Next_Word_Prediction_&_Writing_Assistant

October 24, 2025

[1]:
```
# Step 1: Setup environment

!pip install -q transformers datasets accelerate torch sentencepiece streamlit

import os
os.makedirs("writewise/data/raw", exist_ok=True)
os.makedirs("writewise/data/processed", exist_ok=True)
os.makedirs("writewise/models", exist_ok=True)

print("  Environment ready - now upload or create your dataset in writewise/
 ↪data/raw/")
```

```
 Environment ready - now upload or create your dataset in writewise/data/raw/
```

[2]:
```
# Step 2: Create / upload training data

# Option A: Create a small sample dataset here

sample_text = """
Artificial intelligence is transforming industries and society.
Data science helps us understand patterns in complex data.
Machine learning enables computers to learn from experience.
Generative AI can create new ideas and automate creativity.
Business leaders use data-driven insights for better decisions.
"""

with open("writewise/data/raw/sample.txt", "w", encoding="utf-8") as f:
    f.write(sample_text)

print("  Sample dataset created at writewise/data/raw/sample.txt")

# Option B: You can upload your own text file(s) into writewise/data/raw/
```

```
 Sample dataset created at writewise/data/raw/sample.txt
```

[3]:
```
# Step 3: Dataset preparation

# Splits text into clean sentences (1 per line) for training
```

```python
import re, glob

def clean_text(text):
    text = re.sub(r'\s+', ' ', text)
    return text.strip()

input_dir = "writewise/data/raw"
output_file = "writewise/data/processed/train.txt"

all_lines = []
for fpath in glob.glob(f"{input_dir}/*.txt"):
    with open(fpath, encoding="utf-8") as f:
        raw = f.read()
    for block in raw.splitlines():
        block = block.strip()
        if not block: continue
        parts = re.split(r'(?<=[.!?])\s+', block)
        for p in parts:
            p = clean_text(p)
            if len(p.split()) >= 5:
                all_lines.append(p)

os.makedirs(os.path.dirname(output_file), exist_ok=True)
with open(output_file, "w", encoding="utf-8") as f:
    for line in all_lines:
        f.write(line + "\n")

print(f" Prepared {len(all_lines)} training lines at {output_file}")
```

```
 Prepared 5 training lines at writewise/data/processed/train.txt
```

[4]:
```python
# ============================================================
# Step 4: Fine-tune GPT-2 (small)
# ============================================================
from datasets import load_dataset
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer,␣
 ↪TrainingArguments, DataCollatorForLanguageModeling
import torch, os

# Disable Weights & Biases logging
os.environ["WANDB_DISABLED"] = "true"

train_file = "writewise/data/processed/train.txt"
model_name = "gpt2"
output_dir = "writewise/models/writewise-gpt2"
```

```python
# Load text dataset
dataset = load_dataset("text", data_files={"train": train_file})

tokenizer = AutoTokenizer.from_pretrained(model_name)
if tokenizer.pad_token is None:
    tokenizer.add_special_tokens({'pad_token': '[PAD]'})

model = AutoModelForCausalLM.from_pretrained(model_name)
model.resize_token_embeddings(len(tokenizer))

def tokenize_fn(examples):
    return tokenizer(examples["text"], truncation=True, max_length=128)

tokenized = dataset.map(tokenize_fn, batched=True, remove_columns=["text"])
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

training_args = TrainingArguments(
    output_dir=output_dir,
    overwrite_output_dir=True,
    num_train_epochs=3,
    per_device_train_batch_size=2,
    save_strategy="no",
    logging_steps=10,
    learning_rate=5e-5,
    fp16=torch.cuda.is_available(),
    report_to=[],  #  disables wandb and other reporters
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized["train"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)

trainer.train()
trainer.save_model(output_dir)
print("  Model fine-tuned and saved to:", output_dir)
```

```
Generating train split: 0 examples [00:00, ? examples/s]
```

```
/usr/local/lib/python3.12/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`
```

```
Map:   0%|          | 0/5 [00:00<?, ? examples/s]
```

```
/tmp/ipython-input-826544583.py:43: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
The tokenizer has new PAD/BOS/EOS tokens that differ from the model config and
generation config. The model config and generation config were aligned
accordingly, being updated with the tokenizer's values. Updated tokens:
{'pad_token_id': 50257}.
`loss_type=None` was set in the config but it is unrecognized. Using the default
loss: `ForCausalLMLoss`.
```

```
<IPython.core.display.HTML object>
```

```
 Model fine-tuned and saved to: writewise/models/writewise-gpt2
```

[5]:
```python
# Step 5: Inference - Next-word prediction & text generation

import torch
import torch.nn.functional as F
from transformers import AutoTokenizer, AutoModelForCausalLM

model_dir = "writewise/models/writewise-gpt2"
tokenizer = AutoTokenizer.from_pretrained(model_dir)
model = AutoModelForCausalLM.from_pretrained(model_dir)
model.eval()
if torch.cuda.is_available():
    model.to("cuda")

def top_k_next_tokens(prompt, top_k=5):
    inputs = tokenizer(prompt, return_tensors="pt")
    if torch.cuda.is_available():
        inputs = {k:v.to("cuda") for k,v in inputs.items()}
    with torch.no_grad():
        logits = model(**inputs).logits
    last_logits = logits[0, -1, :]
    probs = F.softmax(last_logits, dim=-1)
    topk = torch.topk(probs, top_k)
    tokens = topk.indices.tolist()
    return [(tokenizer.decode([t]).strip(), float(s)) for t,s in zip(tokens,␣
 ↪topk.values)]

def generate_text(prompt, max_new_tokens=40, temperature=0.8):
    inputs = tokenizer(prompt, return_tensors="pt")
    if torch.cuda.is_available():
        inputs = {k:v.to("cuda") for k,v in inputs.items()}
    output = model.generate(**inputs, max_new_tokens=max_new_tokens,␣
 ↪do_sample=True, temperature=temperature, top_k=50)
    return tokenizer.decode(output[0], skip_special_tokens=True)

prompt = "The future of artificial intelligence"
print("Top next-word suggestions:")
for tok,score in top_k_next_tokens(prompt, top_k=5):
    print(f"{tok}  ({score:.4f})")

print("\nGenerated continuation:")
print(generate_text(prompt))
```

```
Top next-word suggestions:
is  (0.3073)
will  (0.0631)
and  (0.0556)
,  (0.0546)
may  (0.0363)

Generated continuation:
The future of artificial intelligence? It depends on how many ways we use it.

The most obvious possibility is to improve AI with software improvements. One
idea is to improve on a computer's human language processing capabilities. For
```

[6]:
```
from pyngrok import ngrok
ngrok.set_auth_token("34VADzTWUIxvmMrGEBiy0GsBBE5_2kGou76Cb1qvHrSJy2koL")
```

[7]:
```
# Step 6: Streamlit demo inside Colab
# Note: Streamlit runs on a separate port – we’ll use `pyngrok` to tunnel it.

!pip install -q pyngrok
from pyngrok import ngrok
import threading, time

# Create app.py
app_code = '''
import streamlit as st
import torch
import torch.nn.functional as F
```

```
from transformers import AutoTokenizer, AutoModelForCausalLM

st.set_page_config(page_title="WriteWise - Next-Word Assistant")

@st.cache_resource
def load_model(model_dir="writewise/models/writewise-gpt2"):
    tokenizer = AutoTokenizer.from_pretrained(model_dir)
    model = AutoModelForCausalLM.from_pretrained(model_dir)
    model.eval()
    if torch.cuda.is_available():
        model.to("cuda")
    return tokenizer, model

tokenizer, model = load_model()

def top_k_next_tokens(prompt, top_k=5):
    inputs = tokenizer(prompt, return_tensors="pt")
    if torch.cuda.is_available():
        inputs = {k:v.to("cuda") for k,v in inputs.items()}
    with torch.no_grad():
        logits = model(**inputs).logits
    last_logits = logits[0, -1, :]
    probs = torch.nn.functional.softmax(last_logits, dim=-1)
    topk = torch.topk(probs, top_k)
    tokens = topk.indices.tolist()
    return [(tokenizer.decode([t]).strip(), float(s)) for t,s in zip(tokens,␣
 ↪topk.values)]

def generate_text(prompt, max_new_tokens=50, temperature=0.8):
    inputs = tokenizer(prompt, return_tensors="pt")
    if torch.cuda.is_available():
        inputs = {k:v.to("cuda") for k,v in inputs.items()}
    output = model.generate(**inputs, max_new_tokens=max_new_tokens,␣
 ↪do_sample=True, temperature=temperature, top_k=50)
    return tokenizer.decode(output[0], skip_special_tokens=True)

st.title("WriteWise - Next-Word Prediction & Text Generator")
prompt = st.text_area("Enter your prompt:", "The future of AI is")

col1, col2 = st.columns(2)
with col1:
    if st.button("Next-Word Suggestions"):
        results = top_k_next_tokens(prompt, 5)
        for w,p in results:
            st.write(f"**{w}**  ({p:.4f})")
with col2:
    if st.button("Generate Text"):
        out = generate_text(prompt, 60, 0.8)
        st.write(out)
'''

with open("app.py", "w") as f:
    f.write(app_code)

# Launch Streamlit app
public_url = ngrok.connect(8501)
print("Streamlit public URL:", public_url)
!streamlit run app.py --server.port 8501
```

```
Streamlit public URL: NgrokTunnel: "https://astronomical-kohen-
treasurable.ngrok-free.dev" -> "http://localhost:8501"

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to
false.


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://172.28.0.12:8501
  External URL: http://34.143.149.124:8501

2025-10-24 07:04:49.653706: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1761289489.674777   10382 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1761289489.681400   10382 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1761289489.700712   10382 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1761289489.700760   10382 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1761289489.700767   10382 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1761289489.700772   10382 computation_placer.cc:177] computation
```

```
placer already registered. Please check linkage and avoid linking the same
target more than once.
c
  Stopping…
  Stopping…
^C
```