Web Trafik Loglarına Dayalı Yapay Zeka Destekli Soru-Cevap Sistemi

Furkan Bulut

Manisa Celal Bayar University Faculty of Engineering

Manisa, TURKEY

210316011@ogr.cbu.edu.tr

***Özetçe* — Proje, web trafik loglarına dayalı olarak geliştirilen bir yapay zeka destekli soru-cevap sistemi geliştirme süreçini içerir. Sistem, kullanıcılardan gelen doğal dildeki soruları analiz ederek, log verileri analiz ederek yanıt üretmeyi hedeflemektedir. Log verileri veri önişleme adımları ile temizlenmiş, TF-IDF yöntemiyle vektörize edilerek FAISS vektör veri tabanında depolanmıştır. Sorgulara en uygun log kayıtlarını bulmak için Retrieval-Augmented Generation (RAG) modeli ile GPT-2 dil modeli kullanılmıştır. Geliştirilen sistem, testler sonucunda, sorgulara yanıt üretebilmiştir. Geliştirilen proje, web sitelerinin trafik verilerini daha etkili bir şekilde analiz etmeye ve performanslarını iyileştirmeye yönelik olup bu alanda potansiyel olarak iyileştirme süreçlerine yardımcı olabilecek niteliğe sahiptir.**

I. GIRIŞ

Web siteleri, kullanıcı davranışlarını izleyebilmektedir. Bu davranışları trafik logları olarak kaydederek geliştiriciler için sistem performansını değerlendirmeye olanak sağlamaktadır. Bu loglar, manuel olarak analiz edilmesi zor, karmaşık ve zaman alıcı olan yüksek miktarda verilerdir. Log verilerinden anlamlı bilgiler çıkarmak kullanıcı deneyimini iyileştirmek için kritik öneme sahiptir. Geliştirilen proje, web trafik loglarını verilerini analiz eden ve kullanıcıların doğal dildeki sorularına yanıt üreten yapay zeka destekli bir sistemdir. Retrieval-Augmented Generation (RAG) modeli ile geliştirilen bu sistem, üretilen verilerden anlamlı bilgiler çıkararak, trafik analiz süreçlerini hızlandırmaktadır. Bu proje, web sitelerinin performansını artırma ve güvenlik tehditlerini daha hızlı tespit etme konusunda önemli katkı sunma potansiyeline sahiptir.

II. PROJE DOSYA YAPISI

Product.ipynb dosyası ile proje çalısır durumda olup geliştirilen kodların son versiyonu olma özelliğini taşımaktadır.

1) Data

a. Apache: DataGenerator.py dosyası kullanılarak oluşturulan logfiles.log ve data.csv dosyaları bu dizinde bulunmaktadır.

b. WebTraffic
İnternet üzerinde bulanan hazır veri setine ait 5 farklı CSV dosyası bulunmaktadır. Manuel veri üretiminden önce problem anlamak amaçlı kullanılmıştır.

2) Data_generator

Veri üretme ve veriyi kullanılmak üzere csv formatında kaydeden python dosyalarımız bu dizinde bulunmaktadır.

3) Maintence

Proje gelişirme sürecindeki oluşturulan dosyaları bulundurmaktadır.

III.METADOLOJI

B. Veri Hazırlığı ve Ön İşleme:

Veri hazırlığı ve ön işleme aşaması, modelin başarısı için kritik bir adımdır ve bu aşamada gerçekleştirilen işlemler aşağıdaki gibi olup veri setinin doğru ve verimli bir şekilde analiz edilebilmesi için gerekli ön işlemleri sağlar ve modelin performansını doğrudan etkiler.

IP Address	Date	Request	Endpoint	Status Code	Response Size	Referrer	User Agent	Time
56.81.94.161	[24/Nov/2019:03:03:30 +0300]	GET	/usr/admin	502	5014	https://w	Mozilla/5.0	
117.144.149.150	[09/Mar/2018:03:07:53 +0300]	POST	/usr/login	500	5012	-	Mozilla/5.0	
107.193.107.50	[14/Jun/2019:01:16:10 +0300]	DELETE	/usr/admin/d	200	5002	https://w	Mozilla/5.0	
149.25.240.57	[02/Jan/2019:03:14:05 +0300]	PUT	/usr/admin/d	200	4905	https://w	Mozilla/5.0	
38.186.148.29	[29/Apr/2019:12:24:51 +0300]	DELETE	/usr/register	502	4953	https://w	Mozilla/5.0	
133.111.37.173	[03/Nov/2018:06:23:04 +0300]	DELETE	/usr/register	500	4883	https://w	Mozilla/5.0	
130.219.47.9	[24/Nov/2018:05:49:28 +0300]	POST	/usr/login	500	4943	https://w	Mozilla/5.0	
6.43.6.180	[14/Jul/2019:07:00:34 +0300]	PUT	/usr/register	500	4904	-	Mozilla/5.0	
125.250.223.79	[26/Feb/2019:10:40:00 +0300]	POST	/usr/register	304	4994	-	Mozilla/5.0	

Fig. 1. Manuel olarak oluşturulan veri setinden örnek

1) Verin Oluşturma ve Yükleme:

Web trafik logları, Apache sunucusu simüle edilerek manuel olarak proje içerisinde bulunan DataGenerator.py dosyası çalıştırılarak elde edilen CSV formatındaki dosya ile üretilip yüklenmiştir. Bu dosya IP Address,Date,Request,Endpoint,Status Code,Response Size,Referrer,User Agent,Time Taken bilgilerini içermektedir. Elde edilen veri data.csv olarak kaydedilmiş olup Jupyter notebook ile işlenme süreçleri için yüklenmiştir.

2) Veri Temizliği:

a. **Eksik Verilerin Kontrolü:** Verilerin eksik olup olmadığını belirlemek için, her sütundaki eksik değerler kontrol edilmiştir. Bu aşamada, eksik veriler veya hatalı değerler (örneğin, bozuk tarih formatları) temizlenmiştir.

b. **Gereksiz Sütunların Kaldırılması:** Log dosyasında gereksiz olabilecek sütunlar (örneğin, Referrer, User Agent) veri çerçevesinden çıkarılmıştır. Bu işlem, verinin daha temiz ve işlenebilir hale gelmesini sağlamıştır.

3) Tarih ve Zaman Bilgilerinin İşlenmesi:

Tarih Sütununun Dönüştürülmesi: Date sütunu, tarih ve saat bilgisini içermektedir ve datetime formatına dönüştürülmüştür. Bu dönüşüm, zaman damgalarının doğru bir şekilde analiz edilebilmesi için gereklidir. Örneğin, pd.to_datetime fonksiyonu kullanılarak tarih ve saat bilgileri datetime objelerine dönüştürülmüştür.

4) Yeni Özelliklerin Eklenmesi:

Zamanla İlgili Özellikler: *Date sütunundan, yıl (Year), ay (Month), gün (Day), saat (Hour), ve dakika (Minute) gibi yeni özellikler çıkarılmıştır. Bu özellikler, zaman bazlı analizler ve sorgular için kullanılacaktır.*

5) Sayısal Verilerin İşlenmesi:

Durum Kodlarının Sayısal Değerlere Dönüştürülmesi: Status Code sütunu, HTTP yanıt kodlarını temsil eder ve gerekirse sayısal değerlere dönüştürülmüştür. Bu dönüşüm, analitik işlemler ve veri analizi sırasında gereklidir.

6) Boş veya Hatalı Verilerin Temizlenmesi:

Boş Değerlerin Kaldırılması: Request, Endpoint, ve Status Code gibi kritik sütunlarda boş veya hatalı veriler varsa, bu satırlar veri çerçevesinden çıkarılmıştır. Ayrıca, boş olan veya hatalı Request ve Endpoint değerleri filtrelenmiştir.

7) Birleştirilmiş Metin Verisinin Oluşturulması:

Kombine Özelliklerin Oluşturulması: Request, Endpoint, ve Status Code sütunları birleştirilerek yeni bir metin özelliği (Combined) oluşturulmuştur. Bu metin, vektörleştirme ve bilgi alma aşamalarında kullanılacak girdi verilerini sağlamaktadır.

8) Son Kontroller ve Doğrulama:

Boş Verilerin Kontrolü: Combined sütununda boş değerlerin olup olmadığı kontrol edilmiştir. Herhangi bir boş değer varsa, bu veriler temizlenmiştir.

C. RAG Modelinin oluşturulması

RAG (Retrieval-Augmented Generation) modeli, bilgi alma ve jeneratif model bileşenlerini birleştirerek kullanıcı sorgularına anlamlı yanıtlar üreten bir yaklaşımdır. Bu aşamada gerçekleştirilen işlemler aşağıdaki gibi olup RAG modelinin başarılı bir şekilde kurulması ve entegrasyonunu sağlayarak, etkili bir soru-cevap sistemi oluşturmayı hedeflenmiştir.

1) Bilgi Alma Bileşeni (Retrieval):

a. **Vektör Veri Tabanının Kurulumu:** Veri hazırlığı aşamasında elde edilen metin vektörleri, FAISS (Facebook AI Similarity Search) kütüphanesi kullanılarak bir vektör veri tabanına yüklenmiştir. FAISS, büyük ölçekli vektör veri tabanlarında hızlı ve verimli arama yapabilen bir araçtır.

b. **Sorgu Vektörlerinin Hesaplanması:** Kullanıcının sorusunu temsil eden metin, TF-IDF vektörizasyon yöntemi kullanılarak vektör haline getirilmiştir. Bu vektörler, FAISS indeksinde arama yapmak için kullanılır.

c. **En Yakın Komşuların Bulunması:** FAISS kullanılarak, kullanıcı sorgusuna en uygun log kayıtları bulunur. Bu işlem, sorgu vektörü ile vektör veri tabanındaki vektörler arasındaki benzerliği hesaplar ve en yakın komşuları belirler.

2) Jeneratif Model Bileşeni (Generation):

a. **Model ve Tokenizer Yükleme:** Jeneratif model olarak GPT-2 seçilmiştir. Bu model, dil üretiminde güçlü bir performansa sahiptir. GPT-2 modeli ve tokenizer, Hugging Face'in Transformers kütüphanesi kullanılarak yüklenmiştir.

b. **Giriş Metninin Hazırlanması:** FAISS aracılığıyla bulunan log kayıtları, kullanıcı sorgusuna yanıt oluşturmak için GPT-2 modeline sunulacak şekilde birleştirilmiştir. Log kayıtları, tarih, istek, endpoint ve durum kodu gibi bilgileri içerir ve bu veriler, metin haline getirilir.

c. **Yanıt Üretimi:** Hazırlanan giriş metni, GPT-2 modeline verilerek yanıt üretilir. Model, verilen metni dikkate alarak kullanıcı sorgusuna uygun bir yanıt oluşturur. Yanıtın uzunluğu ve içeriği, modelin hiperparametreleri kullanılarak kontrol edilir.

D. Sistem Entegrasyonu ve Test

Sistem entegrasyonu ve test aşaması, bilgi alma ve jeneratif model bileşenlerinin bir araya getirilerek bütünleşik bir sistem oluşturulmasını ve bu sistemin performansının değerlendirilmesini içerir. Bu aşama aşağıdaki adımları içermekte olup bilgi alma ve jeneratif modellerin entegre bir şekilde çalışmasını sağlayarak, etkili ve verimli bir soru-cevap sisteminin kurulmasını ve test edilmesini hedeflemektedir. Entegrasyon ve test süreci, sistemin performansını maksimize etmek için kritik öneme sahiptir.

1) Sistem Entegrasyonu:

a. **Bilgi Alma ve Jeneratif Modellerin Entegrasyonu:** Bilgi alma bileşeni (FAISS ile vektör arama) ve jeneratif model (GPT-2) birleştirilmiştir. Bu entegrasyon, kullanıcıdan gelen sorguların önce bilgi alma aşamasında işlenmesini ve en ilgili log kayıtlarının bulunmasını, ardından bu kayıtların jeneratif model aracılığıyla yanıt üretmek üzere kullanılmasını sağlar.

b. **Veri Akışının Yönetimi:** Sorgular, bilgi alma bileşenine iletilir ve en uygun log kayıtları elde edilir. Elde edilen bu log kayıtları, jeneratif modele aktarılır ve model, bu bilgileri kullanarak yanıt üretir. Bu akış, sistemin doğru çalışmasını sağlamak için dikkatlice yönetilmiştir.

2) Test Senaryolarının Oluşturulması:

a. **Test Sorgularının Belirlenmesi:** Sistemin performansını değerlendirmek amacıyla çeşitli test sorguları oluşturulmuştur. Bu sorgular, farklı türde bilgi taleplerini içerir ve sistemin geniş bir yelpazede yanıt verebilme yeteneğini test eder.

b. **Test Kapsamı:** Testler, yanıt tutarlığına, alakasına göre değerlendirilmiştir. Sorgular, sistemin doğru ve anlamlı yanıtlar verip vermediğini değerlendirmek için kullanılmıştır.

3) Sistem Performansının Değerlendirilmesi:

a. **Yanıt Kalitesinin Analizi:** Üretilen yanıtlar, içerik kalitesi, doğruluk ve alaka düzeyine göre değerlendirilmiştir. Yanıtların kullanıcının sorgularına uygunluğu ve anlamlılığı kontrol edilmiştir.

b. **Yanıt Sürelerinin Ölçülmesi:** Sistemin yanıt süreleri ölçülerek, performans değerlendirilmiştir. Yanıt sürelerinin kabul edilebilir seviyelerde olup olmadığı gözlemlenmiştir.

E. Performans Değerlendirmesi

Performans değerlendirmesi, geliştirilen RAG tabanlı soru-cevap sisteminin etkinliğini ve verimliliğini ölçmek amacıyla yapılan bir dizi analiz ve testten oluşur. Bu değerlendirme, sistemin doğruluğunu, yanıt kalitesini ve yanıt süresini analiz etmeyi içerir.

1) Doğruluk ve Yanıt Kalitesinin Değerlendirilmesi:

a. **Yanıtların Doğruluğu:** Sistem tarafından üretilen yanıtların doğruluğu, verisetimiz içerisinde bulunan log kayıtlarıyla karşılaştırılarak değerlendirilmiştir. Yanıtların, kullanıcının sorguları ile alakası tutarlı olmuştur.

b. **Yanıtların Anlamlılığı:** Üretilen yanıtlar Teknik düzeyde olup yeterince açıklayacı olmayıp sorgu anahtar kelimeleri ile uygundur.

c. **Kapsam ve Relevans:** Yanıtların, kullanıcı sorgularıyla uyumlu olduğu gözlemlenmiştir. Sistem, sorgulara en uygun log kayıtlarını bularak bu verilerle ilgili yanıtlar üretebilmektedir.

2) Yanıt Sürelerinin Ölçülmesi:

Sistem üzerinde yapılan testlerde karşılaşılan hata ve sorunlar analiz edilerek geliştirme süreçleri içinde çözülmüştür. Yanıt sürelerinin mevcut veriler üzerinde verimliği olduğu saptanmıştır.

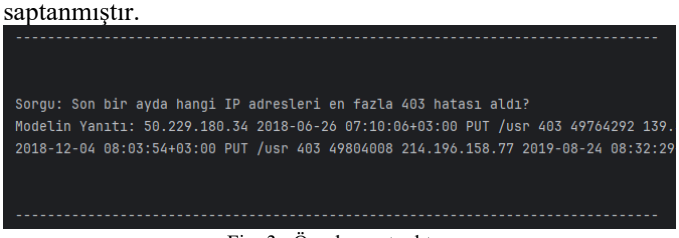

Fig. 3. Örnek yanıt çıktısı

IV. Karşılaşılan Zorluklar

Proje geliştirme süreçlerinde karşılaşılan zorluklar ve süreçler aşağıdaki gibidir:

a. **Problemin Karmaşıklığı:** Verilen problem ve türevleri konusunda tecrübe yetersizliğinden kaynaklanan bir algılama güçlüğü yaşanarak geliştirme süreçlerinin yavaş ilerlemesiyle sonuçlanmıştır.

b. **Veri Seti Oluşturma:** Hazır veri setlerinin bulunmaması nedeniyle, veriler manuel olarak üretilmiştir. Bu durum, veri hazırlığı ve önişleme süreçlerini uzatmıştır.

c. **FAISS Kütüphanesi Problemleri:** FAISS kütüphanesi entegrasyon sürecinde çeşitli import hatalarıyla karşılaşılmış, bu da sistemin stabil çalışmasını engelleyerek proje geliştirme süreçini aksatmıştır.

d. **Model Entegrasyonu:** Bilgi eksikliği ve deneyim yetersizliği, modelin verimli entegrasyonunu ve performansını olumsuz etkilemiştir. Bu nedenle, modelin kıyaslamaları tam anlamıyla gerçekleştirilememiştir. Modelden başlangıçta yeterli verim alınamayarak daha kapsamlı araştırma yapılarak farklı model türleri denenmiştir. Sonuç olarak ‘gpt-2’ modeli kullanılarak geliştirme süreçleri devam etmiştir.

e. **Yanıt Formatı:** Üretilen yanıtların anlamlı, tutarlı ve kullanıcıya uygun formatta sunulması için ek ve manuel bir formatlama çalışmaları gerekmiştir. Bu, kullanıcının aldığı model cevabındaki anlaşılırlığı önemli ölçüde düşürmüştür.

V. Sistemin Cevap Kalitesini Artırmak İçin İyileştirme Önerileri

Sistemin verdiği cevapların kalitesini artırmak amacıyla yapılabilecek iyileştirmeler şu şekilde olup sistemin genel performansını artırarak daha tutarlı ve anlaşılır bir çıktı sunmasına yardımcı olacaktır.

a. **Gelişmiş Model Eğitimi:** Modelin daha geniş, gerçek ve çeşitli veri setleriyle yeniden eğitilmesi, üretilen cevapların doğruluğunu ve karmaşık sorguları anlamlandrma imkanı sunabilir. Özellikle, domain-spesifik verilerle modelin zenginleştirilmesi, sektöre özgü sorgulara daha tutarlı yanıtlar verilmesini sağlayacaktır.

b. **Cevap Formatlama:** Üretilen yanıtların formatlama süreçleri geliştirilmeli ve daha anlamlı, kullanıcı dostu formatlarda sunulabilir.

c. **Yanıt Doğruluk Değerlendirmesi:** Yanıtların doğruluğunu ve bağlamını kontrol eden ek algoritmaların eklenmesi, yanlış veya tutarsız yanıtların önüne geçebilir.

d. **Geribildirim Mekanizması:** Kullanıcıların sistem yanıtları hakkında geribildirim vermesini sağlayacak bir mekanizmanın geliştirilmesi, sistemin sürekli olarak öğrenmesine ve yanıt kalitesinin zamanla artmasına katkıda bulunabilir.

e. **Kullanılan Dil Modeli:** Mevcutta kullanılan ‘gpt-2’ modeli kulanıcıya istediği performansı vermeyebilir. Daha net ve formatlı çıktılar için daha güncel bird il işleme model entegrasyonu sağlanabilir.

f. **Veri önişleme:** Veri önişleme süreçleri, oluşturlan veriye göre, detaylandırılıp genşletilerek model performansı artıöı hedeflenebilir.

VI. Tartışma

Geliştirilen Retrieval-Augmented Generation (RAG) tabanlı soru-cevap sistemi, kullanıcı sorgularına uygun yanıtlar sunma konusunda kabul edilebilir sonuçlar üretebilmiştir. Yanıt kalitesi sorulan sorulardı anahtar kelimeler ile uyumlu olup tatmin edici olmayabilir, karmaşık ve nadir sorgular için doğruluk ve kapsam iyileştirilmelidir.

Problem başlangıç düzeyi için karmaşık bir yapıda olup kıyaslanabilirliği düşük olması nedeniyle geliştirme süreçlerinde verim kaybı yaratmış olup verilen ile geliştirilen projenin tutarlığı hakkında net bir kanıya varılamamaktadır.

Veri temizliği ve model entegrasyonu süreçlerinde karşılaşılan zorluklar, özellikle tarih formatlarının standartlaştırılması ve bilgi alma ile jeneratif modellerin uyumlu çalışması konularında önemli engeller oluşturmuştur.

Gelecekte, model iyileştirmeleri ve veri zenginleştirme yöntemleri, yanıt kalitesini artırabilirken, performans optimizasyonları yanıt sürelerini kısaltabilir. Sistem, büyük ölçekli web siteleri ve işletmeler için kullanıcı destek hizmetleri ve veri analizi alanlarında pratik uygulamalara sahiptir. Ayrıca, daha gelişmiş jeneratif modeller ve veri kaynaklarının entegrasyonu, sistemin kapsamını ve etkinliğini artırabilir. Elde edilen bulgular ve karşılaşılan zorluklar, sistemin verimliliğini ve doğruluğunu artırma yönünde önemli ipuçları sunmaktadır.

VII. Sonuç

Bu çalışmada, web trafik loglarına dayalı bir Retrieval-Augmented Generation (RAG) tabanlı soru-cevap sistemi geliştirilmiştir. Sistem, kullanıcıların doğal dildeki sorularına uygun yanıtlar sunmak amacıyla bilgi alma ve jeneratif model bileşenlerini bir araya getirmiştir. Veri hazırlığı aşamasında, web trafik logları detaylı bir şekilde temizlenmiş ve işlenmiş, TF-IDF vektörizasyonu kullanılarak vektörler oluşturulmuş ve FAISS veri tabanına eklenmiştir. RAG modelinin kurulumu ve entegrasyonu, bilgi alma ve yanıt üretme süreçlerinin uyumlu bir şekilde çalışmasını sağlamıştır.

Testler, sistemin genel olarak tatmin edici yanıtlar ürettiğini, ancak bazı durumlarda yanıt kalitesinin ve performansının iyileştirilmesi gerektiğini göstermiştir. Yüksek sorgu yükleri altında performans düşüşü yaşanmış ve veri temizliği ile model entegrasyonu süreçlerinde çeşitli zorluklar yaşanmıştır. Gelecek çalışmalar, daha gelişmiş modellerin ve veri zenginleştirme tekniklerinin entegrasyonu ile bu zorlukların üstesinden gelebilir ve sistemin yanıt kalitesini ve performansını artırabilir.

Sonuç olarak, geliştirilen sistem, büyük ölçekli web siteleri ve işletmeler için etkili bir kullanıcı destek aracı ve veri analizi çözümü sunmaktadır. Elde edilen bulgular, sistemin mevcut performansını ve pratik uygulamalarını değerlendirmekte ve gelecekteki iyileştirmeler için temel teşkil etmektedir. Sistem, kullanıcı deneyimini artırma potansiyeline sahip olup, sürekli gelişim ve optimizasyon gerektiren dinamik bir yapı olarak değerlendirilmelidir.

Ekstra kodlar

```
import pandas as pd
from sklearn.feature_extraction.text
import TfidfVectorizer
import faiss
import torch
from transformers import GPT2LMHeadModel,
GPT2Tokenizer

data =
pd.read_csv('data/apache/data.csv')

print("Sütun Adları:", data.columns)

data['Date'] =
pd.to_datetime(data['Date'].str.strip('[]
'), format='%d/%b/%Y:%H:%M:%S %z',
errors='coerce')

missing_values = data.isnull().sum()
print("Eksik Veriler:", missing_values)

data = data.drop(columns=['Referrer',
'User Agent'], errors='ignore')

data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
data['Day'] = data['Date'].dt.day
data['Hour'] = data['Date'].dt.hour
data['Minute'] = data['Date'].dt.minute

data['Status Code'] =
pd.to_numeric(data['Status Code'],
errors='coerce')

data = data.dropna(subset=['Request',
'Endpoint', 'Status Code'])
data = data[data['Request'].str.strip()
!= '']
data = data[data['Endpoint'].str.strip()
!= '']
data = data[data['Status Code'].notna()]

data['Combined'] = data['Request'] + ' '
+ data['Endpoint'] + ' ' + data['Status
Code'].astype(str)

print(data['Combined'].head())
print(data['Combined'].isnull().sum())

vectorizer = TfidfVectorizer()
X =
vectorizer.fit_transform(data['Combined']
).toarray()

index = faiss.IndexFlatL2(X.shape[1])
index.add(X.astype('float32'))

def find_relevant_logs(query):
    query_vector =
vectorizer.transform([query]).toarray().a
stype('float32')
    D, I = index.search(query_vector, 5)
    return data.iloc[I[0]]

def generate_response(logs):
    input_text = " ".join(
        logs.apply(lambda row:
f"{row['Date']} {row['Request']}
{row['Endpoint']} {row['Status Code']}",
                   axis=1).tolist())
    input_text = input_text[:1000]  #
Giriş metnini 1000 karakterle sınırlama

    input_ids =
tokenizer.encode(input_text,
return_tensors='pt')
    attention_mask =
torch.ones(input_ids.shape,
dtype=torch.long)

    output = model.generate(input_ids,
attention_mask=attention_mask,
max_length=150, num_return_sequences=1,

pad_token_id=tokenizer.eos_token_id)
    response =
tokenizer.decode(output[0],
skip_special_tokens=True)
    return response

def answer_question(query):
    relevant_logs =
find_relevant_logs(query)
    response =
generate_response(relevant_logs)
    return response

model_name = "gpt2"
model =
GPT2LMHeadModel.from_pretrained(model_nam
e)
tokenizer =
GPT2Tokenizer.from_pretrained(model_name)
#%%
# Test Sorguları Listesi
queries = [
    "Son 24 saatte hangi URL'ler 500
hatası aldı?",
    "Son bir ayda hangi IP adresleri en
fazla 403 hatası aldı?",
    "Son bir yıl içinde en sık kullanılan
POST isteklerinin listesi nedir?",
    "Son 30 gün içinde hangi tarayıcılar
en fazla 404 hatası aldı?",
    "Son haftada hangi endpoint'ler en
yüksek Response Size'a sahipti?",
    "En son 10 istekte hangi User
Agent'lar kullanıldı?",
    "En yüksek zaman alımı (Time Taken)
olan 5 istek nedir?",
    "Son 6 ayda hangi Referrer en çok
```

```
ziyaret edildi?",
    "Son 24 saatte hangi Endpoint'lerde
502 hatası alındı?",
    "Hangi IP adresleri en uzun süre GET
isteği yaptı?",
    "Which IP adress has Longest GET time
?",
]
#%%
# Her bir sorguyu test etme
for query in queries:
    print("-" * 80)
    print("\n")

    response = answer_question(query)
    print(f"Sorgu: {query}")
    print(f"Modelin Yanıtı: {response}")
    print("\n")
```