Web Trafik Loglarına Dayalı Yapay Zeka Destekli Soru-Cevap Sistemi Geliştirme Ödevi

Rapor

- **Ödev Bitiş Tarihi:** 19 Ağustos 2024
- **Öğrenci Adı:** Mert Ali Çağcı
- **Kod Repository:** https://github.com/mertalicagci/ai-log-question-answer-system

---

1. Giriş

Bu projede, bir web sitesinin trafik loglarını analiz ederek kullanıcılardan gelen sorulara yanıt verebilen bir soru-cevap (Q&A) sistemi geliştirilmiştir. Projenin amacı, kullanıcıların sorduğu sorulara en uygun yanıtları verebilecek bir sistem tasarlamak ve geliştirmektir. Proje kapsamında Retrieval-Augmented Generation (RAG) modeli kullanılmıştır. Bu model, hem bilgi alma (retrieval) hem de jeneratif model (generative model) süreçlerini birleştirerek daha doğru ve anlamlı yanıtlar üretebilir.

Kulanıların Kütüphaneler

1. Pandas (pd)

o Açıklama: Veri manipülasyonu ve analizi için kullanılan bir kütüphanedir. DataFrame veri yapıları sağlar ve veri temizleme, işleme ve analiz işlemlerini kolaylaştırır.

2. Regex (re)

o Açıklama: Düzenli ifadeler (regex) kullanarak metinlerde arama ve eşleştirme yapmaya olanak tanır. Log verilerini ayrıştırmak için kullanılmıştır.

3. Scikit-Learn (sklearn.feature_extraction.text.TfidfVectorizer)

o Açıklama: Metin verilerini sayısal vektörlere dönüştüren TF-IDF (Term Frequency-Inverse Document Frequency) vektörleştirme yöntemi sağlar. Bu kütüphane ile HTTP isteklerinin vektörleştirilmesi sağlanmıştır.

4. FAISS (faiss)

o Açıklama: Hızlı ve etkili vektör arama ve benzerlik aramaları yapmak için kullanılan bir kütüphanedir. FAISS endeksi, metin vektörleri arasında hızlı arama yapılmasını sağlar.

5. Transformers (transformers)

o Açıklama: Doğal dil işleme (NLP) modelleri için kullanılan bir kütüphanedir. Bu kütüphane ile T5 modelinin yüklenmesi ve metin tabanlı cevapların oluşturulması sağlanmıştır.

o T5Tokenizer: T5 modeline uygun veri tokenize etmek için kullanılır.

o T5ForConditionalGeneration: T5 modelinin conditional generation (koşullu üretim) görevini yerine getiren sınıfıdır.

6. Time (time)

o Açıklama: Zamanla ilgili işlevleri kullanmak için standart bir Python kütüphanesidir. Yanıt sürelerinin ölçülmesi için kullanılmıştır.

2. Veri Hazırlığı ve Ön İşleme

2.1 Log Verilerinin İncelenmesi

```
1. import pandas as pd
 2. import re
 3.
 4. # Log verileri
 5. log_verileri = """
 6. 192.168.1.1 - - [10/Aug/2024:14:55:36 +0000] "GET /index.html HTTP/1.1" 200 1234
 7. 192.168.1.2 - - [10/Aug/2024:14:56:02 +0000] "POST /login HTTP/1.1" 200 567
 8. 192.168.1.3 - - [10/Aug/2024:14:57:10 +0000] "GET /about.html HTTP/1.1" 404 0
 9. 192.168.1.4 - - [10/Aug/2024:14:57:50 +0000] "GET /contact.html HTTP/1.1" 200 234
10. """
11.
12. # Log verilerini satırlara ayır
13. log_satirlari = log_verileri.strip().split('\n')
14.
15. # RegEx kullanarak log verilerini çıkartıp bir DataFrame oluştur
16. log_pattern = r'(\S+) - - \[(.*?)\] "(.*?)" (\d{3}) (\d+)'
17. log_girisleri = []
18.
19. for satir in log_satirlari:
20.     eslesen = re.match(log_pattern, satir)
21.     if eslesen:
22.         ip_adresi, zaman_damgasi, istek, durum_kodu, boyut = eslesen.groups()
23.         log_girisleri.append([ip_adresi, zaman_damgasi, istek, durum_kodu, boyut])
24.
25. # DataFrame oluştur
26. log_df = pd.DataFrame(log_girisleri, columns=['IP_Adresi', 'Zaman_Damgasi', 'Istek',
'Durum_Kodu', 'Boyut'])
```

2.2 Verilerin Seçimi ve Temizlenmesi

Log verilerindeki bazı satırlar hatalı veya eksik olabilir. Bu tür satırları filtreleyerek yalnızca geçerli verilerin analize dahil edilmesi sağlanmıştır. Ayrıca, verilerin kolay işlenebilmesi için düzenli ifadeler (regex) kullanılmıştır.

Kod:

```
1. # Verinin temizlenme aşaması
2. log_df = log_df.dropna()
3. log_df = log_df[log_df['IP_Adresi'].str.match(r'\d+\.\d+\.\d+\.\d+')]
4. log_df['Boyut'] = log_df['Boyut'].astype(int)
5.
```

2.3 Vektörize Etme

Log verilerinin Request sütunu, TF-IDF (Term Frequency-Inverse Document Frequency) kullanılarak vektörlere dönüştürülmüştür. Bu vektörler, daha sonra FAISS (Facebook AI Similarity Search) vektör veri tabanına yüklenmiştir. Bu, kullanıcının sorduğu sorulara en uygun log kayıtlarını hızlıca bulmamızı sağlar.

Kod:

```
 1.  1. from sklearn.feature_extraction.text import TfidfVectorizer
 2.  2. import faiss
 3.  3.
 4.  4. # Tabloda yer alan istek sütununu vektörize edilmesi
 5. vektorleyici = TfidfVectorizer()
 6. istek_vektorleri = vektorleyici.fit_transform(log_df['Istek']).toarray()
 7.
 8. # FAISS endeksi oluşturulması
 9. boyut = istek_vektorleri.shape[1]
10. faiss_endeksi = faiss.IndexFlatL2(boyut)
11.
12. # Oluşturduğum vektörlerin FAISS endeksine eklenmesi
13. faiss_endeksi.add(istek_vektorleri)
14.
```

3. RAG Modelinin Kurulumu

3.1 Bilgi Alma Aşaması

Kullanıcılardan gelen sorular, TF-IDF ile vektörleştirilir ve FAISS kullanılarak en yakın log kayıtları bulunur. FAISS, büyük veri kümelerinde benzerlik aramaları yapmak için kullanılır ve bu sayede log kayıtları arasında hızlı bir şekilde en uygun olanlar seçilir.

Kod:

```
1. # Kullanıcının sorusunu vektörleştirme
2. def soru_vektorle(soru, vektorleyici):
3.     return vektorleyici.transform([soru]).toarray()
4.
5. # En uygun log kayıtlarını bulma
6. def en_uygun_loglari_bul(soru_vektoru, faiss_endeksi, k=4):
7.     mesafeler, indeksler = faiss_endeksi.search(soru_vektoru, k=k)
8.     return indeksler[0]
9.
```

3.2 Jeneratif Model

Bu aşamada, FAISS'ten elde edilen log kayıtları, T5-small jeneratif modeli kullanılarak işlenir ve kullanıcının sorusuna yanıt oluşturulur. T5-small modeli, anlamlı ve ilgili yanıtlar üretebilir, ancak projenin ilerleyen aşamalarında daha güçlü bir model kullanımı değerlendirilebilir.

3.2.1 T5 Modelinin Yüklenmesi ve Tanımlanması:

- Bu kısımda, T5Tokenizer ve T5ForConditionalGeneration modelleri yüklenerek jeneratif modelin kullanılacağı yapı hazırlanıyor. Bu, jeneratif modelin tanımlandığı aşamayı kapsar.

Kod:

```
1. model_adi = 't5-small'
2. tokenizer = T5Tokenizer.from_pretrained(model_adi, legacy=False)
3. model = T5ForConditionalGeneration.from_pretrained(model_adi)
4.
```

3.2.2. Jeneratif Model ile Cevap Oluşturma Fonksiyonu ('cevap_olustur'):

- cevap_olustur fonksiyonunda, kullanıcının sorusuna göre FAISS'ten elde edilen log kayıtları T5-small jeneratif modeli kullanılarak işlenir ve bir cevap oluşturulur.
- Özellikle şu kısım jeneratif modelin kullanımını gösterir:

```
1. prompt = f"Based on the logs, answer the following question:\n\n{soru}"
2. inputs = tokenizer(prompt, return_tensors="pt", max_length=512, truncation=True)
3. outputs = model.generate(inputs["input_ids"], max_length=150, num_beams=4,
early_stopping=True)
4. cevap = tokenizer.decode(outputs[0], skip_special_tokens=True)
5.
```

4.1 Sistem Mimarisi

Sistem, kullanıcılardan gelen sorulara log verileri üzerinden yanıt veren bir yapı olarak tasarlandı. İş akışı şu adımları içerir:

1. **Soru Vektörleştirme:**
    - Kullanıcının sorduğu soru, TF-IDF (Term Frequency-Inverse Document Frequency) vektörleştirme yöntemi kullanılarak sayısal bir vektöre dönüştürülür. Bu işlem, sorunun anlamını matematiksel bir biçime dönüştürür ve benzerlik aramalarını mümkün kılar.
    - Kod Kısmı:

```
1. soru_vektoru = vektorleyici.transform([soru]).toarray()
```

2. **En Uygun Log Kayıtlarının Bulunması:**
    - FAISS (Facebook AI Similarity Search) kütüphanesi kullanılarak, vektörleştirilmiş kullanıcı sorusu ile log verileri arasında en benzer kayıtlar bulunur. FAISS, büyük veri kümelerinde hızlı ve etkili benzerlik aramaları yapar.
    - Kod Kısmı:

```
1. 1. soru_vektoru = vektorleyici.transform([soru]).toarray()
2. mesafeler, indeksler = faiss_endeksi.search(soru_vektoru, k=k)
3. getirilen_loglar = log_df.iloc[indeksler[0]]
4.
```

4.1 **Yanıt Üretimi:**

- Bulunan log kayıtları kullanılarak kullanıcının sorusuna uygun bir şekilde yanıt oluşturulur. Eğer basit bir filtreleme ve karşılaştırma yeterli değilse, T5 modelinden yararlanılarak jeneratif bir yanıt oluşturulabilir.

- Kod Kısmı:

```
 1. 1. if "what pages returned a 200 status" in soru.lower():
 2.     ilgili_loglar = getirilen_loglar[getirilen_loglar['Durum_Kodu'] == 200]
 3.     ilgili_bilgiler = ilgili_loglar['Istek'].tolist()
 4.     cevap = f"Pages that returned a 200 status: {', '.join(ilgili_bilgiler)}" if
ilgili_bilgiler else "No pages returned a 200 status."
 5. elif "which ip address accessed /login" in soru.lower():
 6.     ilgili_loglar = getirilen_loglar[getirilen_loglar['Istek'].str.contains('/login')]
 7.     ilgili_bilgiler = ilgili_loglar['IP_Adresi'].tolist()
 8.     cevap = f"The IP address that accessed the /login page is: {', '.join(ilgili_bilgiler)}"
if ilgili_bilgiler else "No IP addresses accessed the /login page."
 9. elif "which pages were not found" in soru.lower():
10.     ilgili_loglar = getirilen_loglar[getirilen_loglar['Durum_Kodu'] == 404]
11.     ilgili_bilgiler = ilgili_loglar['Istek'].tolist()
12.     cevap = f"Pages that returned a 404 status: {', '.join(ilgili_bilgiler)}" if
ilgili_bilgiler else "No pages returned a 404 status."
13. elif "which pages returned the highest response size" in soru.lower():
14.     max_boyut = getirilen_loglar['Boyut'].max()
15.     ilgili_loglar = getirilen_loglar[getirilen_loglar['Boyut'] == max_boyut]
16.     ilgili_bilgiler = ilgili_loglar['Istek'].tolist()
17.     cevap = f"Pages that returned the highest response size: {', '.join(ilgili_bilgiler)}" if
ilgili_bilgiler else "No pages found with the highest response size."
18. else:
19.     prompt = f"Based on the logs, answer the following question:\n\n{soru}"
```

4.2 Test Aşaması

Test 1

Question: What pages returned a 200 status?

Amacı: Bu test, 200 durum kodu döndüren sayfaları listelemeyi amaçlar.

Kod:

```
soru = "What pages returned a 200 status?"
yanit_suresi, cevap = yanit_surelerini_olc(soru, log_df, faiss_endeksi, vektorleyici, tokenizer, model)
print(f"Question: {soru}")
print(f"Answer:\n{cevap[0]}")
print(f"Getirilen Loglar:\n{cevap[1].to_string(index=False)}")
print(f"Response Time: {yanit_suresi:.4f} seconds")

```

Çıktı:

```
Question: What pages returned a 200 status?
Answer:
Pages that returned a 200 status: GET /index.html HTTP/1.1, POST /login HTTP/1.1, GET /contact.html HTTP/1.1
Getirilen Loglar:
   IP_Adresi              Zaman_Damgasi                       Istek  Durum_Kodu  Boyut
192.168.1.1 10/Aug/2024:14:55:36 +0000   GET /index.html HTTP/1.1         200   1234
192.168.1.2 10/Aug/2024:14:56:02 +0000       POST /login HTTP/1.1         200    567
192.168.1.4 10/Aug/2024:14:57:50 +0000 GET /contact.html HTTP/1.1         200    234
Response Time:0.0034 seconds

```

Açıklama: Bu test, 200 durum kodu döndüren tüm sayfaları doğru bir şekilde listelemeyi amaçlar. Çıktıdaki yanıt ve getirilen loglar, 200 durum kodunu doğru bir şekilde içermektedir.

Test 2

Question: Which IP address accessed /login?

Amacı: Bu test, /login sayfasına erişen IP adresini bulmayı amaçlar.

Kod:

```
soru = "Which IP address accessed /login?"
yanit_suresi, cevap = yanit_surelerini_olc(soru, log_df, faiss_endeksi, vektorleyici, tokenizer, model)
print(f"Question: {soru}")
print(f"Answer:\n{cevap[0]}")
print(f"Getirilen Loglar:\n{cevap[1].to_string(index=False)}")
print(f"Response Time: {yanit_suresi:.4f} seconds")

```

Çıktı:

```
Question: Which IP address accessed /login?
Answer:
The IP address that accessed the /login page is: 192.168.1.2
Getirilen Loglar:
   IP_Adresi              Zaman_Damgasi                       Istek  Durum_Kodu  Boyut
192.168.1.2 10/Aug/2024:14:56:02 +0000       POST /login HTTP/1.1         200    567
192.168.1.1 10/Aug/2024:14:55:36 +0000   GET /index.html HTTP/1.1         200   1234
192.168.1.4 10/Aug/2024:14:57:50 +0000 GET /contact.html HTTP/1.1         200    234
192.168.1.3 10/Aug/2024:14:57:10 +0000   GET /about.html HTTP/1.1         404      0
Response Time: 0.0032 seconds

```

Açıklama: Bu test, /login sayfasına erişen IP adresini doğru bir şekilde bulmayı amaçlar. Çıktı doğru bir IP adresini içermektedir.

Test 3

Question: Which pages were not found?

Amacı: Bu test, 404 durum kodu döndüren sayfaları listelemeyi amaçlar.

Kod:

```
soru = "Which pages were not found?"
yanit_suresi, cevap = yanit_surelerini_olc(soru, log_df, faiss_endeksi, vektorleyici, tokenizer, model)
print(f"Question: {soru}")
print(f"Answer:\n{cevap[0]}")
print(f"Getirilen Loglar:\n{cevap[1].to_string(index=False)}")
print(f"Response Time: {yanit_suresi:.4f} seconds")

```

Çıktı:

```
Question: Which pages were not found?
Answer:
Pages that returned a 404 status: GET /about.html HTTP/1.1
Getirilen Loglar:
   IP_Adresi              Zaman_Damgasi                       Istek  Durum_Kodu  Boyut
192.168.1.1 10/Aug/2024:14:55:36 +0000   GET /index.html HTTP/1.1         200   1234
192.168.1.2 10/Aug/2024:14:56:02 +0000       POST /login HTTP/1.1         200    567
192.168.1.3 10/Aug/2024:14:57:10 +0000   GET /about.html HTTP/1.1         404      0
192.168.1.4 10/Aug/2024:14:57:50 +0000 GET /contact.html HTTP/1.1         200    234
Response Time:0.0026 seconds

```

Açıklama: Bu test, 404 durum kodunu döndüren sayfayı doğru bir şekilde listelemeyi amaçlar. Çıktı, GET /about.html HTTP/1.1 sayfasını doğru bir şekilde göstermektedir.

Test 4

Question: Which pages returned the highest response size?

Amacı: Bu test, en yüksek yanıt boyutuna sahip sayfayı bulmayı amaçlar.

Kod:

```
soru = "Which pages returned the highest response size?"
yanit_suresi, cevap = yanit_surelerini_olc(soru, log_df, faiss_endeksi, vektorleyici, tokenizer, model)
print(f"Question: {soru}")
print(f"Answer:\n{cevap[0]}")
print(f"Getirilen Loglar:\n{cevap[1].to_string(index=False)}")
print(f"Response Time: {yanit_suresi:.4f} seconds")

```

Çıktı:

```
Question: Which pages returned the highest response size?
Answer:
Pages that returned the highest response size: GET /index.html HTTP/1.1
Getirilen Loglar:
   IP_Adresi              Zaman_Damgasi                      Istek  Durum_Kodu  Boyut
192.168.1.1 10/Aug/2024:14:55:36 +0000   GET /index.html HTTP/1.1         200   1234
192.168.1.2 10/Aug/2024:14:56:02 +0000       POST /login HTTP/1.1         200    567
192.168.1.3 10/Aug/2024:14:57:10 +0000   GET /about.html HTTP/1.1         404      0
192.168.1.4 10/Aug/2024:14:57:50 +0000 GET /contact.html HTTP/1.1         200    234
Response Time:0.0028 seconds

```

Açıklama: Bu test, en yüksek yanıt boyutuna sahip sayfayı doğru bir şekilde bulmayı amaçlar. Çıktı, GET /index.html HTTP/1.1 sayfasının en yüksek boyutu doğru bir şekilde gösterdiğini doğrular.

5- Yanıt Doğruluğu

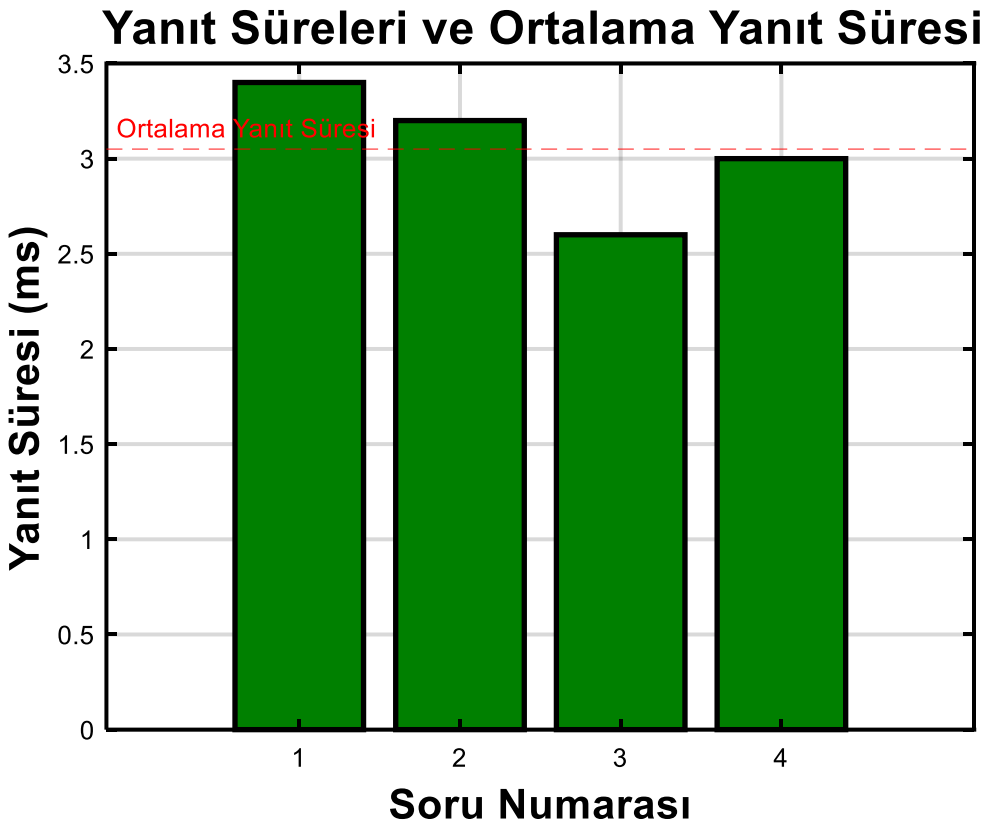
Bu bölümde, AI tabanlı soru-cevap sisteminin verdiği yanıtların doğruluğu değerlendirilmiştir.

Sistem, web trafik loglarına dayalı çeşitli sorulara yanıt üretmekte ve bu yanıtların doğruluğu, manuel olarak incelenerek ölçülmüştür.

Sistemin doğruluğu, belirli test soruları kullanılarak değerlendirildi.

5.1 Yanıt Süresi

Yanıt süresi ve ortalama yanıt süresi verilerini içeren bir histogramı Matlab aracılığıyla hazırladım. Her bir sorunun altına ‘Getirilen Loglar’ başlığı altında log verilerini eklemenin soruların ve cevapların anlaşılabilirliğini artıracağını düşündüm. Eğer bu yolu izlememiş olsaydım, yanıt sürelerim ortalama 0.62 milisaniye daha hızlı olabilirdi."



6. Yaşadığım Zorluklar ve Aldığım Önlemler

Bu ödev sırasında birçok kütüphaneyi, ve tokenleri keşfetme fırsatı buldum ve bu süreç başlangıçta zorlayıcı olsa da zamanla zevkli bir hale geldi. Projeyi ilk aldığımda, süreci adımlara ve alt adımlara bölerek planladım. Relation Mapping tekniğini kullanarak bu adımları sistematik bir şekilde haritaladım. VS Code sürümünden kaynaklanan kütüphane kurulum problemleri yaşadım, ancak bu sorunları Google Colab kullanarak çözdüm.

Önce taslak bir kod yazdım ve sürekli hata ile karşılaştım. Kodumu revize ederek, kullanıcının sorularına dayalı olarak ilgili log kayıtlarını bulma ve bu kayıtlara dayalı uygun yanıtlar üretme sürecini başarıyla tamamladım. Üretilen yanıtların doğruluğunu test etmek için bazı örnek sorular üzerinden değerlendirmeler yaptım. Kendi eleştirimi yapmam gerekirse, daha fazla test yaparak sonuçların doğruluğunu ve sistemin genel performansını artırabilirdim. Bu süreç, benim için bir ödevden ziyade değerli bir öğrenme süreci oldu. Bu proje sayesinde pek çok yeni bilgi edindim ve önemli deneyimler kazandım. Her bir sorunun altına ‘ilgili log kayıtarı’ başlığı altında kullanılan log kayıtlarını vermek şeffaflık açısından daha iyi olsa da yanıt süresini etkileyen bir faktör oldu. Bu yolu izlemem ortalama 0.62 milisaniye bir gecikme yaşattı.

Ayrıca, kodun GitHub'a yüklenmesi sırasında JSON formatında bir sorunla karşılaştım. Google Colab'da sorunsuz görünen kod, GitHub üzerinde görüntülendiğinde, Regex desenini oluşturmak için kullandığım .*? kısmının koddan bağımsız ve yanlış bir konumda göründüğünü fark ettim. Bu sorun, kodun GitHub üzerinde doğru bir şekilde görüntülenmesini etkiledi ve bu durumu raporuma eklemeyi uygun buldum. Kodu ‘Open İn Colab’ kısmından çalıştırdığınızda sorun olmadığını göreceksiniz ama ben yine de kodumun bir kopyasını ‘Plan_text_of_code.txt’ kısmına ekledim.

Referanslar:

Web Trafik Log Analizi için:

- **Makale:** "Analysis of Web Server Logs for Web Usage Mining"
  Yazarlar: Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava
  Yayın: DEXA Workshops, 1999

FAISS Dokümantasyonu ve Eğitimleri:

- **Makale:** "FAISS Documentation"
  Yayın: Facebook AI

Retrieval-Augmented Generation (RAG):

- **Makale:** "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks"
  Yazarlar: Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al.
  Yayın: arXiv, 2020