Proje Raporu: Web Trafik Loglarına Dayalı Yapay Zeka Destekli Soru-Cevap Sistemi Geliştirme

Bu projede, web sitesi trafik loglarını kullanarak bir yapay zeka destekli soru-cevap (Q&A) sistemi geliştirdim. Bu sistem, web sunucusundan gelen günlük kayıtlarını analiz ederek kullanıcılardan gelen doğal dildeki sorulara anlamlı ve doğru cevaplar vermeyi amaçlamaktadır. Projenin temelinde, Retrieval-Augmented Generation (RAG) modeli yer alıyor. Bu model, iki ana bileşeni bir araya getirir: bilgi alma ve jeneratif model. Bilgi alma aşamasında, log verileri vektörlere dönüştürülerek uygun bir vektör veri tabanına yükledim ve kullanıcı sorularına en uygun kayıtları bu veri tabanından çektim. Jeneratif model aşamasında ise, elde edilen log kayıtları kullanılarak dil modeli (örneğin, GPT veya T5) aracılığıyla soruya uygun cevaplar oluşturuldu. Bu proje, verinin analizi, model entegrasyonu ve sistem performansının değerlendirilmesi gibi adımları içeriyor. Amacım, bu süreçler boyunca karşılaşılan zorlukları aşarak etkili ve doğru sonuçlar elde etmek ve sonuçların kalitesini artıracak iyileştirmeler önermektir.

**Aşama 1: Veri Hazırlığı ve Ön İşleme**

!pip install faker

import random

from faker import Faker

from datetime import datetime, timedelt

Gerekli kütüphanelerin içe aktarımı:

* **random**: Rastgele seçimler yapmak için kullanılır (örneğin, HTTP yöntemleri, durum kodları).
* **Faker**: Sahte veriler üretir (örneğin, IP adresleri, kullanıcı ajanları).
* **datetime**: Tarih ve saat bilgilerini yönetir.
* **timedelta**: Tarih ve saat hesaplamaları yapar.

fake = Faker()

LOG\_LINE\_COUNT = 20

HTTP\_METHODS = ['GET', 'POST', 'PUT', 'DELETE']

HTTP\_STATUS\_CODES = [200, 201, 301, 400, 401, 403, 404, 500, 502, 503]

URLS = [

    '/home',

    '/about',

    '/contact',

    '/products',

    '/products/item1',

    '/products/item2',

    '/api/data',

    '/login',

    '/signup'

]

USER\_AGENTS = [

    fake.firefox,

    fake.chrome,

    fake.safari,

    fake.internet\_explorer,

    fake.opera

]

def generate\_log\_line():

    ip\_address = fake.ipv4()

    identity = '-'

    userid = '-'

    time = (datetime.now() - timedelta(seconds=random.randint(0, 86400))).strftime('%d/%b/%Y:%H:%M:%S %z')

    method = random.choice(HTTP\_METHODS)

    resource = random.choice(URLS)

    protocol = 'HTTP/1.1'

    status\_code = random.choice(HTTP\_STATUS\_CODES)

    response\_size = random.randint(200, 5000)

    referrer = fake.url()

    user\_agent = random.choice(USER\_AGENTS)()

    log\_line = f'{ip\_address} {identity} {userid} [{time}] "{method} {resource} {protocol}" {status\_code} {response\_size} "{referrer}" "{user\_agent}"'

    return log\_line

with open('web\_traffic.log', 'w') as log\_file:

    for \_ in range(LOG\_LINE\_COUNT):

        log\_line = generate\_log\_line()

        log\_file.write(log\_line + '\n')

Bu adımda, Faker kütüphanesi kullanılarak sahte veri üretilmiş ve bu veriler bir log dosyasında depolanmıştır.

Oluşturulacak log satırlarının sayısı LOG\_LINE\_COUNT değişkeni ile belirlenmiştir. Ayrıca, HTTP yöntemleri (GET, POST, PUT, DELETE), HTTP durum kodları (200, 404, 500, vb.), URL'ler (/home, /products/item1, vb.) ve kullanıcı ajanları (firefox, chrome, vb.) gibi parametreler tanımlanmıştır.

generate\_log\_line fonksiyonu, her bir log satırını üretir. Fonksiyonun işleyişi şu şekildedir:

* Rastgele bir IP adresi ve zaman damgası oluşturulur.
* Rastgele bir HTTP yöntemi, URL, durum kodu ve yanıt boyutu seçilir.
* Sahte bir referans URL'si ve kullanıcı ajanı oluşturulur.
* Bu bilgiler bir log satırı formatında birleştirilir ve geri döndürülür.

Oluşturulan log satırları web\_traffic.log adlı bir dosyaya yazılır. Dosya, belirtilen sayıda log satırı içerir ve bu veriler, web trafik analizi ve sistem testleri için kullanılabilir.

import pandas as pd

import re

with open("web\_traffic.log") as file:

    log\_lines = file.readlines()

log\_entries = []

for line in log\_lines:

    match = re.match(r'(\S+) - - \[(.\*?)\] "(.\*?)" (\d{3}) (\d+) "(.\*?)" "(.\*?)"', line)

    if match:

        log\_entries.append({

            "ip": match.group(1),

            "timestamp": match.group(2),

            "request": match.group(3),

            "status\_code": match.group(4),

            "response\_size": match.group(5),

            "referrer": match.group(6),

            "user\_agent": match.group(7),

        })

df\_logs = pd.DataFrame(log\_entries)

df\_logs.head()

print(df\_logs)

Bu adımda, web\_traffic.log dosyasındaki web trafik log verileri okunmuş ve işlenmiştir.

* **Log Dosyasının Okunması:** web\_traffic.log dosyası okunarak, log satırları bir listeye alınmıştır.
* **Log Satırlarının Analizi:** Her bir log satırı düzenli ifadeler (regex) kullanılarak analiz edilmiştir. Bu işlem, log satırındaki IP adresi, zaman damgası, HTTP isteği, durum kodu, yanıt boyutu, referans URL ve kullanıcı ajanı gibi bilgileri ayıklamayı sağlar.
* **Verilerin Yapılandırılması:** Elde edilen bilgiler, bir sözlük (dictionary) yapısına dönüştürülerek bir listeye eklenmiştir.
* **DataFrame Oluşturulması:** İşlenen log verileri, pandas kütüphanesi kullanılarak bir DataFrame'e dönüştürülmüştür. Bu DataFrame, verilerin daha kolay analiz edilmesini sağlar ve ilk birkaç satırı df\_logs.head() fonksiyonu ile görüntülenmiştir.

print(df\_logs.isnull().sum())

df\_logs = df\_logs.dropna()

df\_logs["status\_code"] = df\_logs["status\_code"].astype(int)

df\_logs.head()

Veri temizliği aşamasında, önce eksik değerler kontrol edilip, eksik veri içeren satırlar DataFrame'den çıkarılmıştır. Ardından, durum kodları integer veri tipine dönüştürülmüştür. Sonuç olarak, temizlenmiş veriler df\_logs.head() fonksiyonu ile görüntülenmiştir. Bu işlemler, verinin doğruluğunu ve analizin geçerliliğini artırmak için yapılmıştır.

df\_logs.to\_json("cleaned\_web\_traffic.json", orient="records", lines=True)

Temizlenmiş log verileri, JSON formatında bir dosyaya (cleaned\_web\_traffic.json) kaydedilmiştir. Bu işlem, verilerin saklanmasını ve diğer uygulamalarla paylaşılmasını kolaylaştırır. JSON dosyası, her bir log kaydını bir satır olarak içerecek şekilde yapılandırılmıştır.

from sklearn.feature\_extraction.text import TfidfVectorizer

df\_logs["combined"] = (

    df\_logs["ip"] + " " +

    df\_logs["timestamp"] + " " +

    df\_logs["request"] + " " +

    df\_logs["status\_code"].astype(str) + " " +

    df\_logs["response\_size"].astype(str) + " " +

    df\_logs["referrer"] + " " +

    df\_logs["user\_agent"]

)

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(df\_logs["combined"])

print(X.shape)

print(X)

Veri hazırlık sürecinde, log dosyasındaki URL'ler ve kullanıcı ajanları birleştirilerek tek bir metin sütunu oluşturulmuştur. Bu metin sütunu, TfidfVectorizer kullanılarak sayısal vektörlere dönüştürülmüştür. TF-IDF vektörizasyonu, kelimelerin önemini hesaplar ve bu metinleri sayısal forma getirir. Sonuç olarak, vektörlerin boyutu ve içerikleri ekrana yazdırılmıştır.

!pip install faiss-cpu

import faiss

index = faiss.IndexFlatL2(X.shape[1])

index.add(X.toarray())

faiss.write\_index(index, "faiss\_index.index")

FAISS kullanarak bir vektör arama indexi oluşturulmuştur. İlk olarak, faiss-cpu kütüphanesi yüklenmiş ve FAISS modülü import edilmiştir. Ardından, TF-IDF vektörlerinin eklenmesi için bir FAISS indexi oluşturulmuş ve bu indexe vektörler eklenmiştir. Son olarak, oluşturulan FAISS indexi "faiss\_index.index" dosyasına kaydedilmiştir.

import numpy as np

from sklearn.feature\_extraction.text import TfidfVectorizer

def vectorize\_query(query, vectorizer):

    return vectorizer.transform([query]).toarray()

def retrieve\_top\_k(query\_vector, index, k=1):

    D, I = index.search(query\_vector, k)

    return I[0], D[0]

user\_query = "Hangi sayfalar 500  status kodunu almıştır ?"

query\_vector = vectorize\_query(user\_query, vectorizer)

indices, distances = retrieve\_top\_k(query\_vector, index)

matching\_logs = df\_logs.iloc[indices]

matching\_logs

Bu adımda, kullanıcıdan gelen bir sorgu vektör haline dönüştürülmüştür. vectorize\_query fonksiyonu, sorguyu TF-IDF vektörizasyonu kullanarak vektörleştirmiştir. Ardından, bu sorgu vektörü FAISS indexinde arama yapılmış ve en benzer vektörler bulunmuştur. retrieve\_top\_k fonksiyonu, en yakın 5 vektörü bularak ilgili log kayıtlarını seçmiştir. Sonuç olarak, en uygun log kayıtları matching\_logs veri çerçevesinde görüntülenmiştir.

**Aşama 2: RAG Modelinin Kurulumu**

!pip install transformers

!pip install torch

from transformers import T5ForConditionalGeneration, T5Tokenizer

import pandas as pd

model\_name = "t5-small"

model = T5ForConditionalGeneration.from\_pretrained(model\_name)

tokenizer = T5Tokenizer.from\_pretrained(model\_name)

df\_logs = pd.DataFrame(log\_entries)

def generate\_response(logs, user\_query):

    prompt = f"Kullanıcı sorusu: {user\_query}\n"

    prompt += "İlgili log kayıtları:\n"

    for log in logs.to\_dict(orient="records"):

        prompt += f"{log['timestamp']} - {log['request']} - {log['status\_code']}\n"

    prompt += "\nBu bilgilere dayanarak, kullanıcı sorusuna yanıt verin."

    inputs = tokenizer.encode(prompt, return\_tensors="pt", truncation=True, max\_length=512)

    outputs = model.generate(inputs, max\_length=150, num\_beams=4, early\_stopping=True)

    response = tokenizer.decode(outputs[0], skip\_special\_tokens=True)

    return response

Bu adımda, T5 modelini kullanarak kullanıcı sorgusuna uygun yanıtlar oluşturmak için bir sistem kurulmuştur. İlk olarak, T5 modelini ve tokenizer'ını yükleyip yapılandırılmıştır. Ardından, kullanıcı sorgusu ve ilgili log kayıtlarını içeren bir giriş metni oluşturulmuştur. Bu metin, T5 modelinin anlayabileceği şekilde tokenlere dönüştürülüp modelin yanıt üretme fonksiyonu kullanılarak bir yanıt üretilmiştir. Son olarak, modelden alınan yanıt dekode edilerek kullanıcıya sunulacak şekilde düzenlenmiştir.

**Aşama 3: Sistem Entegrasyonu ve Test**

def rag\_system(user\_query, df\_logs):

    response = generate\_response(df\_logs, user\_query)

    return response

user\_query = "500 hata kodları hakkında bilgi ver"

response = rag\_system(user\_query, df\_logs)

print("Yanıt:", response)

Bu adımda, Retrieval-Augmented Generation (RAG) sistemini çalıştırarak kullanıcı sorgusuna yanıt üretmek için bir fonksiyon tanımlanmıştır. rag\_system fonksiyonu, kullanıcı sorgusunu ve log verilerini alır, generate\_response fonksiyonunu kullanarak uygun yanıtı oluşturur ve sonucu döndürür. Örnek bir kullanıcı sorgusu verildiğinde, sistem çalıştırılır ve yanıt ekrana yazdırılır. Bu işlem, modelin log verilerini kullanarak kullanıcı sorgularına yanıt vermesini sağlar.

def command\_line\_interface():

    print("Web Trafik Loglarına Dayalı Soru-Cevap Sistemi")

    print("Çıkmak için 'exit' yazın.")

    while True:

        user\_query = input("\nSorunuzu yazın: ")

        if user\_query.lower() in ['exit', 'quit']:

            print("Çıkılıyor...")

            break

        response = rag\_system(user\_query,df\_logs)

        print("\nYanıt:", response)

command\_line\_interface()

Web Trafik Loglarına Dayalı Soru-Cevap Sistemi

Çıkmak için 'exit' yazın.

Sorunuzu yazın: 404 hatası veren sayfalar nelerdir?

Yanıt: : 404 hatas veren sayfalar nelerdir? lgili log kaytlar: 20/Aug/2024:19:07:47 - POST /products/item2 HTTP/1.1 - 401 21/Aug/2024:17:12:15 - PUT /products/item2 HTTP/1.1 - 301 21/Aug/2024:16:53:49

Sorunuzu yazın: "Son ziyaret edilen sayfalar hangileri?"

Yanıt: : "Son ziyaret edilen sayfalar hangileri?" Kullanc sorusu: "Son ziyaret edilen sayfalar hangileri?" lgili log kaytlar: 20/Aug/2024:19:07:47 - POST /products/item2 HTTP/1.1 - 401 21/Aug/2024:17:12:

Sorunuzu yazın: exit

Çıkılıyor...

Bu adımda, kullanıcıların komut satırından sorgularını girebileceği bir arayüz oluşturulmuştur. command\_line\_interface fonksiyonu, kullanıcıdan sürekli olarak soru alır ve 'exit' veya 'quit' komutları ile çıkış yapana kadar çalışır. Kullanıcıdan alınan her sorgu, rag\_system fonksiyonu aracılığıyla işlenir ve ilgili yanıt ekrana yazdırılır. Bu arayüz, kullanıcıların doğal dildeki sorularına gerçek zamanlı yanıt almasını sağlar.

test\_queries = [

    "503 status code veren sayfalar nelerdir?",

    "Son ziyaret edilen sayfalar hangileri?",

    "Siteye erişim tarihleri nelerdir?",

    "En yüksek yanıt boyutuna sahip istekler nelerdir?",

    "Hangi HTTP yöntemleri en sık kullanılmıştır?",

    "En fazla 403 hata kodu alan istekler nelerdir?",

]

print("Test Senaryoları Başlatılıyor...\n")

for query in test\_queries:

    print(f"Soru: {query}")

    response = rag\_system(query, df\_logs)

    print("Yanıt:", response)

    print("\n" + "-"\*50 + "\n")

print("Testler Tamamlandı.")

Bu adımda, sistemin performansını değerlendirmek için çeşitli test senaryoları uygulanmıştır. Önceden belirlenen test sorguları, rag\_system fonksiyonu kullanılarak işlendi ve her bir sorguya karşılık gelen yanıtlar alındı. Elde edilen yanıtlar ekrana yazdırıldı, bu sayede sistemin farklı sorulara nasıl tepki verdiği ve yanıtlarının doğruluğu değerlendirildi. Bu testler, sistemin genel işlevselliğini ve yanıt kalitesini ölçmeyi amaçlamaktadır.

**Aşama 4: Performans Değerlendirmesi**

evaluation\_queries = {

    "503 status code veren sayfalar nelerdir?": "/home",

    "Son ziyaret edilen sayfalar hangileri?": "/home",

    "Siteye erişim tarihleri nelerdir?": "21/Aug/2024, 22/Aug/2024",

    "En yüksek yanıt boyutuna sahip istekler nelerdir?": "/home, /signup",

    "Hangi HTTP yöntemleri en sık kullanılmıştır?": "PUT, DELETE",

    "En fazla 403 hata kodu alan istekler nelerdir?": "PUT /signup"

}

correct\_answers = 0

total\_queries = len(evaluation\_queries)

print("Doğruluk Değerlendirmesi Başlıyor...\n")

for query, expected\_answer in evaluation\_queries.items():

    print(f"Soru: {query}")

    response = rag\_system(query, df\_logs)

    print("Yanıt:", response)

    if expected\_answer.lower() in response.lower():

        print("Doğru")

        correct\_answers += 1

    else:

        print("Yanlış")

    print("\n" + "-"\*50 + "\n")

accuracy = correct\_answers / total\_queries

print(f"Doğruluk: {accuracy \* 100:.2f}%")

Bu kod, geliştirilen RAG sisteminin doğruluğunu değerlendirmek için kullanılır. Önce belirli örnek senaryolar ve beklenen yanıtlar tanımlanır. Sonra, bu senaryolar üzerinden sistem test edilir ve yanıtlar, beklenen yanıtlarla karşılaştırılır. Her bir test senaryosunda yanıtın doğruluğu kontrol edilir ve doğru yanıt sayısı hesaplanır. Son olarak, doğru yanıtların toplam test senaryolarına oranı hesaplanarak sistemin genel doğruluk oranı belirlenir ve ekranda gösterilir.

* **Sistemi Kurarken Karşılaştığım Zorluklar:**

Bu projede web trafik log dosyalarıyla ilk kez çalışma deneyimim oldu.Projede en çok zorlandığım kısımlar, veri hazırlığı aşamasında log dosyalarının çeşitliliği ve eksik verilerle başa çıkmaktı. FAISS kütüphanesi ile vektör veri tabanı oluştururken performans sorunları yaşadım. RAG modelini entegre etmek de biraz karmaşık oldu çünkü bilgi alma ve jeneratif model bileşenlerini uyumlu hale getirmek zaman aldı. Yanıtların kalitesini iyileştirmek için model üzerinde sürekli denemeler yapmam gerekti. Performans değerlendirmesinde ise doğru yanıtları belirlemek ve objektif ölçümler yapmak beni biraz zorladı. Bu süreç, dikkatli çalışmayı ve sürekli iyileştirme yapmayı gerektirdi.

* **Sistemin Doğruluğu ve Performansı:**

Sistemin doğruluğunu değerlendirdim ve test sonuçlarına göre doğruluk oranı %50 olarak belirlendi. Bazı sorulara doğru yanıt verildi, ancak özellikle tarih ve HTTP yöntemleri gibi konularda sistem beklenen performansı gösteremedi. Bu, projenin hala geliştirme aşamasında olduğunu ve daha fazla iyileştirme gerektirdiğini gösteriyor. Bu sonuç, sistemin genel doğruluğunun artırılması gerektiğine işaret ediyor ve ilerleyen süreçte yapılacak optimizasyonlar için yol gösterici olacak.

* **Sistemin Cevaplarının Kalitesini Artırmak İçin Yapılabilecek İyileştirmeler:**

Sistemin cevaplarının kalitesini artırmak için birkaç iyileştirme yapabiliriz. Özellikle, “503 status code veren sayfalar nelerdir?” ve “Son ziyaret edilen sayfalar hangileri?” gibi sorularda doğru sonuçlar aldık. Ancak “Siteye erişim tarihleri nelerdir?” ve “En yüksek yanıt boyutuna sahip istekler nelerdir?” gibi sorularda daha net ve doğru yanıtlar almak için sistemin analiz yeteneklerini geliştirmemiz gerekiyor. Yanıtların doğruluğunu artırmak adına, veri analizini ve sistemin algoritmalarını gözden geçirebilir,verilere uygun sorular soraraksistemi iyileştirebiliriz. Bu sayede, sistemin daha kesin ve faydalı cevaplar verebilmesini sağlarız.