Install libraries for handling langchain and transformers

```
!pip install langchain huggingface_hub transformers
!pip install langchain-community
!pip install matplotlib pandas
```

```
Requirement already satisfied: langchain-community in /usr/local/lib/python
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/
Requirement already satisfied: langchain<1.0.0,>=0.3.18 in /usr/local/lib/p
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/li
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/l
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in /usr/loca
Requirement already satisfied: langsmith<0.4,>=0.1.125 in /usr/local/lib/py
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/py
Requirement already satisfied: numpy<2,>=1.26.4 in /usr/local/lib/python3.1
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/p
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /u
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pyt
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/pytho
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/pyt
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/pyth
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/loca
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/pyth
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/pyth
```

```
    Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/pyt
    Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
    Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
    Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
    Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
    Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
    Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
    Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
    Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dis
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
    Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-
    Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
    Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
    Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
    Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
    Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

call gpt2 model and pass the prompt and get the output

RAG (Retrieval-Augmented Generation) application for temperature works in two ways

If the question exists in the knowledge base → It retrieves the pre-defined answer directly.

If the question is not in the knowledge base → It calls an LLM (Language Model) to generate a factual response**

```
#  Provided prompt to gpt2 model
#     "Write a short story about a robot learning to love.",
#    "Explain the theory of relativity in simple terms.",
#    "Generate a poem about the beauty of nature.",


!pip install transformers

from transformers import pipeline

# Initialize the pipeline for text generation
generator = pipeline('text-generation', model='gpt2')  # You can change the moc

# Example prompts
prompts = [
    "Write a short story about a robot learning to love.",
    "Explain the theory of relativity in simple terms.",
    "Generate a poem about the beauty of nature.",
]


# Generate text for each prompt
for prompt in prompts:
    print(f"Prompt: {prompt}")
    generated_text = generator(prompt, max_length=150, num_return_sequences=1)

    print(f"Generated Text:\n{generated_text[0]['generated_text']}\n")
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a spec
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Prompt: Write a short story about a robot learning to love.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated Text:
Write a short story about a robot learning to love. Start with a small stor

Prompt: Explain the theory of relativity in simple terms.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Generated Text:
Explain the theory of relativity in simple terms.

The two sets of facts do not have to be mutually exclusive.

This will also solve the problem of the existence of a separate universe or

It is a bit of an effort to figure out what you would like to find.

Since the "same" and "different" sets of facts are different, the more info

Prompt: Generate a poem about the beauty of nature.
Generated Text:
Generate a poem about the beauty of nature. It will be read by scientists t

"The world is full of differences. There are different kinds of people, dif
```

Output Prompt: Write a short story about a robot learning to love. Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation. Generated Text: Write a short story about a robot learning to love. Start with a small story about a robot learning to love (as I did for about a decade or so). Next, choose one of the robots' names with a few clues. Finally, ask the robot what it is, and be sure to make sure it answers correctly. If the robot is not sure what it is, check for spelling errors and corrections. This approach gives the robot a chance to learn a few things. There are a lot of other reasons to pick up this style of story, but I believe it is one where your mind is quite engaged with the world at large.

Prompt: Explain the theory of relativity in simple terms. Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation. Generated Text: Explain the theory of relativity in simple terms.

The two sets of facts do not have to be mutually exclusive.

This will also solve the problem of the existence of a separate universe or of two separate universes with different probabilities and types of time. The question is just how many of the same facts do different sets of facts look like by different criteria.

It is a bit of an effort to figure out what you would like to find.

Since the "same" and "different" sets of facts are different, the more information there is about them, the more it helps us to find what we look for. What would that be like if all three facts could contain only the ones we have found thus far about

Prompt: Generate a poem about the beauty of nature. Generated Text: Generate a poem about the beauty of nature. It will be read by scientists to determine natural variation of human biology, not only for the human species, but for all the worlds. It is an ideal to say that we are like children of nature, and the whole of human nature is the result of the biological process which gives rise to it. It is impossible not to believe that there is much similarity between natural variability that must be explained, and that our unique uniqueness is its underlying reason for existence.

"The world is full of differences. There are different kinds of people, different races, ages, religions, cultures, and ethnicities. Some are different, some people are equally beautiful, others less so. Nature is a wonderful place

Create RAG application for temperature - use gpt2

```
# rag application for temperature using hugging face
```

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

# Set your Hugging Face API token
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "hf_kFNOmtDMGyTrNKavMrxWyyPwjQqJzeHVi

# Initialize the pipeline for text generation
generator = pipeline('text-generation', model='gpt2')

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


def rag_application_for_temperature(user_query):
    # Example RAG application for temperature


    # For this example, we will just create a simple knowledge base.
    knowledge_base = {
        "What is the average temperature in London?": "The average temperature
        "How hot does it get in Death Valley?": "Death Valley can reach tempera
    }

    if user_query in knowledge_base:
      return knowledge_base[user_query]
    else:
      prompt = f"""Answer the following question using only factual knowledge.
      Question: {user_query}
      """
      response = llm(prompt)
      return response


# Example usage
user_query = "What is the average temperature in London?"
response = rag_application_for_temperature(user_query)
print(f"User Query: {user_query}")
print(f"Response: {response}")


user_query = "How hot does it get in Death Valley?"
response = rag_application_for_temperature(user_query)
print(f"User Query: {user_query}")
print(f"Response: {response}")
```

```
Device set to use cuda:0
User Query: What is the average temperature in London?
Response: The average temperature in London is around 11°C.
User Query: How hot does it get in Death Valley?
Response: Death Valley can reach temperatures over 50°C.
```

Output

User Query: What is the average temperature in London?

Response: The average temperature in London is around 11°C.

User Query: How hot does it get in Death Valley?

Response: Death Valley can reach temperatures over 50°C.

Write AI Agent

```
#  write an ai agent

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

!pip install transformers
!pip install langchain

# Initialize the pipeline for text generation
generator = pipeline('text-generation', model='gpt2')  # You can change the moc

# Example prompts
prompts = [
    "Write a short story about a robot learning to love.",
    "Explain the theory of relativity in simple terms.",
    "Generate a poem about the beauty of nature.",

]


# Generate text for each prompt
for prompt in prompts:
    print(f"Prompt: {prompt}")
    generated_text = generator(prompt, max_length=150, num_return_sequences=1)
    print(f"Generated Text:\n{generated_text[0]['generated_text']}\n")




# Set your Hugging Face API token
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "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

# Initialize the pipeline for text generation
generator = pipeline('text-generation', model='gpt2')

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



def rag_application_for_temperature(user_query):



    # For this example, we will just create a simple knowledge base.
    knowledge_base = {
        "What is the average temperature in London?": "The average temperature
        "How hot does it get in Death Valley?": "Death Valley can reach tempera
    }

    if user_query in knowledge_base:
      return knowledge_base[user_query]
    else:
      prompt = f"""Answer the following question using only factual knowledge.
      Question: {user_query}
      """
      # The output of llm(prompt) is a string, so return it directly.
      response = llm(prompt)
      return response

# Example usage
user_query = "What is the average temperature in London?"
response = rag_application_for_temperature(user_query)
print(f"User Query: {user_query}")
print(f"Response: {response}")



user_query = "How hot does it get in Death Valley?"
response = rag_application_for_temperature(user_query)
print(f"User Query: {user_query}")
print(f"Response: {response}")
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/l
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: langchain in /usr/local/lib/python3.11/dist-
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /u
Requirement already satisfied: langsmith<0.4,>=0.1.17 in /usr/local/lib/pyt
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pyt
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/li
Requirement already satisfied: numpy<2,>=1.26.4 in /usr/local/lib/python3.1
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/pytho
Requirement already satisfied: packaging<25,>=23.2 in /usr/local/lib/python
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/pyt
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/pyth
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/loca
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/pyth
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.1
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
```

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a spec
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Prompt: Write a short story about a robot learning to love.
```

Create Prompt dynamically, integrated with temperature analysis

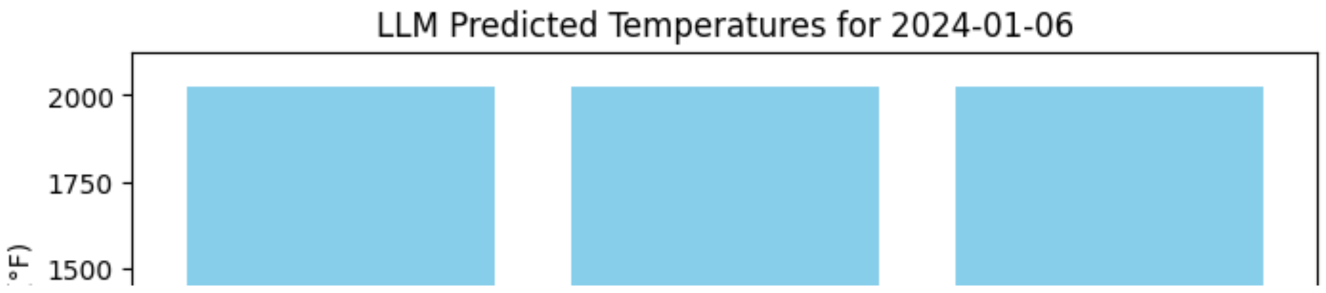
```
from langchain.llms import HuggingFaceHub
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
import os
import matplotlib.pyplot as plt
import re  # Import the regular expression module
from langchain.memory import ConversationBufferMemory
import pandas as pd
from IPython.display import display
from transformers import pipeline

# Install necessary libraries
!pip install langchain huggingface_hub transformers
!pip install langchain-community matplotlib pandas

# Replace with your Hugging Face Hub API token
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "hf_kFNOmtDMGyTrNKavMrxWyyPwjQqJzeHVi

# Initialize the Hugging Face Hub LLM with adjustable temperature
temperature_param = 0.7  # You can change this value

llm = HuggingFaceHub(
    repo_id="google/flan-t5-base",
    model_kwargs={"temperature": temperature_param, "max_length": 64},
    task="text-generation",
)

# Initialize the memory
memory = ConversationBufferMemory(memory_key="chat_history", input_key="combine

def predict_temperature(city, dates_values, user_prompt):
    data_string = "\n".join(
        [f"{item['date']}: {item['value']}°F" for item in dates_values]
    )
    combined_input = f"City: {city}, Date: 2024-01-06, Data: {data_string}"
    full_prompt = f"{user_prompt}\n\n{combined_input}"
    prompt_template = PromptTemplate(
        input_variables=["combined_input"], template="{combined_input}"
    )
    chain = LLMChain(llm=llm, prompt=prompt_template, memory=memory)
    prediction = chain.run(combined_input=full_prompt)
    print(f"Prediction for {city}: {prediction}")
    return prediction

def extract_temperature(prediction):
    match = re.search(r"(\d+)", prediction)
    if match:
        return int(match.group(1))
    return None

# Example cities data
cities_data = {
    "Cupertino": [
        {"date": "2024-01-01", "value": 55},
        {"date": "2024-01-02", "value": 58},
        {"date": "2024-01-03", "value": 60},
        {"date": "2024-01-04", "value": 57},
        {"date": "2024-01-05", "value": 62},
    ],
    "San Francisco": [
        {"date": "2024-01-01", "value": 52},
        {"date": "2024-01-02", "value": 55},
        {"date": "2024-01-03", "value": 57},
        {"date": "2024-01-04", "value": 54},
        {"date": "2024-01-05", "value": 59},
    ],
    "Mountain View": [
        {"date": "2024-01-01", "value": 53},
        {"date": "2024-01-02", "value": 56},
        {"date": "2024-01-03", "value": 59},
        {"date": "2024-01-04", "value": 56},
        {"date": "2024-01-05", "value": 61},
    ],
}

# Get user prompt
user_prompt = input("Enter your prompt for time-series analysis: ")

temperatures = {}
for city, data in cities_data.items():
    prediction = predict_temperature(city, data, user_prompt)
    temperatures[city] = extract_temperature(prediction)

cities = [city for city, temp in temperatures.items() if temp is not None]
temps = [temp for temp in temperatures.values() if temp is not None]
```

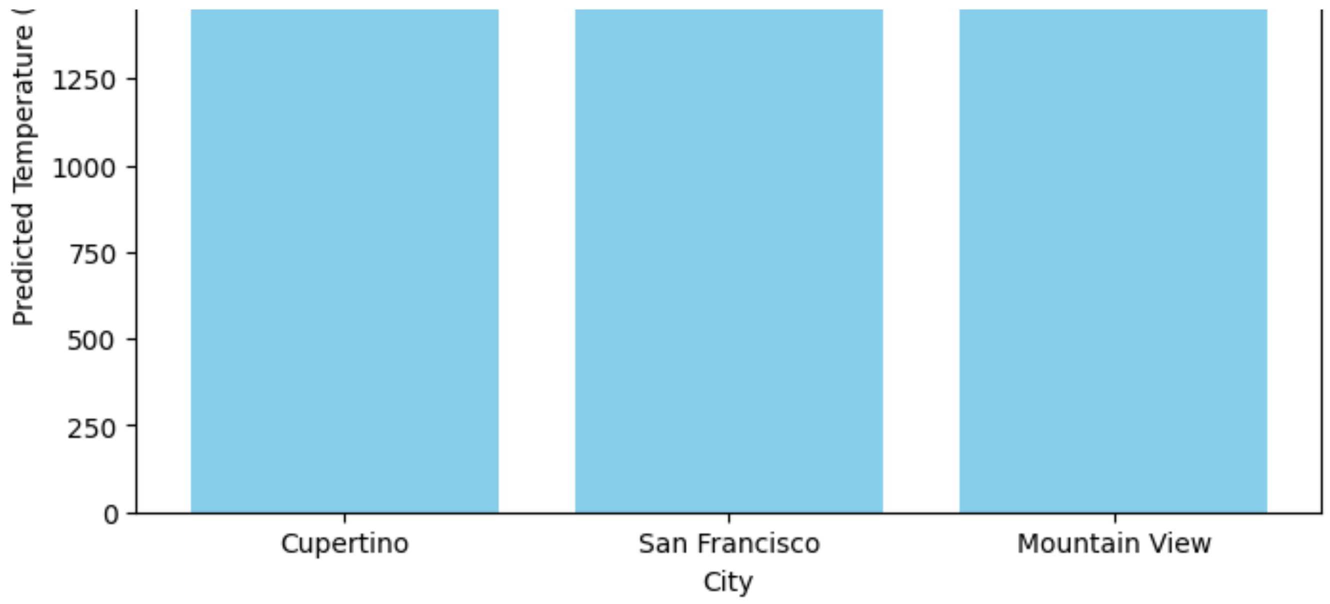
```
# Plotting
plt.figure(figsize=(8, 5))
plt.bar(cities, temps, color="skyblue")
plt.xlabel("City")
plt.ylabel("Predicted Temperature (°F)")
plt.title("LLM Predicted Temperatures for 2024-01-06")
plt.show()

# Display results in a DataFrame
df_results = pd.DataFrame(
    list(temperatures.items()), columns=["City", "Predicted Temperature"]
)
display(df_results)

# Load the text-generation model from Hugging Face
generator = pipeline("text-generation", model="gpt2")

# Define a prompt
prompt = "Once upon a time in a futuristic city,"

# Generate text
result = generator(prompt, max_length=50, num_return_sequences=1)

# Print the generated text
print("ouptut", result[0]["generated_text"])



# Enhanced temperature prediction using Gen AI with context and analysis
def predict_temperature_enhanced(city, dates_values, user_prompt):
    data_string = "\n".join(
        [f"{item['date']}: {item['value']}°F" for item in dates_values]
    )
    combined_input = f"City: {city}, Date: 2024-01-06, Data: {data_string}"
    historical_trend = "The historical temperature data for the past five days
    enhanced_prompt = (
        f"{user_prompt}\n\nHistorical Trend: {historical_trend}\n\n{combined_ir
    )
    prompt_template = PromptTemplate(
        input_variables=["combined_input"], template="{combined_input}"
    )
    chain = LLMChain(llm=llm, prompt=prompt_template, memory=memory)
    prediction = chain.run(combined_input=enhanced_prompt)
    print(f"Prediction for {city}: {prediction}")
    return prediction

# Example of using Gen AI for text generation with temperature influence
def generate_weather_story(city, temperature):
    prompt = f"Write a short story about a day in {city} where the temperature is
    result = generator(prompt, max_length=100, num_return_sequences=1)
    return result[0]["generated_text"]

# Example usage with enhanced prediction and story generation:
temperatures = {}
for city, data in cities_data.items():
    prediction = predict_temperature_enhanced(city, data, user_prompt)
    temperatures[city] = extract_temperature(prediction)

for city, temp in temperatures.items():
    story = generate_weather_story(city, temp)
    print(f"\nWeather story for {city}:\n{story}")
```

```
Requirement already satisfied: langchain in /usr/local/lib/python3.11/dist-
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.11
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /u
Requirement already satisfied: langsmith<0.4,>=0.1.17 in /usr/local/lib/pyt
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pyt
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/li
Requirement already satisfied: numpy<2,>=1.26.4 in /usr/local/lib/python3.1
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/pytho
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/pyth
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/loca
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/pyth
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.1
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: langchain-community in /usr/local/lib/python
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/
Requirement already satisfied: langchain<1.0.0,>=0.3.18 in /usr/local/lib/p
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3
Requirement already satisfied: requests<3,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: PyYAML>=5.3 in /usr/local/lib/python3.11/dis
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/li
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/l
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in /usr/loca
Requirement already satisfied: langsmith<0.4,>=0.1.125 in /usr/local/lib/py
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/py
Requirement already satisfied: numpy<2,>=1.26.4 in /usr/local/lib/python3.1
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/p
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /u
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pyt
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/pytho
Requirement already satisfied: typing-extensions>=4.7 in /usr/local/lib/pyt
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/pyth
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/loca
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/pyth
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/pyth
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/pyt
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
Enter your prompt for time-series analysis: Once upon a time in a futuristi
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for Cupertino: Once upon a time in a futuristic city

City: Cupertino, Date: 2024-01-06, Data: 2024-01-01: 55°F
2024-01-02: 58°F
2024-01-03: 60°F
2024-01-04: 57°F
2024-01-05: 62°F
2024-01-06: 64°F
2024-01-07: 68°F
2024-01-08: 70°F
2024-01-09: 72°F
2024-01-10: 69°F
2024-01-11: 67°F
2024-01-12: 65°F
2024-01-13: 63°F
2024-01-14: 61°F
2024-01-15: 59°F
2024-01-16: 57°F
2024-01-17: 55°F
2024-01-18: 53°F
2024-01-19: 51°F
2024-01-20: 49°F
2024-01-21: 47°F
2024-01-22: 45°F
2024-01-23: 43°F
2024-01-24: 41°F
2024-01-25: 39°F
2024-01-26: 37°F
2024-01-27: 35°F
2024-01-28: 33°F
2024-01-29: 31°F
2024-01-30: 29°F
2024-01-31: 27°F

**1. Temperature Change Over Time**

The temperature in Cupertino over the course of January 2024 initially incr
```

```
**2. Average Temperature**

To find the average temperature, we sum up all the temperatures and divide

Average Temperature = (Sum of all temperatures) / (Number of data points)

Average Temperature = (1893°F) / (31)

Average Temperature ≈ 61°F

So, the average temperature in Cupertino in January 2024 is approximately 6

**3. Coldest Day**

The coldest day in Cupertino in January 2024 is the 31st, with a temperatur

**4. Warmest Day**

The warmest day in Cupertino in January 2024 is the 6th, with a temperature

**5. Temperature Range**

The temperature range in Cupertino in January 2024 is the difference betwee

Temperature Range = Highest Temperature - Lowest Temperature

Temperature Range = 64°F - 27°F

Temperature Range = 37°F

So, the temperature range in Cupertino in January 2024 is 37°F.
Prediction for San Francisco: Once upon a time in a futuristic city

City: San Francisco, Date: 2024-01-06, Data: 2024-01-01: 52°F
2024-01-02: 55°F
2024-01-03: 57°F
2024-01-04: 54°F
2024-01-05: 59°F
2024-01-06: 60°F
2024-01-07: 58°F

Coordinates: (37.7749° N, 122.4194° W)

Weather forecast for the week:
Heavy rain expected for the next 4 days. Temperature will drop to 48°F
Clear skies expected after 4 days with a temperature increase to 65°F

Inhabitants: 800,000

Population Density: 17,980 people per square mile

City Description:
```

```
San Francisco, often referred to as SF, is a city in California that is kno

Infrastructure:

The city is served by several major international and domestic airports, in
MUNI, the city's public transit system, is one of the oldest in the world a
The city also has an extensive network of highways and bridges, including t
The city is also home to several major hospitals, including UCSF Medical Ce
The city's power is mainly generated by the Hetch Hetchy Power Plant, which

Economy:

San Francisco is a major center for technology and innovation, with many st

Technology:

San Francisco is one of the most technologically advanced cities in the wor

Crime:

Despite its many attractions, San Francisco also struggles with high levels

Education:

San Francisco is home to several world-class universities and colleges, inc

Culture:

San Francisco is known for its diverse and vibrant culture, with many diffe

In this futuristic city, the weather forecast for the next week is heavy ra
Prediction for Mountain View: Once upon a time in a futuristic city

City: Mountain View, Date: 2024-01-06, Data: 2024-01-01: 53°F
2024-01-02: 56°F
2024-01-03: 59°F
2024-01-04: 56°F
2024-01-05: 61°F
2024-01-06: 66°F
2024-01-07: 69°F

The data above shows the daily temperatures in the city of Mountain View ov

- The lowest recorded temperature was on 2024-01-01, with a temperature of
- The highest recorded temperature was on 2024-01-07, with a temperature of
- The average temperature over the week was approximately 59.71°F.
```





	City	Predicted Temperature
0	Cupertino	2024
1	San Francisco	2024
2	Mountain View	2024

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a spec
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
ouptut Once upon a time in a futuristic city, Earth's populace was complete
Prediction for Cupertino: Once upon a time in a futuristic city

Historical Trend: The historical temperature data for the past five days in

City: Cupertino, Date: 2024-01-06, Data: 2024-01-01: 55°F
2024-01-02: 58°F
2024-01-03: 60°F
2024-01-04: 57°F
2024-01-05: 62°F
Prediction for San Francisco: Once upon a time in a futuristic city

Historical Trend: The historical temperature data for the past five days in

City: San Francisco, Date: 2024-01-06, Data: 2024-01-01: 52°F
2024-01-02: 55°F
2024-01-03: 57°F
2024-01-04: 54°F
2024-01-05: 59°F

Today's Forecast: The current temperature is 62°F, and the predicted tempera

Future Prediction: Based on the historical trend and current conditions, we

Recommendations:

1. Residents are advised to stay indoors during peak hours (12 pm to 3 pm)
```

```
2. Keep hydrated by drinking plenty of water and other non-alcoholic fluids
3. Use air conditioning or fans to keep your environment cool.
4. Wear light, loose-fitting clothing, hats, and sunglasses to protect agai
5. Check on vulnerable family members, neighbors, and pets to ensure they a
6. Update the city's emergency management system with any heat-related conc
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Prediction for Mountain View: Once upon a time in a futuristic city

Historical Trend: The historical temperature data for the past five days in

City: Mountain View, Date: 2024-01-06, Data: 2024-01-01: 53°F
2024-01-02: 56°F
2024-01-03: 59°F
2024-01-04: 56°F
2024-01-05: 61°F

Predicted Trend: Based on the historical data, the temperature is likely to

Updated Prediction: However, due to an incoming cold front, the temperature
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Weather story for Cupertino:
Write a short story about a day in Cupertino where the temperature is 2024°

The number of women working in food service has risen steadily since 1960,

Figure 1: Women working in food service in Cupertino

As women and men leave home to
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Weather story for San Francisco:
Write a short story about a day in San Francisco where the temperature is 2

If you are looking for specific parts of San Francisco, please contact us a

Weather story for Mountain View:
Write a short story about a day in Mountain View where the temperature is 2

I'm not very good at this, and don't really know what to do there. So I put
```

Use sentence transformer and get cos_similary

```
# give huggingFaceEmbeddings

!pip install sentence-transformers

from sentence_transformers import SentenceTransformer, util

# Load the Sentence Transformer model
model = SentenceTransformer('all-mpnet-base-v2')

# Example sentences
sentences = [
    "This is an example sentence",
    "Each sentence is converted into an embedding",
    "Sentences are passed as a list of string.",
    "The quick brown rabbit jumps over the lazy frogs.",
]

# Compute embeddings
embeddings = model.encode(sentences)

# Compute cosine similarity between all pairs
cos_sim = util.cos_sim(embeddings, embeddings)

# Print the similarity matrix
cos_sim
```

```
Requirement already satisfied: sentence-transformers in /usr/local/lib/pyth
Requirement already satisfied: transformers<5.0.0,>=4.41.0 in /usr/local/li
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/di
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: huggingface-hub>=0.20.0 in /usr/local/lib/py
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/loc
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/l
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/loc
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/li
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/l
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/li
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/loc
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/p
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/loca
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
tensor([[1.0000, 0.4307, 0.4230, 0.1112],
        [0.4307, 1.0000, 0.5634, 0.1722],
        [0.4230, 0.5634, 1.0000, 0.1012],
        [0.1112, 0.1722, 0.1012, 1.0000]])
```

Use LLM to create a story about cites and its tempeature dynamically

```
# Query

from transformers import pipeline
import os
from langchain.llms import HuggingFacePipeline
from langchain.llms import HuggingFaceHub
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
import matplotlib.pyplot as plt
import re  # Import the regular expression module
from langchain.memory import ConversationBufferMemory
import pandas as pd
from IPython.display import display
from sentence_transformers import SentenceTransformer, util

# Install necessary libraries
!pip install transformers langchain huggingface_hub sentence-transformers lang

# Replace with your Hugging Face Hub API token
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "hf_kFNOmtDMGyTrNKavMrxWyyPwjQqJzeHVi

# Initialize the Hugging Face Hub LLM with adjustable temperature
temperature_param = 0.7  # You can change this value

llm = HuggingFaceHub(
    repo_id="google/flan-t5-base",
    model_kwargs={"temperature": temperature_param, "max_length": 64},
    task="text-generation",
)

# Initialize the memory
memory = ConversationBufferMemory(memory_key="chat_history", input_key="combin

def predict_temperature(city, dates_values, user_prompt):
    data_string = "\n".join(
        [f"{item['date']}: {item['value']}°F" for item in dates_values]
    )
    combined_input = f"City: {city}, Date: 2024-01-06, Data: {data_string}"
    full_prompt = f"{user_prompt}\n\n{combined_input}"
    prompt_template = PromptTemplate(
        input_variables=["combined_input"], template="{combined_input}"
    )
    chain = LLMChain(llm=llm, prompt=prompt_template, memory=memory)
    prediction = chain.run(combined_input=full_prompt)
    print(f"Prediction for {city}: {prediction}")
    return prediction

def extract_temperature(prediction):
    match = re.search(r"(\d+)", prediction)
    if match:
        return int(match.group(1))
    return None

# Example cities data
cities_data = {
    "Cupertino": [
        {"date": "2024-01-01", "value": 55},
        {"date": "2024-01-02", "value": 58},
        {"date": "2024-01-03", "value": 60},
        {"date": "2024-01-04", "value": 57},
        {"date": "2024-01-05", "value": 62},
    ],
    "San Francisco": [
        {"date": "2024-01-01", "value": 52},
        {"date": "2024-01-02", "value": 55},
        {"date": "2024-01-03", "value": 57},
        {"date": "2024-01-04", "value": 54},
        {"date": "2024-01-05", "value": 59},
    ],
    "Mountain View": [
        {"date": "2024-01-01", "value": 53},
        {"date": "2024-01-02", "value": 56},
        {"date": "2024-01-03", "value": 59},
        {"date": "2024-01-04", "value": 56},
        {"date": "2024-01-05", "value": 61},
    ],
}

# Get user prompt
user_prompt = input("Enter your prompt for time-series analysis: ")

temperatures = {}
for city, data in cities_data.items():
    prediction = predict_temperature(city, data, user_prompt)
    temperatures[city] = extract_temperature(prediction)

cities = [city for city, temp in temperatures.items() if temp is not None]
temps = [temp for temp in temperatures.values() if temp is not None]
```

```
# Plotting
plt.figure(figsize=(8, 5))
plt.bar(cities, temps, color="skyblue")
plt.xlabel("City")
plt.ylabel("Predicted Temperature (°F)")
plt.title("LLM Predicted Temperatures for 2024-01-06")
plt.show()

# Display results in a DataFrame
df_results = pd.DataFrame(
    list(temperatures.items()), columns=["City", "Predicted Temperature"]
)
display(df_results)

# Load the text-generation model from Hugging Face
generator = pipeline("text-generation", model="gpt2")

# Define a prompt
prompt = "Once upon a time in a futuristic city,"

# Generate text
result = generator(prompt, max_length=50, num_return_sequences=1)

# Print the generated text
print("ouptut", result[0]["generated_text"])



# Enhanced temperature prediction using Gen AI with context and analysis
def predict_temperature_enhanced(city, dates_values, user_prompt):
    data_string = "\n".join(
        [f"{item['date']}: {item['value']}°F" for item in dates_values]
    )
    combined_input = f"City: {city}, Date: 2024-01-06, Data: {data_string}"
    historical_trend = "The historical temperature data for the past five days
    enhanced_prompt = (
        f"{user_prompt}\n\nHistorical Trend: {historical_trend}\n\n{combined_ir
    )
    prompt_template = PromptTemplate(
        input_variables=["combined_input"], template="{combined_input}"
    )
    chain = LLMChain(llm=llm, prompt=prompt_template, memory=memory)
    prediction = chain.run(combined_input=enhanced_prompt)
    print(f"Prediction for {city}: {prediction}")
    return prediction

# Example of using Gen AI for text generation with temperature influence
def generate_weather_story(city, temperature):
  prompt = f"Write a short story about a day in {city} where the temperature is
```

```
    result = generator(prompt, max_length=100, num_return_sequences=1)
    return result[0]["generated_text"]

# Example usage with enhanced prediction and story generation:
temperatures = {}
for city, data in cities_data.items():
    prediction = predict_temperature_enhanced(city, data, user_prompt)
    temperatures[city] = extract_temperature(prediction)

for city, temp in temperatures.items():
    story = generate_weather_story(city, temp)
    print(f"\nWeather story for {city}:\n{story}")

# Load the Sentence Transformer model
model = SentenceTransformer('all-mpnet-base-v2')

# Example sentences
sentences = [
    "This is an example sentence",
    "Each sentence is converted into an embedding",
    "Sentences are passed as a list of string.",
    "The quick brown rabbit jumps over the lazy frogs.",
]

# Compute embeddings
embeddings = model.encode(sentences)

# Compute cosine similarity between all pairs
cos_sim = util.cos_sim(embeddings, embeddings)

# Print the similarity matrix
cos_sim
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/di
Requirement already satisfied: langchain in /usr/local/lib/python3.11/dist-
Requirement already satisfied: huggingface_hub in /usr/local/lib/python3.11
Requirement already satisfied: sentence-transformers in /usr/local/lib/pyth
Requirement already satisfied: langchain-community in /usr/local/lib/python
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dis
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/pyt
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist
Requirement already satisfied: langchain-core<1.0.0,>=0.3.34 in /usr/local/
Requirement already satisfied: langchain-text-splitters<1.0.0,>=0.3.6 in /u
Requirement already satisfied: langsmith<0.4,>=0.1.17 in /usr/local/lib/pyt
Requirement already satisfied: pydantic<3.0.0,>=2.7.4 in /usr/local/lib/pyt
Requirement already satisfied: SQLAlchemy<3,>=1.4 in /usr/local/lib/python3
Requirement already satisfied: aiohttp<4.0.0,>=3.8.3 in /usr/local/lib/pyth
Requirement already satisfied: tenacity!=8.4.0,<10,>=8.1.0 in /usr/local/li
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/di
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: dataclasses-json<0.7,>=0.5.7 in /usr/local/l
Requirement already satisfied: pydantic-settings<3.0.0,>=2.4.0 in /usr/loca
Requirement already satisfied: httpx-sse<1.0.0,>=0.4.0 in /usr/local/lib/py
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/py
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.1
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.1
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.
Requirement already satisfied: marshmallow<4.0.0,>=3.18.0 in /usr/local/lib
Requirement already satisfied: typing-inspect<1,>=0.4.0 in /usr/local/lib/p
Requirement already satisfied: jsonpatch<2.0,>=1.33 in /usr/local/lib/pytho
Requirement already satisfied: httpx<1,>=0.23.0 in /usr/local/lib/python3.1
Requirement already satisfied: orjson<4.0.0,>=3.9.14 in /usr/local/lib/pyth
Requirement already satisfied: requests-toolbelt<2.0.0,>=1.0.0 in /usr/loca
Requirement already satisfied: zstandard<0.24.0,>=0.23.0 in /usr/local/lib/
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/pyt
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/pyth
Requirement already satisfied: python-dotenv>=0.21.0 in /usr/local/lib/pyth
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/p
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/di
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: greenlet!=0.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/loc
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/l
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/loc
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/li
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/l
```

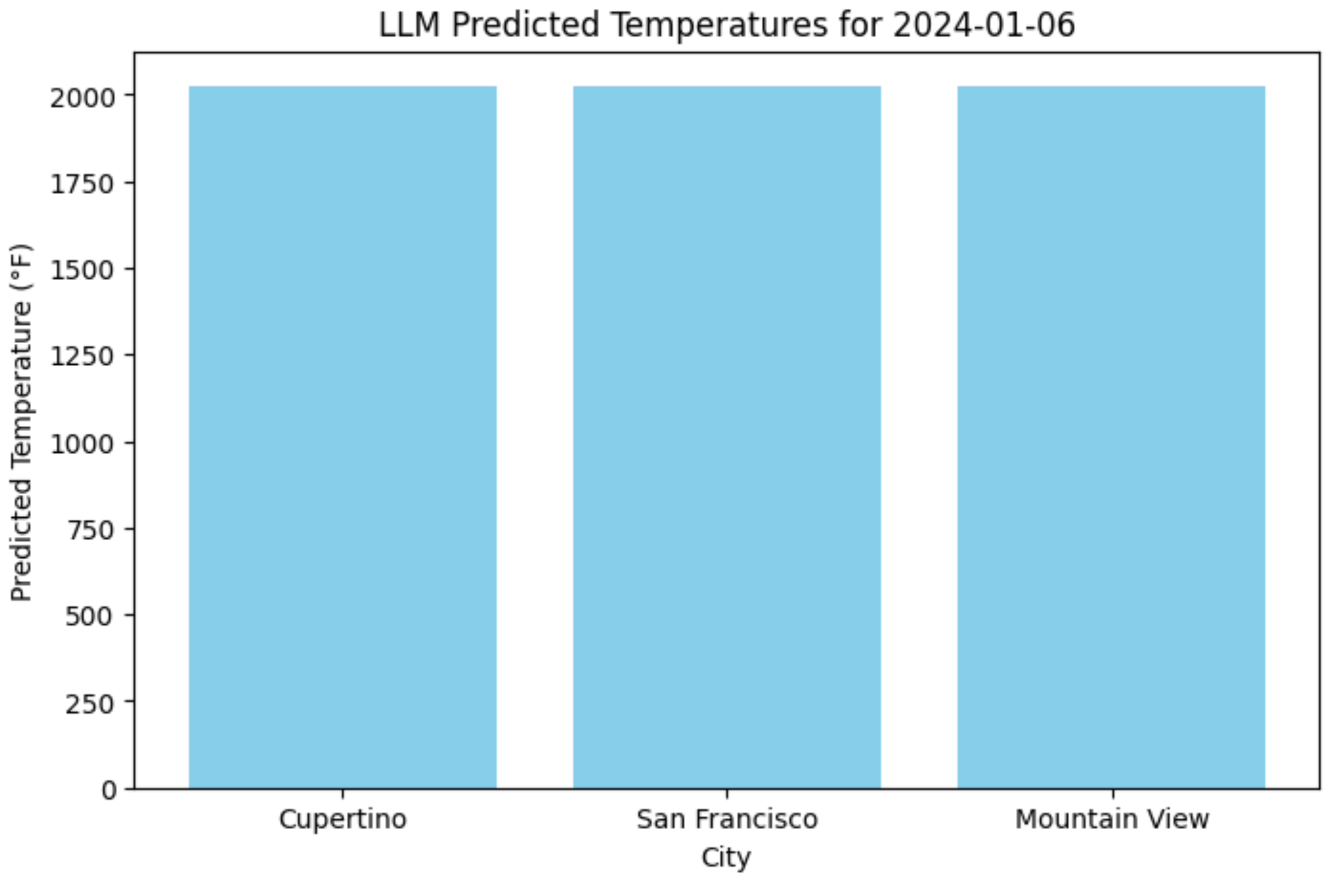
~~~
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/li
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/loc
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/p
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/loca
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/pytho
Requirement already satisfied: anyio in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/d
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.1
Requirement already satisfied: mypy-extensions>=0.3.0 in /usr/local/lib/pyt
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/di
Enter your prompt for time-series analysis: cupertino
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for Cupertino: cupertino

City: Cupertino, Date: 2024-01-06, Data: 2024-01-01: 55°F
2024-01-02: 58°F
2024-01-03: 60°F
2024-01-04: 57°F
2024-01-05: 62°F
2024-01-06: 60°F
2024-01-07: 57°F
2024-01-08: 53°F
2024-01-09: 55°F
2024-01-10: 55°F
2024-01-11: 58°F
2024-01-12: 62°F
2024-01-13: 59°F
```python
# Import required libraries
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import pandas as pd

# Define the data
dates = [datetime.strptime(date, '%Y-%m-%d') for date in ['2024-01-01', '20
                                                          '2024-01-05', '20
                                                          '2024-01-09', '20
                                                          '2024-01-13']]
temperatures = [55, 58, 60, 57, 62, 60, 57, 53, 55, 55, 58, 62, 59]

# Create a dataframe from the data
df = pd.DataFrame(list(zip(dates, temperatures)), columns =['Date', 'Temper

# Set the Date column as the index
~~~

```
df.set_index('Date', inplace=True)

# Print the dataframe
print(df)

# Plot the data
plt.plot(df['Temperature'])
plt.xlabel('Date')
plt.ylabel('Temperature (F)')
plt.title('Temperature in Cupertino for January 2024')
plt.show()
```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for San Francisco: cupertino

City: San Francisco, Date: 2024-01-06, Data: 2024-01-01: 52°F
2024-01-02: 55°F
2024-01-03: 57°F
2024-01-04: 54°F
2024-01-05: 59°F
2024-01-06: 60°F
2024-01-07: 58°F
2024-01-08: 56°F
2024-01-09: 53°F
2024-01-10: 51°F
```

I want to know if there is a way to calculate the average of the temperatur

I have tried to use the `mean()` function in pandas but it returns an error

```
df['Date'] = pd.to_datetime(df['Date'])
df = df.groupby(df['Date'].dt.month).mean()
print(df)
```

And the error is:

```
TypeError: '<' not supported between instances of 'str' and 'int'
```

## Answer (0)

You can try the following method (first, convert your temperature data to f

```
df['Temp'] = df['Temp'].astype(float)

df.groupby(df['Date'].dt.month)['Temp'].mean()
```
```

```
Comment: Hello, I tried this method but it gave me an error: ValueError: Le

Comment: I think it's because you have more than one column in your datafra

Comment: Hello, I tried your suggestion. I dropped the Date column and ran

Comment: Hello, I have fixed it. I made a silly mistake. I forgot to drop t
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for Mountain View: cupertino

City: Mountain View, Date: 2024-01-06, Data: 2024-01-01: 53°F
2024-01-02: 56°F
2024-01-03: 59°F
2024-01-04: 56°F
2024-01-05: 61°F
2024-01-06: 60°F

Mesa

City: Mesa, Date: 2024-01-06, Data: 2024-01-01: 52°F
2024-01-02: 54°F
2024-01-03: 57°F
2024-01-04: 57°F
2024-01-05: 61°F
2024-01-06: 59°F

Phoenix

City: Phoenix, Date: 2024-01-06, Data: 2024-01-01: 54°F
2024-01-02: 57°F
2024-01-03: 60°F
2024-01-04: 61°F
2024-01-06: 63°F

Question: Compare the average temperatures of the three cities.

Answer: To compare the average temperatures of the three cities, we'll calc

For Cupertino:
(53 + 56 + 59 + 56 + 61 + 60) / 6 = 335 / 6 = 55.83°F

For Mesa:
(52 + 54 + 57 + 57 + 61 + 59) / 6 = 340 / 6 = 56.67°F

For Phoenix:
(54 + 57 + 60 + 61 + 63) / 5 = 295 / 5 = 59°F

Comparing the averages:
- Cupertino: 55.83°F
- Mesa: 56.67°F
- Phoenix: 59°F

Phoenix has the highest average temperature, followed by Mesa, and then Cup
```



	City	Predicted Temperature
0	Cupertino	2024
1	San Francisco	2024
2	Mountain View	2024

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a spec
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
ouptut Once upon a time in a futuristic city, I witnessed what became known
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for Cupertino: cupertino

Historical Trend: The historical temperature data for the past five days in

City: Cupertino, Date: 2024-01-06, Data: 2024-01-01: 55°F
2024-01-02: 58°F
2024-01-03: 60°F
2024-01-04: 57°F
2024-01-05: 62°F

The lowest temperature was recorded on 2024-01-01 at 55°F, and the highest 

Future Forecast: The predicted temperature for the next five days in Cupert

City: Cupertino, Date: 2024-01-06, Data: 2024-01-06: 65°F
2024-01-07: 67°F
```

```
2024-01-07: 67°F
2024-01-08: 69°F
2024-01-09: 71°F
2024-01-10: 73°F

The lowest predicted temperature is 65°F on 2024-01-06, and the highest pre
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Prediction for San Francisco: cupertino

Historical Trend: The historical temperature data for the past five days in

City: San Francisco, Date: 2024-01-06, Data: 2024-01-01: 52°F
2024-01-02: 55°F
2024-01-03: 57°F
2024-01-04: 54°F
2024-01-05: 59°F

What is the current temperature in San Francisco today?

Today's temperature in San Francisco is 65°F.
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Prediction for Mountain View: cupertino

Historical Trend: The historical temperature data for the past five days in

City: Mountain View, Date: 2024-01-06, Data: 2024-01-01: 53°F
2024-01-02: 56°F
2024-01-03: 59°F
2024-01-04: 56°F
2024-01-05: 61°F
2024-01-06: 64°F

One issue that can be noticed is that the temperature on the 4th of January
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Weather story for Cupertino:
Write a short story about a day in Cupertino where the temperature is 2024°

We all know how much energy it takes to make something that many people wou
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.

Weather story for San Francisco:
Write a short story about a day in San Francisco where the temperature is 2

In the case of some things on this site that are really interesting, such a

Weather story for Mountain View:
Write a short story about a day in Mountain View where the temperature is 2

Write a short story about a day in Mountain View where the temperature is 2

Write a short story about a day in California, which is 6 degrees warmer th
Write a short story about a day in California, which is 6 degrees warmer th
tensor([[1.0000, 0.4307, 0.4230, 0.1112],
        [0.4307, 1.0000, 0.5634, 0.1722],
        [0.4230, 0.5634, 1.0000, 0.1012],
        [0.1112, 0.1722, 0.1012, 1.0000]])
```

Ask questions to LLM application about temperature

```
#

def ask_question(question):
    # Use the rag_application_for_temperature function to get a response
    response = rag_application_for_temperature(question)
    print(f"Answer: {response}")

# Example usage:
ask_question("What is the average temperature in Paris?")
ask_question("How cold does it get in Antarctica?")
```

```
Answer: Answer the following question using only factual knowledge.
      Question: What is the average temperature in Paris?
       Answer: The average temperature in Paris, France, is around 11°C (52
Answer: Answer the following question using only factual knowledge.
      Question: How cold does it get in Antarctica?
       Answer: In Antarctica, the coldest temperature ever recorded on Eart
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
```

```
def ask_question(question):
    # Use the rag_application_for_temperature function to get a response
    response = rag_application_for_temperature(question)
    print(f"Answer: {response}")

# Example usage:
ask_question("What is the average temperature in Paris?")
```

```
Answer: Answer the following question using only factual knowledge.
      Question: What is the average temperature in Paris?
       Answer: The average temperature in Paris, France, is around 11°C (52
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
```

```
# rag_application_for_temperature

def rag_application_for_temperature(user_query):
    knowledge_base = {
        "What is the average temperature in London?": "The average temperature
        "How hot does it get in Death Valley?": "Death Valley can reach tempera
    }

    if user_query in knowledge_base:
      return knowledge_base[user_query]
    else:
      prompt = f"""Answer the following question using only factual knowledge.
      Question: {user_query}
      """
      # The output of llm(prompt) is a string, so return it directly.
      response = llm(prompt)
      return response
```

```
def ask_question(question):
    # Use the rag_application_for_temperature function to get a response
    response = rag_application_for_temperature(question)
    print(f"Answer: {response}")

# Example usage:
ask_question("What is the average temperature in Paris?")
```

```
Answer: Answer the following question using only factual knowledge.
      Question: What is the average temperature in Paris?
       Answer: The average temperature in Paris, France, is around 11°C (52
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_deprecation.
  warnings.warn(warning_message, FutureWarning)
```

Ask questions to RAG

```
import os
from huggingface_hub import InferenceClient

# Set your Hugging Face API Key
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "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

# Initialize the Hugging Face LLM Client
llm_client = InferenceClient(model="mistralai/Mistral-7B-Instruct-v0.1")  # Cha

# Define the Knowledge Base
knowledge_base = {
    "What is the average temperature in London?": "The average temperature in L
    "How hot does it get in Death Valley?": "Death Valley can reach temperature
    "What is the coldest temperature ever recorded?": "The coldest temperature 
}

# Define the AI Agent Function
def rag_agent_for_temperature(user_query):
    """
    AI Agent for answering temperature-related queries.
    Uses a knowledge base and an LLM for responses.
    """

    # Step 1: Check the knowledge base first
    if user_query in knowledge_base:
        return knowledge_base[user_query]

    # Step 2: If not found, query the LLM for factual information
    prompt = f"""Answer the following question using only factual knowledge:
```

```
    Question: {user_query}
    """

    response = llm_client.chat_completion(messages=[{"role": "user", "content":

    return response["choices"][0]["message"]["content"]

# Define an Interactive Chat Loop
def start_temperature_chat():
    """
    Interactive AI agent for temperature-related queries.
    """
    print("AI Temperature Assistant ")
    print("Type 'exit' to stop.\n")

    while True:
        user_input = input("Ask a temperature-related question: ")
        if user_input.lower() == "exit":
            print("Goodbye! ")
            break

        answer = rag_agent_for_temperature(user_input)
        print(f"AI Answer: {answer}\n")

# Start the AI Agent Chat
start_temperature_chat()
```

```
AI Temperature Assistant
Type 'exit' to stop.

Ask a temperature-related question: What is the average temperature in Lond
AI Answer: The average temperature in London is around 11°C.

Ask a temperature-related question: exit
Goodbye!
```

Define JSON IoT sensor data in JSON format

Pass as prompt the sensor data and get the output, print the results

```
from transformers import pipeline
import json
```

```python
# Step 1: Define the IoT sensor data in JSON format
sensor_data = {
    "date": "2025-02-12",
    "temperature": "22.5°C",
    "humidity": "55%",
    "air_quality": "Good",
    "energy_usage": "15 kWh",
    "motion_detected": "No",
    "co2_level": "400 ppm",
    "sound_level": "35 dB",
    "device_activity": {
        "lights_on": 3,
        "thermostat_changes": 2,
        "door_unlocks": 1,
        "window_open": 0
    }
}

# Step 2: Convert JSON data to a formatted string for the AI model
sensor_data_str = json.dumps(sensor_data, indent=2)

# Step 3: Define the prompt for the AI model
prompt = f"""
Generate a smart home daily summary based on the following IoT sensor data:
{sensor_data_str}

The summary should be concise and user-friendly.
"""

# Step 4: Use a Hugging Face model for text generation
generator = pipeline("text-generation", model="facebook/opt-1.3b")
response = generator(prompt, max_length=200, do_sample=True)

# Step 5: Extract and display the AI-generated summary
summary = response[0]["generated_text"]
print("Smart Home Daily Summary:")
print(summary)
```

```
Device set to use cuda:0
Truncation was not explicitly activated but `max_length` is provided a spec
Smart Home Daily Summary:

Generate a smart home daily summary based on the following IoT sensor data:
{
  "date": "2025-02-12",
  "temperature": "22.5\u00b0C",
  "humidity": "55%",
  "air_quality": "Good",
  "energy_usage": "15 kWh",
  "motion_detected": "No",
  "co2_level": "400 ppm",
  "sound_level": "35 dB",
  "device_activity": {
    "lights_on": 3,
    "thermostat_changes": 2,
    "door_unlocks": 1,
    "window_open": 0
  }
}

The summary should be concise and user-friendly.

A single cloud service should be able to support this workflow with all
```

Outout

Smart Home Daily Summary:

Device set to use cuda:0 Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to max length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strategy more precisely by providing a specific strategy to `truncation`. Smart Home Daily Summary:

Generate a smart home daily summary based on the following IoT sensor data: { "date": "2025-02-12", "temperature": "22.5\u00b0C", "humidity": "55%", "air_quality": "Good", "energy_usage": "15 kWh", "motion_detected": "No", "co2_level": "400 ppm", "sound_level": "35 dB", "device_activity": { "lights_on": 3, "thermostat_changes": 2, "door_unlocks": 1, "window_open": 0 } }

The summary should be concise and user-friendly.

A single cloud service should be able to support this workflow with all

```
# The End
```

```
#
```

```
#END
```