ASSIGNMENT 6.1 - PRE-TRAINED GENERATIVE MODEL USAGE

Pre-trained generative models are powerful tools that can be accessed via APIs to perform tasks like tex Add text cell ı, summarization, translation, and more. Here are some popular pre-trained generative models and their respective APIs: Below are some of the available PRe-trained Generative models and APIs.

---

1. OpenAI GPT (Generative Pre-trained Transformer)

- **API**: OpenAI API
- **Models**: GPT-3, GPT-4
- **Use Cases**:
  - Text generation
  - Summarization
  - Question answering
  - Code generation
- **Advantages**:
  - State-of-the-art performance on natural language tasks.
  - Easy integration via REST API.

---

2. Hugging Face Transformers

- **API**: Hugging Face Inference API
- **Models**:
  - BART (e.g., `facebook/bart-large-cnn`) for summarization.
  - T5 (Text-to-Text Transfer Transformer) for text-to-text tasks.
  - GPT-2 and GPT-Neo for text generation.
- **Use Cases**:
  - Summarization
  - Translation
  - Text classification
- **Advantages**:
  - Open-source models with a wide variety of options.
  - Hosted inference API for easy access.

---

3. Google PaLM (Pathways Language Model)

- **API**: Google Cloud Generative AI
- **Models**: PaLM 2

- **Use Cases**:
  - Text generation and summarization.
  - Chatbots and conversational AI.
  - Code generation and debugging.

Add text cell

- **Advantages**:
  - High performance on multilingual and multimodal tasks.
  - Seamless integration with Google Cloud services.

---

4. Cohere

- **API**: Cohere API
- **Models**:
  - Generate for text generation tasks.
- **Use Cases**:
  - Content creation
  - Semantic search
  - Text summarization
- **Advantages**:
  - Focused on enterprise use cases with fine-tuning options.

---

5. Anthropic Claude

- **API**: Anthropic Claude API
- **Models**: Claude (Claude Instant, Claude Pro)
- **Use Cases**:
  - Conversational AI
  - Document summarization
  - Ethical AI applications
- **Advantages**:
  - Designed with safety and ethical considerations in mind.

---

6. Microsoft Azure OpenAI Service

- **API**: Azure OpenAI Service
- **Models**: GPT models (via OpenAI partnership)
- **Use Cases**:
  - Enterprise-level text generation and summarization.
  - Integration with Microsoft services like Power BI and Azure Cognitive Search.

- **Advantages**:
  - Scalable infrastructure for enterprise needs.

---

7. IBM Watson Natural Language Understanding

Add text cell

- **API**: IBM Watson NLU
- **Models**: Custom Watson NLP models
- **Use Cases**:
  - Sentiment analysis
  - Keyword extraction
  - Summarization
- **Advantages**:
  - Strong focus on enterprise analytics and insights.

Advantages of Pre-trained Generative Models Accessed via APIs

1. **Reduced Development Time**
   - Leverage existing, powerful models without the need for extensive training from scratch, saving significant time and resources.
2. **Accessibility and Scalability**
   - APIs provide easy access to these models, abstracting away complex infrastructure and allowing for scalable usage as needed.
3. **Cost-Effectiveness**
   - Avoid the high costs associated with training and hosting large models; pay-as-you-go pricing for API usage makes it more affordable.
4. **State-of-the-Art Performance**
   - Benefit from models trained on massive datasets, often achieving superior performance compared to custom-trained models, especially for complex tasks.
5. **Focus on Application Logic**
   - Concentrate on building your application's specific features rather than dealing with the intricacies of model training and deployment.
6. **Continuous Improvement**
   - API providers often update their models, ensuring you have access to the latest advancements in generative AI.
7. **Simplified Integration**
   - APIs offer a standardized and straightforward way to integrate powerful generative capabilities into your existing workflows and applications.

STEP BY STEP PROCEDURE TO SETUP ENVIRONMENT

1. Uninstall unsupported versions of PyTorch, TorchVision, and Torchaudio from the Development environment

Add text cell

2. Installs PyTorch, TorchVision, and Torchaudio using the specified CUDA version (cu118) in this case

CUDA And PyTorch:

CUDA is NVIDIA's parallel computing platform that allows PyTorch to leverage GPU acceleration.

The installed PyTorch must match the installed CUDA version on the environment for GPU support to work correctly.

```
# Force reinstall PyTorch with correct CUDA version (Colab specific)
!pip uninstall torch torchvision torchaudio -y
!pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu
```

```
Found existing installation: torch 2.5.1+cu124
Uninstalling torch-2.5.1+cu124:
  Successfully uninstalled torch-2.5.1+cu124
Found existing installation: torchvision 0.20.1+cu124
Uninstalling torchvision-0.20.1+cu124:
  Successfully uninstalled torchvision-0.20.1+cu124
Found existing installation: torchaudio 2.5.1+cu124
Uninstalling torchaudio-2.5.1+cu124:
  Successfully uninstalled torchaudio-2.5.1+cu124
Looking in indexes: https://download.pytorch.org/whl/cu118
Collecting torch
  Downloading https://download.pytorch.org/whl/cu118/torch-2.6.0%2Bcu118-cp311-cp311-
Collecting torchvision
  Downloading https://download.pytorch.org/whl/cu118/torchvision-0.21.0%2Bcu118-cp311
Collecting torchaudio
  Downloading https://download.pytorch.org/whl/cu118/torchaudio-2.6.0%2Bcu118-cp311-c
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: typing-extensions>=4.10.0 in /usr/local/lib/python3.11
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (fro
Collecting nvidia-cuda-nvrtc-cu11==11.8.89 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cuda_nvrtc_cu11-11.8.89-p
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.2/23.2 MB 88.3 MB/s eta 0:00:00
Collecting nvidia-cuda-runtime-cu11==11.8.89 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cuda_runtime_cu11-11.8.89
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 875.6/875.6 kB 55.2 MB/s eta 0:00:00
Collecting nvidia-cuda-cupti-cu11==11.8.87 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cuda_cupti_cu11-11.8.87-p
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 106.1 MB/s eta 0:00:00
Collecting nvidia-cudnn-cu11==9.1.0.70 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cudnn_cu11-9.1.0.70-py3-n
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 663.9/663.9 MB 2.8 MB/s eta 0:00:00
Collecting nvidia-cublas-cu11==11.11.3.6 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cublas_cu11-11.11.3.6-py3
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 417.9/417.9 MB 3.7 MB/s eta 0:00:00
Collecting nvidia-cufft-cu11==10.9.0.58 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cufft_cu11-10.9.0.58-py3-
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 168.4/168.4 MB 6.8 MB/s eta 0:00:00
Collecting nvidia-curand-cu11==10.3.0.86 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_curand_cu11-10.3.0.86-py3
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.1/58.1 MB 13.8 MB/s eta 0:00:00
Collecting nvidia-cusolver-cu11==11.4.1.48 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cusolver_cu11-11.4.1.48-p
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 128.2/128.2 MB 7.4 MB/s eta 0:00:00
Collecting nvidia-cusparse-cu11==11.7.5.86 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_cusparse_cu11-11.7.5.86-p
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 204.1/204.1 MB 6.1 MB/s eta 0:00:00
Collecting nvidia-nccl-cu11==2.21.5 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_nccl_cu11-2.21.5-py3-none
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 147.8/147.8 MB 7.7 MB/s eta 0:00:00
Collecting nvidia-nvtx-cu11==11.8.86 (from torch)
  Downloading https://download.pytorch.org/whl/cu118/nvidia_nvtx_cu11-11.8.86-py3-non
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 9.6 MB/s eta 0:00:00
Collecting triton==3.2.0 (from torch)
  Downloading https://download.pytorch.org/whl/triton-3.2.0-cp311-cp311-manylinux_2_2
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.11/dis
```

Add text cell

```
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-pack
Downloading https://download.pytorch.org/whl/cu118/torch-2.6.0%2Bcu118-cp311-cp311-li
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 848.7/848.7 MB 1.8 MB/s eta 0:00:00
Downloading https://download.pytorch.org/whl/triton-3.2.0-cp311-cp311-manylinux_2_27_
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.7/166.7 MB 6.8 MB/s eta 0:00:00
Downloading https://download.pytorch.org/whl/cu118/torchvision-0.21.0%2Bcu118-cp311-c
   ━━━━ Add text cell ━━━━━━━━━━━━━━━━━━━━━━ 6.5/6.5 MB 101.5 MB/s eta 0:00:00
Downloading https://download.pytorch.org/whl/cu118/torchaudio-2.6.0%2Bcu118-cp311-cp3
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.3/3.3 MB 91.6 MB/s eta 0:00:00
Installing collected packages: triton, nvidia-nvtx-cu11, nvidia-nccl-cu11, nvidia-cus
  Attempting uninstall: triton
    Found existing installation: triton 3.1.0
    Uninstalling triton-3.1.0:
      Successfully uninstalled triton-3.1.0
ERROR: pip's dependency resolver does not currently take into account all the package
fastai 2.7.18 requires torch<2.6,>=1.10, but you have torch 2.6.0+cu118 which is inco
Successfully installed nvidia-cublas-cu11-11.11.3.6 nvidia-cuda-cupti-cu11-11.8.87 nv
WARNING: The following packages were previously imported in this runtime:
  [torchgen]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

An Example usage of Pre-Trained Model API - EleutherAI/gpt-neo-2.7B is a large language model

```
# Import the pipeline function, which provides a high-level, easy-to-use interface for va
from transformers import pipeline

# Ensure the transformers library is installed
!pip install transformers

# Hugging Face Transformers pipeline for text generation
# EleutherAI/gpt-neo-2.7B is a large language model from the GPT-Neo family.
# It has 2.7 billion parameters. This line downloads the model if it's not already cached
# Calls the pipeline with the prompt and other parameters
generator = pipeline('text-generation', model='EleutherAI/gpt-neo-2.7B')

# Prompt the model to generate a 2-line description of the Pythagorean theorem
prompt = "Describe the Pythagorean theorem in 2 lines."

# The generator function returns a list of dictionaries. Since we only asked for one sequ
response = generator(prompt, max_length=50, num_return_sequences=1, truncation=True)[0]['

#Print the Prompt respnse
print(f"Description of Pythagorean Theorem:\n{response}")
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: huggingface-hub<1.0,>=0.26.0 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages
Require Add text cell ly satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/di
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarnin
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public mo
  warnings.warn(
```

config.json: 100% 1.46k/1.46k [00:00<00:00, 82.7kB/s]

model.safetensors: 100% 10.7G/10.7G [01:00<00:00, 202MB/s]

tokenizer_config.json: 100% 200/200 [00:00<00:00, 13.9kB/s]

vocab.json: 100% 798k/798k [00:00<00:00, 1.28MB/s]

merges.txt: 100% 456k/456k [00:00<00:00, 5.80MB/s]

special_tokens_map.json: 100% 90.0/90.0 [00:00<00:00, 6.61kB/s]

```
Device set to use cuda:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Description of Pythagorean Theorem:
Describe the Pythagorean theorem in 2 lines.

A Pythagorean Square is a square with equal sides.

1

2

3

12

1x1

x2-x1
```

An Example usage of Pre-Trained Model API - GPT2:

Prompt to the AI model: What is the synonym of Euphoric?

```
from transformers import pipeline
from langchain.llms import HuggingFacePipeline

os.environ[" Add text cell HUB_API_TOKEN"] = "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"

# Initialize the pipeline for text generation
generator = pipeline('text-generation', model='gpt2')  # Or a smaller model if needed

# Create a LangChain wrapper around the Hugging Face pipeline
llm = HuggingFacePipeline(pipeline=generator)

# Prompt the model to generate a description of the Pythagorean theorem
prompt = "What is synonym of Euphoric?"

response = llm(prompt)  # Use the LangChain LLM to get the response

print(response)
```

```
Device set to use cuda:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
What is synonym of Euphoric? Is it possible? And yet when we talk of Euphoric we mean

The Euphoric [is a Greek Greek adjective meaning "that of the sun"] — where the sun i
```

Install LangChain Community

langchain_community is a collection of community-contributed integrations for the LangChain library. LangChain is a framework for developing applications powered by language models. The langchain_community package often provides integrations with specific LLMs, vector databases, or other tools that are not included in the core langchain package.

```
!pip install langchain_community
```

```
Collecting langchain_community
  Downloading langchain_community-0.3.17-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/lib/pyt
Requirement already satisfied: langchain<1.0.0,>=0.3.18 in /usr/local/lib/python3
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3.11/dis
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/python3.11/
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/lib/pytho
Collecting dataclasses-json<0.7,>=0.5.7 (from langchain_community)
  Downloading dataclasses_json-0.6.7-py3-none-any.whl.metadata (25 kB)
Collecting pydantic-settings<3.0.0,>=2.4.0 (from langchain_community)
  Downloading pydantic_settings-2.7.1-py3-none-any.whl.metadata (3.5 kB)
Requirement already satisfied: langsmith<0.4,>=0.1.125 in /usr/local/lib/python3.1
Collecting httpx-sse<1.0.0,>=0.4.0 (from langchain_community)
  Downloading httpx_sse-0.4.0-py3-none-any.whl.metadata (9.0 kB)
```

```
Requirement already satisfied: numpy<2,>=1.26.4 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.1
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/di
Require Add text cell ly satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist
Collecting marshmallow<4.0.0,>=3.18.0 (from dataclasses-json<0.7,>=0.5.7->langchai
  Downloading marshmallow-3.26.1-py3-none-any.whl.metadata (7.3 kB)
Collecting typing-inspect<1,>=0.4.0 (from dataclasses-json<0.7,>=0.5.7->langchain_
  Downloading typing_inspect-0.9.0-py3-none-any.whl.metadata (1.5 kB)
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /usr/loca
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/python3.11/d
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python3.11/di
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/python3.11
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/python3.11/
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/local/lib/p
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/python3
Collecting python-dotenv>=0.21.0 (from pydantic-settings<3.0.0,>=2.4.0->langchain_
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-packages (4
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/
Collecting mypy-extensions>=0.3.0 (from typing-inspect<1,>=0.4.0->dataclasses-jsor
  Downloading mypy_extensions-1.0.0-py3-none-any.whl.metadata (1.1 kB)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-pack
Downloading langchain_community-0.3.17-py3-none-any.whl (2.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 87.0 MB/s eta 0:00:00
Downloading dataclasses_json-0.6.7-py3-none-any.whl (28 kB)
```

SET UP TEXT GENERATION PIPELINE USING facebook/blenderbot-1B-distill MODEL

This code sets up a text generation pipeline using the facebook/blenderbot-1B-distill model and then wraps it with LangChain's HuggingFacePipeline for easier integration with LangChain workflows.

```
# Import the LangChain class to wrap Hugging Face pipelines.
from langchain.llms import HuggingFacePipeline

#Imports the PyTorch library (used by the model)
```

```
import torch

#Import necessary classes from the Hugging Face transformers library:
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline, AutoModelForSeq2S

model_id = 'facebook/blenderbot-1B-distill'
tokenizer = AutoTokenizer.from_pretrained(model_id) #  For loading the correct tokenizer

# For loading sequence-to-sequence language models (used here). It loads the pretrained m
model = AutoModelForSeq2SeqLM.from_pretrained(model_id)

#Creates the text2text-generation pipeline
pipe = pipeline(
    "text2text-generation", #Specifies the task as text-to-text generation (suitable for
    model=model, # passes the loaded model to pipeline
    tokenizer=tokenizer, #Passes the loaded tokenizer.
    max_length=100 #Sets the maximum length of the generated output to 100 tokens
)

# Creates a LangChain HuggingFacePipeline object (local_llm) by wrapping the Hugging Face
# This makes the model easily usable within LangChain chains and agents. local_llm is now
local_llm = HuggingFacePipeline(pipeline=pipe)
```

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

Add text cell

USE LANGCHAIN FRAMEWORK FOR BUILDING LLM APPLICATIONS

Use LangChain, a framework for building applications with large language models (LLMs), to answer a question using a pre-existing language model represented by local_llm

```
# Import necessary tools from LangChain
from langchain import PromptTemplate, HuggingFaceHub, LLMChain #LLMChain - For creating a

# This defines a template for how the question should be formatted when sent to the LLM.
# It's like a pre-written message with a blank space{question}` where the actual question
# The "Let's think step by step" part encourages the LLM to provide a more detailed reaso
template = """Question: {question}

Answer: Let's think step by step."""

# Create a PromptTemplate object from the template string. It specifies that the template
prompt = PromptTemplate(template=template, input_variables=["question"])

# creates an LLMChain object, which connects the prompt template with the local_llm (your
# local_llm would have been defined earlier in your code (not shown here) and is assumed
llm_chain = LLMChain(prompt=prompt,
                     llm=local_llm
                     )

question = "what is Artificial Intelligence"
```

```
print(llm_chain.run(question))
```

```
I'm not sure what you mean by that, but I do know that artificial intelligence is a
```

Add text cell

QUESTION TO LLM using API: "Which country is best for growing wine?"

```
question = "Which country is best for growing wine?"

print(llm_chain.run(question))
```

```
I would have to say the United States. It is the world's largest producer of wine.
```

QUESTION TO LLM USING API: Which country is largest in the world?

```
question = "Which country is largest in the world?"

print(llm_chain.run(question))
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Question: What country is largest in the world?

Answer: Let's think step by step. Russia, Saudi Arabia, Turkey is also the largest, a
```

QUESTION TO LLM USING API: Which is the most populous country in the world?

```
from langchain import PromptTemplate, HuggingFaceHub, LLMChain

template = """Question: {question}

Answer: Let's think step by step."""

prompt = PromptTemplate(template=template, input_variables=["question"]

llm_chain = LLMChain(prompt=prompt,
                     llm=local_llm
                     )

question = "Which is the most populous country in world?"
```

```
You seem to be using the pipelines sequentially on GPU. In order to maximize efficien
 India is the most populous country in the world with a population of over 1.3 billio
```

Add text cell

QUESTION TO LLM USING API: What do you see in the night sky?

```
question = "What do you see in the night sky?"

print(llm_chain.run(question))
```

```
 I see a lot of stars when I'm out at night. Do you see many stars?
```

QUESTION TO LLM USING API: What do you feel when you see a moon?

```
question = "What do you feel when you see a moon?"

print(llm_chain.run(question))
```

```
 That's a good way to look at it. I feel the same way when I see the moon.
```

QUESTION TO LLM USING API: What is the most beautiful place in the world?

```
question = "What is the most beautiful place in the world?"

print(llm_chain.run(question))
```

```
 That is a good question. I would have to say the ocean. I love the water.
```