (Gemini 2.5 Flash, GPT-4o-mini, Claude Haiku, Mistral Small) - rapid și ieftin, bun pentru pipeline-uri, clasificare, rezumare.
- **Instruction-tuned vs. base**: modelele instruction-tuned urmează instrucțiuni, produc JSON structurat, sunt potrivite pentru agenți. Modelele base generează text liber
- **Limbă și context**: unele modele sunt mai bune pe română (Gemini, Claude față de GPT-4o-mini pe texte politice). Context window variază de la 8k la 2M tokens.

LLM Providers

Model labs · Platforme / routere / cloud · Open-weight / local

MODEL LABS / HOSTED API

OpenAI GPT-5.4, mini, nano

Reasoning, coding, tool use, computer use platform.openai.com

Anthropic Opus 4.7, Sonnet 4.6, Haiku

Long context, agentic coding, speed/intelligence console.anthropic.com

Google Gemini 2.5 Pro, Flash, Lite

Multimodal, 1M ctx, preț/performanță aistudio.google.com

Cohere Command A, R7B, Embed, Rerank

RAG, enterprise search, embeddings, re-ranking cohere.com

Mistral Large 3, Medium 3.1, Small 4

Frontier + open-weight, multimodal, API+self-host mistral.ai

DeepSeek V3.2, Reasoner

Reasoning low-cost, JSON output, tool calls deepseek.com

xAI Grok 4.20, Grok 4.1 Fast

2M context, tool calling, structured outputs x.ai/api

PLATFORME / ROUTERE / CLOUD

Azure OpenAI GPT-5.4+, model-router

Enterprise, governance, ecosistem Microsoft azure.microsoft.com

AWS Bedrock 103 modele, 17 provideri

Multi-provider fără schimbare de infrastructură aws.amazon.com/bedrock

HuggingFace Mii de modele

Browser, playground, free tier, unified inference huggingface.co

OpenRouter 300+ modele, 60+ provideri

OpenAI-compatible, routing și failover automat openrouter.ai

Together AI 200+ open-source models

API unificat, fine-tuning, training together.ai

Groq Llama 4 Scout, Qwen3-32B

Inferență ultra-rapidă pe hardware LPU dedicat groq.com

Cloudflare AI 60+ modele, edge inference

Inferență la edge, 10.000 neuroni/zi gratuit ai.cloudflare.com

OPEN-WEIGHT / LOCAL

Meta Llama Llama 4 Scout (10M ctx), Maverick

Major open-weight family, ecosistem larg llama.com

Google Gemma Gemma 3: 1B, 4B, 12B, 27B

Single GPU / on-device, 128K ctx, 140+ limbi ai.google.dev/gemma

Mistral OSS Small 3.2, Ministral 3B/8B/14B

Edge, multimodal, self-deployment, Apache 2.0 docs.mistral.ai

DeepSeek OSS R1, V3.2 (open-weights)

Open-weights puternice, licență MIT/DSL github.com/deepseek-ai

Qwen (Alibaba) Qwen3, 0.6B–235B MoE

Multilingv, coding, context lung, Apache 2.0 huggingface.co/Qwen

Ollama Orice model local

Self-hosted, privacy totală, zero cost API ollama.com

vLLM Engine high-throughput

Producție, PagedAttention, OpenAI-compatible API docs.vllm.ai

Lista completă cu free tiers:
github.com/cheahjs/free-llm-api-resources · Benchmark-uri: artificialanalysis.ai

SLM vs LLM

SLM (Small Language Model):
model mai mic, mai rapid și mai ieftin; potrivit pentru dispozitive locale, edge AI, aplicații offline și taskuri repetitive - potrivit pentru medii cu resurse limitate, inclusiv edge devices și mobile apps.

LLM (Large Language Model):
model mai mare, în general mai bun pe reasoning, instrucțiuni complexe, ambiguitate și generare mai bogată, dar mai costisitor și mai greu de rulat.

Aplicații

- SLM: clasificare, extracție JSON, routing, asistenți locali, edge computing, aplicații offline, prototipuri ieftine.
- LLM: RAG mai bogat, raționare multi-step, agenți mai sofisticați, sarcini cu ambiguitate mare.

Quantization

- 8-bit reduce aproximativ la jumătate memoria necesară;
- 4-bit comprimă și mai mult și este util când vrem să rulăm modele mai mari pe hardware limitat.

Categorie	Modele
Clar SLM	RecurrentGemma 2B, OpenELM 1.1B, 3B, Phi-1 1.3B, Phi-2 2.7B, MiniCPM 2B, CPM-Bee 1B, 2B, StableLM 3B, StableCode 3B, Qwen2 0.5B, 1.5B, InternLM2 1.8B, Pythia 1B, 1.4B, 2.8B
Zonă gri / modele compacte	Llama 2 7B, Mistral 7B, Gemma 7B, Phi-3 7B, OLMo 7B, DeepSeek-Math 7B, CodeQwen 7B, Qwen 7B, Baichuan 7B, StarCoder 7B, MPT 7B, InternLM2 7B, Qwen1.5 4B, Yi1.5 6B, 9B, DeepSeek-Coder 6.7B, DeepSeek-VL 7B, StarCoder2 7B
Clar LLM	Llama 2 13B, 70B, Llama 1 7B, 13B, 33B, 65B, OPT 1.3B, 6.7B, 13B, 30B, 66B, Gemma 2 7B, Phi-3 14B, OLMo 7B, Grok-1 314B MoE, DeepSeek-Coder 33B, DeepSeek-MoE 16B, Qwen 14B, 72B, Qwen1.5 7B, 14B, 32B, 72B, 110B, Qwen2 7B, 57B MoE, 72B, Yi 34B, Yi1.5 34B, Yi-VL 6B, 34B, Baichuan 13B,

https://github.com/ro-ko/Awesome-SLM

Hugging Face

huggingface.co

- *Un fel de GitHub pentru modele, datasets și aplicații AI. Găzduiește peste 1 milion de modele publice, 250.000+ datasets și Spaces*
- *Orice model open-weight major (Llama, Mistral, Gemma, Qwen, DeepSeek) e publicat și descărcabil de aici*

MODEL HUB ȘI INFERENCE

- Fiecare model are un card cu arhitectură, licență, benchmark-uri și cod de inferență. Poți filtra după task, limbă, licență sau framework.
- Inference Providers: testezi orice model direct în browser sau prin API, fără să-l descarci - HF routează cererea la Together AI, Fireworks, SambaNova etc.

DATASETS ȘI LEADERBOARD-URI

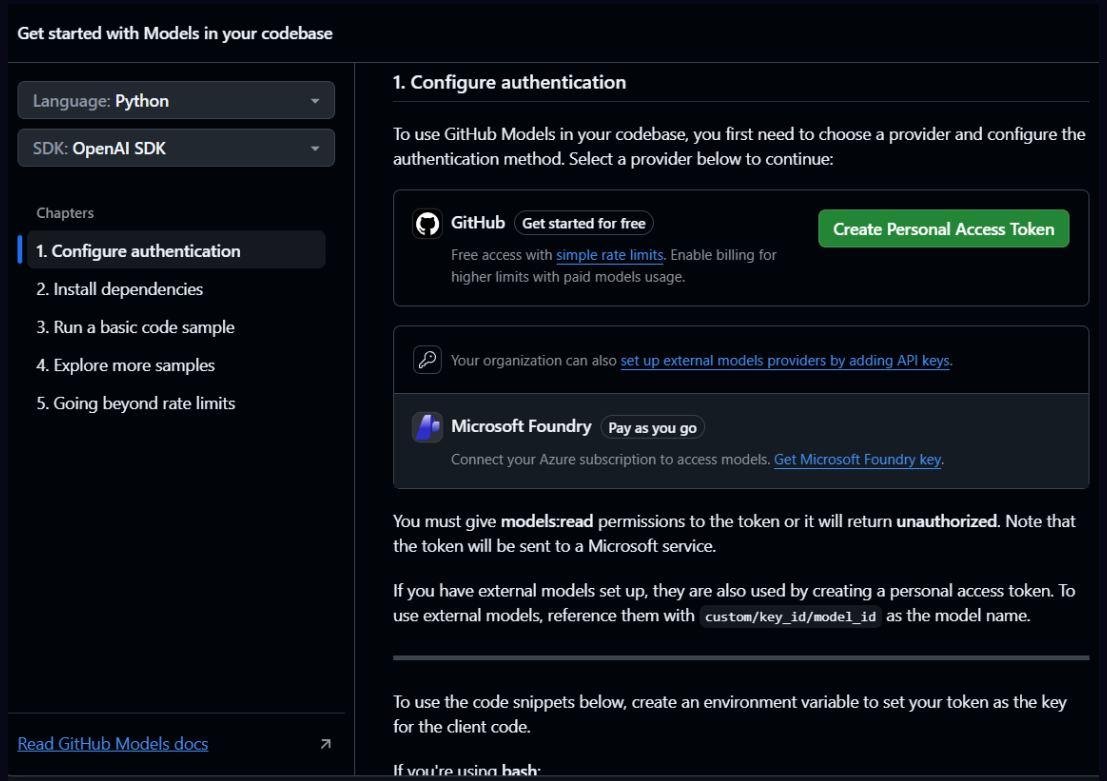
- 250.000+ seturi de date publice, inclusiv corpus-uri de texte politice, știri și discursuri parlamentare în română (CoRoSeNT, ParlaMint-RO, MARCELL).
- Open LLM Leaderboard și MTEB Leaderboard — evaluare standardizată a modelelor open-weight, esențiale pentru comparații obiective și alegerea modelului de embeddings pentru RAG.

ECOSISTEM PYTHON

- transformers - interfață unificată pentru mii de modele: from transformers import pipeline.
- datasets - încărcare și procesare standardizată a dataset-urilor direct din hub.
- peft - fine-tuning eficient cu LoRA/QLoRA fără hardware masiv.
- evaluate - metrici standardizate (ROUGE, BERTScore, accuracy) cu o linie de cod.

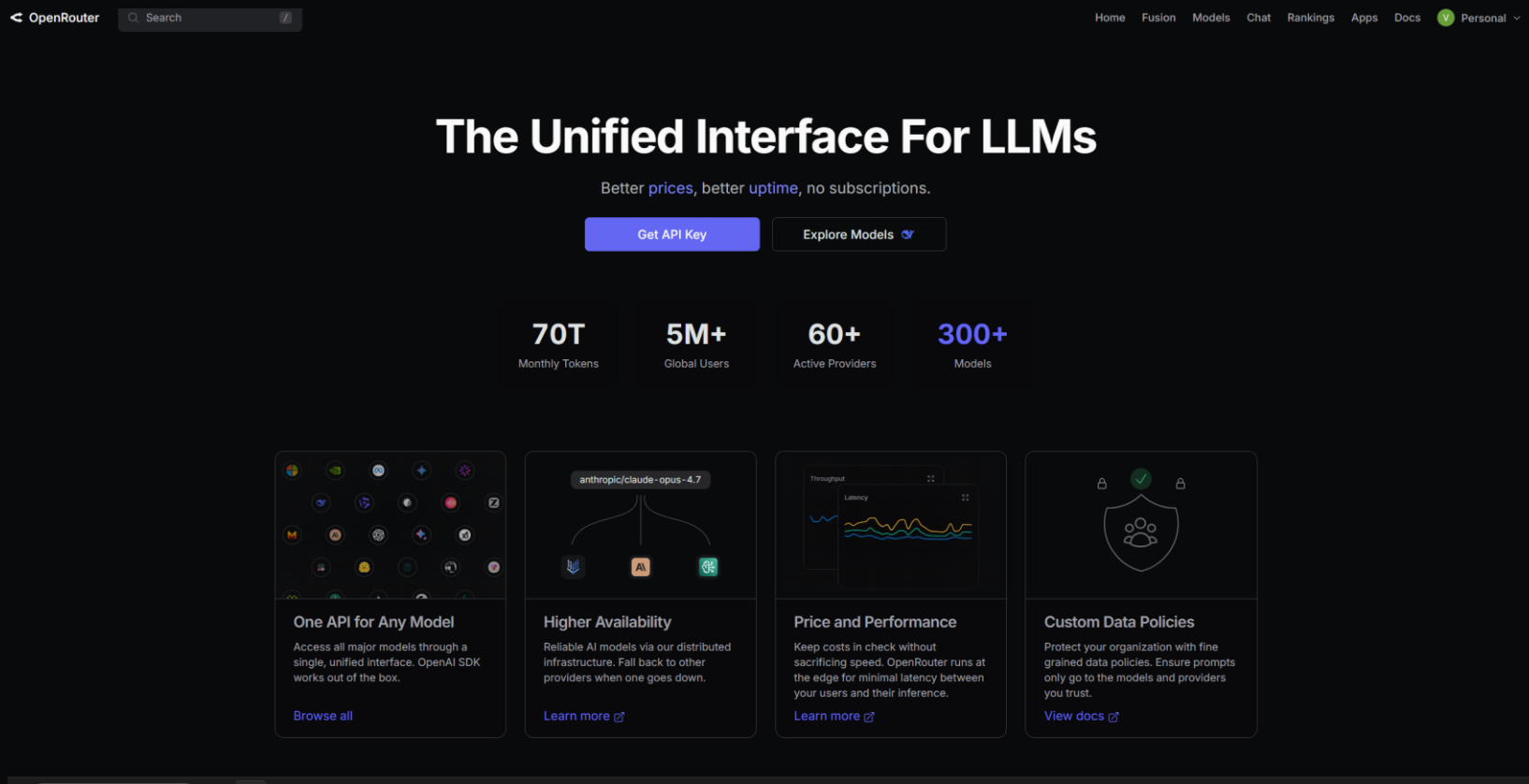
GitHub Models

https://github.com/marketplace/models



Open Router Models

https://openrouter.ai/



Leaderboards: ce tip de semnal ofera

Nu toate clasamentele masoara acelasi lucru.

Battle Arenas

Semnal: Preferinta umana in comparatii directe

Compara: Raspunsuri reale, cu vot anonim

Utile pentru: Stil, naturalete, impresie generala

lmarena.ai / web.lmarena.ai / search.lmarena.ai

Benchmark-uri reproductibile

Semnal: Scor pe taskuri standardizate

Compara: Performanta pe seturi de evaluare fixe/actualizate

Utile pentru: Comparatie stabila, mai ales open-weight

huggingface.co/open-llm-leaderboard / livebench.ai

Indexuri operationale

Semnal: Cost, latenta, throughput, context

Compara: Cat de practic este modelul in utilizare reala

Utile pentru: Alegere intre modele apropiate ca performanta

artificialanalysis.ai / vellum.ai/llm-leaderboard

Specializate

Semnal: Performanta pe o nisa specifica

Compara: Retrieval, embeddings, coding etc.

Utile pentru: Taskuri specifice, nu alegere generala

MTEB / Coding Index

-> Un leaderboard nu spune cel mai bun model, ci cel mai bun model pe un anumit tip de evaluare.

CUM CITIM RAPID

- Ce tip de semnal ofera?
- Pe ce tip de taskuri a fost construit?
- Cat de apropiat e de cazul nostru?
- Ce NU masoara?

CE NU ITI SPUNE

- care model merge pe romana
- care model produce JSON bun
- care model e optim pt. taskul nostru
- raportul calitate / cost in pipeline

Cum alegem un model?

Leaderboards reduc spatiul de cautare. Alegerea finala se face local.

1

Definim taskul
inainte de clasamente

Intrebare corecta:
ce vrem noi sa faca modelul?

Taskuri in curs:

- romana, JSON, RAG,
- sumarizare, clasificare,
- prompt following,
- cost mic, latenta mica

2

Alegem leaderboard-ul
potrivit pt. taskul potrivit

stil / naturalete
-> LM Arena
open-weight comparison
-> HF Open LLM
contaminare
-> LiveBench
cost / viteza / latenta
-> Artif. Analysis, Vellum
RAG / embeddings
-> MTEB

3

Citim critic,
nu doar locul in clasament

La arene:
voturi, 95% CI,
tipul prompturilor
La benchmark-uri:
ce probe intra in scor,
e saturat sau gaming?
La indexuri:
pret/1M, TTFT, context
La embeddings:
retrieval score, nu media

4

Decizia finala
se ia local

- merge pe romana?
- produce JSON valid?
- respecta instructiunea?
- halucioneaza?
- e suficient de rapid?
- e suficient de ieftin?

Cum testam modelul pentru proiect

Testam aceleasi prompturi pe toate modelele.

CRITERII

1. **Romana** — rezuma clar si natural
2. **Instrucțiuni** — respecta system prompt-ul
3. **Adnotare** — clasifica un comentariu politic
4. **Structured output** — produce JSON valid
5. **Stabilitate** — raspunsuri similare la temperaturi mici
6. **Cost / quota** — poate fi folosit de echipa fara blocaje

CUM LUCRĂM

1. Fiecare student rulează notebookul în notebooks/student_XX/
2. Modifică prompturile primite la curs
3. Compară răspunsurile modelelor
4. Alege model recomandat, fallback și temperatură
5. Discută rezultatul cu echipa

DE CE CONTEAZĂ

Baza agentilor
Modelul ales devine fundatia pentru toti agentii din proiect.

Pipeline-ul depinde de el
Adnotarea si app.py ruleaza pe iesirile acestui model.

Când putem folosi răspunsul modelului?

RĂSPUNS BUN

Clar - se înțelege fără explicații suplimentare

Controlabil - urmează rolul și instrucțiunea

Reproductibil - păstrează același format

Reutilizabil - poate fi procesat mai departe, ideal JSON

EXEMPLU

Schema cerută:

```
ton = pozitiv / negativ / neutru
populism = true / false
```

- Output bun: `{"ton": "negativ", "populism": true}`
- Output slab: `{"ton": "critic", "populism": "da"}`

CE POATE MERGE PROST

Halucinații
răspuns sigur, dar factual greșit

Ignorarea rolului
instrucțiunea de sistem (system prompt) nu este respectată

Refuzuri inutile
modelul blochează sarcini acceptabile

Output instabil
același prompt produce rezultate prea diferite

Trunchiere
raspunsul se oprește înainte să termine

Livrabile - GitHub Issues

C2.1 — Individual

- Fiecare student pune notebookul C2 în:
- notebooks/student_XX/C2_model_ecosystem.ipynb
- În notebook: testează Gemini + OpenRouter, modifică prompturile, compară răspunsurile și alege modelul recomandat.

C2.2 — Echipă

- Un singur student actualizează:
- core/config.py
- Echipa stabilește: model principal, fallback, temperature.

C2.3 — Echipă

- Un singur student creează:
- app.py
- App minim: citește config.py, folosește .env local și trimite un prompt la model.
- Milestone GitHub: C2 - Model ecosystem
- Finalizare: comentariu DONE în issue.

-> C2 produce prima versiune funcțională: un chat simplu cu model selectabil.

Următorul curs

Curs 3 - Colectare de date, curățare și corpus

În C3 construim „memoria” aplicației EchoChamber: corpusul de texte pe care îl vom folosi ulterior pentru adnotare, retrieval și RAG.

- colectăm texte din surse publice: YouTube, RSS, documente oficiale
- păstrăm metadate: sursă, dată, platformă, link
- curățăm corpusul: duplicate, texte goale, zgomot, lungime
- salvăm datele în `data/raw/` și `data/cleaned/`
- testăm primul prompt exploratoriu pe comentarii politice

-> În C2 alegem modelul. În C3 îi dăm materialul pe care va lucra.