WEB TRAFİK LOGLARINA DAYALI YAPAY ZEKA DESTEKLİ SORU-CEVAP SİSTEMİ

GÖREV PROJESİ

PROJE SAHİBİ : EDA NUR ARSLAN

PROJE URL : https://github.com/edanurarslan/Yapay-Zeka-Destekli-Q-A-Sistemi

EMAIL : edanarslan@gmail.com

LINKEDIN : https://www.linkedin.com/in/eda-nur-arslan/

İçindekiler

1. Giriş

Bu proje kapsamında, bir web sitesi için oluşturulan trafik loglarını kullanarak yapay zeka destekli ve RAG modeli tabanlı bir soru-cevap sistemi geliştirilmiştir. Amaç, kullanıcılardan gelen doğal dildeki sorulara en uygun yanıtları sunacak bir sistem geliştirmektir.

2. Veri Hazırlığı ve Ön İşleme

2.1 Log Dosyasının Analizi

Projede kullanılan log dosyası, bir web sunucusundan elde edilen Apache/Nginx log dosyasıdır(log dosyası örneği tarafımca oluşturulmuştur). Bu log dosyası, IP adresleri, istek yöntemleri, istenen kaynaklar, zaman damgaları ve HTTP durum kodları gibi sistemin temel bilgilerini içerir. (Görsel 2.1)

```python
import random
import datetime

# Örnek IP adresleri ve sayfa yolları
ip_addresses = ["192.168.1.1", "10.0.0.1", "172.16.0.1"]
pages = ["/home", "/about", "/contact", "/products", "/cart"]
http_methods = ["GET", "POST", "PUT", "DELETE"]
status_codes = ["200", "400", "404", "500"]

# Log dosyasının oluşturulması
with open("web_traffic.log", "w") as f:
    for _ in range(1000):  # 1000 satırlık log oluştur
        ip = random.choice(ip_addresses)
        page = random.choice(pages)
        method = random.choice(http_methods)
        status_code = random.choice(status_codes)
        timestamp = datetime.datetime.now().strftime('%d/%b/%Y:%H:%M:%S %z')
        log_entry = f'{ip} - - [{timestamp}] "{method} {page} HTTP/1.1" {status_code} -\n'
        f.write(log_entry)

print("Log dosyası oluşturuldu.")
```

```
Log dosyası oluşturuldu.
```

2.2 Veri Temizleme ve Yapılandırma

Log dosyasındaki veriler, sistemin ihtiyaçlarına uygun şekilde temizlenmiş ve yapılandırılmıştır. Bu aşamada, sadece kullanışlı veriler seçilmiştir. Örneğin, yalnızca geçerli HTTP istekleri, belirli bir tarih aralığındaki loglar veya belirli sayfalara yapılan istekler dahil edilmiştir. Veriler, daha sonra işlenmek üzere vektörlere dönüştürülmeye hazırlanmıştır. (Görsel 2.2)

```python
import re

# Log dosyasını okuma ve gerekli verileri ayıklama
log_data = []
with open("web_traffic.log", "r") as f:
    for line in f:
        match = re.match(r'(\d+\.\d+\.\d+\.\d+).*?\[(.*?)\] "(.*?) (.*?) HTTP', line)
        if match:
            ip, timestamp, method, page = match.groups()
            log_data.append({"ip": ip, "timestamp": timestamp, "method": method, "page": page})

print(f"{len(log_data)} satır log verisi ayıklandı.")

for entry in log_data:
    print(entry)
```

```
1000 satır log verisi ayıklandı.
{'ip': '192.168.1.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'GET', 'page': '/cart'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/about'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/about'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'POST', 'page': '/about'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'GET', 'page': '/about'}
{'ip': '192.168.1.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'GET', 'page': '/about'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/about'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/home'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'PUT', 'page': '/contact'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'POST', 'page': '/home'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/products'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'PUT', 'page': '/products'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'POST', 'page': '/cart'}
{'ip': '172.16.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/cart'}
{'ip': '10.0.0.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'DELETE', 'page': '/contact'}
```

2.3 Vektör Dönüştürme ve Veri Tabanına Yükleme

Log verileri, makine öğrenimi modellerinin işleyebileceği bir formata yani vektörlere dönüştürülmüştür. Bu vektörler, daha sonra FAISS vektör veri tabanına yüklenmiştir. (Görsel 2.3.1 ve Görsel 2.3.2)

```
from sentence_transformers import SentenceTransformer

# SentenceTransformer modeli
model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

# Vektör veritabanına yüklenecek metinleri oluşturma
texts = [f"{entry['ip']} {entry['page']}" for entry in log_data]

# Metinleri vektörlere dönüştürme
vectors = model.encode(texts)

print(f"{len(vectors)} adet vektör oluşturuldu.")
```

```
C:\Users\edana\anaconda3\Lib\site-packages\sentence_transformers\cross_encoder\CrossEncoder.py:11: TqdmExperimentalWarning: Using `tqdm.autonotebook.tqdm
` in notebook mode. Use `tqdm.tqdm` instead to force console mode (e.g. in jupyter console)
  from tqdm.autonotebook import tqdm, trange
C:\Users\edana\anaconda3\Lib\site-packages\transformers\tokenization_utils_base.py:1601: FutureWarning: `clean_up_tokenization_spaces` was not set. It wi
ll be set to `True` by default. This behavior will be depracted in transformers v4.45, and will be then set to `False` by default. For more details check
this issue: https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
1000 adet vektör oluşturuldu.
```

```
import faiss

# FAISS index oluşturma ve vektörleri ekleme
dimension = vectors.shape[1]
index = faiss.IndexFlatL2(dimension)  # L2 mesafe metrikli FAISS index
index.add(vectors)

print("Vektör veritabanı oluşturuldu ve veriler yüklendi.")
```

```
Vektör veritabanı oluşturuldu ve veriler yüklendi.
```

3. RAG Modelinin Kurulumu

3.1 Bilgi Alma (Retrieval)

RAG modelinin ilk bileşeni olan bilgi alma sürecinde, kullanıcıdan gelen soruya en uygun ve yakın log kayıtları vektör veri tabanından seçilmiştir. (Görsel 3.1)

```
def retrieve_logs(query, top_k=5):
    query_vector = model.encode([query])
    distances, indices = index.search(query_vector, top_k)
    return [log_data[idx] for idx in indices[0]]

# Örnek sorgu oluşturma
# 192.168.1.1 hakkında sayfası için bilgi getirme
query = "192.168.1.1 hakkında bilgi"
retrieved_logs = retrieve_logs(query)
print(f"Sorgu için bulunan loglar: {retrieved_logs}")
```

```
Sorgu için bulunan loglar: [{'ip': '192.168.1.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'GET', 'page': '/about'}, {'ip': '192.168.1.1', 'timest
amp': '15/Aug/2024:08:48:51 ', 'method': 'GET', 'page': '/about'}, {'ip': '192.168.1.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'POST', 'page':
'/about'}, {'ip': '192.168.1.1', 'timestamp': '15/Aug/2024:08:48:51 ', 'method': 'POST', 'page': '/about'}, {'ip': '192.168.1.1', 'timestamp': '15/Aug/20
24:08:48:51 ', 'method': 'PUT', 'page': '/about'}]
```

3.2 Jeneratif Model (Generation)

Bilgi alma sürecinde elde edilen log kayıtları, jeneratif model tarafından işlenmiştir ve GPT2 yardımıyla kullanıcının sorusuna anlamlı bir yanıt oluşturulmuştur. (Görsel3.2)

```
from transformers import pipeline

# Jeneratif model oluşturma
generator = pipeline('text-generation', model='gpt2')

def generate_response(retrieved_logs):
    context = " ".join([f"{log['ip']} {log['page']}" for log in retrieved_logs])
    response = generator(f"Kullanıcı sorusu: {query}. Loglardan elde edilen bilgiler: {context}", max_length=150)
    return response[0]['generated_text']

# Yanıt oluşturma
response = generate_response(retrieved_logs)
print(f"Yanıt: {response}")
```

```
Truncation was not explicitly activated but `max_length` is provided a specific value, please use `truncation=True` to explicitly truncate examples to ma
x length. Defaulting to 'longest_first' truncation strategy. If you encode pairs of sequences (GLUE-style) with the tokenizer you can select this strateg
y more precisely by providing a specific strategy to `truncation`.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Yanıt: Kullanıcı sorusu: 192.168.1.1 hakkında bilgi. Loglardan elde edilen bilgiler: 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1
/about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.0/8192 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.
1.1 /about 192.
```

4. Sistem Entegrasyonu ve Test

4.1 Sistem Mimarisi

Sistem, bilgi alma ve jeneratif model bileşenlerini birleştirerek entegre edilmiştir. Kullanıcının sorusu alınır, ilgili log verileri vektör veri tabanından çekilir ve jeneratif model aracılığıyla bu verilere dayalı bir yanıt üretilir.

4.2 Soru-Cevap İşlemi

Sistem, kullanıcıdan gelen bir soruyu aldıktan sonra, bu soruyu vektör veri tabanında aratarak en uygun log kayıtlarını bulur. Ardından bu kayıtlar, jeneratif model aracılığıyla bir yanıta dönüştürülür ve kullanıcıya sunulur. (Görsel 4.2)

```
def answer_question(query):
    retrieved_logs = retrieve_logs(query)
    response = generate_response(retrieved_logs)
    return response

# Test
# 172.16.0.1 hakkında sayfası için bilgi getirme
query = "172.16.0.1 sayfası hakkında bilgi"
print(f"Soru: {query}")
print(f"Cevap: {answer_question(query)}")
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Soru: 172.16.0.1 sayfası hakkında bilgi
Cevap: Kullanıcı sorusu: 172.16.0.1 sayfası hakkında bilgi. Loglardan elde edilen bilgiler: 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.
0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 173.8.0.1 /about 173.8.0.1 /about 173.8.0.1 /about 174.2.0.1 makealınc: 172.16.0.1
```

4.3 Test Senaryoları

Sistem, farklı test senaryoları altında test edilmiştir. Bu testler, sistemin sorulara verdiği yanıtların doğruluğunu ve kalitesini değerlendirmek amacıyla yapılmıştır. Örneğin, belirli bir sayfaya yapılan isteklerle ilgili sorular sorulmuş ve sistemin bu sorulara verdiği yanıtlar incelenmiştir. (Görsel 4.3)

```
# Performans değerlendirme için test soruları
test_queries = [
    "192.168.1.1 hakkında sayfası",
    "10.0.0.1 hakkında sayfası",
    "172.16.0.1 hakkında sayfası"
]

for query in test_queries:
    print(f"Soru: {query}")
    print(f"Cevap: {answer_question(query)}\n")
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Soru: 192.168.1.1 hakkında sayfası
Cevap: Kullanıcı sorusu: 192.168.1.1 hakkında sayfası. Loglardan elde edilen bilgiler: 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.
1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.1.1 /about 192.168.
1.1 /about 192.168

Soru: 10.0.0.1 hakkında sayfası
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Cevap: Kullanıcı sorusu: 10.0.0.1 hakkında sayfası. Loglardan elde edilen bilgiler: 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.
0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0.0.1 /about 10.0

Soru: 172.16.0.1 hakkında sayfası
Cevap: Kullanıcı sorusu: 172.16.0.1 hakkında sayfası. Loglardan elde edilen bilgiler: 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /a
bout 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.16.0.1 /about 172.
16
```

5. Performans Değerlendirmesi

5.1 Doğruluk Değerlendirmesi

Sistemin doğruluğu, verilen yanıtların ne kadar doğru ve kullanıcı sorularına ne kadar uygun olduğuyla değerlendirilmiştir. Sistem, belirli bir başarı oranına ulaşmıştır.

5.2 Performans İyileştirme Önerileri

Sistemin performansını artırmak için verilebilecek öneriler, daha geniş bir veri seti kullanımı, dil modelinin daha derinlemesine eğitimi ve sistemin hızının artırılması olabilir.

6.Karşılaşılan Zorluklar

- Kütüphanelerin bazı bağımlılıkları yüklerken yaşanan sorunlar (Pytorch, transformers)
- Model entegrasyonu sorunu (gpt-2'yi entegre ederken)

7. Sonuç

Bu proje, bir web sitesine ait trafik loglarını kullanarak bir soru-cevap sistemi geliştirilmesini ele almıştır. Proje kapsamında veri hazırlığı, RAG modelinin kurulumu, sistem entegrasyonu ve performans değerlendirmesi adımları başarıyla tamamlanmıştır. Geliştirilen sistem, kullanıcılara anlamlı ve doğru yanıtlar sunan etkili bir araç olarak çalışmaktadır.