

Ödev 4: İkinci kısım

Busenur Gökler

Chatbot için teknik gereksinimlerin sağlanarak kurulumların yapılması. İlk denemelerin yapılması.

Github: <https://github.com/Buse-ng/newmind-ai-bootcamp-learning/tree/main/homeworks>

Proje Kurulumu ve API Entegrasyonu

Projenin başlangıcında, OpenAI platformundan bir API key aldım. Bu API key ve projenin gizli bilgilerini güvenli bir şekilde saklamak amacıyla bir .env dosyasına kaydettim. Jupyter Notebook (.ipynb) dosyası içerisinde, bu .env dosyasında tanımlanan API key'e erişim sağlanarak OpenAI servislerinin kullanımı mümkün hale getirildi.

NAME	SECRET KEY
bootcamp	sk-...SkwA

.env Dosyası İçeriği

.env dosyası, OpenAI API key ve Neo4j database bağlantısı için gerekli olan özel bilgileri (URI, kullanıcı adı ve şifre) içermektedir.

requirements.txt Dosyası

Projede kullanılacak Python kütüphaneleri requirements.txt dosyasına eklenmiştir. Ardından, bu dosya kullanılarak gerekli kütüphanelerin kurulumu model_preparing.ipynb dosyasında gerçekleştirilmiştir.

Paket Açıklamaları

- **langchain**: LangChain framework'ü
- **langchain-core**: LangChain'in temel bileşenleri.
- **langchain-community**: Topluluk entegrasyonları.
- **langchain-openai**: OpenAI modelleri için LangChain entegrasyonu.
- **langchain-neo4j**: Neo4j veritabanı entegrasyonu.
- **openai**: OpenAI API'sine doğrudan erişim.
- **tiktoken**: OpenAI tokenizer'ı.
- **neo4j**: Neo4j Python driver'ı.
- **python-dotenv**: ortam değişkenlerini .env dosyasından yüklemek için.

Dosyalama Sistemim ve dosyalarım:

▼ **HOMEWORKS_4**

- ⚙️ .env
- 🔒 .gitignore
- 🔄 model_preparing....
- 📄 requirements.txt

⚙️ .env

```
1 OPENAI_API_KEY="sk-
  KDX_NUfUbWUE103963g
2
3 NEO4J_URI="bolt://5
4 NEO4J_USERNAME="neo
5 NEO4J_PASSWORD="bil
```

📄 requirements.txt

```
1 # Core LangChain packages
2 langchain
3 langchain-core
4 langchain-community
5 langchain-openai
6 langchain-neo4j
7
8 # OpenAI integration
9 openai
10 tiktoken
11
12 # Neo4j database
13 neo4j
14
15 # Environment management
16 python-dotenv
17
```

Model Hazırlığı ve Uygulama (model_preparing.ipynb)

İlk olarak, requirements.txt dosyasında belirtilen kütüphaneler yüklenmiştir. Ardından, dotenv kütüphanesi kullanılarak .env dosyasındaki ortam değişkenlerine erişim sağlanmıştır.

```
import os
from dotenv import load_dotenv
load_dotenv()

✓ 0.0s
```

langchain_neo4j kütüphanesi aracılığıyla Neo4j database'e bağlantı kurulmuştur. Bağlantı bilgileri (URI, kullanıcı adı, şifre) .env dosyasından çekilmiştir.

```
from langchain_neo4j import Neo4jGraph

graph = Neo4jGraph(
    url = os.getenv("NEO4J_URI"),
    username = os.getenv("NEO4J_USERNAME"),
    password = os.getenv("NEO4J_PASSWORD"),
    database="neo4j"
)

✓ 20.0s
```

langchain_openai kütüphanesi kullanılarak LLM başlatılmıştır. Bu aşamada, .env dosyasından alınan OpenAI API key ve "gpt-4.1-nano-2025-04-14" modeli kullanılmıştır.

Kullanıcı ile etkileşim kurmak için bir prompt template (ChatPromptTemplate) oluşturulmuştur. Bu template'de, sisteme "Sen bir gurmesin." rolü atanmış ve kullanıcıdan gelecek soru için bir {question} tanımlanmıştır.

Oluşturulan prompt template, OpenAI language modeli ve bir StrOutputParser birleştirilerek bir chat_chain meydana getirildi.

```
from langchain_core.prompts import ChatPromptTemplate
from langchain.schema import StrOutputParser
from langchain_openai import ChatOpenAI
```

✓ 0.0s

```
llm = ChatOpenAI(
    openai_api_key=os.getenv("OPENAI_API_KEY"),
    model="gpt-4.1-nano-2025-04-14"
)
prompt = ChatPromptTemplate.from_messages(
    [
        (
            "system",
            "Sen bir gurmesin.",
        ),
        (
            "human",
            "{question}"
        ),
    ]
)

chat_chain = prompt | llm | StrOutputParser()
response = chat_chain.invoke({"question": "Sence en hafif tatlı nedir?"})
print(response)
```

✓ 4.3s

En hafif tatlılar arasında genellikle meyve bazlı tatlılar öne çıkar. Örneğin, meyve salataları,

OpenAI API'leri arasında fiyatı en düşük olan model **GPT-4.1 nano**'dur. Bu nedenle deneme projesi olduğu için bu modeli kullanmaya karar verdim. Modelin fiyatlandırması şu şekildedir:

- **Input:** 1 milyon token başına \$0.1
- **Cached input:** 1 milyon token başına \$0.025
- **Output:** 1 milyon token başına \$0.4