Ajustar un Modelo

```python
import os
import glob
import json
from dotenv import load_dotenv
from transformers import AutoTokenizer, AutoModelForCausalLM, Trainer, TrainingArguments, DataCollatorForLangu
from datasets import Dataset, load_dataset
import torch

load_dotenv()

NOMBRE_MODEL = "deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B"  # Cambiado al modelo especificado
DIRECTORIO_SALIDA = "modelo_entrenado"
ARCHIVO_ENTRENAMIENTO = "train.jsonl"
MAX_LONGITUD = 512
TAMAÑO_LOTE = 8
EPÓCAS = 3

def crear_datos_entrenamiento(directorio_posts):
    todos_texto = []
    for dir_leng in os.listdir(directorio_posts):
        ruta_leng = os.path.join(directorio_posts, dir_leng)
        if not os.path.isdir(ruta_leng):
            continue
        for ruta_archivo in glob.glob(os.path.join(ruta_leng, "*.md")):
            try:
                with open(ruta_archivo, 'r', encoding='utf-8') as f:
                    contenido = f.read()
                    # Eliminar front matter
                    contenido = contenido.split("---", 2)[-1].strip()
                    todos_texto.append(contenido)
            except Exception as e:
                print(f"Error leyendo archivo {ruta_archivo}: {e}")
    return todos_texto

def preparar_conjunto_datos(textos, tokenizador):
    codificaciones = tokenizador(textos, truncation=True, padding=True, max_length=MAX_LONGITUD, return_tenso
    return Dataset.from_dict(codificaciones)
```

```
def entrenar_modelo(conjunto_datos, tokenizador):
    argumentos_entrenamiento = TrainingArguments(
        output_dir=DIRECTORIO_SALIDA,
        overwrite_output_dir=True,
        num_train_epochs=EPÓCAS,
        per_device_train_batch_size=TAMAÑO_LOTE,
        save_steps=10_000,
        save_total_limit=2,
        prediction_loss_only=True,
        remove_unused_columns=False,
    )
    modelo = AutoModelForCausalLM.from_pretrained(NOMBRE_MODEL, trust_remote_code=True)
    data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizador, mlm=False)
    entrenador = Trainer(
        model=modelo,
        args=argumentos_entrenamiento,
        train_dataset=conjunto_datos,
        data_collator=data_collator,
    )
    entrenador.train()
    entrenador.save_model(DIRECTORIO_SALIDA)

def main():
    directorio_posts = "_posts"
    textos = crear_datos_entrenamiento(directorio_posts)
    tokenizador = LlamaTokenizerFast.from_pretrained(NOMBRE_MODEL, trust_remote_code=True, use_fast=True)
    tokenizador.pad_token = tokenizador.eos_token
    conjunto_datos = preparar_conjunto_datos(textos, tokenizador)
    entrenar_modelo(conjunto_datos, tokenizador)

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