```
pip install transformers torch torchvision pillow

Requirement already satisfied: transformers in
/usr/local/lib/python3.11/dist-packages (4.47.1)
Requirement already satisfied: torch in
/usr/local/lib/python3.11/dist-packages (2.5.1+cu121)
Requirement already satisfied: torchvision in
/usr/local/lib/python3.11/dist-packages (0.20.1+cu121)
Requirement already satisfied: pillow in
/usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.11/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.27.1)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.11/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.11/dist-packages (from transformers)
(2024.11.6)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.5.2)
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.11/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in
/usr/local/lib/python3.11/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.11/dist-packages (from torch) (3.1.5)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.11/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105
in /usr/local/lib/python3.11/dist-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in
/usr/local/lib/python3.11/dist-packages (from torch) (9.1.0.70)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in
```

```
/usr/local/lib/python3.11/dist-packages (from torch) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in
/usr/local/lib/python3.11/dist-packages (from torch) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in
/usr/local/lib/python3.11/dist-packages (from torch) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from torch) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in
/usr/local/lib/python3.11/dist-packages (from torch) (12.1.105)
Requirement already satisfied: triton==3.1.0 in
/usr/local/lib/python3.11/dist-packages (from torch) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in
/usr/local/lib/python3.11/dist-packages (from torch) (1.13.1)
Requirement already satisfied: nvidia-nvjitlink-cu12 in
/usr/local/lib/python3.11/dist-packages (from nvidia-cusolver-
cu12==11.4.5.107->torch) (12.6.85)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch)
(1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2024.12.14)

# Using the BLIP model from Hugging Face provides a powerful,
#  pre-trained solution for generating captions for images.
#   It abstracts away the complexity of training an image captioning
model,
#    allowing you to focus on using the model for practical
applications.
#    This approach works well without needing a custom dataset, and
#    it gives you high-quality captions based on the visual content of
the image.

from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image
import requests
import matplotlib.pyplot as plt
```

```python
# Load pre-trained BLIP model and processor from Hugging Face
processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")

# Function to display the image
def display_image(image_path):
    image = Image.open(image_path)
    plt.imshow(image)
    plt.axis('off')
    plt.show()

# Function to generate a caption for an image using BLIP
def generate_caption(image_path):
    # Open the image
    image = Image.open(image_path)

    # Preprocess the image and prepare input for BLIP model
    inputs = processor(images=image, return_tensors="pt")

    # Generate caption
    out = model.generate(**inputs)

    # Decode the generated tokens to a readable caption
    caption = processor.decode(out[0], skip_special_tokens=True)
    return caption

# Image path (replace with your own image)
image_path = "/content/download (1).jpg"  # Replace with your image path

# Display the image
display_image(image_path)

# Generate and print the caption
caption = generate_caption(image_path)
print(f"Generated Caption: {caption}")
```

```
Generated Caption: two pup sitting in a field of flowers
```

```
from datasets import load_dataset

ds = load_dataset("roneneldan/TinyStories")
```


```
{"model_id":"0281359bb38b429ca0d57cb4dac250cd","version_major":2,"vers
ion_minor":0}

{"model_id":"1d95cc2cce6546f1a43683fe233aef36","version_major":2,"vers
ion_minor":0}

{"model_id":"9c6df2b29aeb482eb59efc514254b11d","version_major":2,"vers
ion_minor":0}

{"model_id":"c543d69a45764abab500b441733b58cb","version_major":2,"vers
ion_minor":0}

{"model_id":"3bef4059004f4d228d5526449dc49428","version_major":2,"vers
ion_minor":0}

{"model_id":"867d6a6e16a44a8e81a101180ae90291","version_major":2,"vers
ion_minor":0}

{"model_id":"944030586b0945ec87fc69d3491318e5","version_major":2,"vers
ion_minor":0}
```

{"model_id":"fa7a049131d449f9a65f2db9c02bf150","version_major":2,"version_minor":0}

```
!pip install datasets
```

```
Collecting datasets
  Downloading datasets-3.2.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in
/usr/local/lib/python3.10/dist-packages (from datasets) (4.67.1)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2
kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from
fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in
/usr/local/lib/python3.10/dist-packages (from datasets) (3.11.10)
Requirement already satisfied: huggingface-hub>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.27.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(2.4.4)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.3.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(4.0.3)
Requirement already satisfied: attrs>=17.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(24.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(6.1.0)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(0.2.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets)
(1.18.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0-
>datasets) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.32.2-
>datasets) (2024.12.14)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.2)
Requirement already satisfied: tzdata>=2022.7 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets)
(2024.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->datasets) (1.17.0)
Downloading datasets-3.2.0-py3-none-any.whl (480 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 8.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 8.7 MB/s eta
0:00:00
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 14.3 MB/s eta
0:00:00
ultiprocess-0.70.16-py310-none-any.whl (134 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 11.2 MB/s eta
0:00:00
anylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 13.8 MB/s eta
0:00:00
ultiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec
2024.9.0 which is incompatible.
Successfully installed datasets-3.2.0 dill-0.3.8 fsspec-2024.9.0
multiprocess-0.70.16 xxhash-3.5.0
```

```
!pip install python-dotenv
```

```
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM

# Load the pre-trained model and tokenizer
model_name = "EleutherAI/gpt-neo-1.3B"  # Larger model for better
generation
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Add a padding token to the tokenizer
if tokenizer.pad_token is None:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})
    model.resize_token_embeddings(len(tokenizer))  # Resize model
embeddings to include the new token


def generate_kids_story_offline(keywords, max_length=300,
temperature=0.8):
    """
    Generate a short, simple kids' story using a large offline
language model.
```

```
    Args:
        keywords (list): List of keywords for the story.
        max_length (int): Maximum length of the generated story.
        temperature (float): Creativity factor in generation.

    Returns:
        str: A kids' story with clear paragraphs and a happy ending.
    """
    # Create a prompt with clear instructions
    prompt = (
        f"Write a short and simple story for kids using these words: {', '.join(keywords)}. "
        "Make it fun, easy to read, and include a beginning, middle, and happy ending. "
        "Divide the story into paragraphs."
    )

    # Encode the prompt
    inputs = tokenizer(prompt, return_tensors="pt", padding=True, truncation=True)

    # Generate the story
    outputs = model.generate(
        inputs.input_ids,
        max_length=max_length,
        temperature=temperature,
        top_k=50,
        top_p=0.95,
        pad_token_id=tokenizer.eos_token_id,
        do_sample=True,
        no_repeat_ngram_size=2
    )

    # Decode and format the story
    story = tokenizer.decode(outputs[0], skip_special_tokens=True)

    # Extract the actual story (after the prompt)
    story_content = story.split("Write a short and simple story for kids", 1)[-1].strip()
    paragraphs = story_content.split('. ')
    formatted_story = "\n\n".join([" ".join(paragraphs[i:i + 2]) for i in range(0, len(paragraphs), 2)])

    return formatted_story

# Example keywords
keywords = ["castle", "dragon", "princess", "friendship"]

# Generate the story
```

```
kids_story = generate_kids_story_offline(keywords)
print("Generated Kids' Story:\n")
print(kids_story)
```

```
The new embeddings will be initialized from a multivariate normal
distribution that has old embeddings' mean and covariance. As
described in this article: https://nlp.stanford.edu/~johnhew/vocab-
expansion.html. To disable this, use `mean_resizing=False`
The attention mask is not set and cannot be inferred from input
because pad token is same as eos token. As a consequence, you may
observe unexpected behavior. Please pass your input's `attention_mask`
to obtain reliable results.

Generated Kids' Story:

using these words: castle, dragon, princess, friendship Make it fun,
easy to read, and include a beginning, middle, and happy ending

Divide the story into paragraphs Share your story with the kids by
telling them the different parts of the castle.

“One day you will find out that the prince is not a bad guy

He is really good at what he does and he has good friends.”
—‘Princess,’ by Alina F Martin
A fairy tale, Cinderella

The story of a princess who gets a prince and his evil stepsister She
is taken to a castle and told that her prince will marry her

When they arrive, they discover that she is in fact a fairy They are
separated and the princess is sent to live with her wicked stepmother,
the wicked witch

As the witch plans to get rid of her, she has the help of an evil
fairy and three evil knights One knight is the evil one, while the
other two are the good knights and they are all planning to help the
witches plan to take the royal princess away

Their plan is to save the kingdom from evil With each new twist in the
plot, it,Äôs more and more difficult to figure out who is who

But when one of them is found out to be the king‘Æ‡, everything
changes and this story has a happy
```

```
from huggingface_hub import login

# Replace this with your actual token
token = "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"
login(token)
```

```python
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer,
TrainingArguments
from datasets import load_dataset
from huggingface_hub import login

# Authenticate with Hugging Face
token = "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"
login(token)

# Load the pre-trained model and tokenizer
model_name = "EleutherAI/gpt-neo-1.3B"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(model_name)

# Add a padding token to the tokenizer if necessary
if tokenizer.pad_token is None:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})
    model.resize_token_embeddings(len(tokenizer))

# Load the TinyStories dataset
dataset = load_dataset("roneneldan/TinyStories")

# Tokenize the dataset
def tokenize_function(examples):
    return tokenizer(examples['text'], truncation=True,
padding="max_length", max_length=512)

tokenized_datasets = dataset.map(tokenize_function, batched=True)

# Split the dataset into train and validation sets
train_dataset = tokenized_datasets["train"]
validation_dataset = tokenized_datasets["validation"]

# Define training arguments
training_args = TrainingArguments(
    output_dir="./gpt_neo_finetuned_tinystories",
    overwrite_output_dir=True,
    eval_strategy="epoch",  # Updated to avoid deprecation warning
    learning_rate=5e-5,
    weight_decay=0.01,
    per_device_train_batch_size=4,
    per_device_eval_batch_size=4,
    num_train_epochs=3,
    save_strategy="epoch",
    save_total_limit=2,
    logging_dir="./logs",
    logging_steps=50,
    fp16=True,  # Use mixed precision training for faster performance
    report_to="none",
)
```

```python
# Define a Trainer instance
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=validation_dataset,
)

# Fine-tune the model
trainer.train()

# Save the fine-tuned model
trainer.save_model("./gpt_neo_finetuned_tinystories")
tokenizer.save_pretrained("./gpt_neo_finetuned_tinystories")

print("Fine-tuning complete. Model saved to
'./gpt_neo_finetuned_tinystories'")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/
_auth.py:94: UserWarning:
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://huggingface.co/settings/tokens), set it as
secret in your Google Colab and restart your session.
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 models or datasets.
  warnings.warn(
The new embeddings will be initialized from a multivariate normal
distribution that has old embeddings' mean and covariance. As
described in this article: https://nlp.stanford.edu/~johnhew/vocab-
expansion.html. To disable this, use `mean_resizing=False`
```

{"model_id":"da0f257513f448af851aac66e5191845","version_major":2,"vers
ion_minor":0}

```python
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load the fine-tuned model
fine_tuned_model_path = "./gpt_neo_finetuned_tinystories"
model = AutoModelForCausalLM.from_pretrained(fine_tuned_model_path)
tokenizer = AutoTokenizer.from_pretrained(fine_tuned_model_path)

# Generate a sample story
def generate_story(prompt, model, tokenizer, max_length=100):
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate(
        inputs.input_ids,
        max_length=max_length,
```

```
        num_return_sequences=1,
        do_sample=True,
        temperature=0.7,
    )
    return tokenizer.decode(outputs[0], skip_special_tokens=True)

# Example usage
keywords = ["castle", "dragon", "princess", "friendship"]
prompt = f"Once upon a time, {', '.join(keywords)},"
story = generate_story(prompt, model, tokenizer)
print("Generated Story:\n", story)


---------------------------------------------------------------------------
-----
OSError                                   Traceback (most recent call
last)
<ipython-input-6-daf76e48f88f> in <cell line: 0>()
      3 # Load the fine-tuned model
      4 fine_tuned_model_path = "./gpt_neo_finetuned_tinystories"
----> 5 model =
AutoModelForCausalLM.from_pretrained(fine_tuned_model_path)
      6 tokenizer =
AutoTokenizer.from_pretrained(fine_tuned_model_path)
      7

/usr/local/lib/python3.11/dist-packages/transformers/models/auto/auto_
factory.py in from_pretrained(cls, pretrained_model_name_or_path,
*model_args, **kwargs)
    562         elif type(config) in cls._model_mapping.keys():
    563             model_class = _get_model_class(config,
cls._model_mapping)
--> 564             return model_class.from_pretrained(
    565                 pretrained_model_name_or_path, *model_args,
config=config, **hub_kwargs, **kwargs
    566             )

/usr/local/lib/python3.11/dist-packages/transformers/modeling_utils.py
in from_pretrained(cls, pretrained_model_name_or_path, config,
cache_dir, ignore_mismatched_sizes, force_download, local_files_only,
token, revision, use_safetensors, weights_only, *model_args, **kwargs)
   3777                     )
   3778                 else:
-> 3779                     raise EnvironmentError(
   3780                         f"Error no file named
{_add_variant(WEIGHTS_NAME, variant)},
{_add_variant(SAFE_WEIGHTS_NAME, variant)},"
   3781                         f" {TF2_WEIGHTS_NAME},
{TF_WEIGHTS_NAME + '.index'} or {FLAX_WEIGHTS_NAME} found in
directory"
```

```
OSError: Error no file named pytorch_model.bin, model.safetensors,
tf_model.h5, model.ckpt.index or flax_model.msgpack found in directory
./gpt_neo_finetuned_tinystories.
```

```python
from transformers import AutoModelForCausalLM, AutoTokenizer

# Load the fine-tuned model
fine_tuned_model_path = "./gpt_neo_finetuned_tinystories"
model = AutoModelForCausalLM.from_pretrained(fine_tuned_model_path)  #
Define model in global scope
tokenizer = AutoTokenizer.from_pretrained(fine_tuned_model_path) #
Define tokenizer in global scope

# Generate a sample story
def generate_story(prompt, max_length=100): # Removed model and
tokenizer as arguments
    inputs = tokenizer(prompt, return_tensors="pt")
    outputs = model.generate( # Using model from global scope
        inputs.input_ids,
        max_length=max_length,
        num_return_sequences=1,
        do_sample=True,
        temperature=0.7,
    )
    return tokenizer.decode(outputs[0], skip_special_tokens=True) #
Using tokenizer from global scope

# Example usage
keywords = ["castle", "dragon", "princess", "friendship"]
prompt = f"Once upon a time, {', '.join(keywords)},"
story = generate_story(prompt)
print("Generated Story:\n", story)


model.save_pretrained("./gpt_neo_finetuned_tinystories")
tokenizer.save_pretrained("./gpt_neo_finetuned_tinystories")
```

```
---------------------------------------------------------------------
-----
OSError                                   Traceback (most recent call
last)
<ipython-input-8-286d3e326164> in <cell line: 0>()
      3 # Load the fine-tuned model
      4 fine_tuned_model_path = "./gpt_neo_finetuned_tinystories"
----> 5 model =
AutoModelForCausalLM.from_pretrained(fine_tuned_model_path)  # Define
model in global scope
      6 tokenizer =
AutoTokenizer.from_pretrained(fine_tuned_model_path) # Define
```

```
tokenizer in global scope
      7

/usr/local/lib/python3.11/dist-packages/transformers/models/auto/auto_
factory.py in from_pretrained(cls, pretrained_model_name_or_path,
*model_args, **kwargs)
    562         elif type(config) in cls._model_mapping.keys():
    563             model_class = _get_model_class(config,
cls._model_mapping)
--> 564             return model_class.from_pretrained(
    565                 pretrained_model_name_or_path, *model_args,
config=config, **hub_kwargs, **kwargs
    566             )

/usr/local/lib/python3.11/dist-packages/transformers/modeling_utils.py
in from_pretrained(cls, pretrained_model_name_or_path, config,
cache_dir, ignore_mismatched_sizes, force_download, local_files_only,
token, revision, use_safetensors, weights_only, *model_args, **kwargs)
   3777                     )
   3778                 else:
-> 3779                     raise EnvironmentError(
   3780                         f"Error no file named
{_add_variant(WEIGHTS_NAME, variant)},
{_add_variant(SAFE_WEIGHTS_NAME, variant)},"
   3781                         f" {TF2_WEIGHTS_NAME},
{TF_WEIGHTS_NAME + '.index'} or {FLAX_WEIGHTS_NAME} found in
directory"

OSError: Error no file named pytorch_model.bin, model.safetensors,
tf_model.h5, model.ckpt.index or flax_model.msgpack found in directory
./gpt_neo_finetuned_tinystories.
```