Deepseek’i yerel bilgisayarımızda nasıl özelleştirebiliriz?

1. Prompt Engineering (Kolay Seviye Özelleştirme)

Prompt Engineering, modeli **eğitmeden**, sadece **sorduğun soruların (promptların) yapısını değiştirerek** modelin davranışını kontrol etmektir. Bu yöntemle, model seninle farklı tonlarda, biçimlerde veya kurallara göre konuşabilir.Ancak benim yorumum elimizdeki güçlü bilgisayarlar ve altyapı için oldukça basit iş , klasik chatbotla konuşmaktan çokta büyük bir farkı yoktur.

2. RAG (Retrieval-Augmented Generation)

Modeli **eğitmeden**, kendi belirlediğin belgeleri (PDF, TXT, CSV vs.) bir bilgi kaynağı olarak kullanarak yanıt üretmesini sağlarsın. Bu yöntem, modeli bir "asistan" gibi eğitmeden belgeye bağlar.

Bu model benim tercih ettiğim yöntem olan LM Studio’da doğrudan kullanılamıyor bu durum şöyle : LM Studio üzerinden .gguf formatında indirilen model Python ortamında **llama-cpp-python** ile bağlanarak RAG yapılabilir.

Aşağıdaki komutları terminalde (Git Bash veya VSCode Terminal) adım adım çalıştırarak Deepseek modelimizi kendi verilerimiz vasıtasıyla bilgilendirip istediğimiz bilgi ve yoruma kendi öz kaynaklarımızla ulaşacağız.

1. Python Ortamını Hazırla : pip install llama-cpp-python llama-index langchain
2. Dosya Yapısını Kur :
   Terminalde veya dosya gezgininde bir klasör oluştur:
   mkdir deepseek_rag
   cd deepseek_rag

Ardından terminalde:

mkdir verilerim

verilerim klasörünün içine kendi .txt ya da .pdf belgelerini kopyala.
Örn: notlar.txt, rapor.pdf gibi.

3. Python Kodunu Yaz
   code deepseek_rag.py
   İçine şu kodu **yapıştır**:

```
from llama_index import VectorStoreIndex, SimpleDirectoryReader,
ServiceContext
from llama_index.llms import LlamaCPP

# GGUF model yolunu buraya yaz (LM Studio'dan indirdiğin modelin gerçek
konumu)
MODEL_PATH = "C:\Users\ismail\.lmstudio\models\lmstudio-
community\DeepSeek-R1-Distill-Qwen-7B-GGUF"

# LLM motorunu başlat
llm = LlamaCPP(
    model_path=MODEL_PATH,
    temperature=0.5,
    max_new_tokens=512,
    context_window=4096
)

# Belgeleri yükle (verilerim klasörüne koyduğun PDF/TXT dosyaları)
documents = SimpleDirectoryReader("verilerim/").load_data()

# Servis context'i oluştur
service_context = ServiceContext.from_defaults(llm=llm)

# Vektör index'i oluştur
index = VectorStoreIndex.from_documents(documents,
service_context=service_context)

# Sorgu motoru
query_engine = index.as_query_engine()

# Soru sor
query = "Bu dökümanlar bana ne anlatıyor?"
response = query_engine.query(query)

print("\n Yanıt:\n", response)
```

4. Çalıştırmak İçin:

Terminalde şunu yaz: python deepseek_rag.py

Bu yönerge ile Python kullanarak çeşitli terminaller vasıtasıyla indirdiğimiz modele kendi dosyalarımızla yol gösterebiliriz.

Klasik prompt yönteminden daha iyi sonuçlar versede tam bir şekilde özgür custom haline getiremedik.Bunun için bir yöntemimiz daha var.

3. Fine-Tuning / LoRA / QLoRA – İleri Seviye Özelleştirme

Fine-tuning, büyük bir dil modelini (LLM) kendi verilerinle tekrar eğiterek **özelleştirmen** anlamına gelir.
Ama tam modeli eğitmek çok RAM ve zaman ister. İşte burada devreye **LoRA** ve onun daha hafif versiyonu **QLoRA** girer.

Fine-Tuning Yöntemleri

A) Tam Fine-Tuning

- Tüm model parametreleri yeniden eğitilir.
- **Çok yüksek VRAM gerekir** (30GB+)
- Genelde akademik ya da kurum düzeyinde yapılır.

B) LoRA (Low-Rank Adaptation)

- Modelin tamamını eğitmek yerine **bazı katmanlara küçük "adapter" modüller eklenir**.
- VRAM dostudur (12-16GB ile yapılabilir).
- Hugging Face peft kütüphanesi ile yapılır.

 C) QLoRA (Quantized LoRA)

- 4-bit veya 8-bit quantize edilmiş model üzerine LoRA uygulanır.
- **Çok düşük RAM ile çalışabilir** (8GB VRAM bile yeterli olabilir!)
- Yavaş ama en garanti çözüm.