Practical Inference

EN.705.743: ChatGPT from Scratch

Outline

- Recap prompting ideas
- LLM inference as a subroutine
- RAG
- Popular Inference Providers + Libraries
- Huggingface Overview
  - Inference API
  - Datasets
  - Transformers

Prompting Recap

Last time we saw that introducing a pattern to the model can induce a specific behavior:

I love the beach / Ich liebe den Strand

What time is it? / Wie spät ist es?

The beer is ten dollars. / Das Bier kostet zehn Dollar.

Translation examples. Clear English / German pattern.

Where can I get a coffee? /

Incomplete pattern continuation for what we want to translate.

Prompting Recap

Last time we saw that introducing a pattern to the model can induce a specific behavior:

I love the beach / Ich liebe den Strand

What time is it? / Wie spät ist es?

The beer is ten dollars. / Das Bier kostet zehn Dollar.

Where can I get a coffee? / Wo kann ich einen Kaffee bekommen?

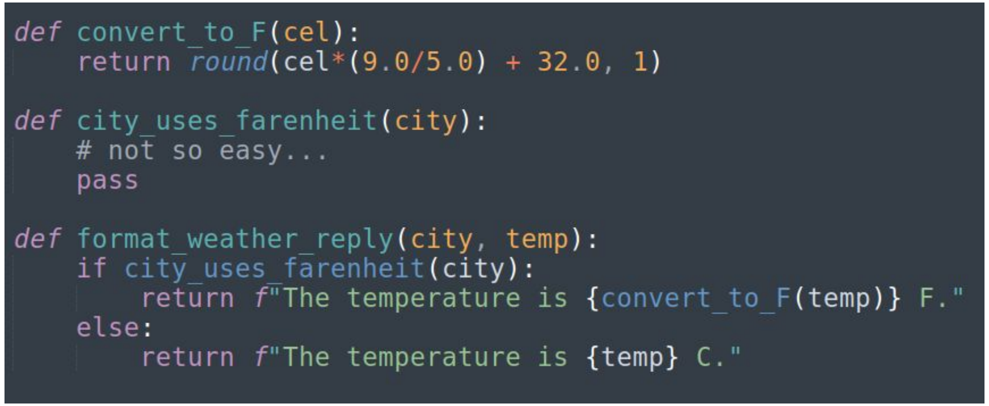
By continuing the text, the model serves as a translator.

Prompting + LLM as Subroutine

Taking this further, we can store a variety of these prompts to call forth a given behavior as needed. These can replace or work alongside traditional software modules.

Prompting + LLM as Subroutine

Taking this further, we can store a variety of these prompts to call forth a given behavior as needed. These can replace or work alongside traditional software modules.



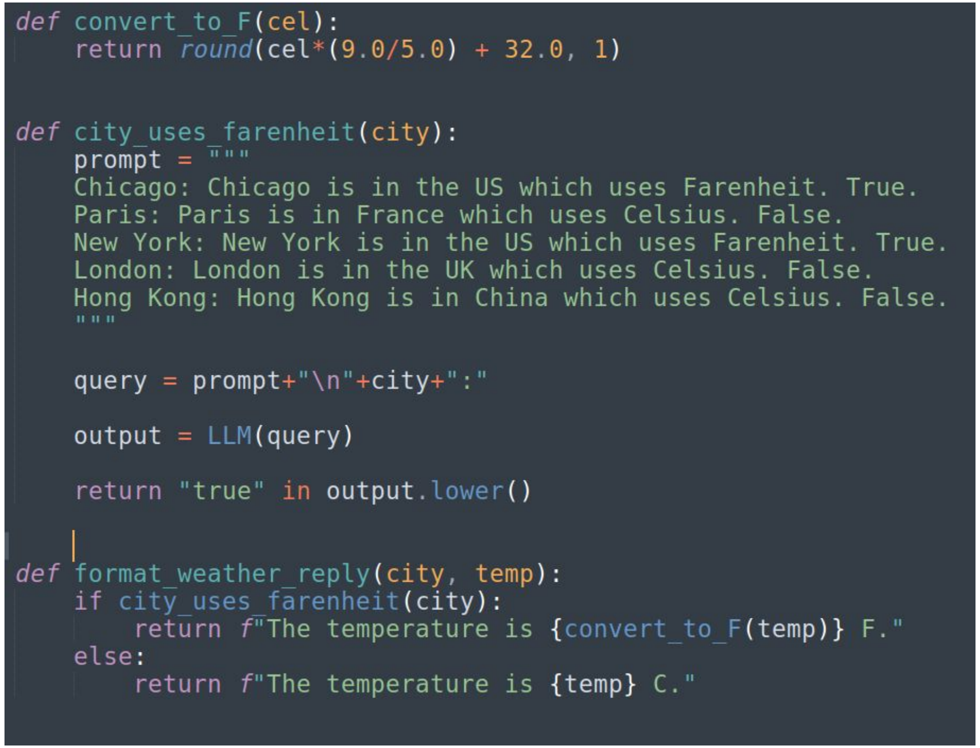
```python
def convert_to_F(cel):
    return round(cel*(9.0/5.0) + 32.0, 1)

def city_uses_farenheit(city):
    # not so easy...
    pass

def format_weather_reply(city, temp):
    if city_uses_farenheit(city):
        return f"The temperature is {convert_to_F(temp)} F."
    else:
        return f"The temperature is {temp} C."
```

Prompting + LLM as Subroutine

Taking this further, we can store a variety of these prompts to call forth a given behavior as needed. These can replace or work alongside traditional software modules.

In the example, we have replaced a typical method with a call to an LLM, and we are trusting that the LLM knows which cities use which system.

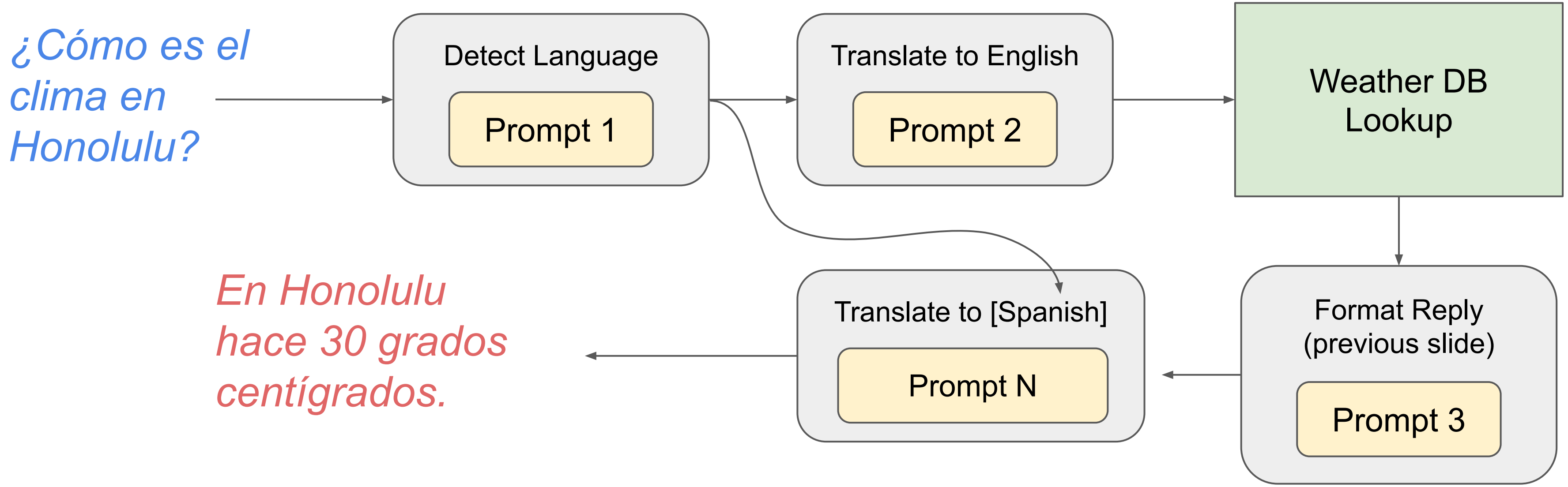
Note that pre/post processing and error handling is important.



```
def convert_to_F(cel):
    return round(cel*(9.0/5.0) + 32.0, 1)


def city_uses_farenheit(city):
    prompt = """
    Chicago: Chicago is in the US which uses Farenheit. True.
    Paris: Paris is in France which uses Celsius. False.
    New York: New York is in the US which uses Farenheit. True.
    London: London is in the UK which uses Celsius. False.
    Hong Kong: Hong Kong is in China which uses Celsius. False.
    """

    query = prompt+"\n"+city+":"

    output = LLM(query)

    return "true" in output.lower()


def format_weather_reply(city, temp):
    if city_uses_farenheit(city):
        return f"The temperature is {convert_to_F(temp)} F."
    else:
        return f"The temperature is {temp} C."
```

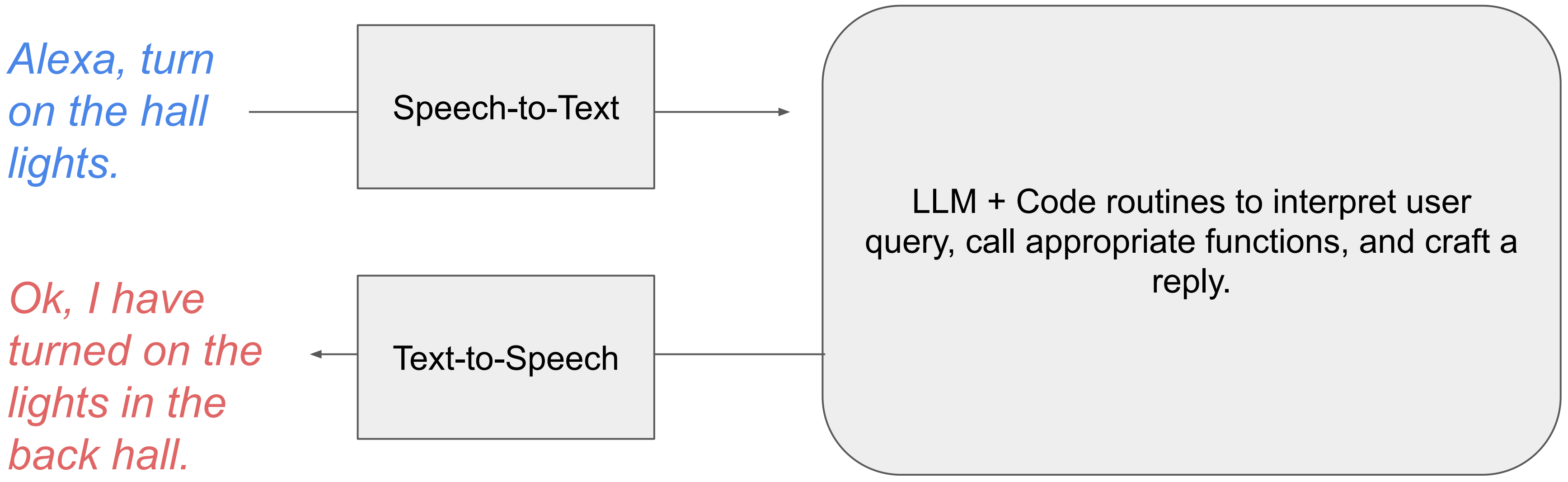
Prompting + LLM as Subroutine

We could string together a variety of these methods to create complex software systems. Since our own code is still orchestrating everything, we can route prompts and responses as needed.



Conversational Interfaces

This is especially useful when creating a system that will use a conversational interface, for example a voice-controlled app.


Alexa, turn on the hall lights.
Speech-to-Text
LLM + Code routines to interpret user query, call appropriate functions, and craft a reply.
Text-to-Speech
Ok, I have turned on the lights in the back hall.

Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG)

What if we need to include specialized data in the LLM query? In the prior examples, the LLM itself would not know the current weather or the state of the hallway lighting.

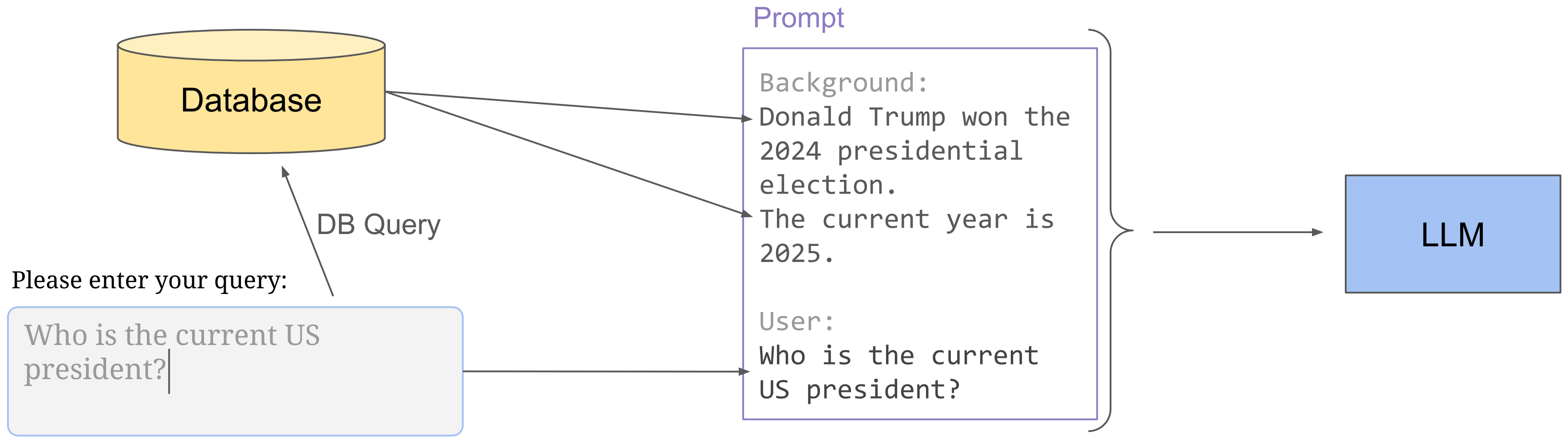
More broadly, an LLM will not have access to live information or non-public data sources.

To solve this, we can inject information from another data source into our prompts.

Retrieval Augmented Generation (RAG)

Although we cannot include all news and events (or all information from a target domain) in our prompt, we can include pieces of information that are relevant.

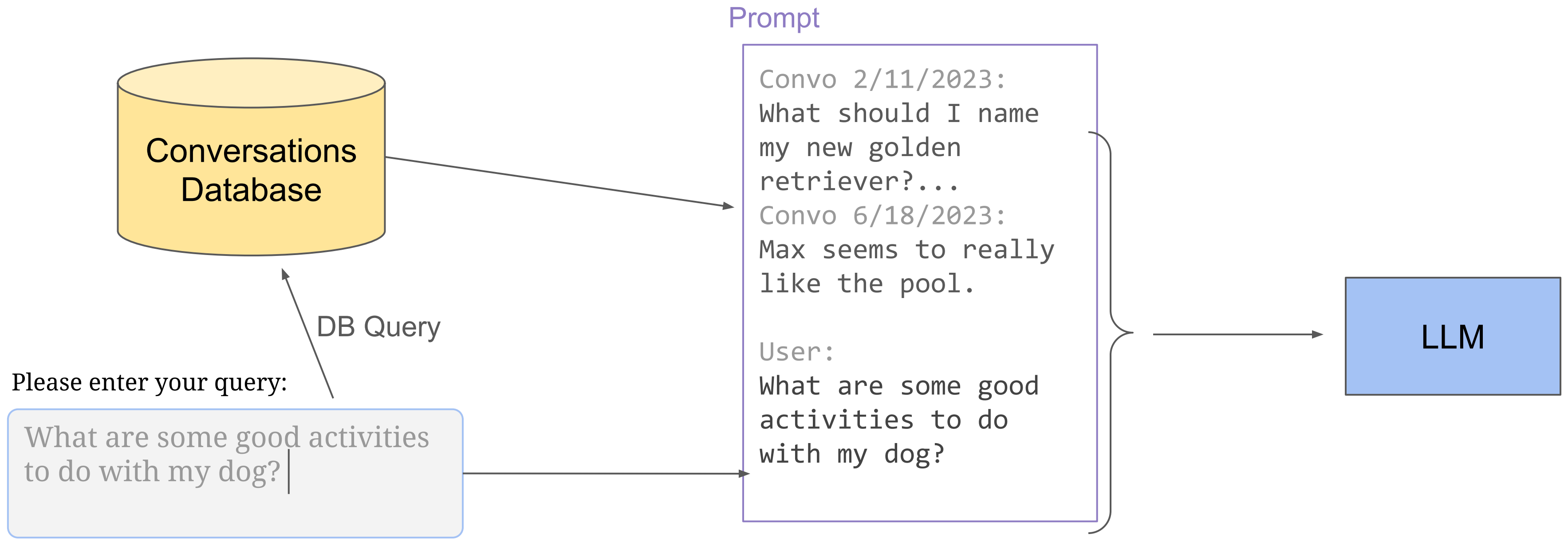
Combined with a software middleman (similar to shadow prompting), we can pull critical info into our prompt before it is processed:



```
Background:
Donald Trump won the
2024 presidential
election.
The current year is
2025.

User:
Who is the current
US president?
```

Retrieval Augmented Generation (RAG)

A really common use-case of RAG is pulling in relevant information from past user conversations. This is how ChatGPT “knows about you”.



```
Convo 2/11/2023:
What should I name
my new golden
retriever?...
Convo 6/18/2023:
Max seems to really
like the pool.

User:
What are some good
activities to do
with my dog?
```

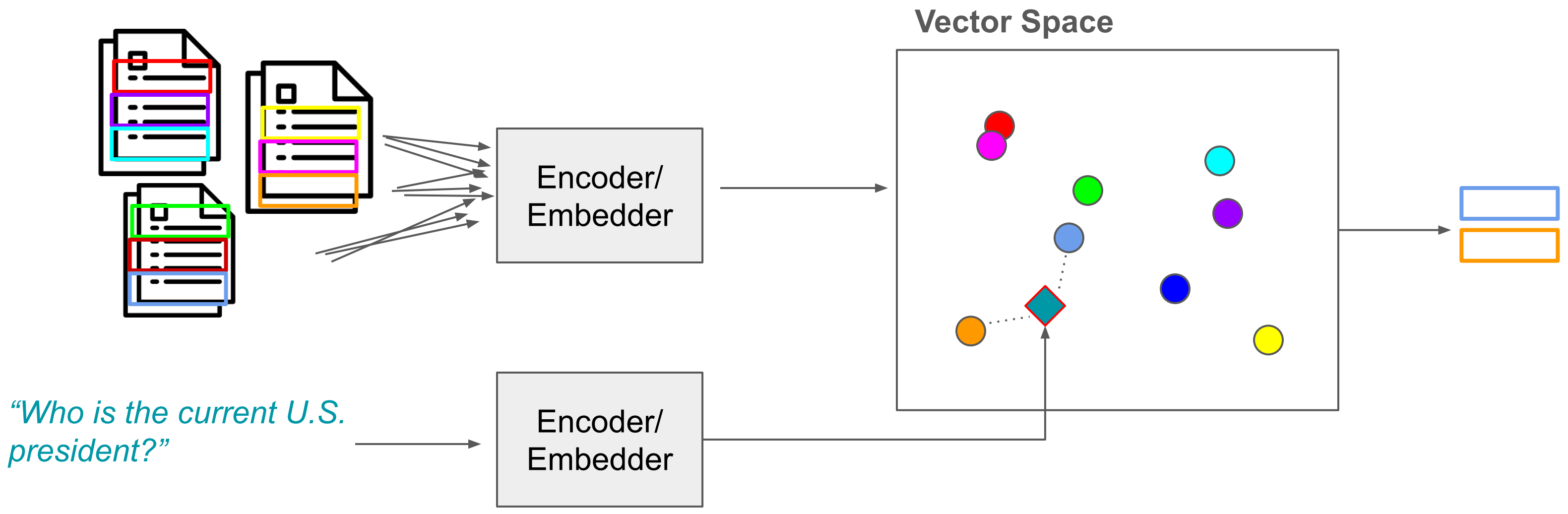
RAG

There are 4 basic steps:

1) Capture the user’s query before sending to the LLM
2) Look up the most relevant pieces of information in a database
3) Construct a prompt that includes both the relevant information and the user’s query
4) Feed all of this into the LLM

RAG Database Query

Typically the database for RAG is a vector store. Chunks of source documents are encoded as vectors in a database. The user’s query is also encoded, and nearest-neighbors is used to find the N closest pieces of information.

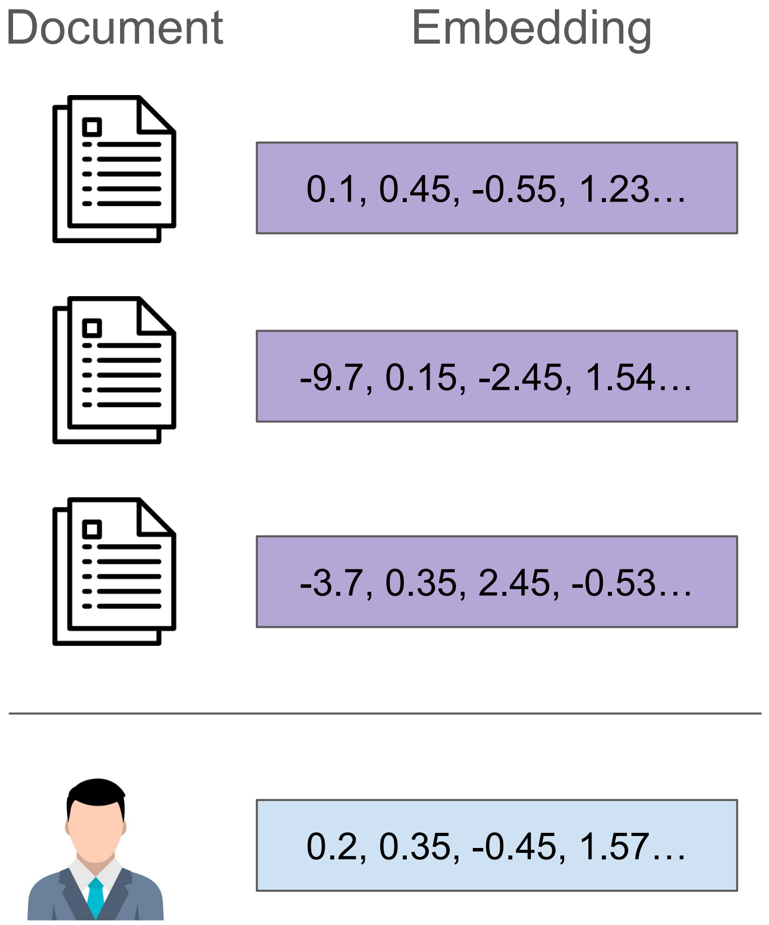


Simple RAG Algorithm

Organize a database of documents or pieces of documents.

Using an embedding model (not a word embedding), embed all samples in the database.

When the user queries the system, embed the user query with the same embedding model.

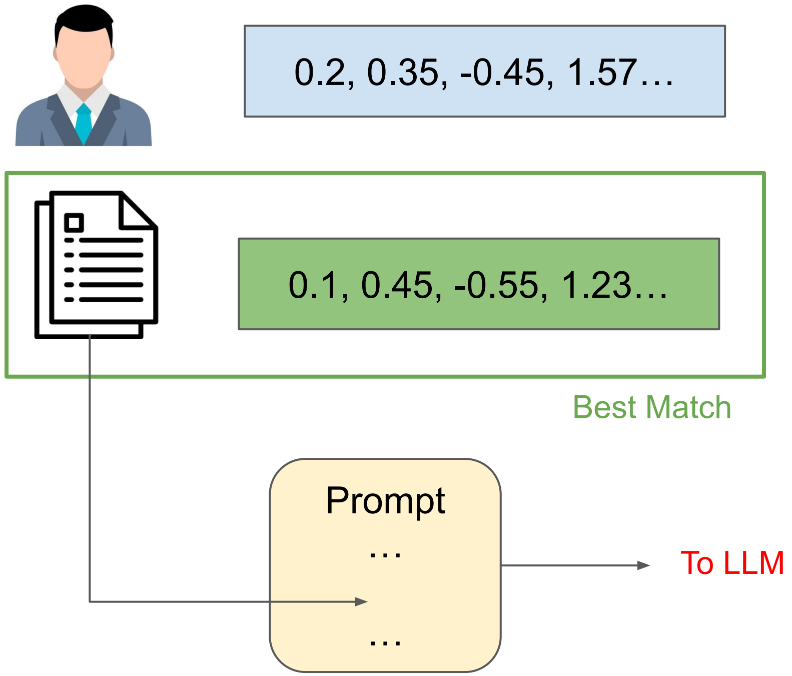


Simple RAG Algorithm (Cont.)

Select the documents with vector representations most similar to the user query.

- Small set of documents: use something like `torch.nn.functional.cosine_similarity`
- Many documents: tree-based data structure for fast lookup.

Copy the relevant text into a prompt, and proceed with LLM query.



LLM Inference Services + Tools

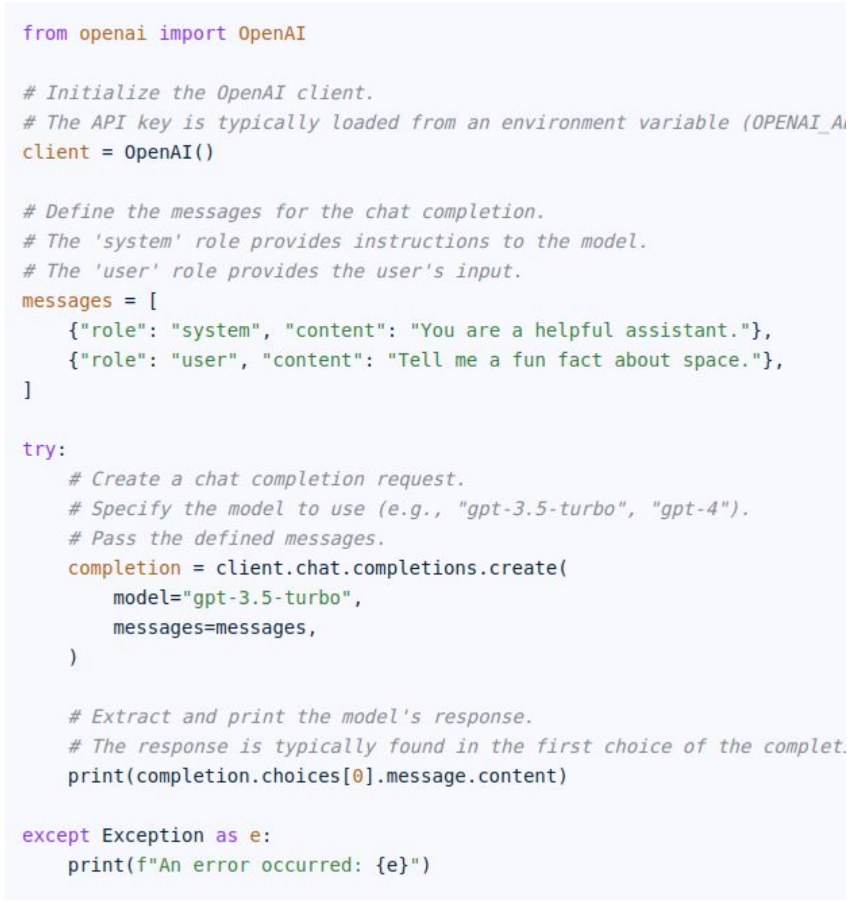
Inference Services

Large LLMs are often provided as an external service rather than run locally, due to size and cost.

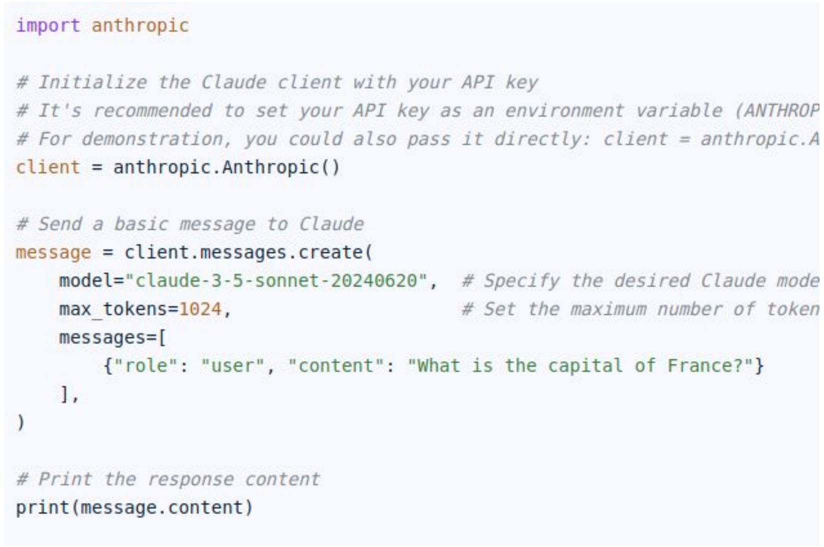
In the preceding examples, invocations of an LLM would probably be API requests sent over the web, with the reply of the LLM coming back as a response.

Major LLM companies will provide APIs that you can hit from your software.

Two Python Examples

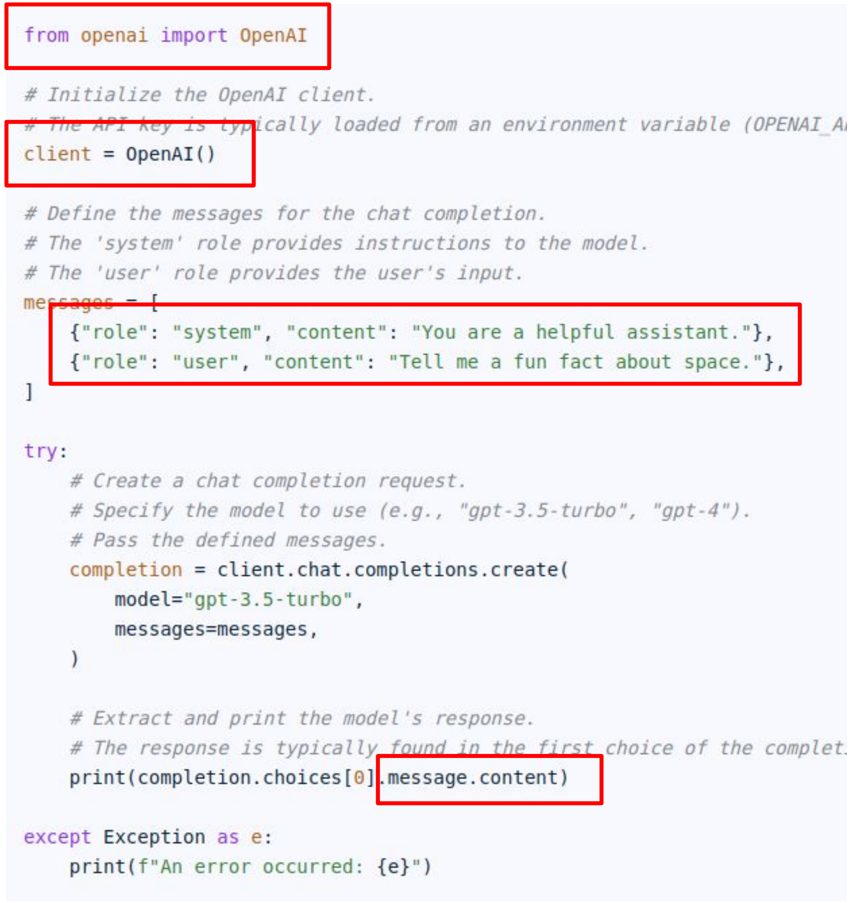


```python
from openai import OpenAI

# Initialize the OpenAI client.
# The API key is typically loaded from an environment variable (OPENAI_A
client = OpenAI()

# Define the messages for the chat completion.
# The 'system' role provides instructions to the model.
# The 'user' role provides the user's input.
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Tell me a fun fact about space."},
]

try:
    # Create a chat completion request.
    # Specify the model to use (e.g., "gpt-3.5-turbo", "gpt-4").
    # Pass the defined messages.
    completion = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=messages,
    )

    # Extract and print the model's response.
    # The response is typically found in the first choice of the complet
    print(completion.choices[0].message.content)

except Exception as e:
    print(f"An error occurred: {e}")
```

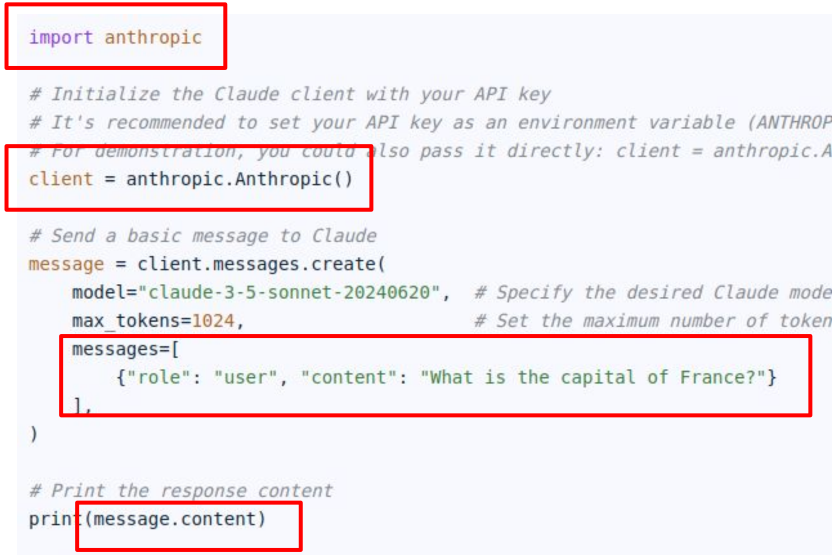


```python
import anthropic

# Initialize the Claude client with your API key
# It's recommended to set your API key as an environment variable (ANTHROP
# For demonstration, you could also pass it directly: client = anthropic.A
client = anthropic.Anthropic()

# Send a basic message to Claude
message = client.messages.create(
    model="claude-3-5-sonnet-20240620",  # Specify the desired Claude mode
    max_tokens=1024,                     # Set the maximum number of token
    messages=[
        {"role": "user", "content": "What is the capital of France?"}
    ],
)

# Print the response content
print(message.content)
```

Two Python Examples



```python
from openai import OpenAI

# Initialize the OpenAI client.
# The API key is typically loaded from an environment variable (OPENAI_A
client = OpenAI()

# Define the messages for the chat completion.
# The 'system' role provides instructions to the model.
# The 'user' role provides the user's input.
messages = [
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Tell me a fun fact about space."},
]

try:
    # Create a chat completion request.
    # Specify the model to use (e.g., "gpt-3.5-turbo", "gpt-4").
    # Pass the defined messages.
    completion = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=messages,
    )

    # Extract and print the model's response.
    # The response is typically found in the first choice of the complet
    print(completion.choices[0].message.content)

except Exception as e:
    print(f"An error occurred: {e}")
```



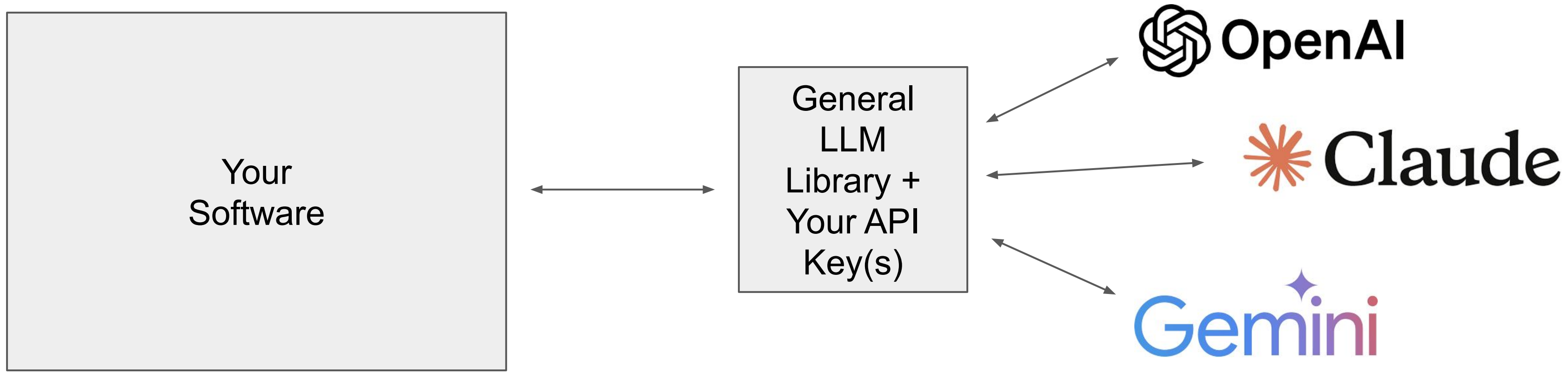
```python
import anthropic

# Initialize the Claude client with your API key
# It's recommended to set your API key as an environment variable (ANTHROP
# For demonstration, you could also pass it directly: client = anthropic.A
client = anthropic.Anthropic()

# Send a basic message to Claude
message = client.messages.create(
    model="claude-3-5-sonnet-20240620",  # Specify the desired Claude mode
    max_tokens=1024,                     # Set the maximum number of token
    messages=[
        {"role": "user", "content": "What is the capital of France?"}
    ],
)

# Print the response content
print(message.content)
```

Note the similar setup between these two providers. There has been a slow convergence towards standard API formats.

Libraries

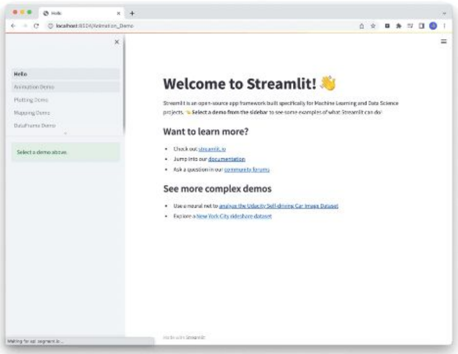
Additionally, many libraries exist which help handle the management of prompts, routing, etc and often abstract away any vendor-specific API formats. Popular options include **LangChain, DSPy, and Pydantic.**

Using these libraries, you typically supply your preferred LLM and the API key associated with that LLM, and the library backend will handle the rest.



Browser UIs

Alongside these libraries for communicating with LLMs, tools have also appeared to make nice UIs that go on top of them. These usually run in the browser and connect to python on the backend.



Welcome to Streamlit! 👋

Streamlit is an open-source app framework built specifically for Machine Learning and Data Science projects. 👈 Select a demo from the sidebar to see some examples of what Streamlit can do!

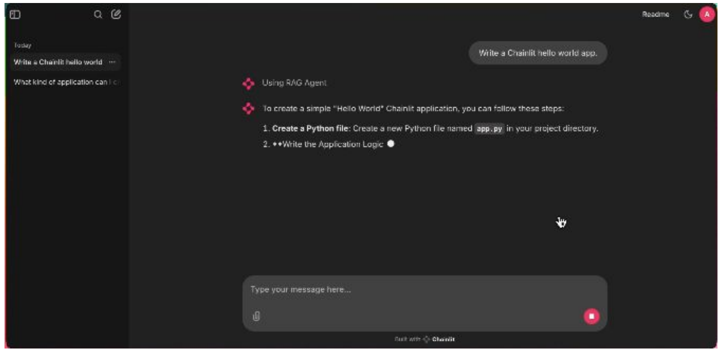
Want to learn more?

- Check out streamlit.io
- Jump into our documentation
- Ask a question in our community forums

See more complex demos

- Use a neural net to analyze the Udacity Self-driving Car Image Dataset
- Explore a New York City rideshare dataset

Streamlit

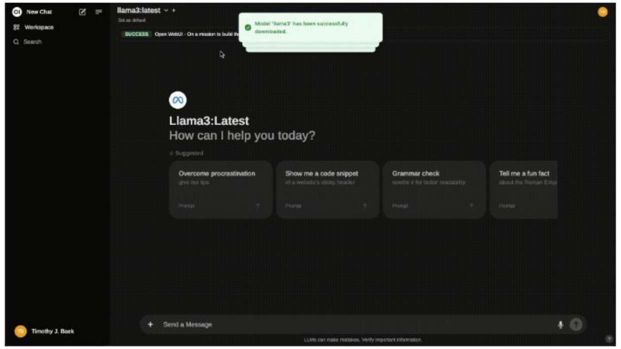

Today
Write a Chainlit hello world
What kind of application can I c
Readme
Using RAG Agent
Type your message here...
Built with Chainlit

Write a Chainlit hello world app.

To create a simple "Hello World" Chainlit application, you can follow these steps:

1. **Create a Python file:** Create a new Python file named `app.py` in your project directory.
2. **Write the Application Logic ●

Chainlit



Model 'llama3' has been successfully downloaded.

LLMs can make mistakes. Verify important information.

Open WebUI

Note / Opinion

General libraries like LangChain may become less useful as API formats converge.

LLM-agnositc libraries can also be incompatible with provider-specific offerings. Although, these will probably converge too.

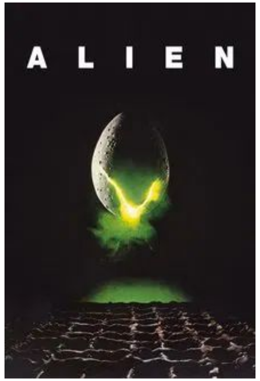
Pydantic seems to be used by some LLM providers as the default system for packaging up messages, so I recommend using that.

Hugging Face

For our assignment this week we will use Hugging Face for inference, because it comes with some free credits (a staggering 10 cents of credit per month).

We have already used Hugging Face here and there in our assignments. It has become the go-to for model and data hosting (which we will look at in a few slides), and lately is also pushing inference services.

Hugging Face is named after the hugging face emoji and uses it as their logo. It is not (like I thought) named after the creepy thing from alien.



Hugging Face Inference

To use HF Inference, you will need to an account (https://huggingface.co/join)

Just like other LLM providers, you will need to use a special key or token when calling their API in python. Instructions for this are in the appendix of this lecture.

```python
import os
from openai import OpenAI # pip install openai

client = OpenAI(
    base_url="https://router.huggingface.co/v1",
    api_key=os.environ["HF_TOKEN"],
)

completion = client.chat.completions.create(
    model="Qwen/Qwen3-Next-80B-A3B-Instruct:novita",
    messages=[
        {
            "role": "user",
            "content": "What is the capital of France?"
        }
    ],
)

print(completion.choices[0].message)
```

Hugging Face Inference

To use HF Inference, you will need to an account (https://huggingface.co/join)

Just like other LLM providers, you will need to use a special key or token when calling their API in python. Instructions for this are in the appendix of this lecture.

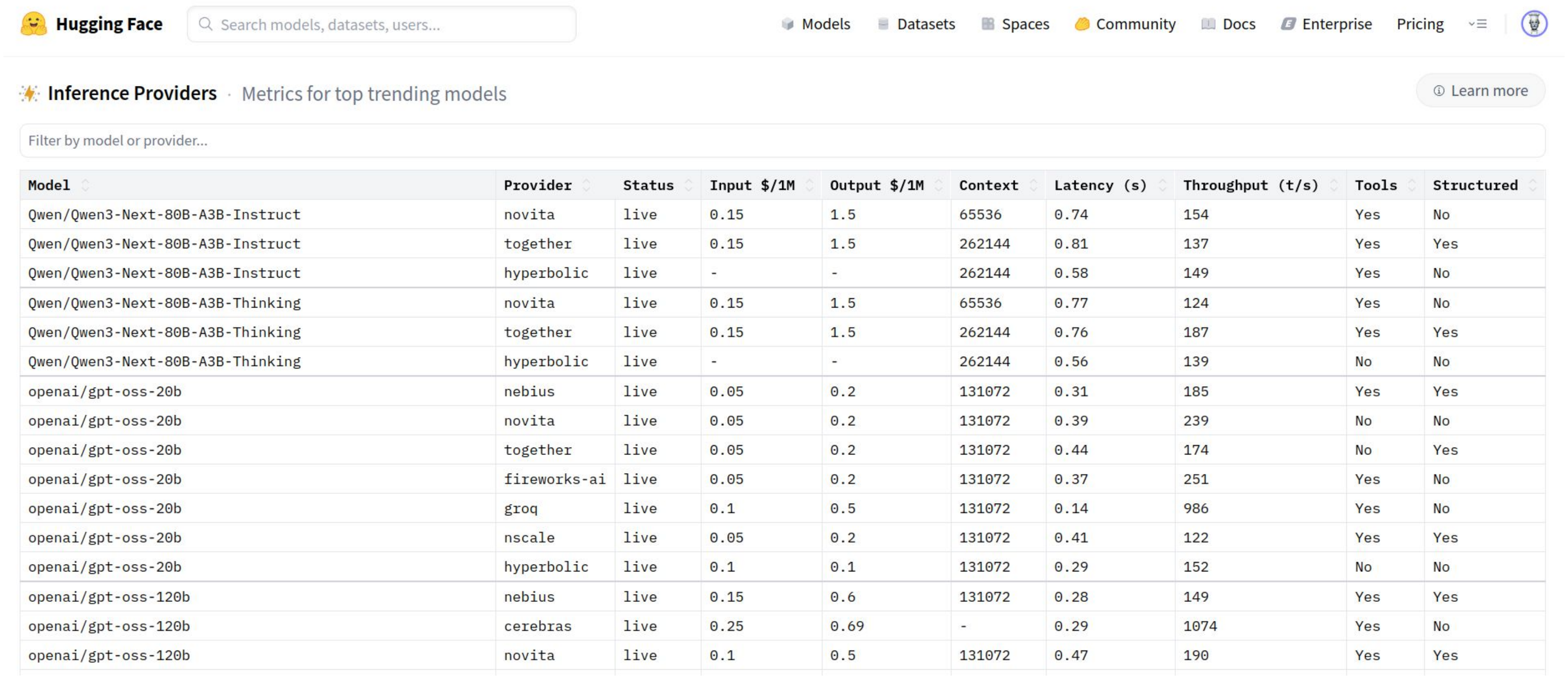
```python
import os
from openai import OpenAI # pip install openai

client = OpenAI(
    base_url="https://router.huggingface.co/v1",
    api_key=os.environ["HF_TOKEN"],
)

completion = client.chat.completions.create(
    model="Qwen/Qwen3-Next-80B-A3B-Instruct:novita",
    messages=[
        {
            "role": "user",
            "content": "What is the capital of France?"
        }
    ],
)

print(completion.choices[0].message)
```

Look familiar?
HF uses the OpenAI API to handle things, so again the format is the same.

Note that we are not actually communicating with OpenAI's servers here. We are just using their python classes to package and transmit messages.

Inference Details

This is a very helpful link: https://huggingface.co/inference/models



Inference Providers · Metrics for top trending models

Model	Provider	Status	Input $/1M	Output $/1M	Context	Latency (s)	Throughput (t/s)	Tools	Structured
Qwen/Qwen3-Next-80B-A3B-Instruct	novita	live	0.15	1.5	65536	0.74	154	Yes	No
Qwen/Qwen3-Next-80B-A3B-Instruct	together	live	0.15	1.5	262144	0.81	137	Yes	Yes
Qwen/Qwen3-Next-80B-A3B-Instruct	hyperbolic	live	-	-	262144	0.58	149	Yes	No
Qwen/Qwen3-Next-80B-A3B-Thinking	novita	live	0.15	1.5	65536	0.77	124	Yes	No
Qwen/Qwen3-Next-80B-A3B-Thinking	together	live	0.15	1.5	262144	0.76	187	Yes	Yes
Qwen/Qwen3-Next-80B-A3B-Thinking	hyperbolic	live	-	-	262144	0.56	139	No	No
openai/gpt-oss-20b	nebius	live	0.05	0.2	131072	0.31	185	Yes	Yes
openai/gpt-oss-20b	novita	live	0.05	0.2	131072	0.39	239	No	No
openai/gpt-oss-20b	together	live	0.05	0.2	131072	0.44	174	No	Yes
openai/gpt-oss-20b	fireworks-ai	live	0.05	0.2	131072	0.37	251	Yes	No
openai/gpt-oss-20b	groq	live	0.1	0.5	131072	0.14	986	Yes	No
openai/gpt-oss-20b	nscale	live	0.05	0.2	131072	0.41	122	Yes	Yes
openai/gpt-oss-20b	hyperbolic	live	0.1	0.1	131072	0.29	152	No	No
openai/gpt-oss-120b	nebius	live	0.15	0.6	131072	0.28	149	Yes	Yes
openai/gpt-oss-120b	cerebras	live	0.25	0.69	-	0.29	1074	Yes	No
openai/gpt-oss-120b	novita	live	0.1	0.5	131072	0.47	190	Yes	Yes

More Hugging Face

Main HF Libraries

HF has lots of offerings that are not tied to an account. Primarily, HF serves as a repository for open source models and datasets.

For this week's assignment we will be using:

- datasets (pip install datasets==3.1.0)
- transformers (pip install transformers)

The 4.0.0 release of datasets still has some bugs going on (Sept 2025). I'd stick to 3.X for now.

HF Datasets

Browsing

Downloading / using

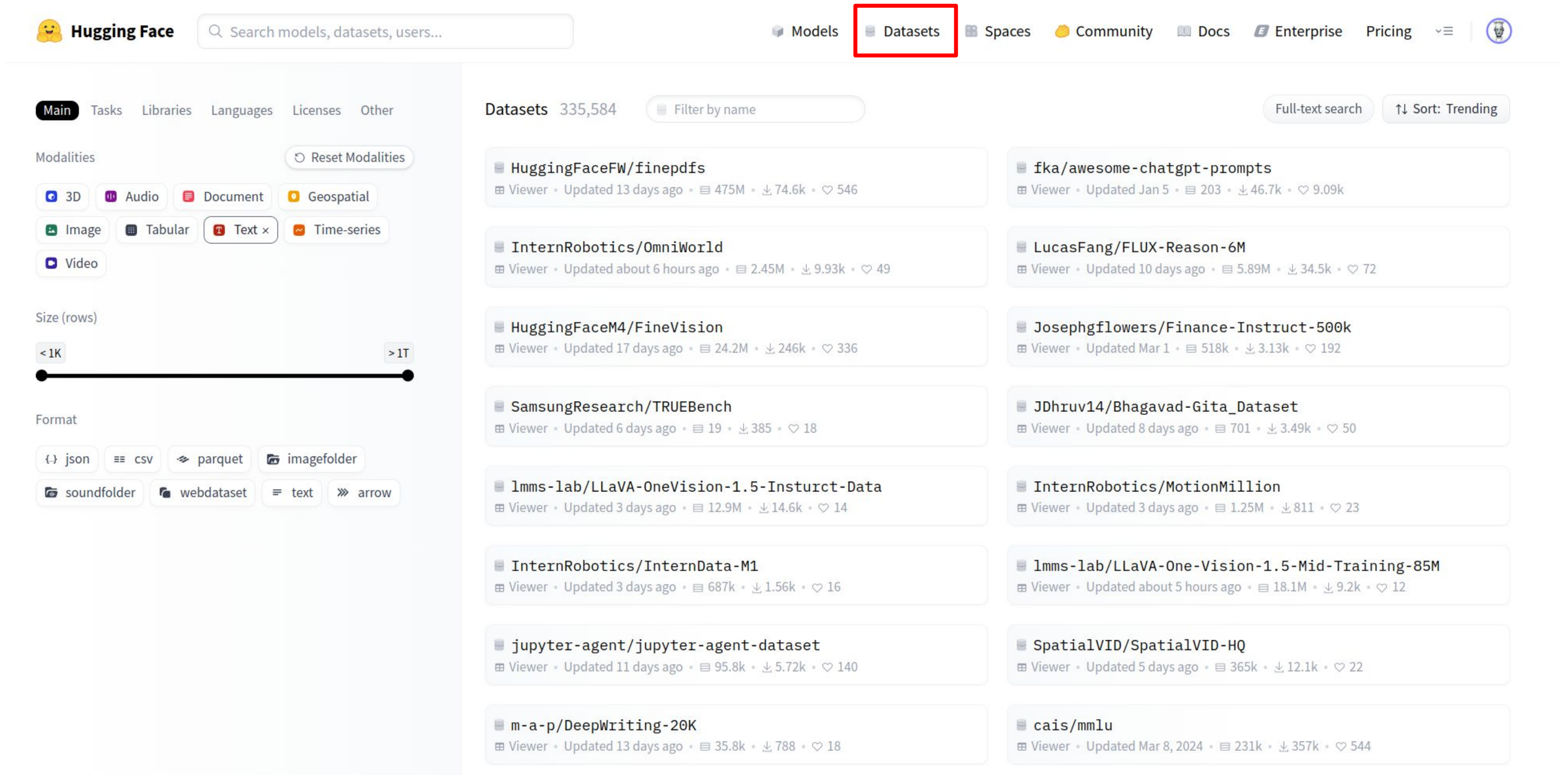
Format of the dataset (train/test, columns, etc)

Process

Streaming

https://huggingface.co/datasets?modality=modality:text&sort=trending

Datasets



Datasets 335,584

HuggingFaceFW/finepdfs
Viewer • Updated 13 days ago • 475M • 74.6k • 546

fka/awesome-chatgpt-prompts
Viewer • Updated Jan 5 • 203 • 46.7k • 9.09k

InternRobotics/OmniWorld
Viewer • Updated about 6 hours ago • 2.45M • 9.93k • 49

LucasFang/FLUX-Reason-6M
Viewer • Updated 10 days ago • 5.89M • 34.5k • 72

HuggingFaceM4/FineVision
Viewer • Updated 17 days ago • 24.2M • 246k • 336

Josephgflowers/Finance-Instruct-500k
Viewer • Updated Mar 1 • 518k • 3.13k • 192

SamsungResearch/TRUEBench
Viewer • Updated 6 days ago • 19 • 385 • 18

JDhruv14/Bhagavad-Gita_Dataset
Viewer • Updated 8 days ago • 701 • 3.49k • 50

lmms-lab/LLaVA-OneVision-1.5-Insturct-Data
Viewer • Updated 3 days ago • 12.9M • 14.6k • 14

InternRobotics/MotionMillion
Viewer • Updated 3 days ago • 1.25M • 811 • 23

InternRobotics/InternData-M1
Viewer • Updated 3 days ago • 687k • 1.56k • 16

lmms-lab/LLaVA-One-Vision-1.5-Mid-Training-85M
Viewer • Updated about 5 hours ago • 18.1M • 9.2k • 12

jupyter-agent/jupyter-agent-dataset
Viewer • Updated 11 days ago • 95.8k • 5.72k • 140

SpatialVID/SpatialVID-HQ
Viewer • Updated 5 days ago • 365k • 12.1k • 22

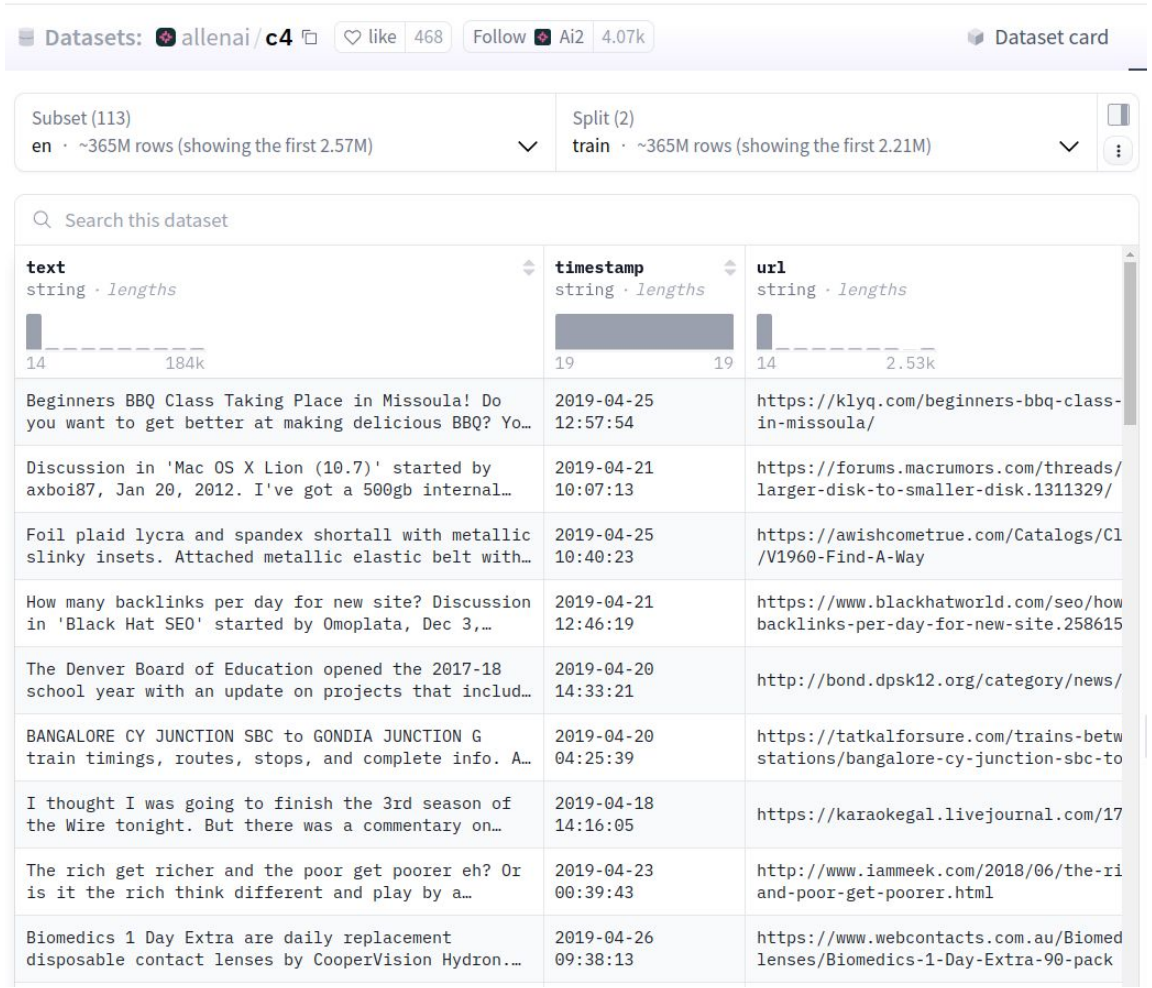
m-a-p/DeepWriting-20K
Viewer • Updated 13 days ago • 35.8k • 788 • 18

cais/mmlu
Viewer • Updated Mar 8, 2024 • 231k • 357k • 544

Datasets

A huggingface dataset is a collection of samples (rows) where each sample has entries in various columns.

Datasets can be further organized into subsets, and are often split into a train and test division.



text string · *lengths* 14 184k	timestamp string · *lengths* 19 19	url string · *lengths* 14 2.53k
Beginners BBQ Class Taking Place in Missoula! Do you want to get better at making delicious BBQ? Yo…	2019-04-25 12:57:54	https://klyq.com/beginners-bbq-class-in-missoula/
Discussion in 'Mac OS X Lion (10.7)' started by axboi87, Jan 20, 2012. I've got a 500gb internal…	2019-04-21 10:07:13	https://forums.macrumors.com/threads/larger-disk-to-smaller-disk.1311329/
Foil plaid lycra and spandex shortall with metallic slinky insets. Attached metallic elastic belt with…	2019-04-25 10:40:23	https://awishcometrue.com/Catalogs/Cl /V1960-Find-A-Way
How many backlinks per day for new site? Discussion in 'Black Hat SEO' started by Omoplata, Dec 3,…	2019-04-21 12:46:19	https://www.blackhatworld.com/seo/how backlinks-per-day-for-new-site.258615
The Denver Board of Education opened the 2017-18 school year with an update on projects that includ…	2019-04-20 14:33:21	http://bond.dpsk12.org/category/news/
BANGALORE CY JUNCTION SBC to GONDIA JUNCTION G train timings, routes, stops, and complete info. A…	2019-04-20 04:25:39	https://tatkalforsure.com/trains-betw stations/bangalore-cy-junction-sbc-to
I thought I was going to finish the 3rd season of the Wire tonight. But there was a commentary on…	2019-04-18 14:16:05	https://karaokegal.livejournal.com/17
The rich get richer and the poor get poorer eh? Or is it the rich think different and play by a…	2019-04-23 00:39:43	http://www.iammeek.com/2018/06/the-ri and-poor-get-poorer.html
Biomedics 1 Day Extra are daily replacement disposable contact lenses by CooperVision Hydron.…	2019-04-26 09:38:13	https://www.webcontacts.com.au/Biomed lenses/Biomedics-1-Day-Extra-90-pack

Datasets

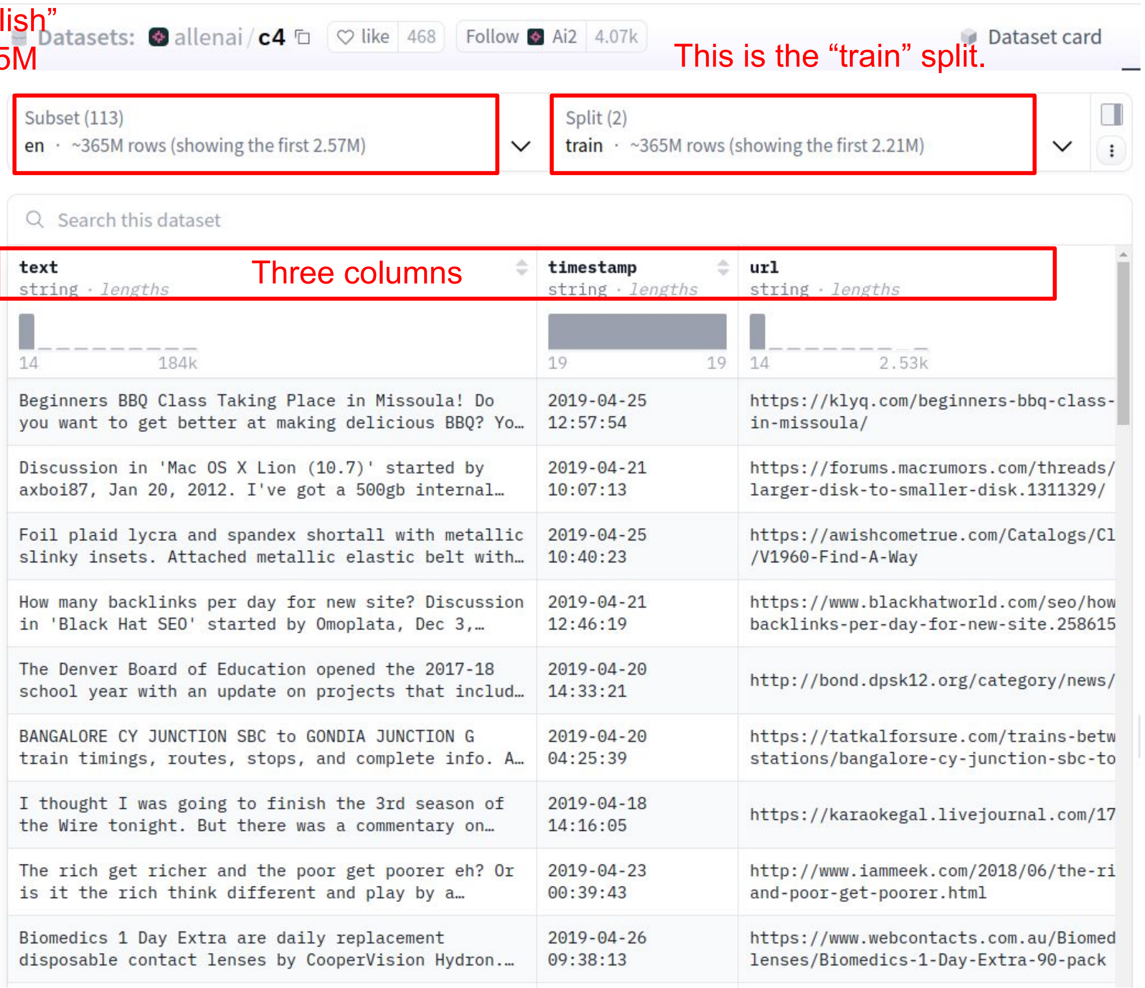
A huggingface dataset is a collection of samples (rows) where each sample has entries in various columns.

Datasets can be further organized into subsets, and are often split into a train and test division.

This is the “english” subset, with 365M entries.

This is the “train” split.

Three columns



Datasets: allenai / c4 like 468 Follow Ai2 4.07k Dataset card

Subset (113)
en · ~365M rows (showing the first 2.57M)

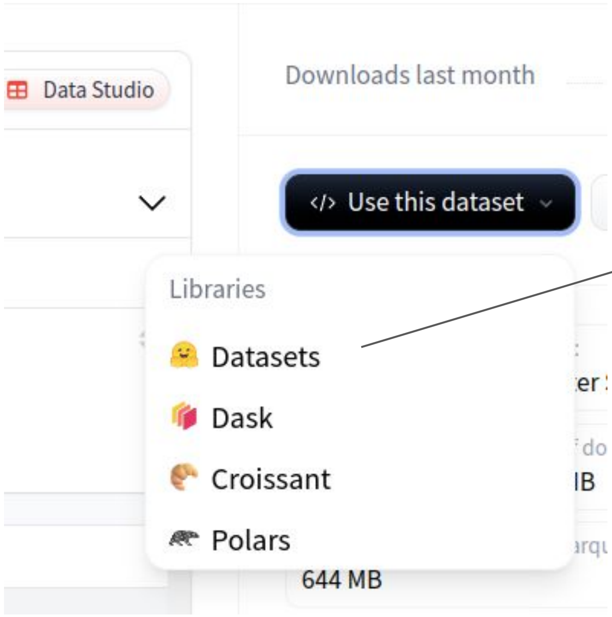
Split (2)
train · ~365M rows (showing the first 2.21M)

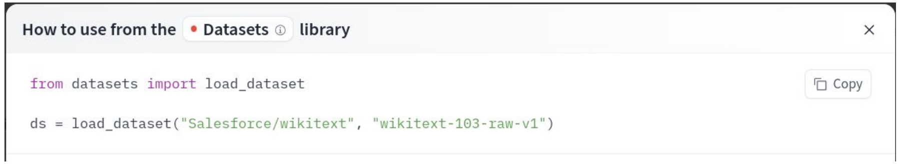
Search this dataset

text string · lengths 14 184k	timestamp string · lengths 19 19	url string · lengths 14 2.53k
Beginners BBQ Class Taking Place in Missoula! Do you want to get better at making delicious BBQ? Yo…	2019-04-25 12:57:54	https://klyq.com/beginners-bbq-class-in-missoula/
Discussion in 'Mac OS X Lion (10.7)' started by axboi87, Jan 20, 2012. I've got a 500gb internal…	2019-04-21 10:07:13	https://forums.macrumors.com/threads/larger-disk-to-smaller-disk.1311329/
Foil plaid lycra and spandex shortall with metallic slinky insets. Attached metallic elastic belt with…	2019-04-25 10:40:23	https://awishcometrue.com/Catalogs/Cl /V1960-Find-A-Way
How many backlinks per day for new site? Discussion in 'Black Hat SEO' started by Omoplata, Dec 3,…	2019-04-21 12:46:19	https://www.blackhatworld.com/seo/how backlinks-per-day-for-new-site.258615
The Denver Board of Education opened the 2017-18 school year with an update on projects that includ…	2019-04-20 14:33:21	http://bond.dpsk12.org/category/news/
BANGALORE CY JUNCTION SBC to GONDIA JUNCTION G train timings, routes, stops, and complete info. A…	2019-04-20 04:25:39	https://tatkalforsure.com/trains-betw stations/bangalore-cy-junction-sbc-to
I thought I was going to finish the 3rd season of the Wire tonight. But there was a commentary on…	2019-04-18 14:16:05	https://karaokegal.livejournal.com/17
The rich get richer and the poor get poorer eh? Or is it the rich think different and play by a…	2019-04-23 00:39:43	http://www.iammeek.com/2018/06/the-ri and-poor-get-poorer.html
Biomedics 1 Day Extra are daily replacement disposable contact lenses by CooperVision Hydron.…	2019-04-26 09:38:13	https://www.webcontacts.com.au/Biomed lenses/Biomedics-1-Day-Extra-90-pack

Using a dataset in Python

Smaller datasets often have an example of loading with the datasets library:



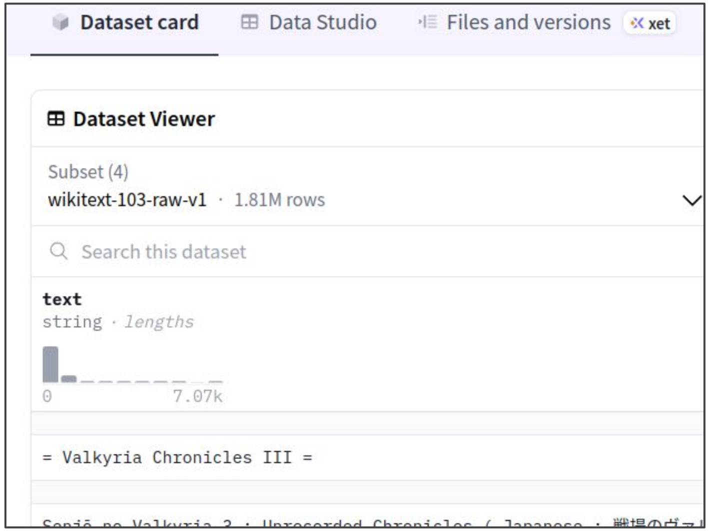


```python
from datasets import load_dataset

ds = load_dataset("Salesforce/wikitext", "wikitext-103-raw-v1")
```

This example is from https://huggingface.co/datasets/Salesforce/wikitext

Using a dataset in python

Pasting the code into a script, we can examine the dataset. Note that it will need to download the first time you run.



Dataset Viewer

Subset (4)
wikitext-103-raw-v1 · 1.81M rows

text
string · lengths

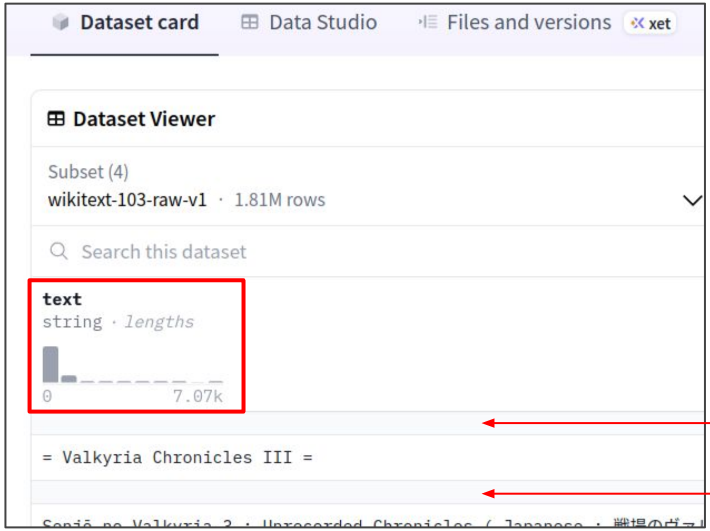
= Valkyria Chronicles III =

Senjō no Valkyria 3 : Unrecorded Chronicles (Japanese : 戦場のヴァ

```python
from datasets import load_dataset

# loads the whole dataset
ds = load_dataset("Salesforce/wikitext",
"wikitext-2-raw-v1")

# look at the available columns
print(ds.column_names)
# prints: {'test': ['text'], 'train': ['text'],
'validation': ['text']}
# we can see there are three splits, each with only one
column called "text".

ds = ds["train"]
print("Samples:", len(ds))
# prints: Samples: 36718

# print out the first few entries
print(ds[0])
print(ds[1])
print(ds[2])
# prints:
# {'text': ''}
# {'text': ' = Valkyria Chronicles III = \n'}
# {'text': ''}

# The first and third rows are empty.
```

Using a dataset in python

Pasting the code into a script, we can examine the dataset. Note that it will need to download the first time you run.



wikitext-103-raw-v1 · 1.81M rows

text
string · lengths
0 7.07k

= Valkyria Chronicles III =

Senjō no Valkyria 3 : Unrecorded Chronicles (Japanese : 戦場のヴァ

```python
from datasets import load_dataset

# loads the whole dataset
ds = load_dataset("Salesforce/wikitext",
"wikitext-2-raw-v1")

# look at the available columns
print(ds.column_names)
# prints: {'test': ['text'], 'train': ['text'],
'validation': ['text']}
# we can see there are three splits, each with only one
column called "text".

ds = ds["train"]
print("Samples:", len(ds))
# prints: Samples: 36718

# print out the first few entries
print(ds[0])
print(ds[1])
print(ds[2])
# prints:
# {'text': ''}
# {'text': ' = Valkyria Chronicles III = \n'}
# {'text': ''}

# The first and third rows are empty.
```

This is a peculiarity of the dataset. We can confirm by looking on the HF page.

Processing the Dataset

There are all sorts of things you can do to refine or modify a dataset once it is loaded. See: https://huggingface.co/docs/datasets/en/process

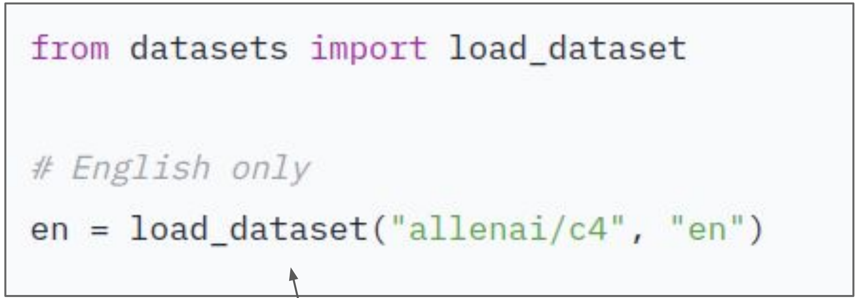
For example, some really common operations might be things like:

- Remove columns that you do not need
- Shuffle the dataset order
- For testing purposes, only use the first N samples
- etc

What about that first dataset?

The initial example, c4, is huge (https://huggingface.co/datasets/allenai/c4). For this reason, there is no “Use this Dataset” button. However, there are options to download it directly.

If we scroll down the page, there are some examples anyway. What would happen if we used them?



```python
from datasets import load_dataset

# English only
en = load_dataset("allenai/c4", "en")
```

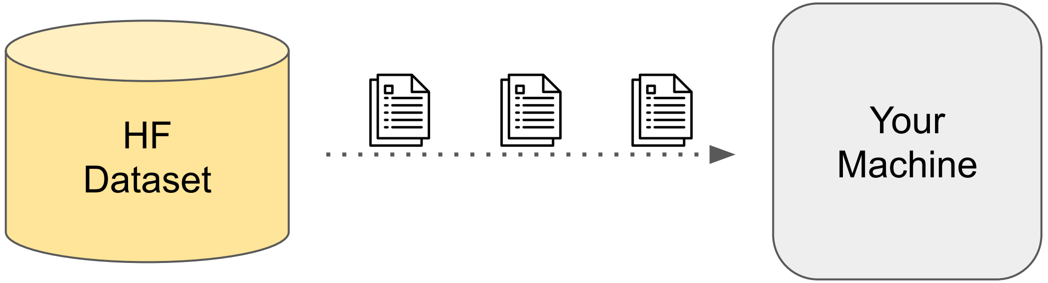
This command will download maybe 100GB to your computer. The entire dataset (all subsets) is 13GB.

Streaming

Hugging Face has an awesome alternative called streaming datasets which makes this much easier:

```
en = load_dataset("allenai/c4", "en", streaming=True)
```

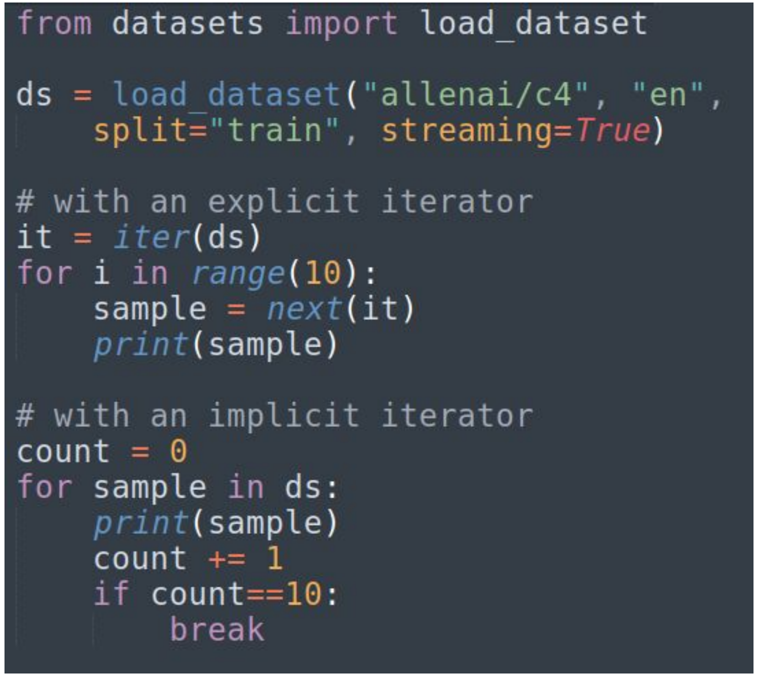
This will create an **IterableDataset**, which is a dataset of undefined length that can be iterated. When your python script needs the next item from the dataset, it is downloaded on the fly (and deleted after use). This allows you to use giant datasets without ever storing them locally.



Streaming

If we set a dataset to stream, we can then iterate through it and samples are pulled as needed.

Note that this may start up a little slower than you expect, since the loader will pull down data in batches as well as supporting files.



```python
from datasets import load_dataset

ds = load_dataset("allenai/c4", "en",
    split="train", streaming=True)

# with an explicit iterator
it = iter(ds)
for i in range(10):
    sample = next(it)
    print(sample)

# with an implicit iterator
count = 0
for sample in ds:
    print(sample)
    count += 1
    if count==10:
        break
```

Streaming + Training?

Streaming can be used in a training loop, but it is not recommended.

There are a variety of practical issues with streaming when trying to train on a large amount of data:

- Download speed can easily be a bottleneck
- Need consistent internet connection
- Cannot shuffle data except for in a rolling pool
    - We can maintain a local buffer of N samples and sample randomly from this, but it is a coarse approximation to true shuffling.
- **IterableDataset** has no len(), which is needed to set up learning rate schedules.

HF Transformers

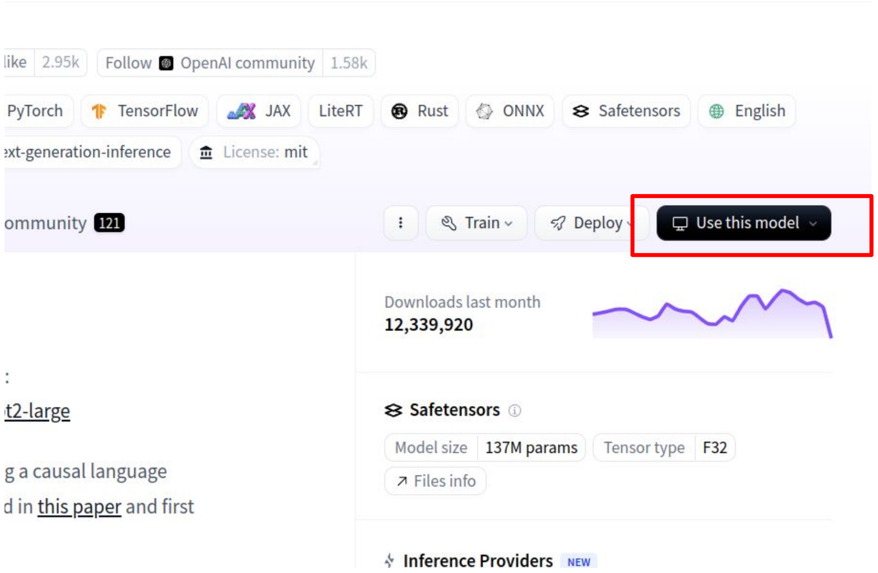
Hugging Face is also a repository for models. Most popular open-source models are available. Small-ish models can be downloaded and used locally via the **transformers** library (pip install transformers).

Note that this also includes relevant tokenizers. To process text, you need to load a model and its corresponding tokenizer.

Fortunately, this is very easy!

AutoModels and AutoTokenizers

HF Models and Tokenizers can be loaded by name. Similar to datasets, you will see a “Use this Model” button. For example, the GPT2 model (https://huggingface.co/openai-community/gpt2):



```python
# Load model directly
from transformers import AutoTokenizer, AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("openai-community/gpt2")
model = AutoModelForCausalLM.from_pretrained("openai-community/gpt2")
```

AutoTokenizers

Calling AutoTokenizer() will return an object that can convert strings into text and vice versa.

You can call the object or use encode() to tokenizer.encode text, and tokenizer.decode() to convert back.

```
from transformers import AutoTokenizer,
AutoModelForCausalLM

tokenizer =
AutoTokenizer.from_pretrained("openai-community/gpt2")

text = "This is some example text."

tokens = tokenizer(text, return_tensors='pt')

# important
tokens = tokens["input_ids"][0]
print(tokens)

text_again = tokenizer.decode(tokens)
print(text_again)

# prints:
# tensor([1212,  318,  617, 1672, 2420,   13])
# This is some example text.
```

AutoTokenizers

Notes:

- We can pass “return_tensors” to select pytorch as the data scheme used
- The output of encoding is a dictionary with keys “input_ids” and “attention_mask”, which are used for the model forward pass.
- The values are given a batch dimension, so we also need to select the first “item in the batch”, hence the [0].

```
from transformers import AutoTokenizer,
AutoModelForCausalLM

tokenizer =
AutoTokenizer.from_pretrained("openai-community/gpt2")

text = "This is some example text."

tokens = tokenizer(text, return_tensors='pt')

# important
tokens = tokens["input_ids"][0]
print(tokens)

text_again = tokenizer.decode(tokens)
print(text_again)

# prints:
# tensor([1212,  318,  617, 1672, 2420,   13])
# This is some example text.
```

AutoModels

We can similarly create a model by name to run locally. Only do this for small models.

The reason the tokenizer output is an unwieldy dictionary is so that it can be directly passed into the model as arbitrary keyword arguments.

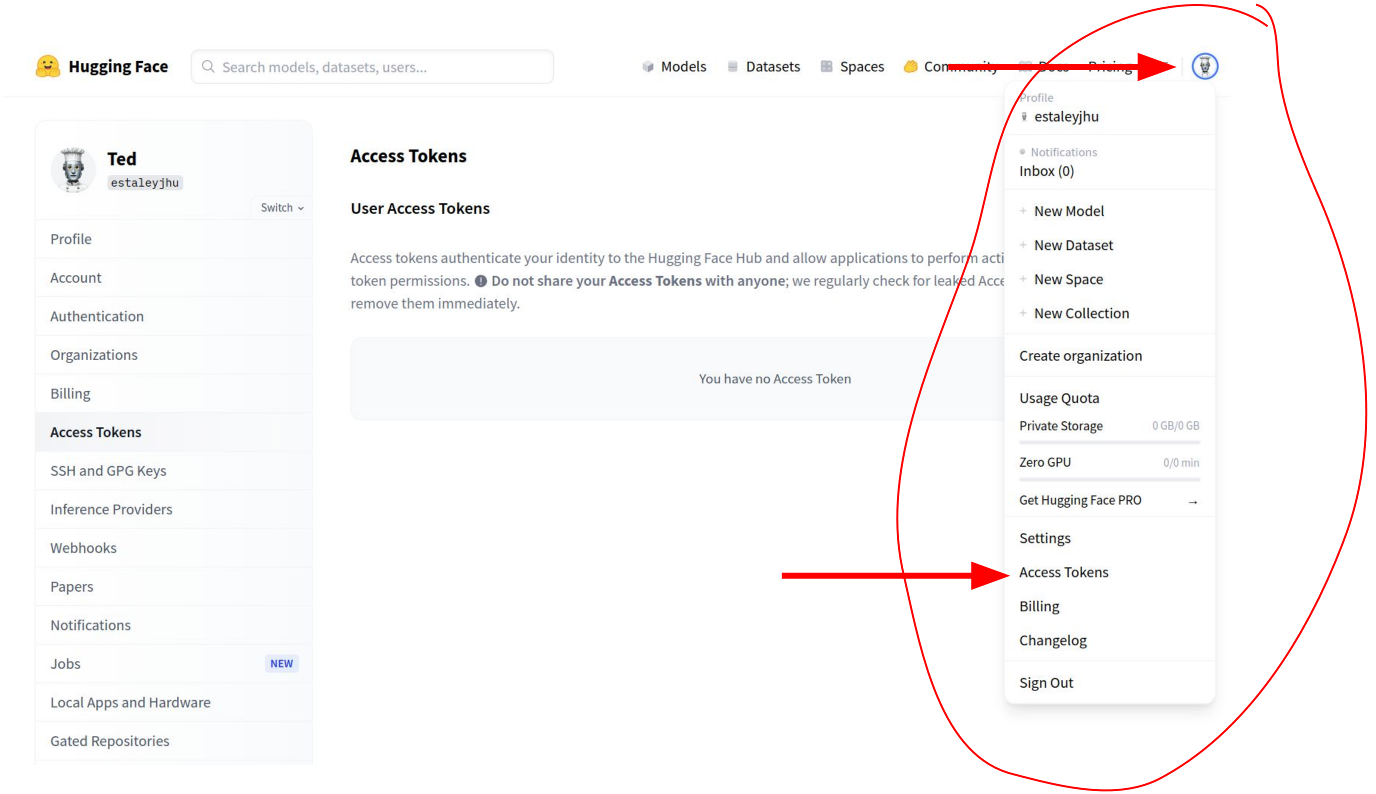
```
from transformers import AutoTokenizer,
AutoModelForCausalLM

tokens = tokenizer(text, return_tensors='pt')
model =
AutoModelForCausalLM.from_pretrained("openai-community/gpt
2")
output = model(**tokens)

print(output["logits"].shape)
# prints (1, 6, 50257)
# this is the raw output distribution at each position
```

Note that **output** is also a dictionary, here with fields “logits” and “past_key_values”. You should recognize these as the inputs to a sampler!

Review Assignment

Appendices

Access Token



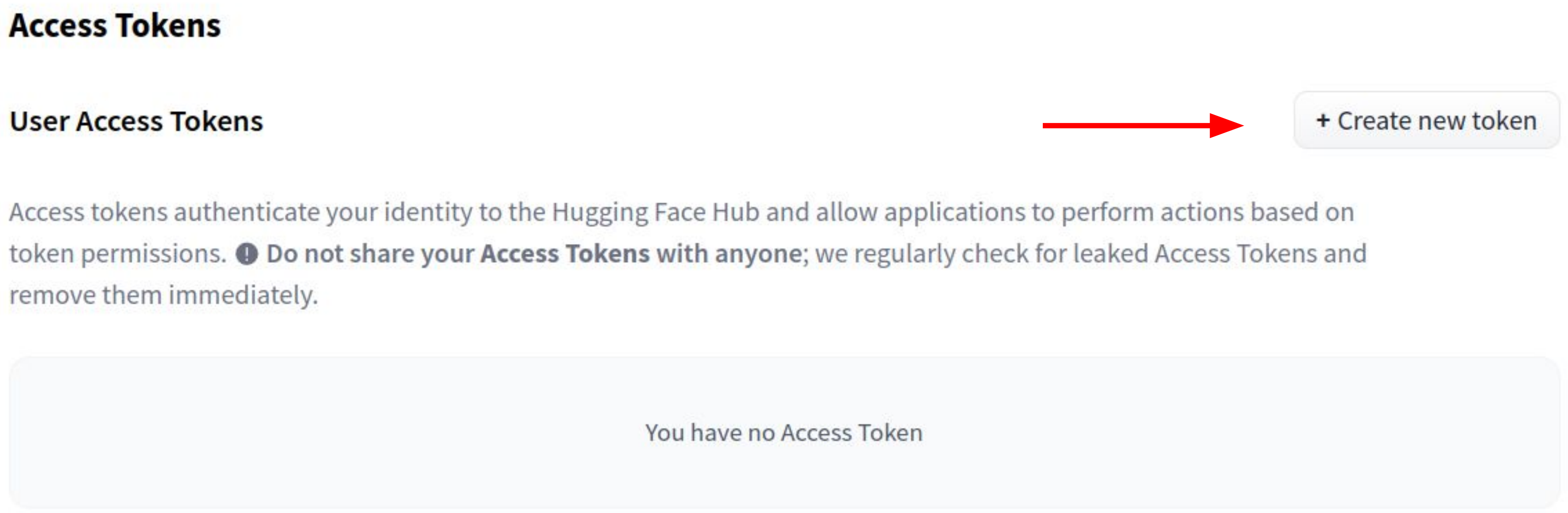
Access Tokens

User Access Tokens

Access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform acti
token permissions. **Do not share your Access Tokens with anyone**; we regularly check for leaked Acce
remove them immediately.

You have no Access Token

Access Token



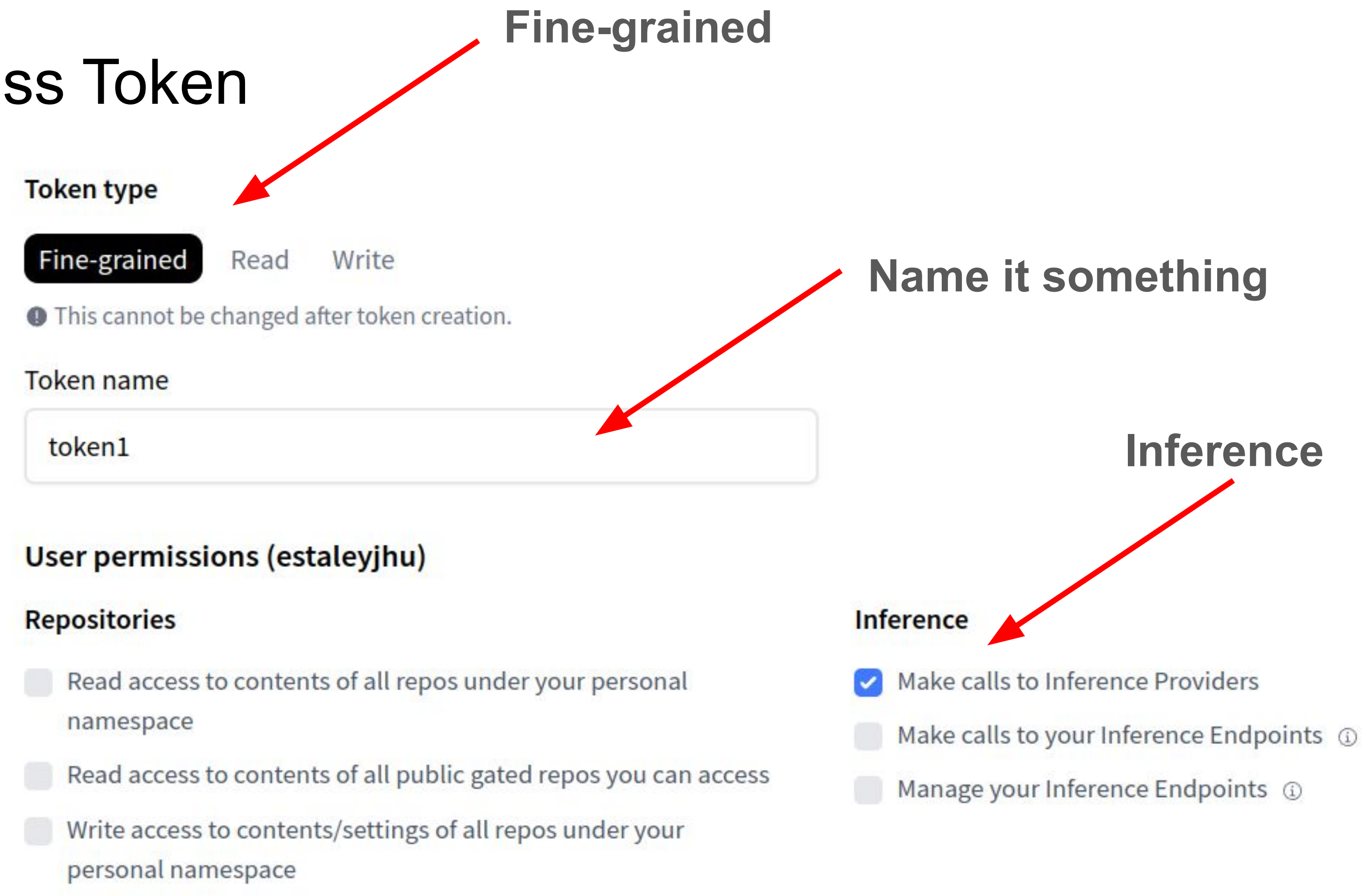
Access Tokens

User Access Tokens

Access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform actions based on token permissions. **Do not share your Access Tokens with anyone**; we regularly check for leaked Access Tokens and remove them immediately.

You have no Access Token

Access Token



Token type

[✓ Fine-grained] Read Write

This cannot be changed after token creation.

Token name

token1

User permissions (estaleyjhu)

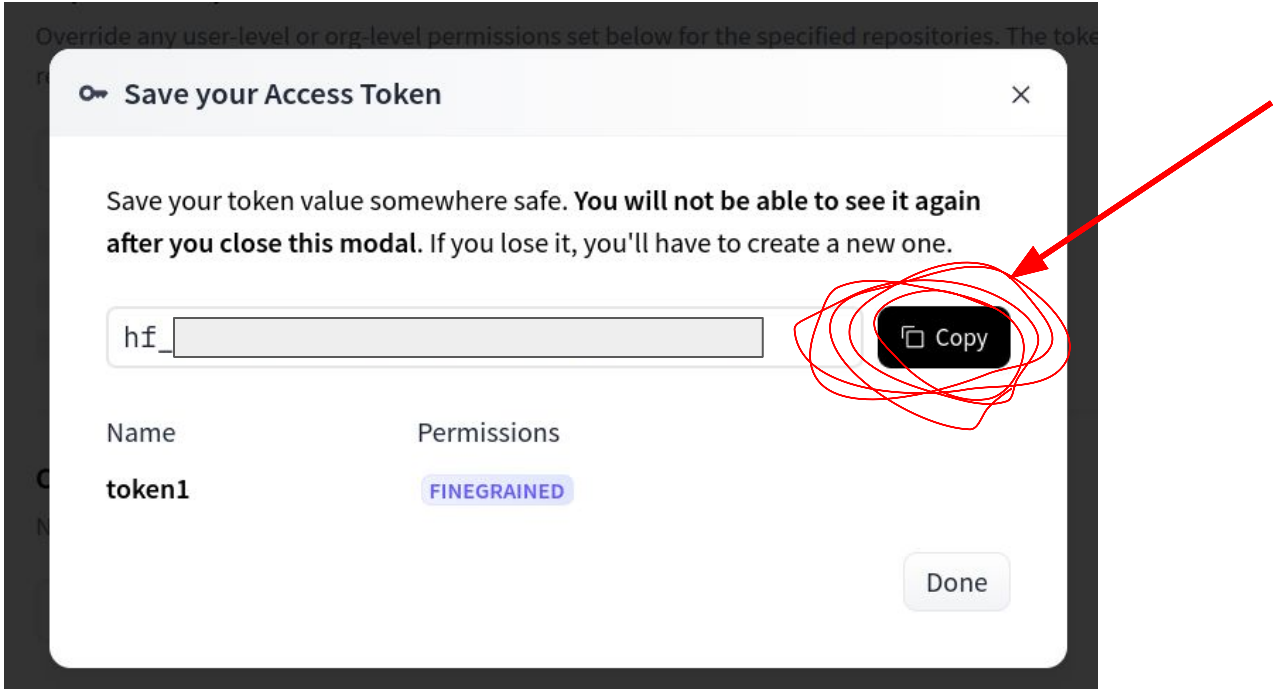
Repositories

- ☐ Read access to contents of all repos under your personal namespace
- ☐ Read access to contents of all public gated repos you can access
- ☐ Write access to contents/settings of all repos under your personal namespace

Inference

- ☑ Make calls to Inference Providers
- ☐ Make calls to your Inference Endpoints
- ☐ Manage your Inference Endpoints

Access Token



Override any user-level or org-level permissions set below for the specified repositories. The toke

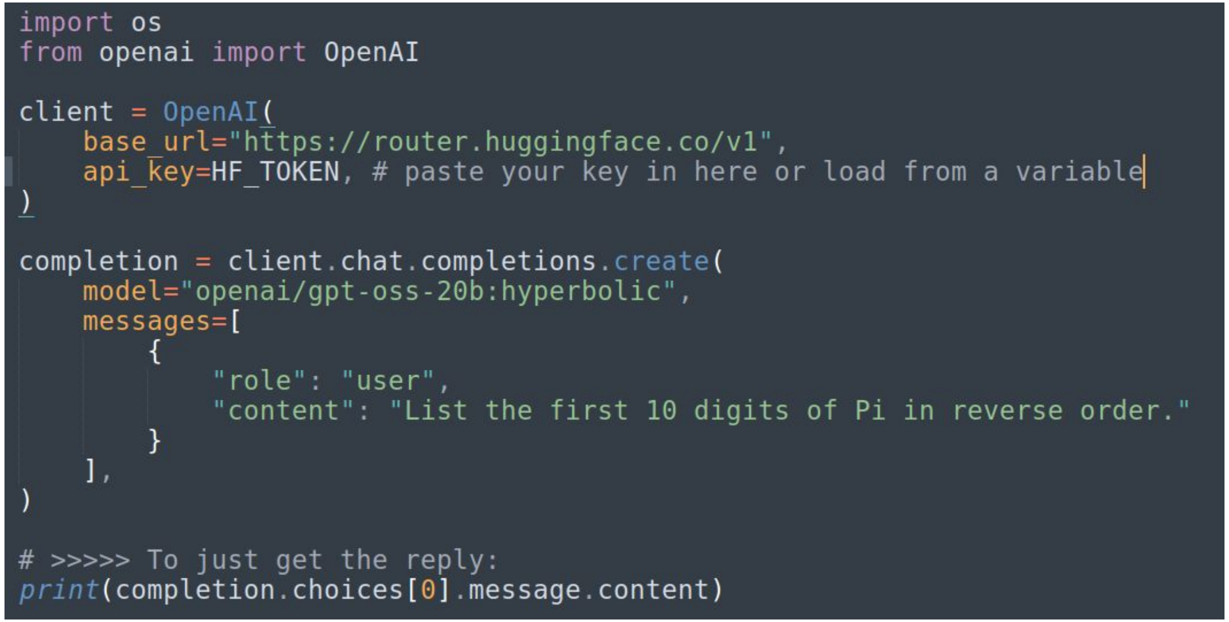
Save your Access Token

Save your token value somewhere safe. **You will not be able to see it again after you close this modal**. If you lose it, you'll have to create a new one.

hf_[redacted]

Name	Permissions
token1	FINEGRAINED

Example Screenshot



```python
import os
from openai import OpenAI

client = OpenAI(
    base_url="https://router.huggingface.co/v1",
    api_key=HF_TOKEN, # paste your key in here or load from a variable
)

completion = client.chat.completions.create(
    model="openai/gpt-oss-20b:hyperbolic",
    messages=[
        {
            "role": "user",
            "content": "List the first 10 digits of Pi in reverse order."
        }
    ],
)

# >>>>> To just get the reply:
print(completion.choices[0].message.content)
```

Note: This costs about $0.00005, which is 0.05% of your free quota on HF. As long as you do not ask for a novel or stick this inside some sort of loop, you should be fine.