Affiner un Modèle

```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()

MODEL_NAME = "deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B"  # Changé pour le modèle spécifié
OUTPUT_DIR = "modèle_entraîné"
TRAIN_FILE = "train.jsonl"
MAX_LENGTH = 512
BATCH_SIZE = 8
EPOCHS = 3

def créer_données_entraînement(répertoire_posts):
    tous_les_textes = []
    for langue_dir in os.listdir(répertoire_posts):
        chemin_langue = os.path.join(répertoire_posts, langue_dir)
        if not os.path.isdir(chemin_langue):
            continue
        for chemin_fichier in glob.glob(os.path.join(chemin_langue, "*.md")):
            try:
                with open(chemin_fichier, 'r', encoding='utf-8') as f:
                    contenu = f.read()
                    # Supprimer la matière frontale
                    contenu = contenu.split("---", 2)[-1].strip()
                    tous_les_textes.append(contenu)
            except Exception as e:
                print(f"Erreur de lecture du fichier {chemin_fichier}: {e}")
    return tous_les_textes

def préparer_jeu_de_données(textes, tokeniseur):
    encodages = tokeniseur(textes, truncation=True, padding=True, max_length=MAX_LENGTH, return_tensors="pt")
    return Dataset.from_dict(encodages)
```

```
def entraîner_modèle(jeu_de_données, tokeniseur):
    arguments_entraînement = TrainingArguments(
        output_dir=OUTPUT_DIR,
        overwrite_output_dir=True,
        num_train_epochs=EPOCHS,
        per_device_train_batch_size=BATCH_SIZE,
        save_steps=10_000,
        save_total_limit=2,
        prediction_loss_only=True,
        remove_unused_columns=False,
    )
    modèle = AutoModelForCausalLM.from_pretrained(MODEL_NAME, trust_remote_code=True)
    data_collator = DataCollatorForLanguageModeling(tokenizer=tokeniseur, mlm=False)
    entraîneur = Trainer(
        model=modèle,
        args=arguments_entraînement,
        train_dataset=jeu_de_données,
        data_collator=data_collator,
    )
    entraîneur.train()
    entraîneur.save_model(OUTPUT_DIR)

def main():
    répertoire_posts = "_posts"
    textes = créer_données_entraînement(répertoire_posts)
    tokeniseur = LlamaTokenizerFast.from_pretrained(MODEL_NAME, trust_remote_code=True, use_fast=True)
    tokeniseur.pad_token = tokeniseur.eos_token
    jeu_de_données = préparer_jeu_de_données(textes, tokeniseur)
    entraîner_modèle(jeu_de_données, tokeniseur)

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