モデルを微調整

```
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"  # 指定されたモデルに変更しました
OUTPUT_DIR = "trained_model"
TRAIN_FILE = "train.jsonl"
MAX_LENGTH = 512
BATCH_SIZE = 8
EPOCHS = 3

def create_training_data(posts_dir):
    all_texts = []
    for lang_dir in os.listdir(posts_dir):
        lang_path = os.path.join(posts_dir, lang_dir)
        if not os.path.isdir(lang_path):
            continue
        for file_path in glob.glob(os.path.join(lang_path, "*.md")):
            try:
                with open(file_path, 'r', encoding='utf-8') as f:
                    content = f.read()
                    # 前書きを削除
                    content = content.split("---", 2)[-1].strip()
                    all_texts.append(content)
            except Exception as e:
                print(f" ファイル {file_path} の読み取りエラー: {e}")
    return all_texts

def prepare_dataset(texts, tokenizer):
    encodings = tokenizer(texts, truncation=True, padding=True, max_length=MAX_LENGTH, return_tensors="pt")
    return Dataset.from_dict(encodings)
```

```
def train_model(dataset, tokenizer):
    training_args = 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,
    )
    model = AutoModelForCausalLM.from_pretrained(MODEL_NAME, trust_remote_code=True)
    data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)
    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=dataset,
        data_collator=data_collator,
    )
    trainer.train()
    trainer.save_model(OUTPUT_DIR)

def main():
    posts_dir = "_posts"
    texts = create_training_data(posts_dir)
    tokenizer = LlamaTokenizerFast.from_pretrained(MODEL_NAME, trust_remote_code=True, use_fast=True)
    tokenizer.pad_token = tokenizer.eos_token
    dataset = prepare_dataset(texts, tokenizer)
    train_model(dataset, tokenizer)

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