Machine Learning and Reasoning with Built Environment Data

Work with LLMs

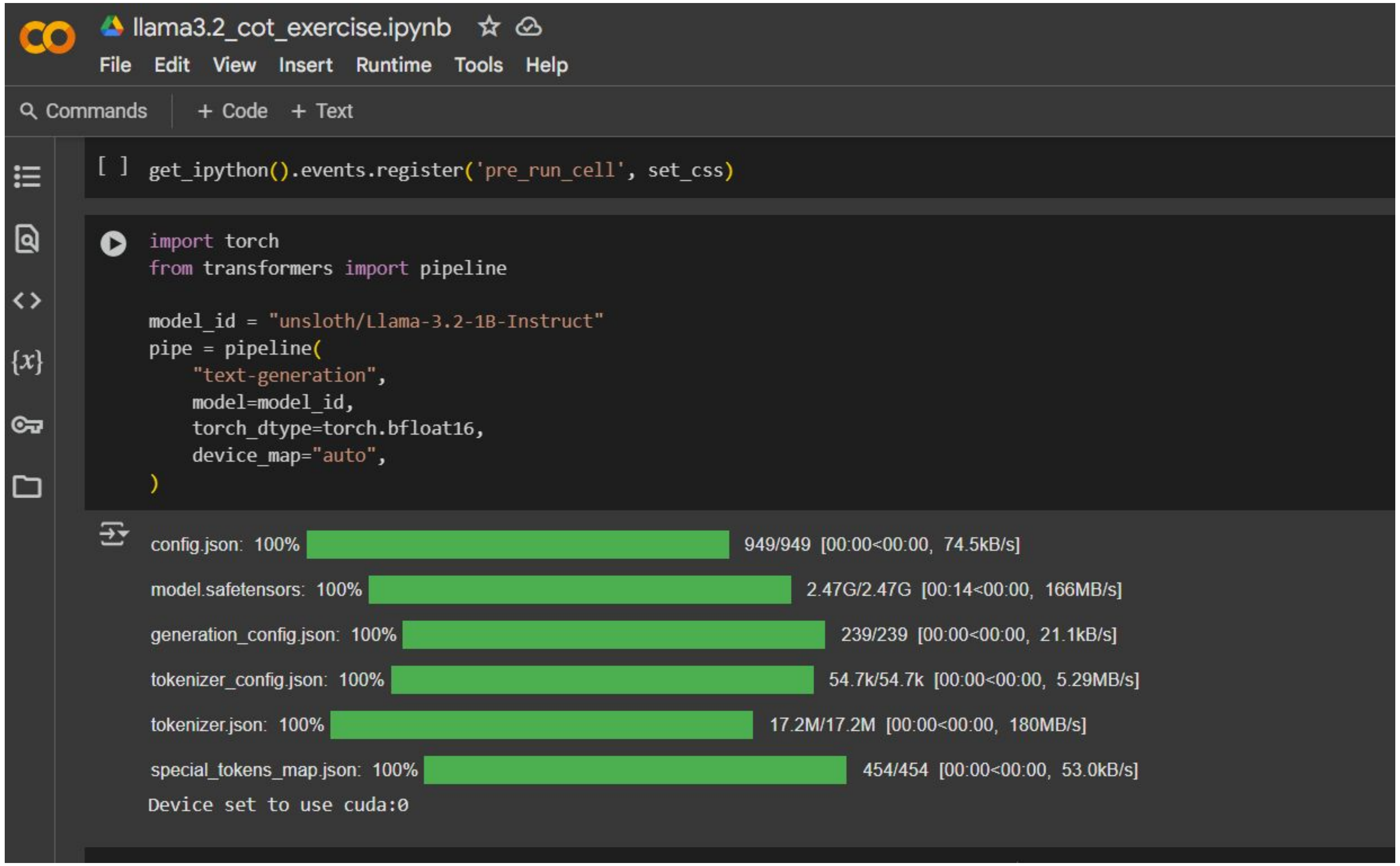
Computing in Construction

Metropolia
University of Applied Sciences

Update of Earlier Script

- Instructions of Token settings added

- model_id changed, no separate permission needed


llama3.2_cot_exercise.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```
[ ] get_ipython().events.register('pre_run_cell', set_css)
```

```
import torch
from transformers import pipeline

model_id = "unsloth/Llama-3.2-1B-Instruct"
pipe = pipeline(
    "text-generation",
    model=model_id,
    torch_dtype=torch.bfloat16,
    device_map="auto",
)
```

```
config.json: 100% 949/949 [00:00<00:00, 74.5kB/s]
model.safetensors: 100% 2.47G/2.47G [00:14<00:00, 166MB/s]
generation_config.json: 100% 239/239 [00:00<00:00, 21.1kB/s]
tokenizer_config.json: 100% 54.7k/54.7k [00:00<00:00, 5.29MB/s]
tokenizer.json: 100% 17.2M/17.2M [00:00<00:00, 180MB/s]
special_tokens_map.json: 100% 454/454 [00:00<00:00, 53.0kB/s]
Device set to use cuda:0
```

Metropolia University of Applied Sciences

Limitations

New information later than the cutoff dates off LLMs?

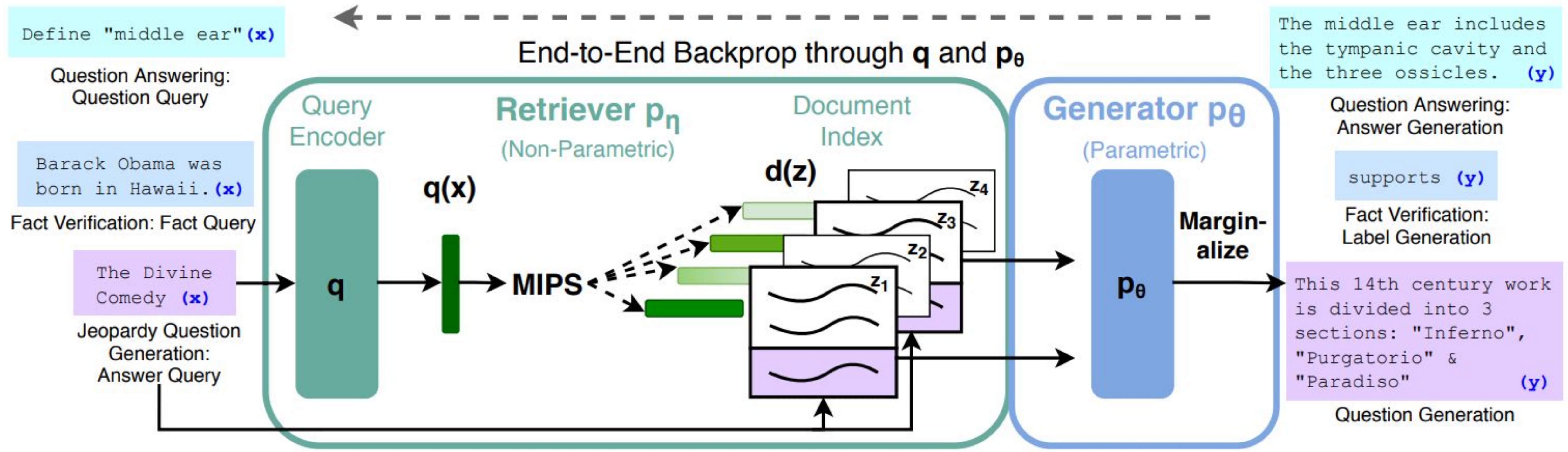
Information not included in the training dataset of LLMs?

Data from local files which are not supposed to share?

Longer texts with more tokens than the context window?

Metropolia
University of Applied Sciences

Retrieval-Augmented Generation (RAG)



End-to-End Backprop through q and pθ
Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
Query Encoder
q
q(x)
MIPS
Retriever pη
(Non-Parametric)
Document Index
d(z)
z4
z3
z2
z1
Generator pθ
(Parametric)
pθ
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

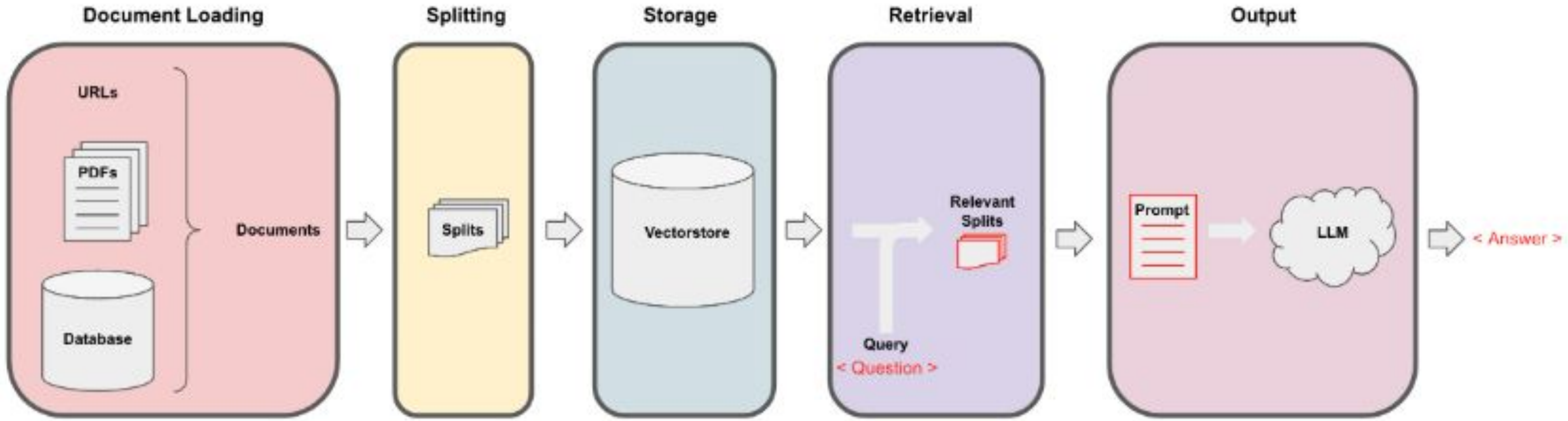
Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

How does it work?

Retrieval augmented generation

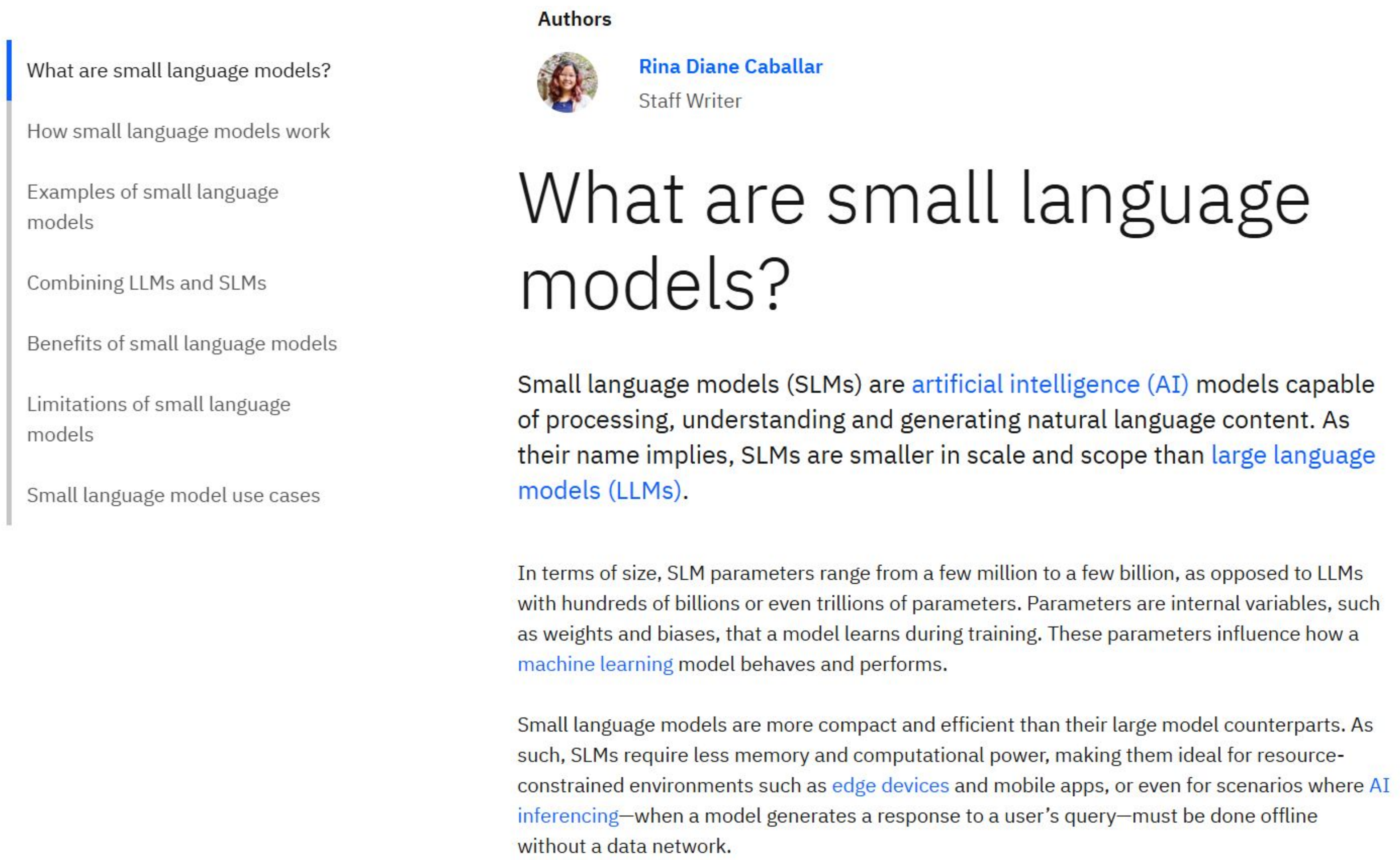
In retrieval augmented generation (RAG), an LLM retrieves contextual documents from an external dataset as part of its execution.

This is useful if we want to ask question about specific documents (e.g., our PDFs, a set of videos, etc).



Why Use Open Source LLMs

Large or Small?



What are small language models?

How small language models work

Examples of small language models

Combining LLMs and SLMs

Benefits of small language models

Limitations of small language models

Small language model use cases

Authors

Rina Diane Caballar
Staff Writer

What are small language models?

Small language models (SLMs) are artificial intelligence (AI) models capable of processing, understanding and generating natural language content. As their name implies, SLMs are smaller in scale and scope than large language models (LLMs).

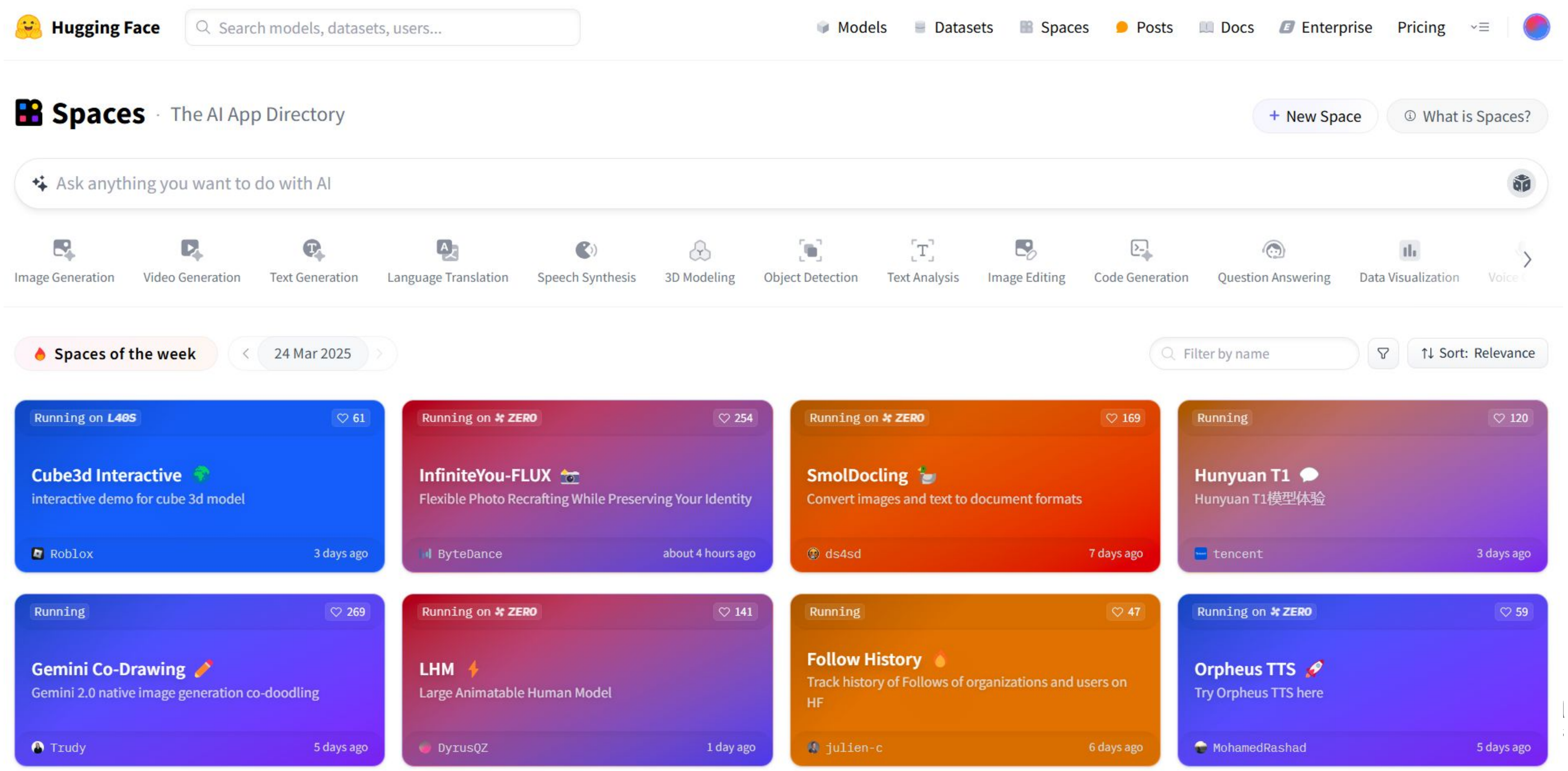
In terms of size, SLM parameters range from a few million to a few billion, as opposed to LLMs with hundreds of billions or even trillions of parameters. Parameters are internal variables, such as weights and biases, that a model learns during training. These parameters influence how a machine learning model behaves and performs.

Small language models are more compact and efficient than their large model counterparts. As such, SLMs require less memory and computational power, making them ideal for resource-constrained environments such as edge devices and mobile apps, or even for scenarios where AI inferencing—when a model generates a response to a user's query—must be done offline without a data network.

AI FIGHTER
VS

Airtrain
Large Language Models
LLM quantization!?
quantization
performance
inference speed

Explore Hugging Face



Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Posts | Docs | Enterprise | Pricing

Spaces · The AI App Directory

+ New Space | What is Spaces?

Ask anything you want to do with AI

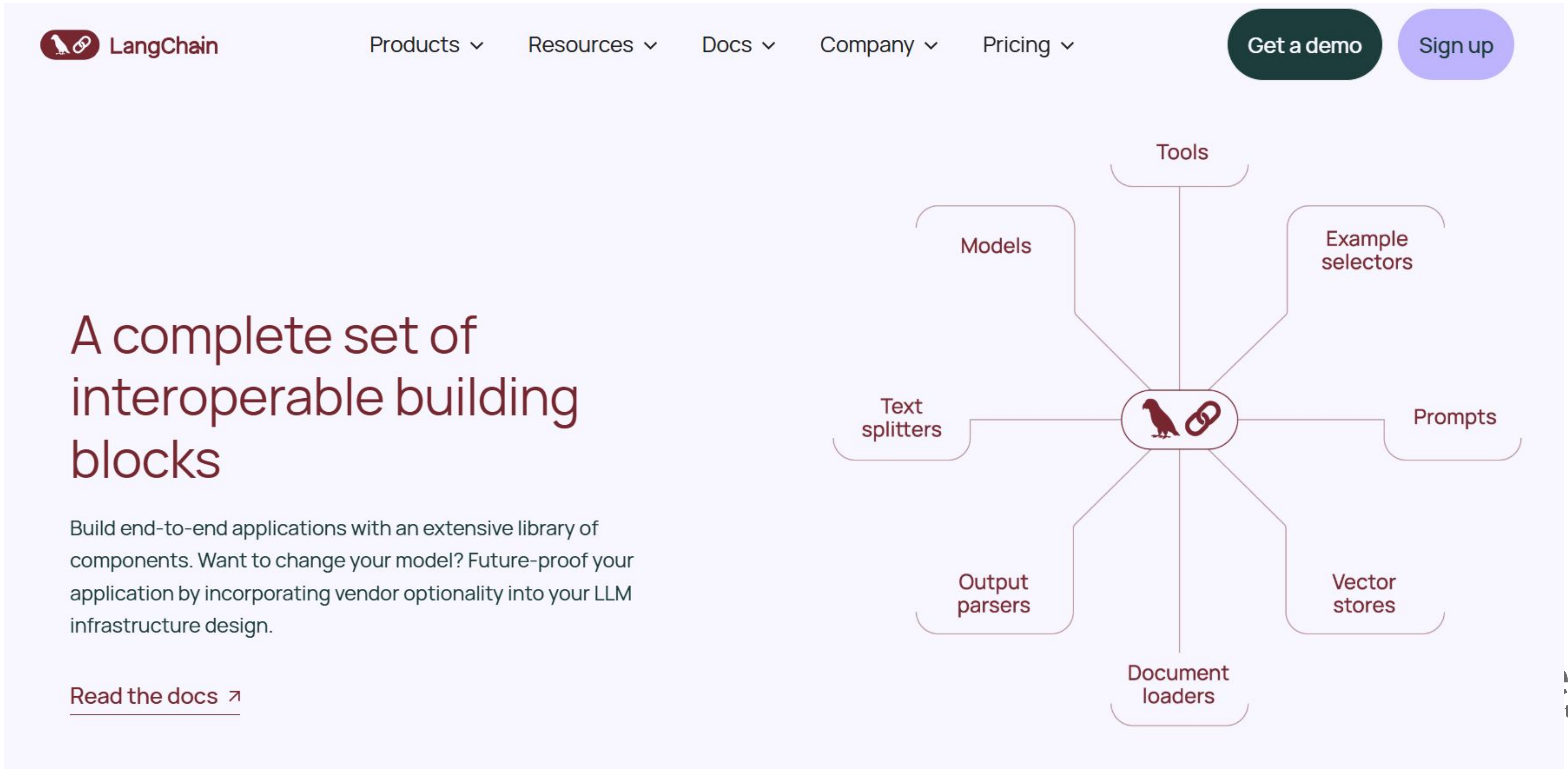
Image Generation | Video Generation | Text Generation | Language Translation | Speech Synthesis | 3D Modeling | Object Detection | Text Analysis | Image Editing | Code Generation | Question Answering | Data Visualization | Voice

Spaces of the week | 24 Mar 2025 | Filter by name | Sort: Relevance

Status	Likes	Name	Description	Author	Updated
Running on L40S	61	Cube3d Interactive	interactive demo for cube 3d model	Roblox	3 days ago
Running on ZERO	254	InfiniteYou-FLUX	Flexible Photo Recrafting While Preserving Your Identity	ByteDance	about 4 hours ago
Running on ZERO	169	SmolDocling	Convert images and text to document formats	ds4sd	7 days ago
Running	120	Hunyuan T1	Hunyuan T1模型体验	tencent	3 days ago
Running	269	Gemini Co-Drawing	Gemini 2.0 native image generation co-doodling	Trudy	5 days ago
Running on ZERO	141	LHM	Large Animatable Human Model	DyrusQZ	1 day ago
Running	47	Follow History	Track history of Follows of organizations and users on HF	julien-c	6 days ago
Running on ZERO	59	Orpheus TTS	Try Orpheus TTS here	MohamedRashad	5 days ago

Popular Python Libraries

LangChain is an open-source framework designed to help developers build applications powered by large language models (LLMs). It focuses on streamlining the process of creating complex, language-driven workflows.

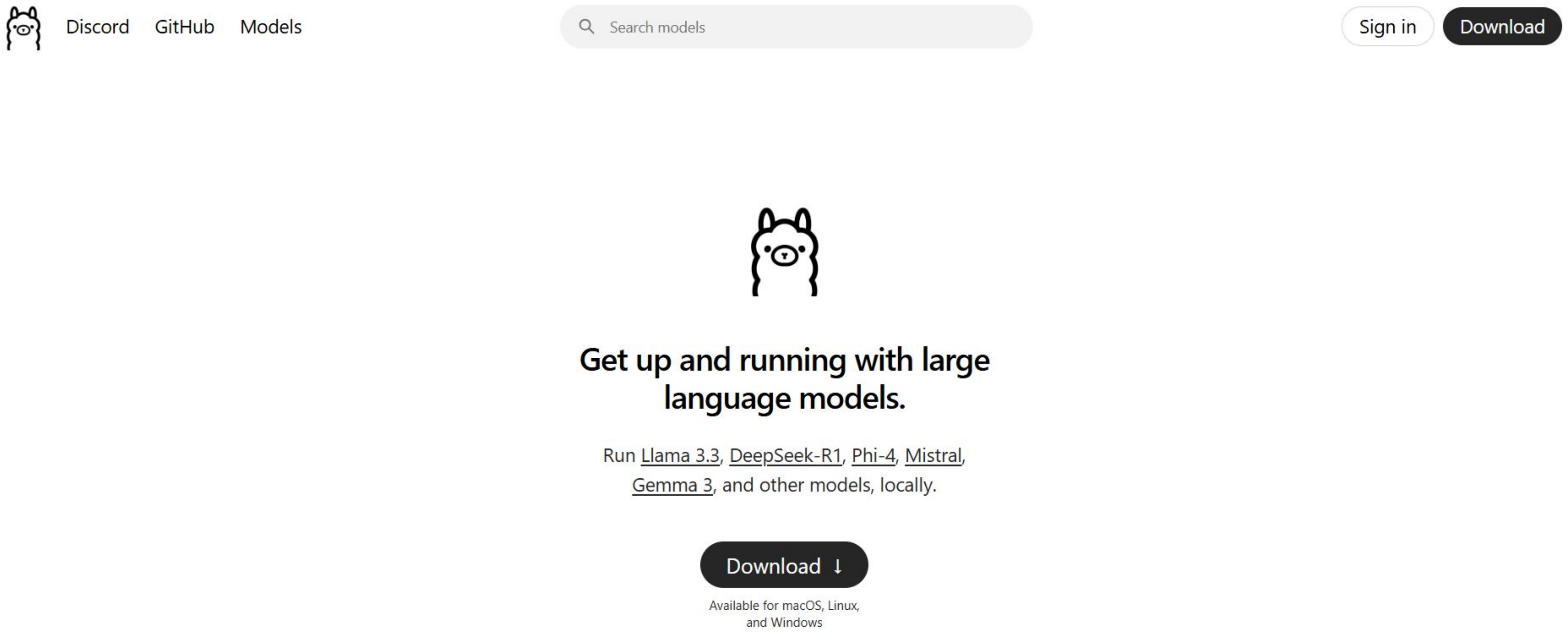


A complete set of interoperable building blocks

Build end-to-end applications with an extensive library of components. Want to change your model? Future-proof your application by incorporating vendor optionality into your LLM infrastructure design.

Popular Python Libraries

Ollama is a platform designed to simplify running and deploying large language models locally. It focuses on providing a streamlined experience for developers who want to leverage LLMs without relying solely on cloud-based APIs.



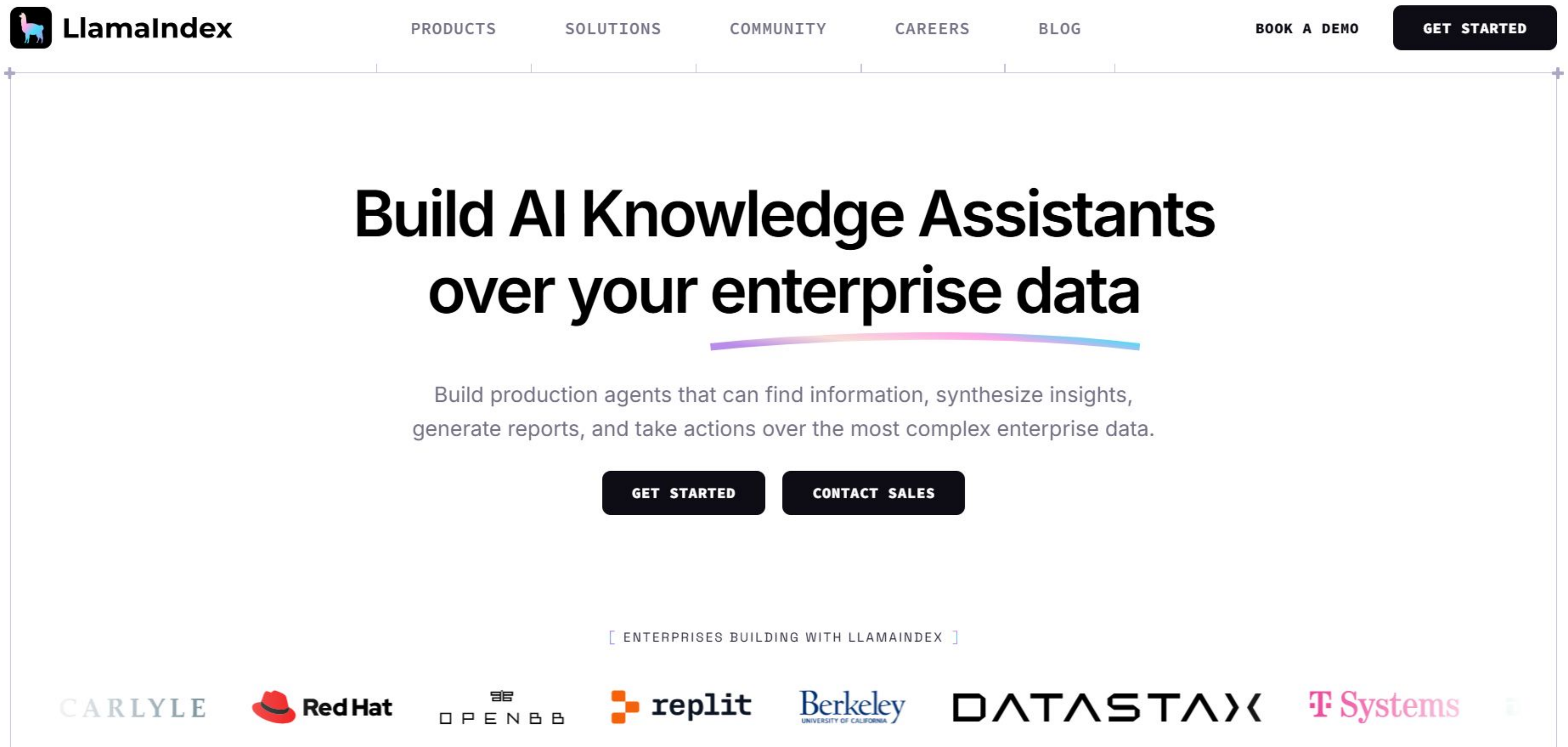
Get up and running with large language models.

Run Llama 3.3, DeepSeek-R1, Phi-4, Mistral, Gemma 3, and other models, locally.

Available for macOS, Linux, and Windows

Popular Python Libraries

LlamaIndex (formerly known as GPT Index) is a data framework that helps connect unstructured data sources (like documents, databases, and APIs) with large language models (LLMs). It enables efficient retrieval and reasoning over private or structured data, making it a powerful tool for Retrieval-Augmented Generation (RAG) applications.

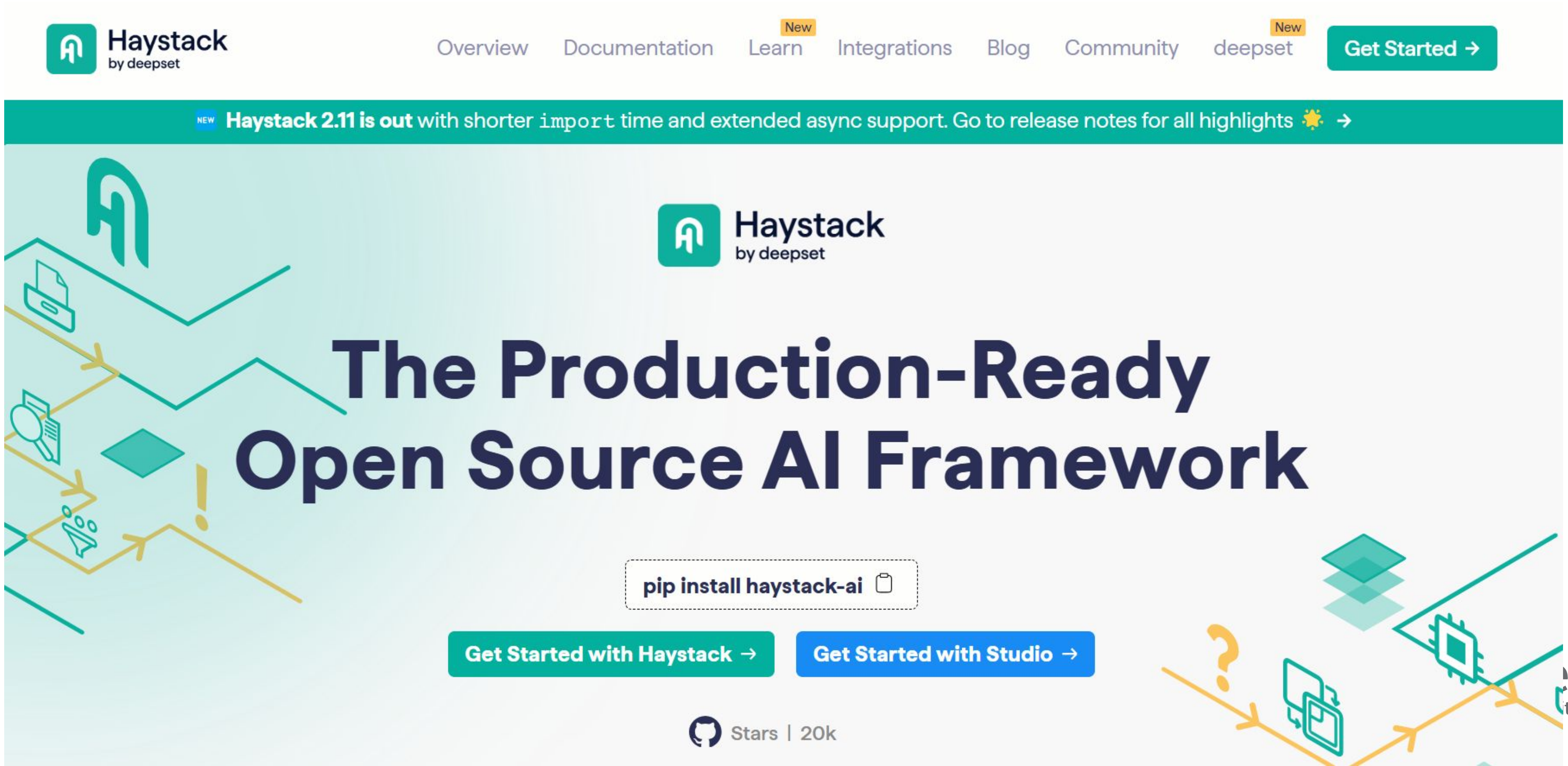


Build AI Knowledge Assistants over your enterprise data

Build production agents that can find information, synthesize insights, generate reports, and take actions over the most complex enterprise data.

Popular Python Libraries

Haystack is an open-source framework focused on building end-to-end search systems and question-answering (QA) applications using natural language processing techniques and language models.



NEW **Haystack 2.11 is out** with shorter `import` time and extended async support. Go to release notes for all highlights →

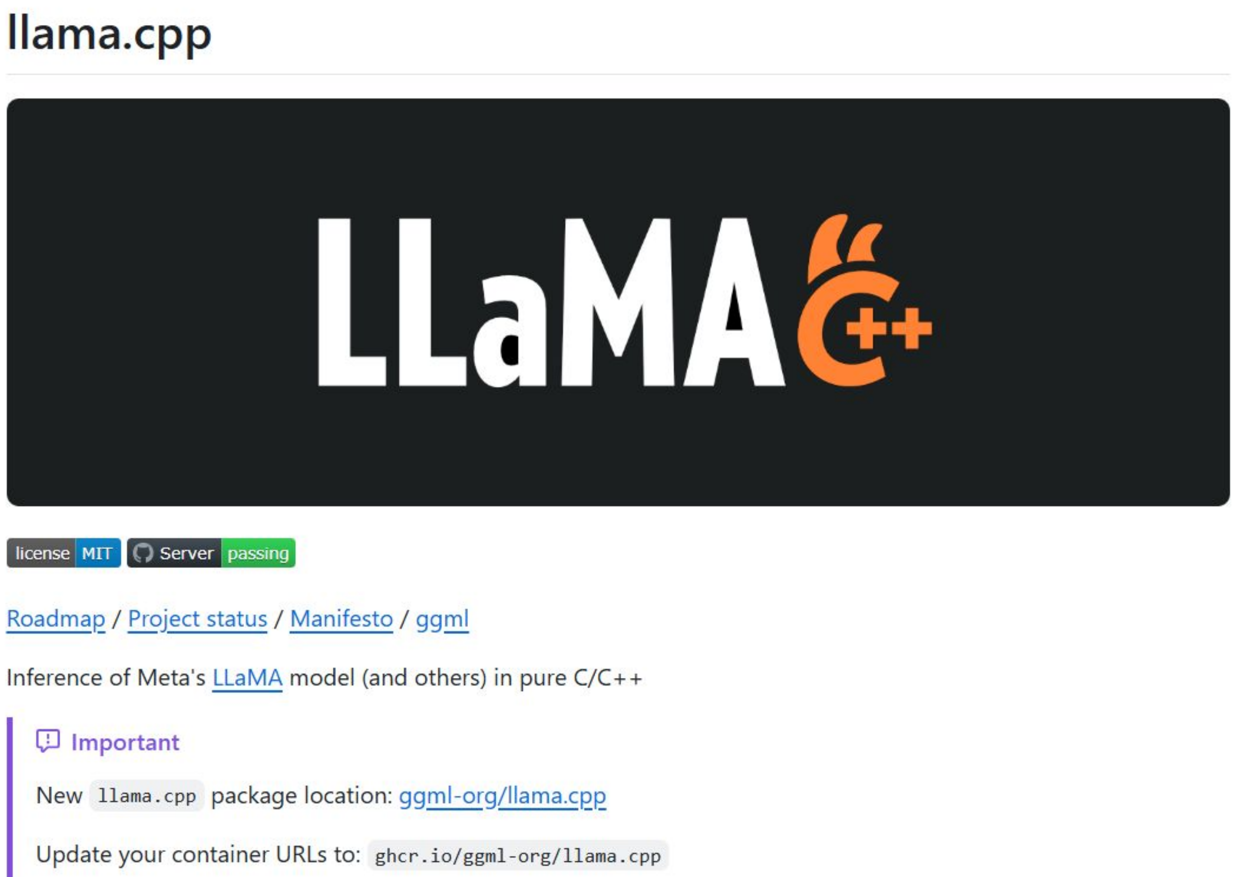
Haystack by deepset

The Production-Ready Open Source AI Framework

```
pip install haystack-ai
```

Popular Python Libraries

Llama.cpp is a lightweight, efficient C++ implementation of Meta's LLaMA (Large Language Model Meta AI) models, designed for running large language models (LLMs) on local devices with minimal dependencies. It enables fast inference on CPUs and GPUs without requiring cloud services or heavy frameworks like PyTorch.



llama.cpp

license MIT | Server passing

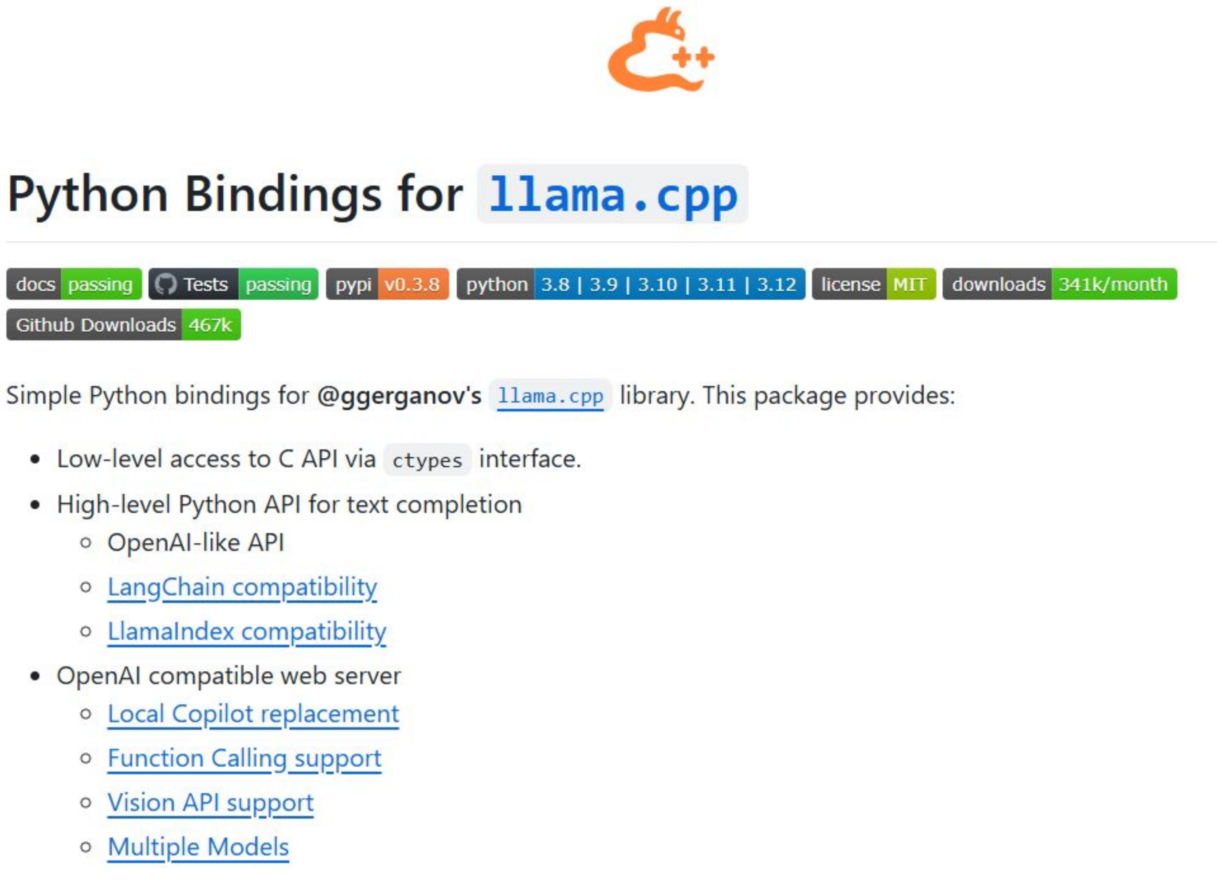
Roadmap / Project status / Manifesto / ggml

Inference of Meta's LLaMA model (and others) in pure C/C++

Important

New `llama.cpp` package location: ggml-org/llama.cpp

Update your container URLs to: `ghcr.io/ggml-org/llama.cpp`



Python Bindings for `llama.cpp`

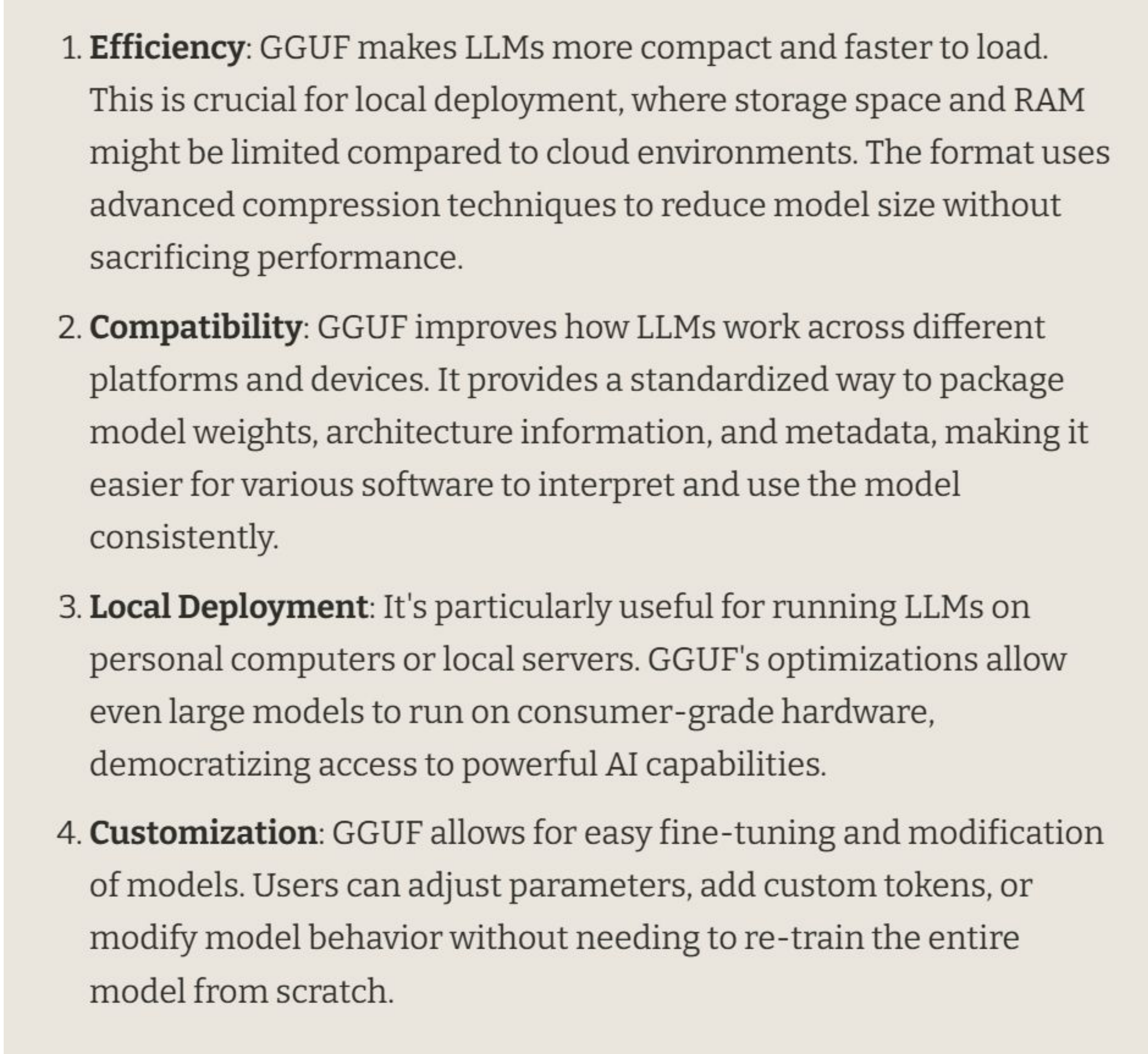
docs passing | Tests passing | pypi v0.3.8 | python 3.8 | 3.9 | 3.10 | 3.11 | 3.12 | license MIT | downloads 341k/month | Github Downloads 467k

Simple Python bindings for **@ggerganov's** `llama.cpp` library. This package provides:

- Low-level access to C API via `ctypes` interface.
- High-level Python API for text completion
  - OpenAI-like API
  - LangChain compatibility
  - LlamaIndex compatibility
- OpenAI compatible web server
  - Local Copilot replacement
  - Function Calling support
  - Vision API support
  - Multiple Models

GGUF

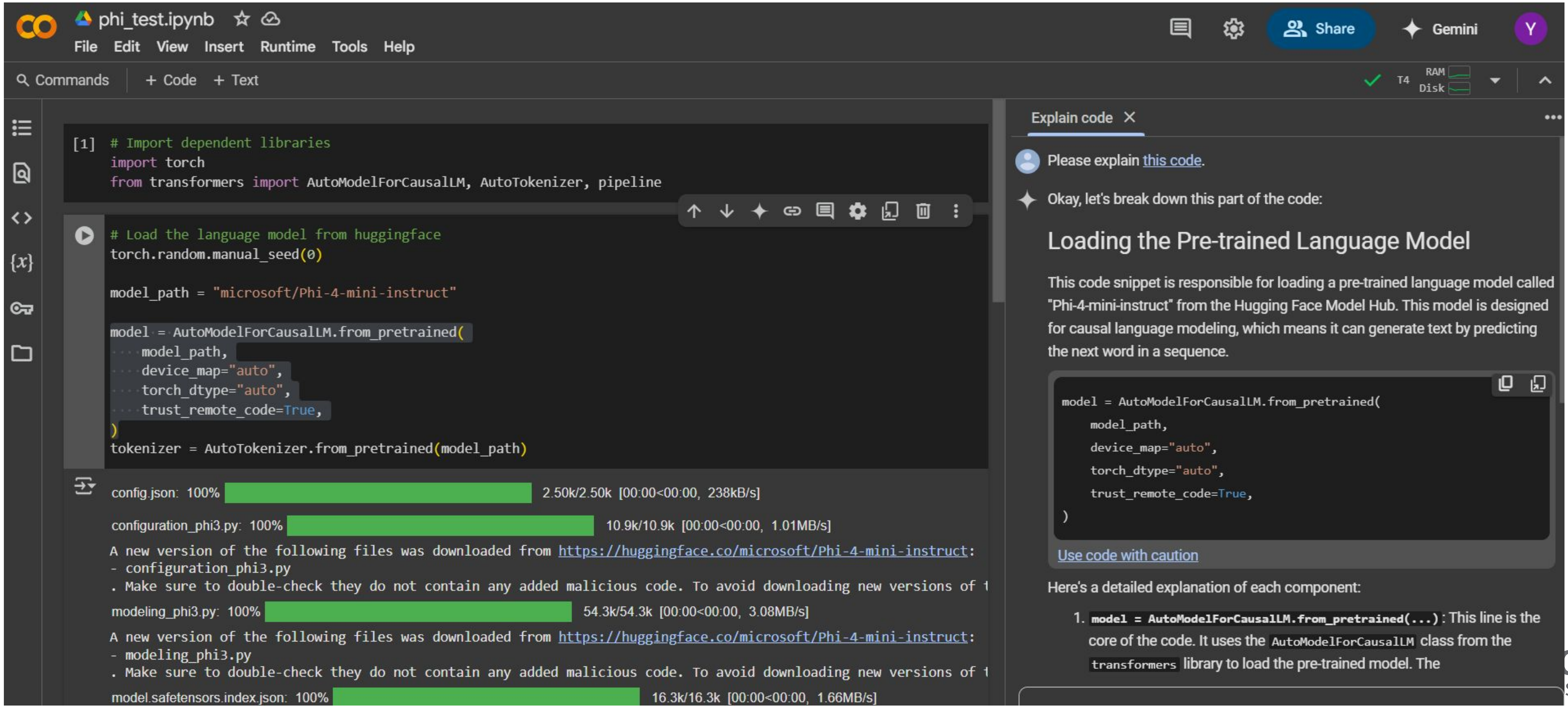
Running LLMs with CPUs



1. **Efficiency**: GGUF makes LLMs more compact and faster to load. This is crucial for local deployment, where storage space and RAM might be limited compared to cloud environments. The format uses advanced compression techniques to reduce model size without sacrificing performance.
2. **Compatibility**: GGUF improves how LLMs work across different platforms and devices. It provides a standardized way to package model weights, architecture information, and metadata, making it easier for various software to interpret and use the model consistently.
3. **Local Deployment**: It's particularly useful for running LLMs on personal computers or local servers. GGUF's optimizations allow even large models to run on consumer-grade hardware, democratizing access to powerful AI capabilities.
4. **Customization**: GGUF allows for easy fine-tuning and modification of models. Users can adjust parameters, add custom tokens, or modify model behavior without needing to re-train the entire model from scratch.

Hands-on Practice

Transformers and Colab AI Features



```
[1] # Import dependent libraries
    import torch
    from transformers import AutoModelForCausalLM, AutoTokenizer, pipeline
```

```
# Load the language model from huggingface
torch.random.manual_seed(0)

model_path = "microsoft/Phi-4-mini-instruct"

model = AutoModelForCausalLM.from_pretrained(
    model_path,
    device_map="auto",
    torch_dtype="auto",
    trust_remote_code=True,
)
tokenizer = AutoTokenizer.from_pretrained(model_path)
```

```
config.json: 100%  2.50k/2.50k [00:00<00:00, 238kB/s]
configuration_phi3.py: 100%  10.9k/10.9k [00:00<00:00, 1.01MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Phi-4-mini-instruct:
- configuration_phi3.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of t
modeling_phi3.py: 100%  54.3k/54.3k [00:00<00:00, 3.08MB/s]
A new version of the following files was downloaded from https://huggingface.co/microsoft/Phi-4-mini-instruct:
- modeling_phi3.py
. Make sure to double-check they do not contain any added malicious code. To avoid downloading new versions of t
model.safetensors.index.json: 100%  16.3k/16.3k [00:00<00:00, 1.66MB/s]
```

Please explain this code.

Okay, let's break down this part of the code:

Loading the Pre-trained Language Model

This code snippet is responsible for loading a pre-trained language model called "Phi-4-mini-instruct" from the Hugging Face Model Hub. This model is designed for causal language modeling, which means it can generate text by predicting the next word in a sequence.

```
model = AutoModelForCausalLM.from_pretrained(
    model_path,
    device_map="auto",
    torch_dtype="auto",
    trust_remote_code=True,
)
```

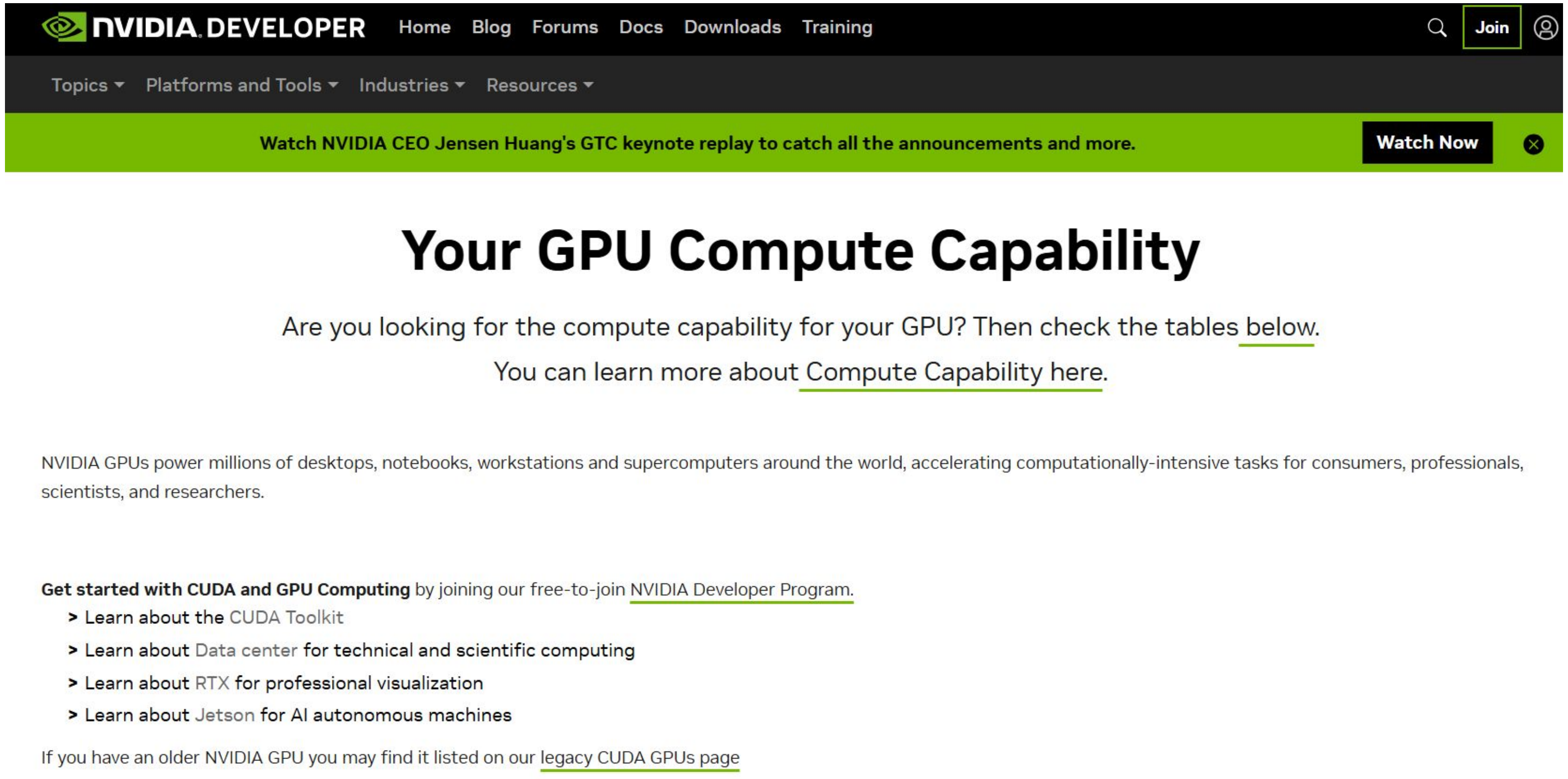
Use code with caution

Here's a detailed explanation of each component:

1. `model = AutoModelForCausalLM.from_pretrained(...)`: This line is the core of the code. It uses the `AutoModelForCausalLM` class from the `transformers` library to load the pre-trained model. The

Python with LLMs on Local Computer

Nvidia GPU



Watch NVIDIA CEO Jensen Huang's GTC keynote replay to catch all the announcements and more.

Your GPU Compute Capability

Are you looking for the compute capability for your GPU? Then check the tables below.

You can learn more about Compute Capability here.

NVIDIA GPUs power millions of desktops, notebooks, workstations and supercomputers around the world, accelerating computationally-intensive tasks for consumers, professionals, scientists, and researchers.

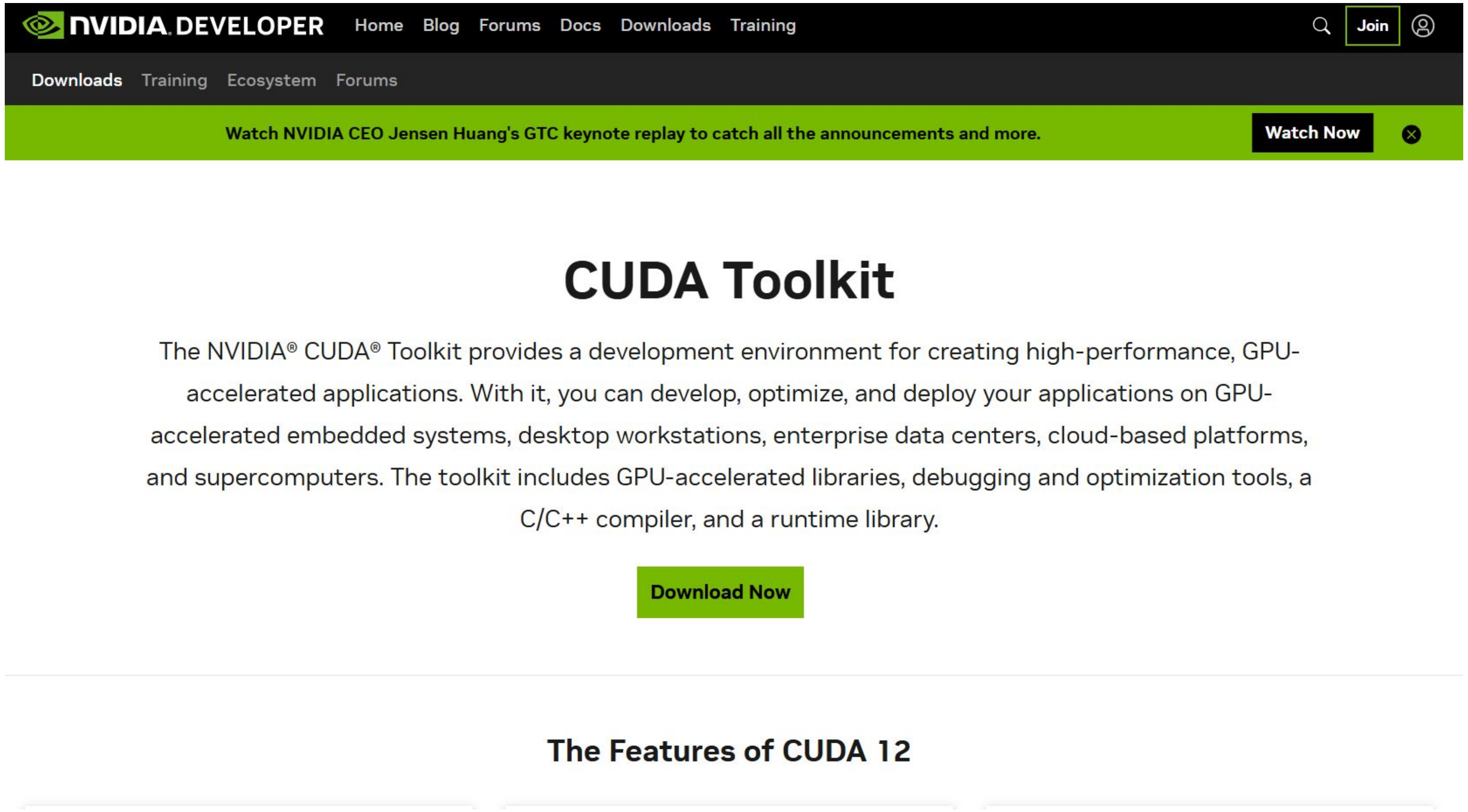
Get started with CUDA and GPU Computing by joining our free-to-join NVIDIA Developer Program.

- > Learn about the CUDA Toolkit
- > Learn about Data center for technical and scientific computing
- > Learn about RTX for professional visualization
- > Learn about Jetson for AI autonomous machines

If you have an older NVIDIA GPU you may find it listed on our legacy CUDA GPUs page

Python with LLMs on Local Computer

Install CUDA Toolkit


NVIDIA DEVELOPER
Home Blog Forums Docs Downloads Training
Join
Downloads Training Ecosystem Forums
Watch Now

Watch NVIDIA CEO Jensen Huang's GTC keynote replay to catch all the announcements and more.

CUDA Toolkit

The NVIDIA® CUDA® Toolkit provides a development environment for creating high-performance, GPU-accelerated applications. With it, you can develop, optimize, and deploy your applications on GPU-accelerated embedded systems, desktop workstations, enterprise data centers, cloud-based platforms, and supercomputers. The toolkit includes GPU-accelerated libraries, debugging and optimization tools, a C/C++ compiler, and a runtime library.

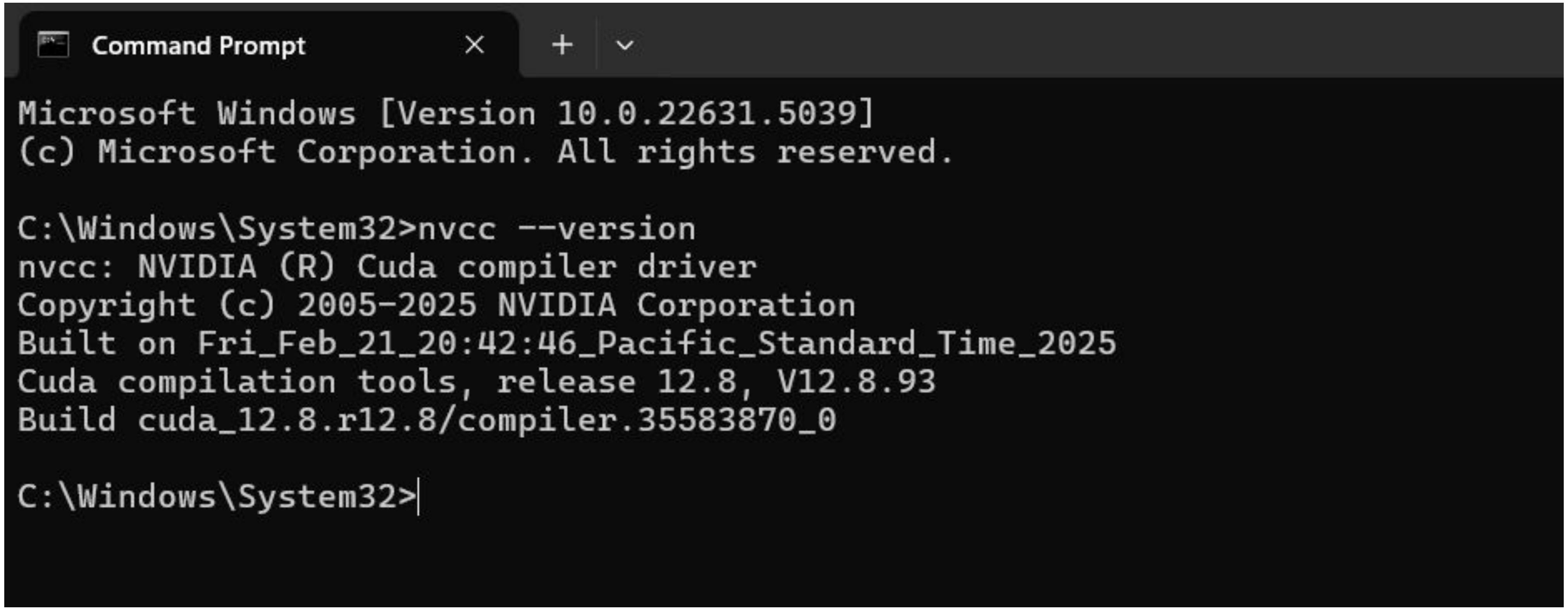
Download Now

The Features of CUDA 12

Metropolia University of Applied Sciences

Python with LLMs on Local Computer

Check CUDA version



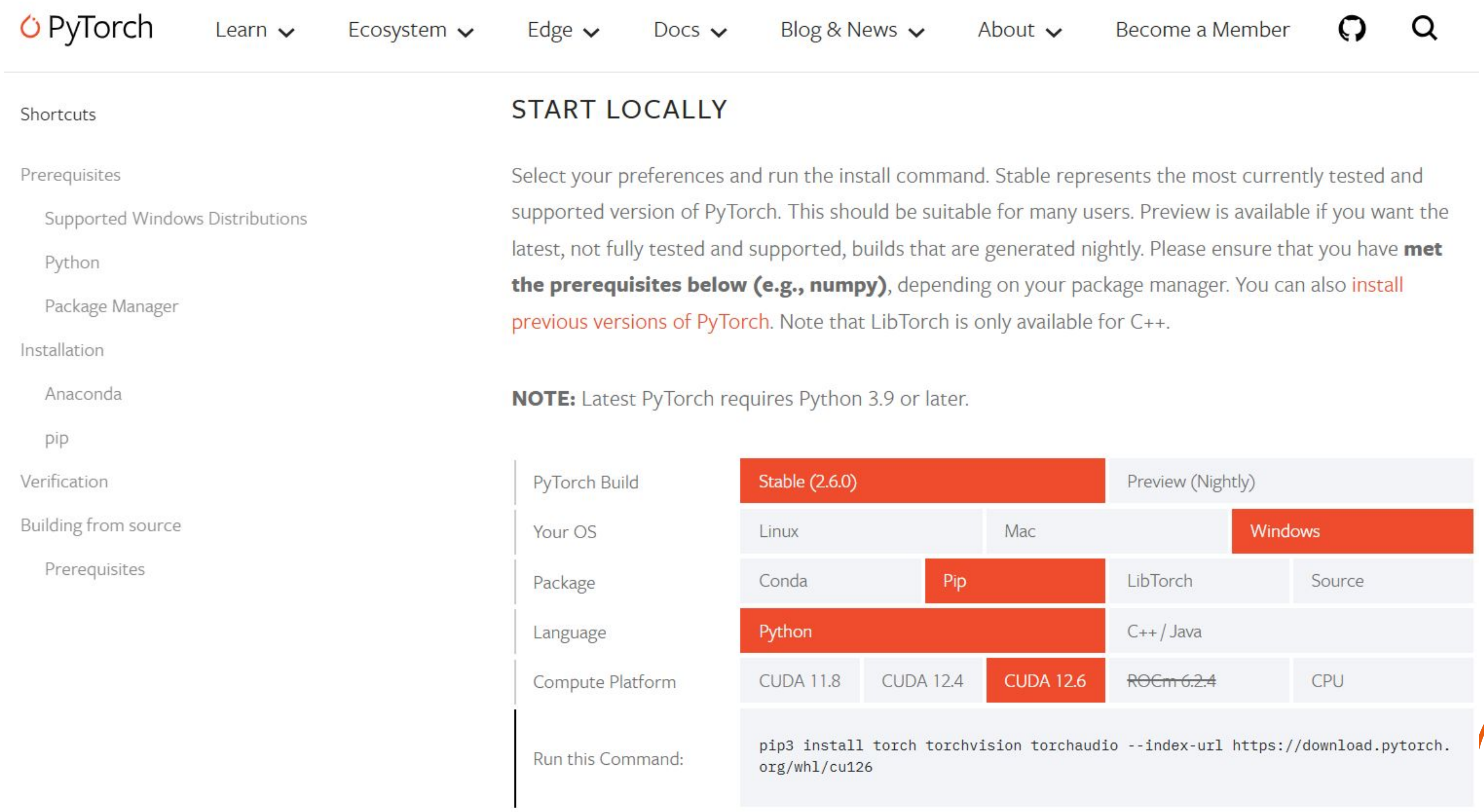
```
Microsoft Windows [Version 10.0.22631.5039]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Fri_Feb_21_20:42:46_Pacific_Standard_Time_2025
Cuda compilation tools, release 12.8, V12.8.93
Build cuda_12.8.r12.8/compiler.35583870_0

C:\Windows\System32>
```

Python with LLMs on Local Computer

Install pytorch according to your system and CUDA version



Shortcuts

- Prerequisites
  - Supported Windows Distributions
  - Python
  - Package Manager
- Installation
  - Anaconda
  - pip
- Verification
- Building from source
  - Prerequisites

START LOCALLY

Select your preferences and run the install command. Stable represents the most currently tested and supported version of PyTorch. This should be suitable for many users. Preview is available if you want the latest, not fully tested and supported, builds that are generated nightly. Please ensure that you have **met the prerequisites below (e.g., numpy)**, depending on your package manager. You can also install previous versions of PyTorch. Note that LibTorch is only available for C++.

NOTE: Latest PyTorch requires Python 3.9 or later.

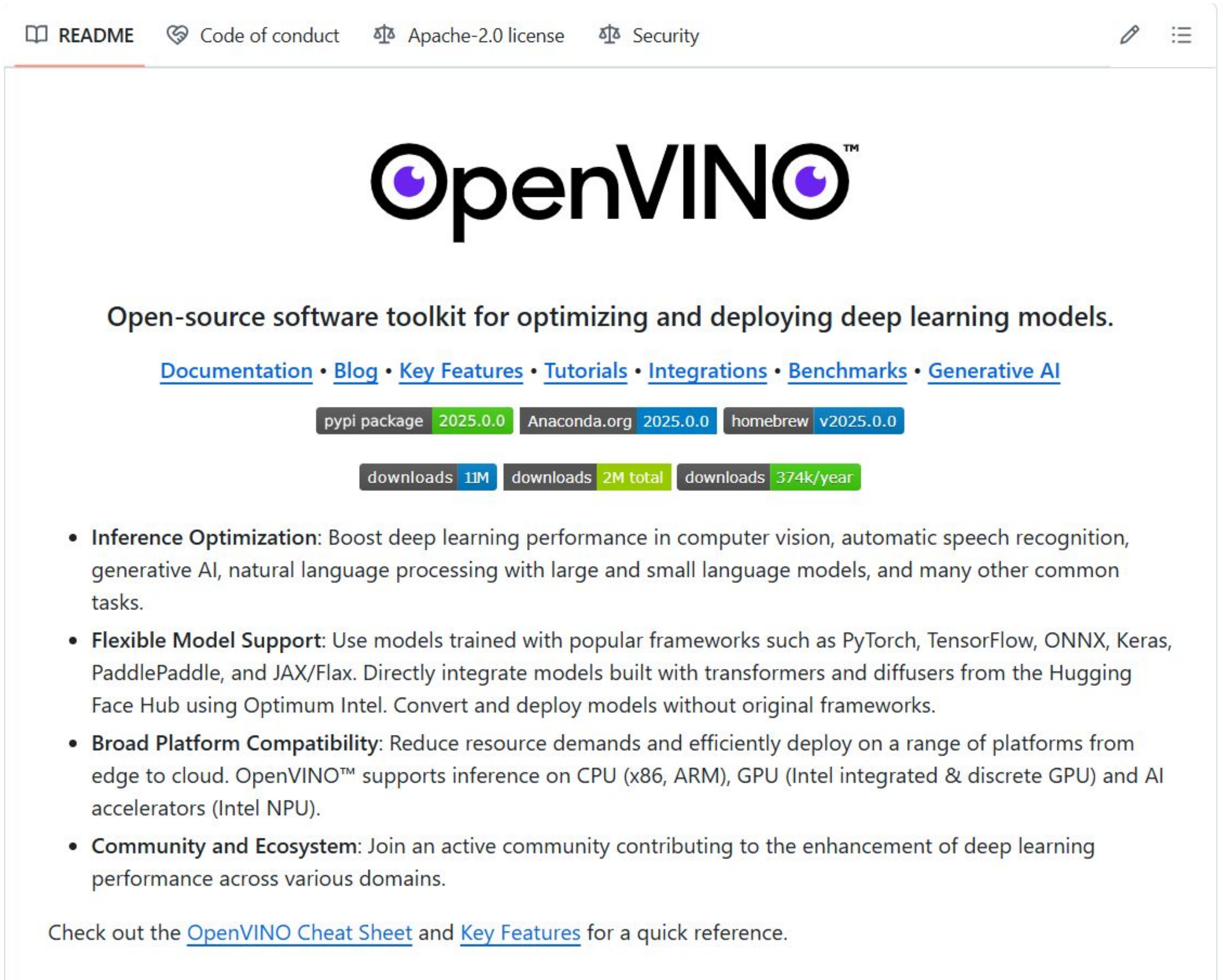
PyTorch Build	[✓ Stable (2.6.0)]	Preview (Nightly)			
Your OS	Linux	Mac	[✓ Windows]		
Package	Conda	[✓ Pip]	LibTorch	Source	
Language	[✓ Python]	C++ / Java			
Compute Platform	CUDA 11.8	CUDA 12.4	[✓ CUDA 12.6]	~~ROCm 6.2.4~~	CPU

Run this Command:

```
pip3 install torch torchvision torchaudio --index-url https://download.pytorch.
org/whl/cu126
```

Python with LLMs on Local Computer

OpenVINO for CPU, other GPUs or NPU



Open-source software toolkit for optimizing and deploying deep learning models.

Documentation • Blog • Key Features • Tutorials • Integrations • Benchmarks • Generative AI

pypi package 2025.0.0 | Anaconda.org 2025.0.0 | homebrew v2025.0.0

downloads 11M | downloads 2M total | downloads 374k/year

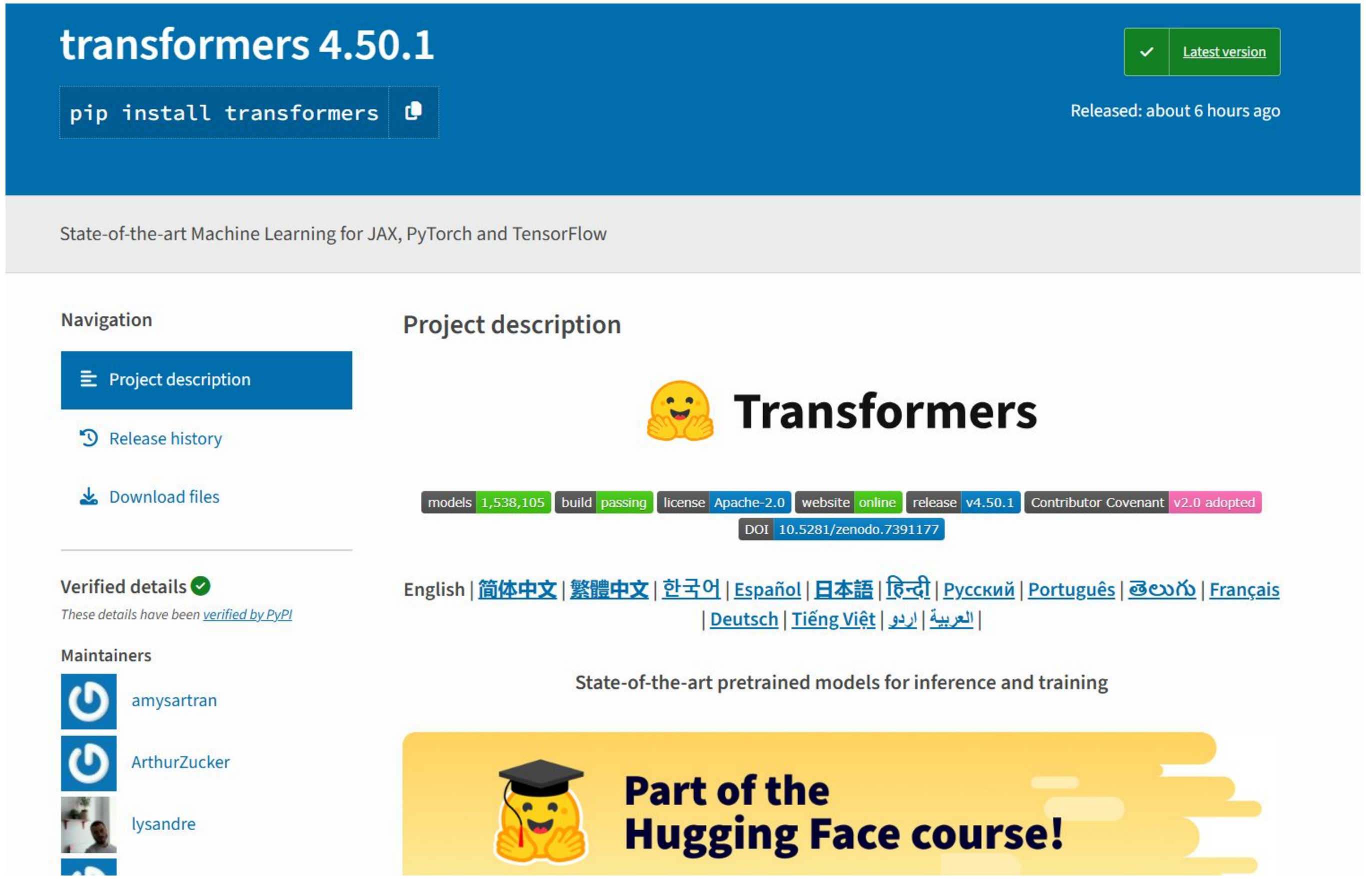
- **Inference Optimization**: Boost deep learning performance in computer vision, automatic speech recognition, generative AI, natural language processing with large and small language models, and many other common tasks.
- **Flexible Model Support**: Use models trained with popular frameworks such as PyTorch, TensorFlow, ONNX, Keras, PaddlePaddle, and JAX/Flax. Directly integrate models built with transformers and diffusers from the Hugging Face Hub using Optimum Intel. Convert and deploy models without original frameworks.
- **Broad Platform Compatibility**: Reduce resource demands and efficiently deploy on a range of platforms from edge to cloud. OpenVINO™ supports inference on CPU (x86, ARM), GPU (Intel integrated & discrete GPU) and AI accelerators (Intel NPU).
- **Community and Ecosystem**: Join an active community contributing to the enhancement of deep learning performance across various domains.

Check out the OpenVINO Cheat Sheet and Key Features for a quick reference.

Metropolia
University of Applied Sciences

Python with LLMs on Local Computer

Install Transformers



transformers 4.50.1

```
pip install transformers
```

Latest version

Released: about 6 hours ago

State-of-the-art Machine Learning for JAX, PyTorch and TensorFlow

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Maintainers

amysartran

ArthurZucker

lysandre

Project description

Transformers

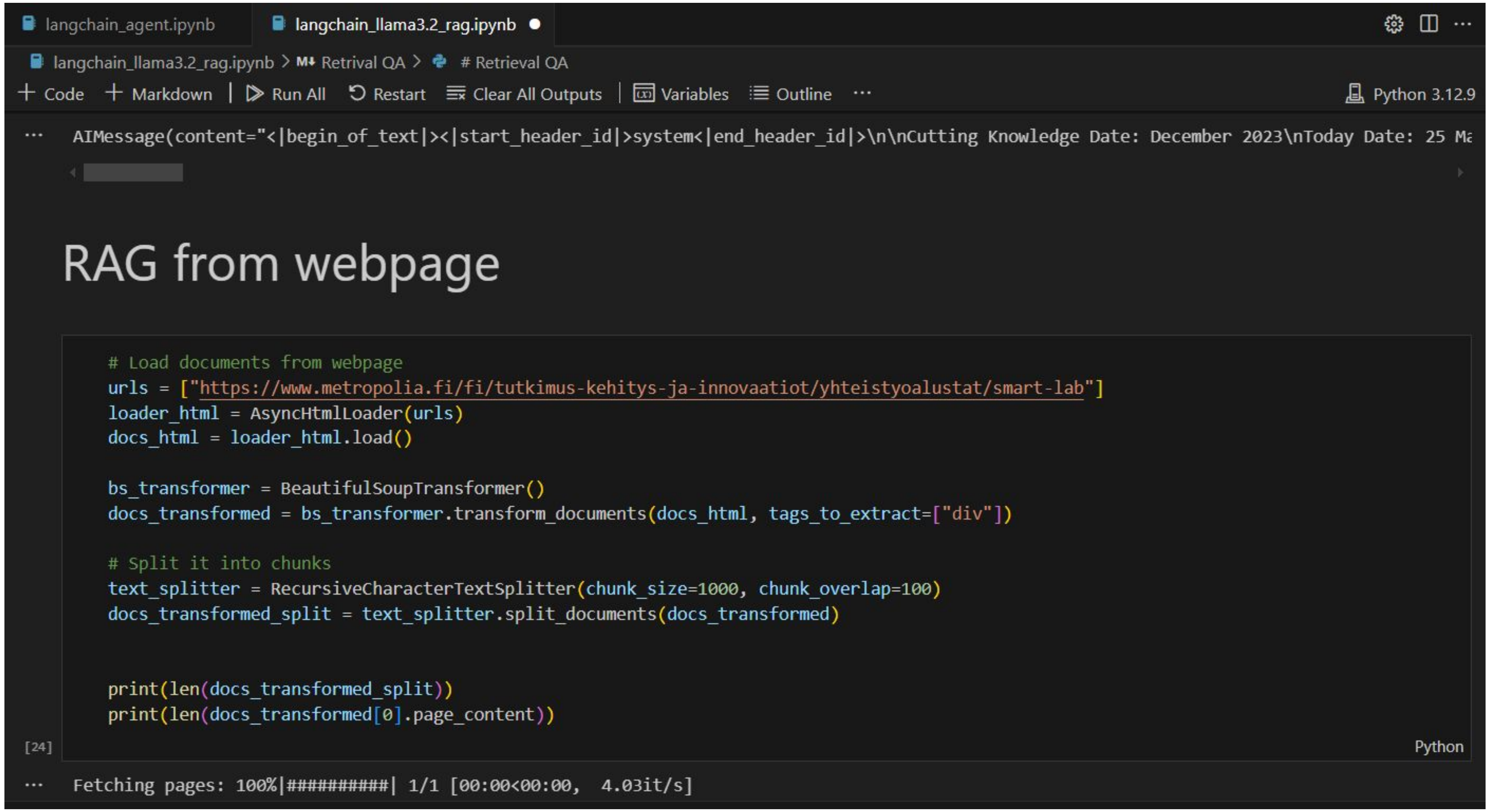
models 1,538,105 | build passing | license Apache-2.0 | website online | release v4.50.1 | Contributor Covenant v2.0 adopted

DOI 10.5281/zenodo.7391177

English | 简体中文 | 繁體中文 | 한국어 | Español | 日本語 | हिन्दी | Русский | Português | తెలుగు | Français | Deutsch | Tiếng Việt | اردو | العربية |

State-of-the-art pretrained models for inference and training

RAG with LangChain



```
AIMessage(content="<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n\nCutting Knowledge Date: December 2023\nToday Date: 25 Ma
```

RAG from webpage

```python
# Load documents from webpage
urls = ["https://www.metropolia.fi/fi/tutkimus-kehitys-ja-innovaatiot/yhteistyoalustat/smart-lab"]
loader_html = AsyncHtmlLoader(urls)
docs_html = loader_html.load()

bs_transformer = BeautifulSoupTransformer()
docs_transformed = bs_transformer.transform_documents(docs_html, tags_to_extract=["div"])

# Split it into chunks
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=100)
docs_transformed_split = text_splitter.split_documents(docs_transformed)


print(len(docs_transformed_split))
print(len(docs_transformed[0].page_content))
```

[24] Python

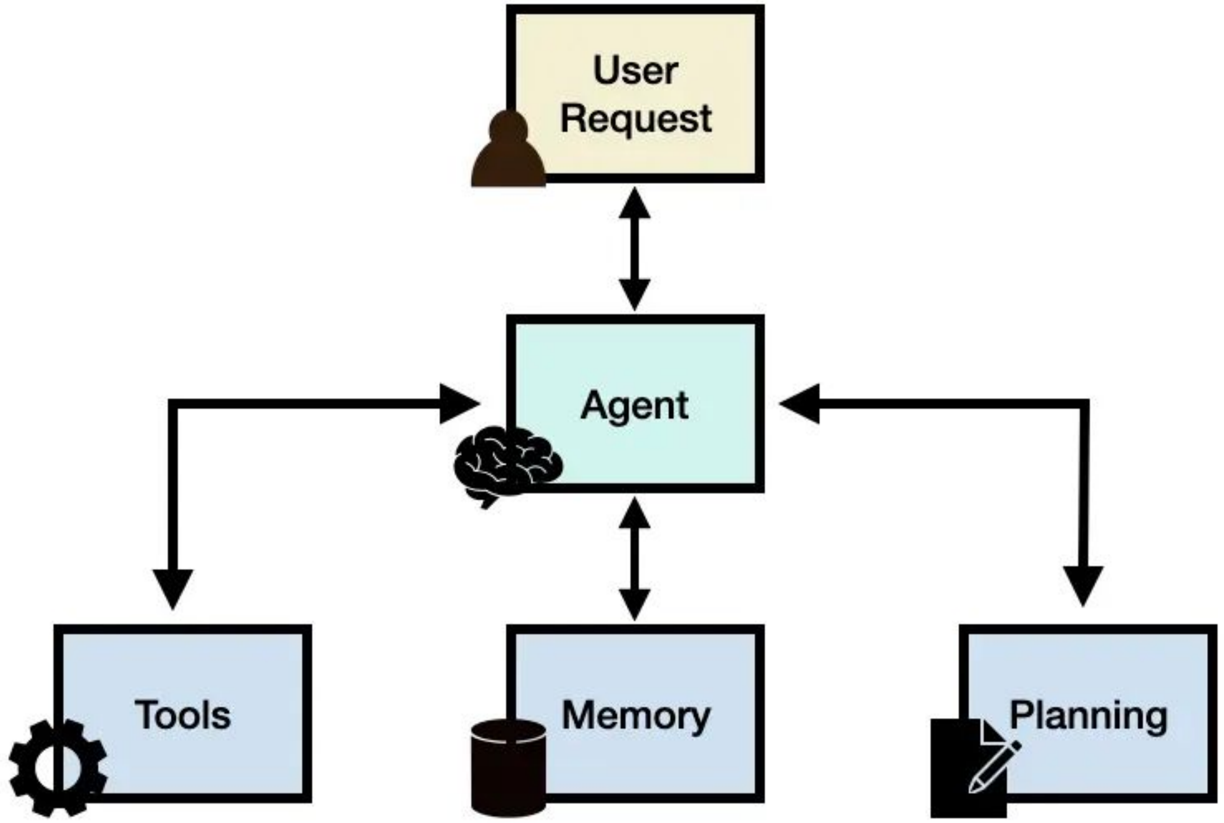
```
Fetching pages: 100%|##########| 1/1 [00:00<00:00,  4.03it/s]
```

AI Ascent
Andrew Ng

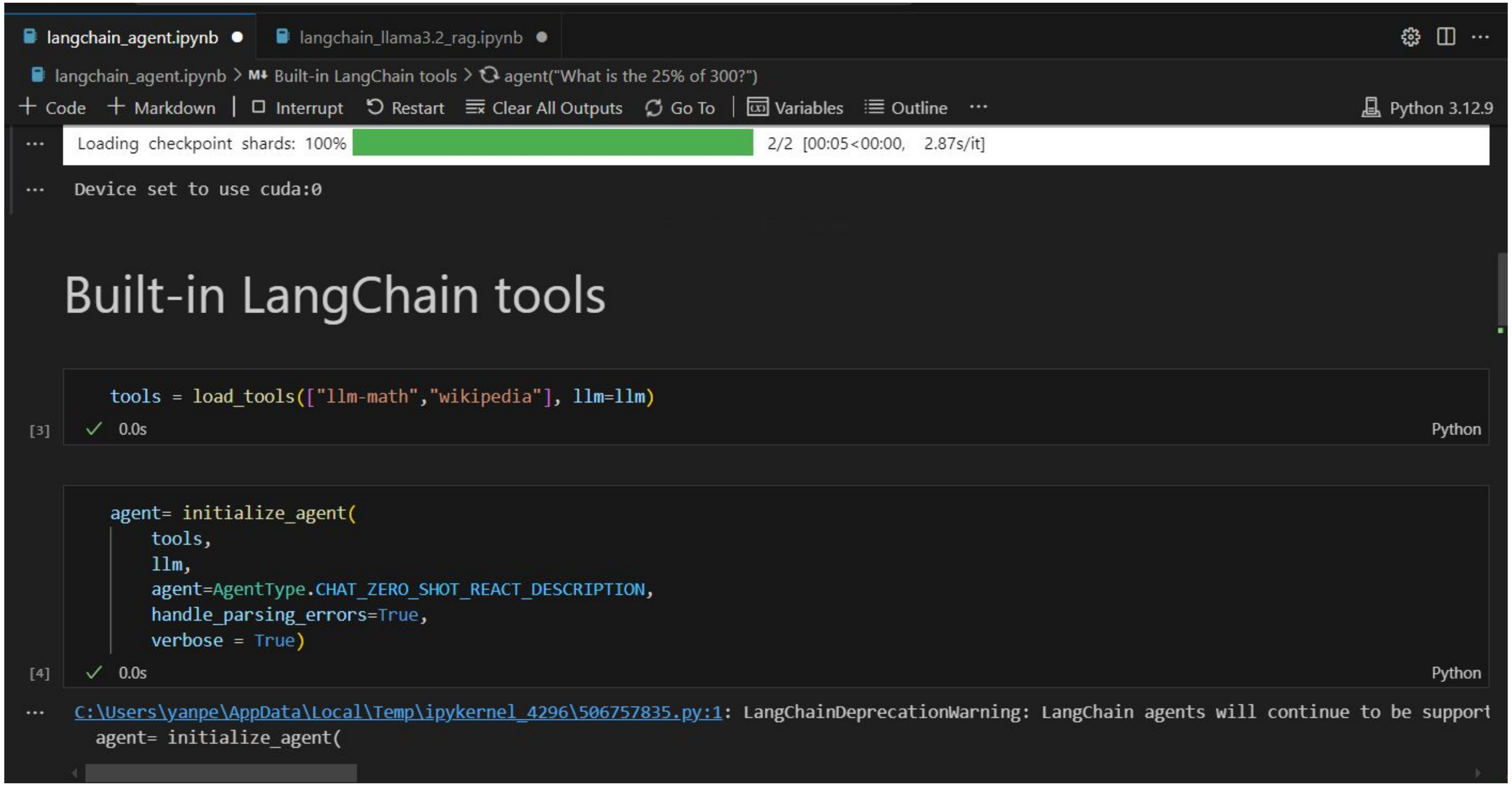
LLM Agent Framework

Generally speaking, an LLM agent framework can consist of the following core components:

- **User Request** - a user question or request
- **Agent/Brain** - the agent core acting as coordinator
- **Planning** - assists the agent in planning future actions
- **Memory** - manages the agent's past behaviors



LangChain Agents


langchain_agent.ipynb
langchain_llama3.2_rag.ipynb
langchain_agent.ipynb > Built-in LangChain tools > agent("What is the 25% of 300?")
Code
Markdown
Interrupt
Restart
Clear All Outputs
Go To
Variables
Outline
Python 3.12.9

```
Loading checkpoint shards: 100% 2/2 [00:05<00:00,  2.87s/it]
```

```
Device set to use cuda:0
```

Built-in LangChain tools

```python
tools = load_tools(["llm-math","wikipedia"], llm=llm)
```

[3] ✓ 0.0s Python

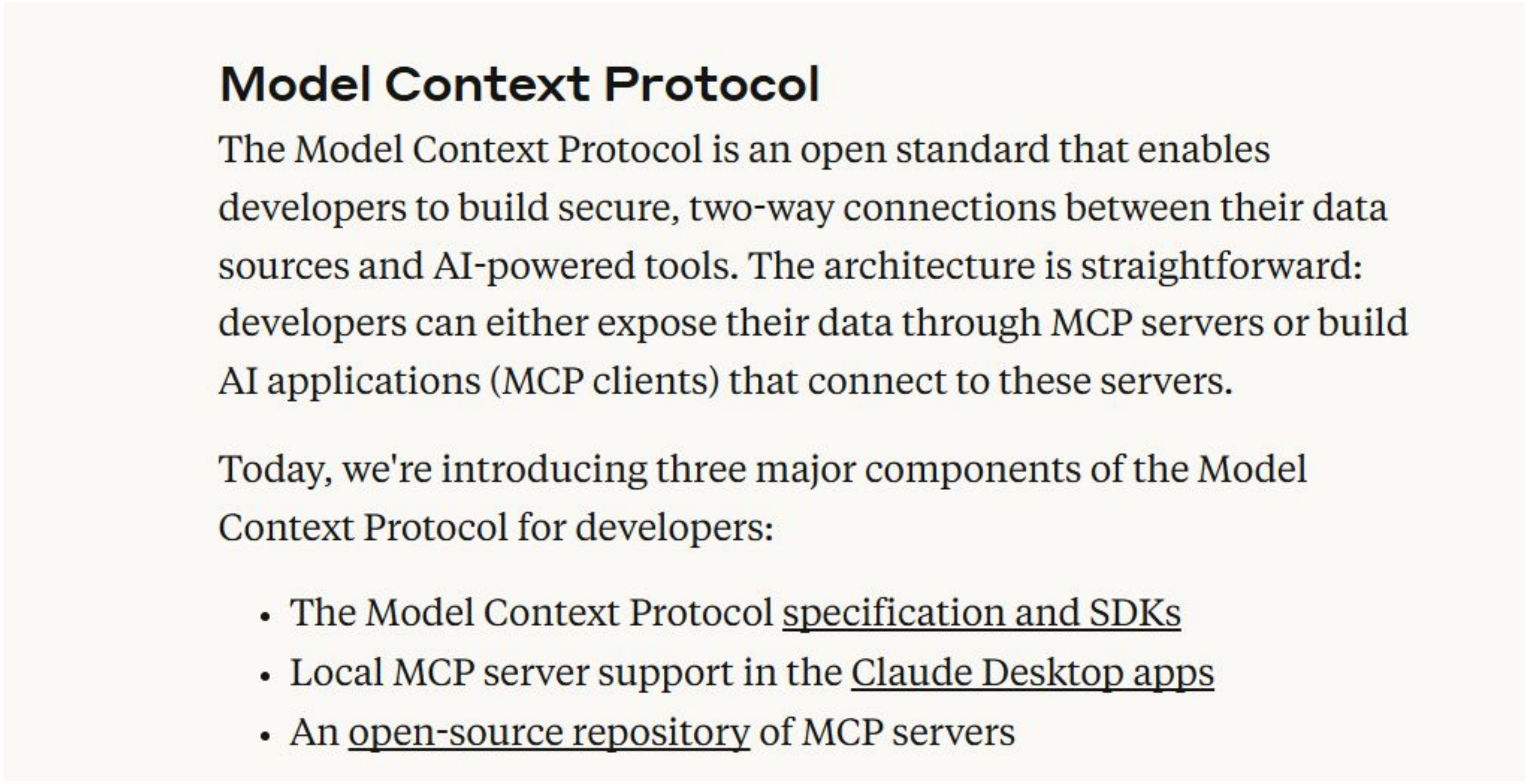
```python
agent= initialize_agent(
    tools,
    llm,
    agent=AgentType.CHAT_ZERO_SHOT_REACT_DESCRIPTION,
    handle_parsing_errors=True,
    verbose = True)
```

[4] ✓ 0.0s Python

```
C:\Users\yanpe\AppData\Local\Temp\ipykernel_4296\506757835.py:1: LangChainDeprecationWarning: LangChain agents will continue to be support
  agent= initialize_agent(
```

Introducing Manus
manus

Model Context Protocol (MCP)



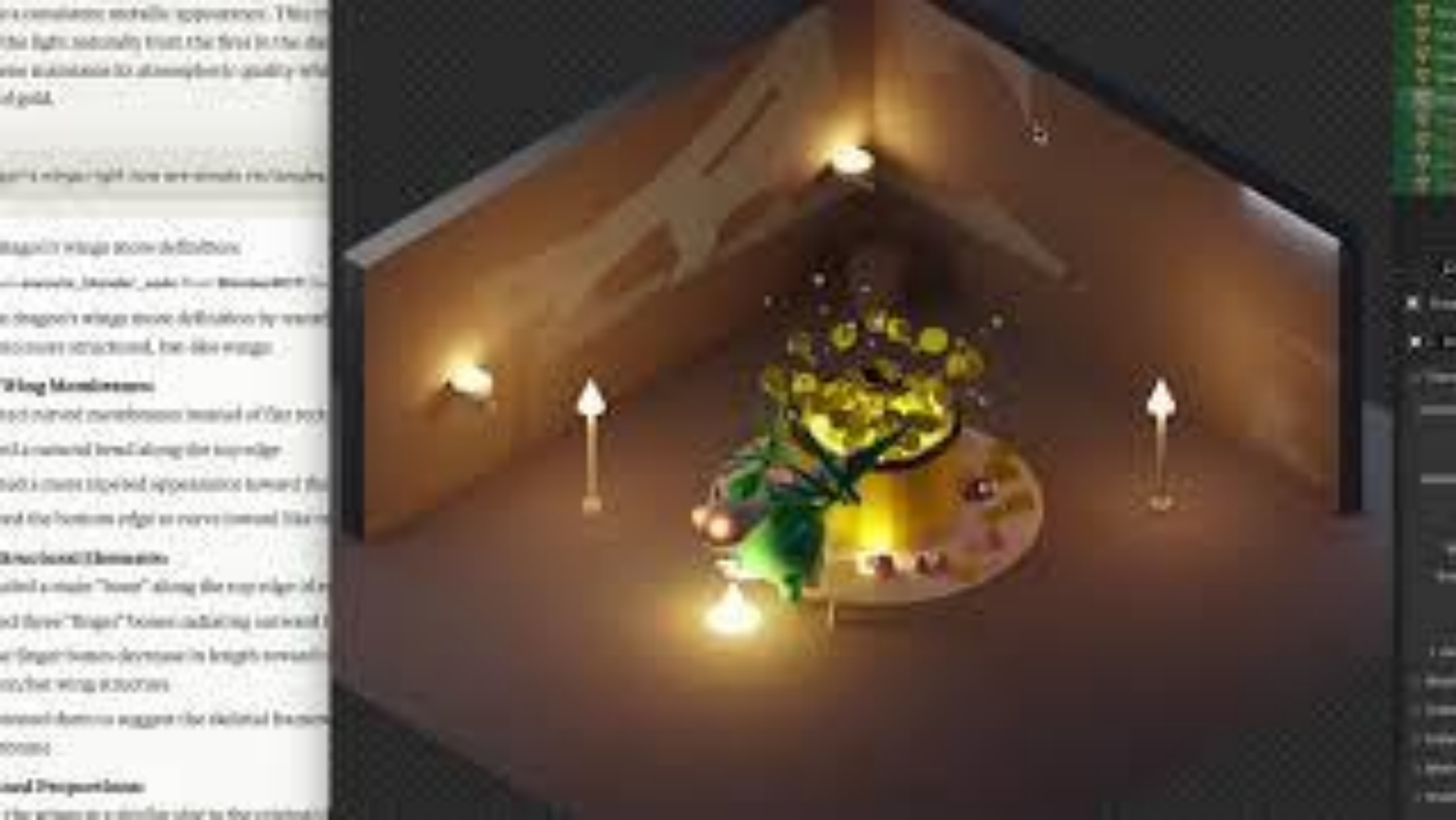
Model Context Protocol

The Model Context Protocol is an open standard that enables developers to build secure, two-way connections between their data sources and AI-powered tools. The architecture is straightforward: developers can either expose their data through MCP servers or build AI applications (MCP clients) that connect to these servers.

Today, we're introducing three major components of the Model Context Protocol for developers:

- The Model Context Protocol specification and SDKs
- Local MCP server support in the Claude Desktop apps
- An open-source repository of MCP servers

How to build a Model Context Protocol
- chat
- IDE
MCP Server
DB
API
Code
</>



a consistent metallic appearance. This m
the light naturally from the first in the d
ene assistance to atmospheric quality wh
of gold.

gon's wings right now are simple rectangles

dragon's wings more definition

execute_blender_code from BlenderMCP

e dragon's wings more definition by creat
e more structured, bat-like wings:

Wing Membranes:

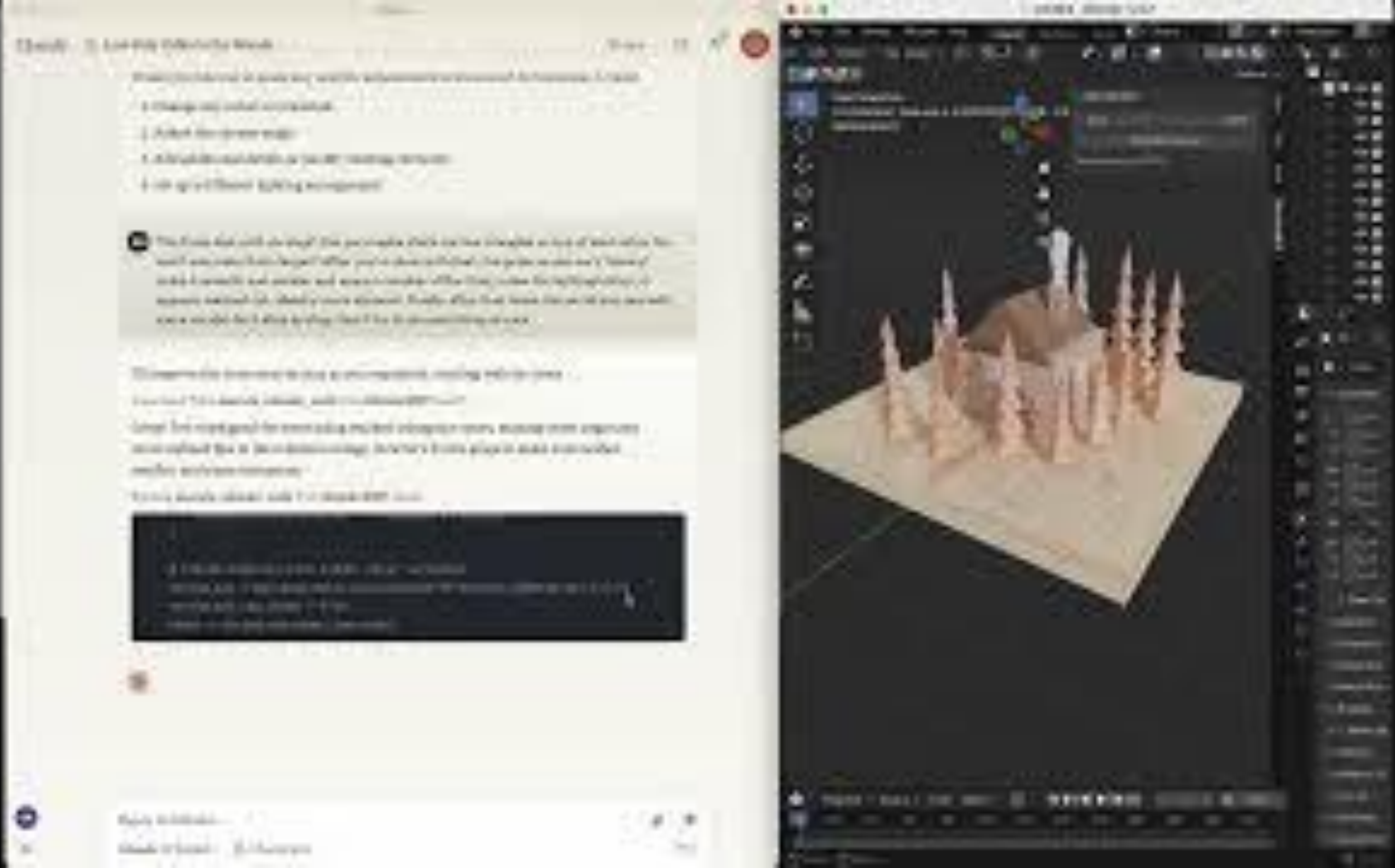
eated curved membranes instead of flat rect
ed a natural bend along the top edge
ade a more tapered appearance toward th
ed the bottom edge to curve inward like

Structural Elements:

dded a main "bone" along the top edge of
ed three "finger" bones radiating outward
e finger bones decrease in length toward
ing bat wing structure
ed elements suggest the skeletal framew
ctions

and Proportions:

[illegible]


Claude

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]

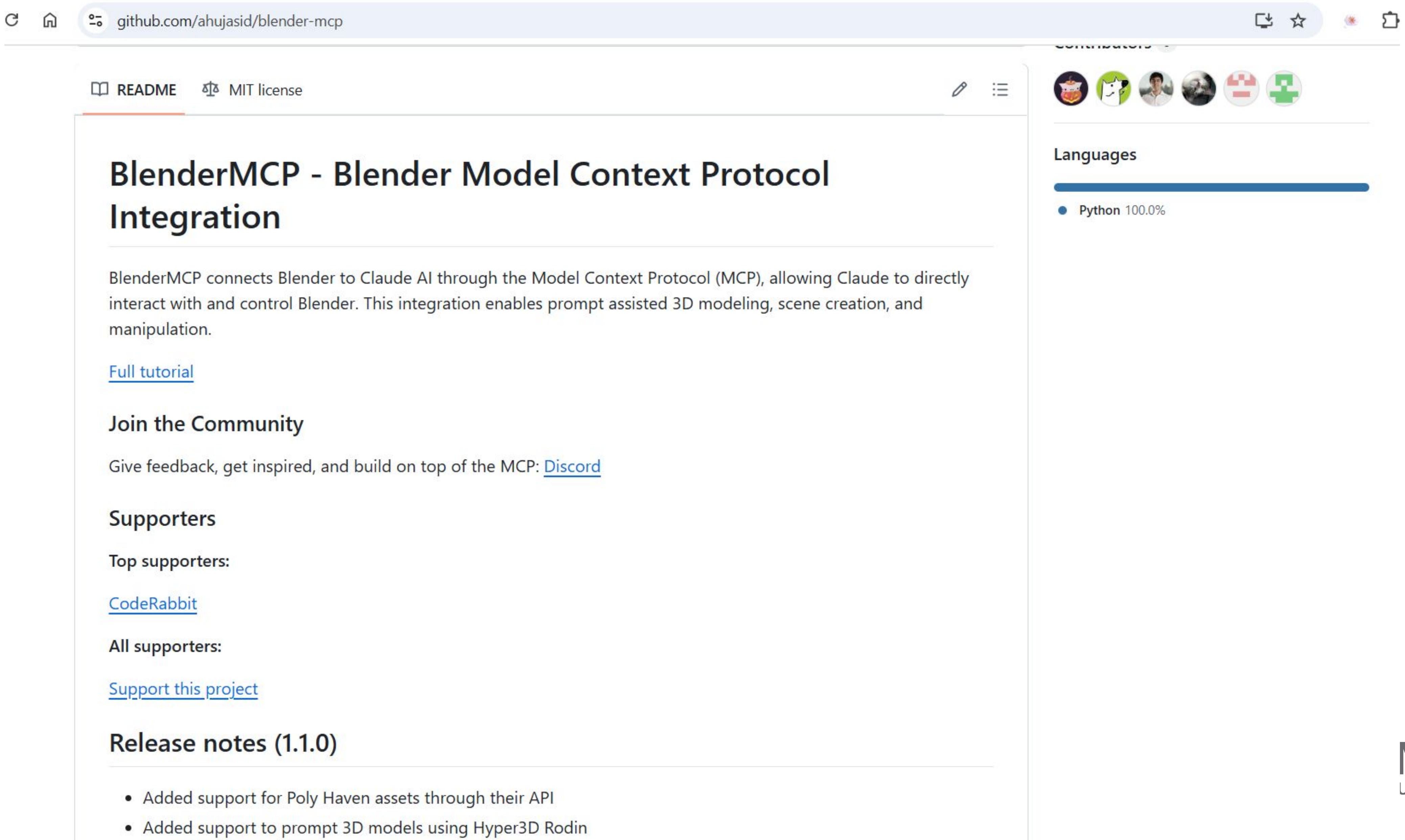
[illegible]

[illegible]

[illegible]

```
[illegible]
```

BlenderMCP



BlenderMCP - Blender Model Context Protocol Integration

BlenderMCP connects Blender to Claude AI through the Model Context Protocol (MCP), allowing Claude to directly interact with and control Blender. This integration enables prompt assisted 3D modeling, scene creation, and manipulation.

Full tutorial

Join the Community

Give feedback, get inspired, and build on top of the MCP: Discord

Supporters

Top supporters:

CodeRabbit

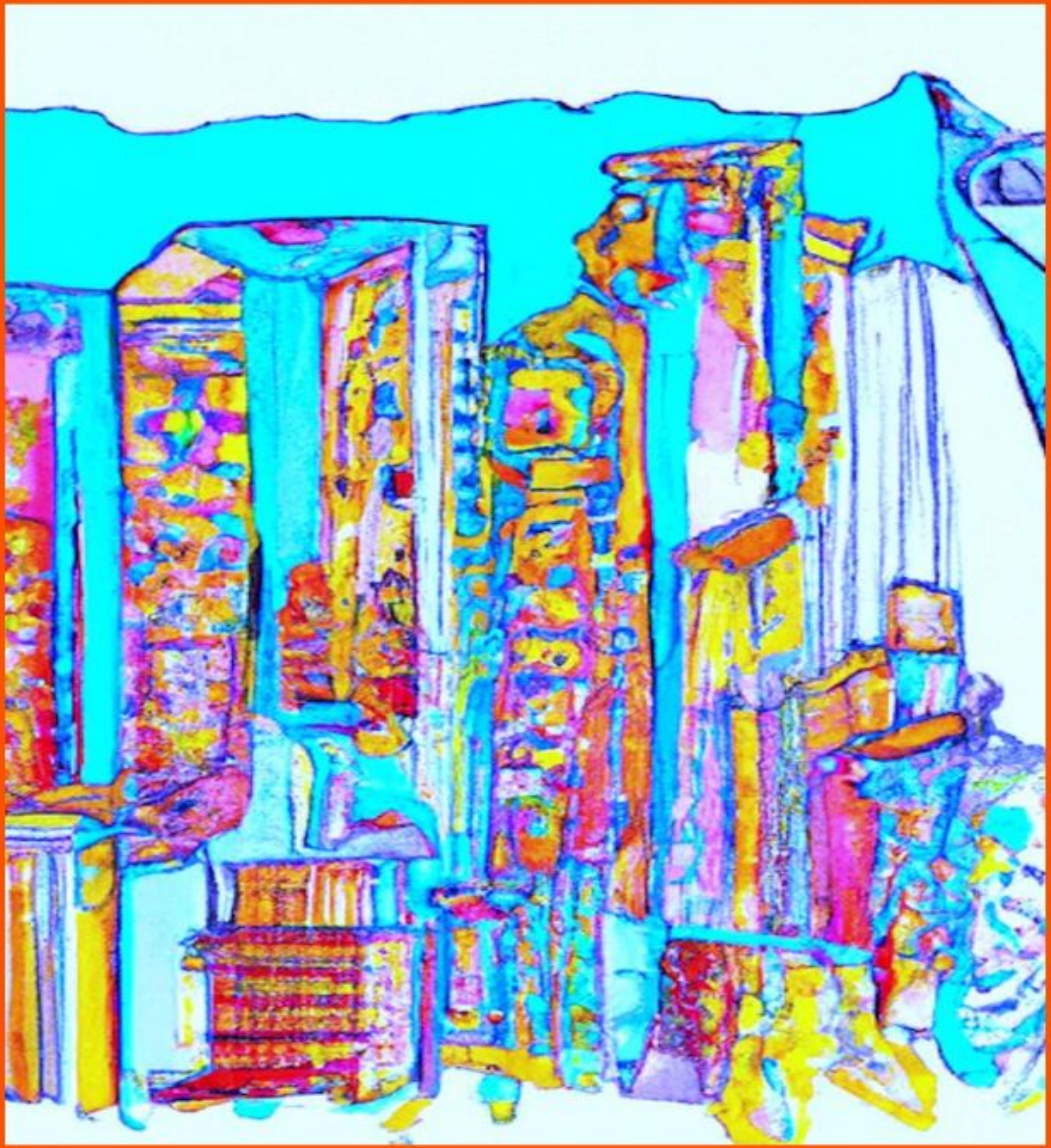
All supporters:

Support this project

Release notes (1.1.0)

- Added support for Poly Haven assets through their API
- Added support to prompt 3D models using Hyper3D Rodin

Metropolia
University of Applied Sciences



Thank you!