Sun_30_Nov copy

December 1, 2025

0.1 (train_sentence_check.py), run with mac's terminal

max_length 512 256 Epochs 3 input_text x

[ ]:
```
import os
import torch
import pandas as pd
import numpy as np
from datasets import Dataset
from transformers import (
    AutoTokenizer,
    AutoModelForSeq2SeqLM,
    DataCollatorForSeq2Seq,
    Seq2SeqTrainingArguments,
    Seq2SeqTrainer,
)
from peft import LoraConfig, get_peft_model, TaskType

# ==========================================
# 1.
# ==========================================
#
DATA_PATH = "pseudopairs_sentence_noisy_only.jsonl"
OUTPUT_DIR = "mac_t5_sentence_model" #

MODEL_NAME = "google/flan-t5-base"
#       512 256
MAX_LEN = 256

# Mac
os.environ["TOKENIZERS_PARALLELISM"] = "false"
DEVICE = "mps" if torch.backends.mps.is_available() else "cpu"
print(f"  Running on: {DEVICE}")

# ==========================================
# 2.
# ==========================================
def load_and_process_data():
```

```python
    print(f"Reading {DATA_PATH}...")
    df = pd.read_json(DATA_PATH, lines=True)
    print(f"   : {len(df)}  ")

    # 
    if 'input_text' in df.columns:
        pass # 
    elif 'x' in df.columns and 'y' in df.columns:
        print("     (x, y)    ...")
        df = df.rename(columns={'x': 'input_text', 'y': 'target_text'})

    #    9:1
    train_df = df.sample(frac=0.9, random_state=42)
    val_df = df.drop(train_df.index)

    print(f"Train: {len(train_df)} | Val: {len(val_df)}")

    return Dataset.from_pandas(train_df), Dataset.from_pandas(val_df)

# ==========================================
# 3. 
# ==========================================
def main():
    train_ds, val_ds = load_and_process_data()
    tokenizer = AutoTokenizer.from_pretrained(MODEL_NAME)

    def tokenize_fn(batch):
        model_inputs = tokenizer(
            batch["input_text"], max_length=MAX_LEN, truncation=True,␣
 ↪padding=False
        )
        labels = tokenizer(
            text_target=batch["target_text"], max_length=MAX_LEN,␣
 ↪truncation=True, padding=False
        )
        model_inputs["labels"] = labels["input_ids"]
        return model_inputs

    print("Tokenizing...")
    train_tok = train_ds.map(tokenize_fn, batched=True)
    val_tok = val_ds.map(tokenize_fn, batched=True)

    print("Loading Model...")
    model = AutoModelForSeq2SeqLM.from_pretrained(MODEL_NAME)
    model.to(DEVICE)

    # LoRA   ( )
    lora_config = LoraConfig(
        r=16, lora_alpha=32, target_modules=["q", "v"],
        lora_dropout=0.05, bias="none", task_type=TaskType.SEQ_2_SEQ_LM
    )
    model = get_peft_model(model, lora_config)
    model.print_trainable_parameters()

    #     (M4 Pro   )
    args = Seq2SeqTrainingArguments(
        output_dir=OUTPUT_DIR,
        num_train_epochs=3,              #  3
        per_device_train_batch_size=16, #   Batch
        per_device_eval_batch_size=16,
        gradient_accumulation_steps=2,  #   Batch 32
        learning_rate=3e-4,
        fp16=False, bf16=False,
        logging_steps=10,
        eval_strategy="epoch",
        save_strategy="epoch",
        save_total_limit=1,
        load_best_model_at_end=True,
        metric_for_best_model="eval_loss",
        dataloader_num_workers=0,
        report_to="none"
    )

    trainer = Seq2SeqTrainer(
        model=model, args=args,
        train_dataset=train_tok, eval_dataset=val_tok,
        tokenizer=tokenizer,
        data_collator=DataCollatorForSeq2Seq(tokenizer, model=model)
    )

    print("    ...")
    trainer.train()

    print("       ...")
    trainer.save_model(OUTPUT_DIR)
    tokenizer.save_pretrained(OUTPUT_DIR)
    print(f"    : {OUTPUT_DIR}")

if __name__ == "__main__":
    main()
```

0.2 (test_sentence_check.py)

Flan-T5 vs fine-tuned

```
[ ]: import torch
     from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
     from peft import PeftModel

     #
     ADAPTER_PATH = "mac_t5_sentence_model" #
     BASE_MODEL = "google/flan-t5-base"
     DEVICE = "mps" if torch.backends.mps.is_available() else "cpu"

     print(f"     {BASE_MODEL} ...")
     tokenizer = AutoTokenizer.from_pretrained(BASE_MODEL)
     base_model = AutoModelForSeq2SeqLM.from_pretrained(BASE_MODEL)

     print(f"      {ADAPTER_PATH} ...")
     model = PeftModel.from_pretrained(base_model, ADAPTER_PATH)
     model.to(DEVICE)
     model.eval()

     def generate(text):
         #       prompt   (  simplify to...),
         #    noisy sentence      prompt
         #      "simplify: "         raw text
         #   pseudopairs    input_text      prompt (    )
         #           prompt

         input_text = text #     f"simplify: {text}"

         inputs = tokenizer(input_text, return_tensors="pt").to(DEVICE)
         with torch.no_grad():
             outputs = model.generate(
                 input_ids=inputs["input_ids"],
                 max_new_tokens=128,
                 temperature=0.7,
                 do_sample=True
             )
         return tokenizer.decode(outputs[0], skip_special_tokens=True)

     #    (      )
     test_sentences = [
         "The inherent complexity of the mechanism precludes simple explanation.",
         "Despite the inclement weather conditions, the event proceeded as scheduled.
       ↪",
         "The proliferation of digital devices has precipitated a paradigm shift in␣
       ↪communication."
     ]

     print("\n" + "="*40)
```

```python
print("     ")
print("="*40)

for sent in test_sentences:
    print(f"\n[ ]: {sent}")
    print(f"  [   ]: {generate(sent)}")
    print("-" * 20)
```