1. Installation and Setup

```
# ! pip install transformers
# ! pip install requests
# ! pip install pillow
```

2. Loading the data

```
# from google.colab import files
# uploaded = files.upload()
```

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving pexels-freestockpro-12944684.jpg to pexels-freestockpro-12944684.jpg
```

```
import requests
from PIL import Image
from transformers import BlipProcessor, BlipForConditionalGeneration

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-large")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-large")

image_source = "/content/pexels-eugenia-remark-5767088-18977906.jpg"
# image_source = "https://static.vecteezy.com/system/resources/thumbnails/011/903/128/small_2x/
```

```
import matplotlib.pyplot as plt
plt.imshow(Image.open(image_source))
plt.axis('off')
plt.show()

if image_source.startswith("http"):
    image_path = image_source
    raw_image = Image.open(requests.get(image_path, stream=True).raw).convert('RGB')
else:
    image_path = image_source
    raw_image = Image.open(image_path).convert('RGB')
```

```
# image_path = image_source
# raw_image = Image.open(requests.get(image_path, stream=True).raw).convert('RGB')
# raw_image = Image.open(image_path).convert('RGB')

# conditional image captioning
text = "This is a photography of"
inputs = processor(raw_image, text, return_tensors="pt")

out = model.generate(**inputs)
print(processor.decode(out[0], skip_special_tokens=True))

# # unconditional image captioning
# inputs = processor(raw_image, return_tensors="pt")

# out = model.generate(**inputs)
# print(processor.decode(out[0], skip_special_tokens=True))
```

```
# print(processor.decode(out[0], skip_special_tokens=True))
```

```
/usr/local/lib/python3.10/dist-packages/transformers/generation/utils.py:1168: UserWarning
  warnings.warn(
this is a photography of a table setting with silverware and utensils
```

```
from transformers import pipeline


def image2text(image_path):
  image_to_text = pipeline("image-to-text", model="Salesforce/blip-image-captioning-large")
  text = image_to_text(image_path)[0]["generated_text"]

  # import matplotlib.pyplot as plt
  # plt.imshow(Image.open(image_source))
  # plt.axis('off')
  # plt.show()

  # print(text)
  return text



scenario = image2text(image_source)
print(scenario)
```

```
there is a tray with silverware and a plate with a spoon and fork
```

2. Text to Story with GPT

```
# !pip install openai==0.28
# !pip install langchain
# !pip install langchain_community

# OR

!pip install openai==0.28 langchain langchain_community
```

Show hidden output

```
# About langchain integration with OpenAI: https://python.langchain.com/v0.2/docs/integrations

import os
import openai
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain_community.llms import OpenAI

# Initializing the Openai API key
os.environ["OPENAI_API_KEY"] = "sk-proj-5J1ofPXEpKa5oddXG4FrT3BlbkFJLF9bcehUr8572CZHaTdE"

def generate_story(scenario):
  template = f"""
  You are a great storyteller;
  Generate a story based on a simple narrative, let the story be less than or equalto 100 word
  CONTEXT: {scenario}
  STORY:

  """
  # Initializing the Openai model
  prompt = PromptTemplate(template=template, input_variables=["scenario"])

  vincent_story = LLMChain(
      llm=OpenAI(model_name="gpt-3.5-turbo", temperature=1),
      prompt=prompt,
      verbose=True,)


  story = vincent_story.predict(scenario=scenario)
  # print(story)
  return story



story = generate_story(scenario)
print(story)
```

```
/usr/local/lib/python3.10/dist-packages/langchain_community/llms/openai.py:253: UserWarnir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/langchain_community/llms/openai.py:1076: UserWarni
  warnings.warn(


> Entering new LLMChain chain...
Prompt after formatting:

  You are a great storyteller;
  Generate a story based on a simple narrative,
  CONTEXT: there is a tray with silverware and a plate with a spoon and fork
  STORY:



> Finished chain.
Once upon a time in a small village nestled in the mountains, there lived a young girl nam

One day, Mia was invited to a grand feast at the village square. As she approached the lor

Mia began to tell a story of a kingdom ruled by a wicked queen who had cast a spell on the

As Mia weaved her tale, the villagers listened in rapt attention, hanging on her every wor
```

```
When Mia finished her story, the villagers erupted into applause, amazed by her ability to
Once upon a time in a small village nestled in the mountains, there lived a young girl nam

One day, Mia was invited to a grand feast at the village square. As she approached the lor

Mia began to tell a story of a kingdom ruled by a wicked queen who had cast a spell on the

As Mia weaved her tale, the villagers listened in rapt attention, hanging on her every wor

When Mia finished her story, the villagers erupted into applause, amazed by her ability to
```

3. Text to Audio

```
import requests
import os


# Add Huggingface access token
HUGGINGFACEHUB_API_TOKEN = "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"
os.environ["HUGGINGFACEHUB_API_TOKEN"] = HUGGINGFACEHUB_API_TOKEN

# Defining text to speech function
def text_to_speech(text):
  API_URL = "https://api-inference.huggingface.co/models/espnet/kan-bayashi_ljspeech_vits"
  headers = {"Authorization": f"Bearer {HUGGINGFACEHUB_API_TOKEN}"}

  payload = {
    "inputs": text,
  }

  response = requests.post(API_URL, headers=headers, json=payload)

  if response.status_code == 200:
    with open("output.flac", "wb") as file:
      file.write(response.content)
      print("The audio file generated successfully.")
  else:
    print(f"Error: {response.status_code}")




text_to_speech(story)
```

```
The audio file generated successfully.
```

```
from google.colab import files
uploaded = files.upload()
```

```
Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving pexels-eugenia-remark-5767088-18977913.jpg to pexels-eugenia-remark-57(
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Double-click (or enter) to edit

```
Start coding or generate with AI.
```