1.Web trafiği oluşturma:
Projenin ilk kısmında hali hazırda Mac kullandığım için Apache web sunucusunu kullanarak erişim günlüğü dosyalarınızı oluşturdum. Log biçimini Apache yapılandırma dosyasıyla aşağıdaki gibi ayarladım:

LogFormat: "%h %l %u %t \"%r\" %>s %b" combined
CustomLog: "/var/log/apache2/access_log" combined

Formatı bu şekilde ayarladıktan sonra gelen HTTP istekleri oluşturarak bu access'lerin kaydedileceği bir log dosyası oluşturdum. Daha sonra bu dosyayı farklı URL'lere karşı GET, POST, PUT, DELETE gibi HTTP istekleri göndererek doldurdurmaya çalıştım. Daha sonra, bu log dosyasını gerçek zamanlı takip etmek için şu komutu kullandım ve yeterli kayıt olduşturduğumu düşündüğüm zaman durdurdum.

sudo tail -f /var/log/apache2/access_log

```
::1 - - [12/Aug/2024:09:34:16 +0300] "GET / HTTP/1.1" 200 45
::1 - - [12/Aug/2024:09:39:32 +0300] "GET / HTTP/1.1" 200 45
::1 - - [12/Aug/2024:09:39:32 +0300] "GET /favicon.ico HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:39:53 +0300] "-" 408 -
::1 - - [12/Aug/2024:09:45:25 +0300] "GET /about HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:45:45 +0300] "-" 408 -
::1 - - [12/Aug/2024:09:48:28 +0300] "GET /contact HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:48:52 +0300] "GET /services HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:48:58 +0300] "GET /nonexistentpage HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:49:14 +0300] "POST /submit HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:49:21 +0300] "PUT /update HTTP/1.1" 405 220
::1 - - [12/Aug/2024:09:49:27 +0300] "DELETE /delete HTTP/1.1" 405 223
::1 - - [12/Aug/2024:09:49:36 +0300] "GET /protected HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:49:36 +0300] "GET / HTTP/1.1" 200 45
::1 - - [12/Aug/2024:09:50:07 +0300] "GET /protected HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:50:07 +0300] "GET / HTTP/1.1" 200 45
::1 - - [12/Aug/2024:09:50:16 +0300] "GET /search?q=keyword HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:50:59 +0300] "GET /search?q=MELIHSINAN HTTP/1.1" 404 196
::1 - - [12/Aug/2024:09:51:54 +0300] "PUT /update HTTP/1.1" 405 220
::1 - - [12/Aug/2024:09:52:19 +0300] "POST /submit HTTP/1.1" 404 196
```

2. FAISS kütüphanesi:

Vektörizasyon hakkında daha önce bir bilgim olmadığı için bu siteden yararladım ve olabildiğince mantığını kavramaya çalıştım:
https://www.matillion.com/blog/step-by-step-guide-building-a-rag-model-with-open-source-llm-llama-2-and-vector-store-faiss

3. Log Dosyası Biçimlendirmesi

Log dosyasını FAISS veritabanına olduğu gibi girmeye çalışırken oluşan access_log'un içerisinde yer alan özel karakterlerden dolayı sorun yaşadım (Örneğin, "-" karakterlerini ve "HTTP/1.1" ifadeleri). Bu sorunu gidermek için access_log'a erişim sağlayıp bu dosyayı daha temiz bir txt dosyasına çevirecek bir method oluşturdum. ".txt" formatı üzerinde

daha kolay çalıştığım ve şu ana kadar yaptığım projelerde dışardan aldığım verileri .txt formatında saklayı sevdiğim için bu formatı kullandım. “.txt” formatını sürekli kullanmayı tercih etmemin sebebi aslında aynı nesne-tabanlı programlar geliştirmek gibi, dışarıdan ana verinin geldiği dosyayı aslında bir yedek olarak kullanıyorum, olabildiğince direkt olarak üzerinde bir işlem yapmıyorum ve bu sayede ilerde hata yaparsam düzeltmem daha kolay oluyor.

```python
import re

input_file_path = "/Users/melihsinancubukcuoglu/Desktop/access_log"
output_file_path = "/Users/melihsinancubukcuoglu/Desktop/output.txt"

def parse_log_line(line):
    pattern = r'(\S+) - - \[(.*?)\] "(.*?)"'
    match = re.search(pattern, line)

    if match:
        ip_address = match.group(1)
        timestamp = match.group(2)
        request = match.group(3)
        page = request.split(" ")[1] if len(request.split(" ")) > 1 else "-"
        formatted_output = f"IP Address: {ip_address}, Timestamp: {timestamp}, Page: {page}"
        return formatted_output

    return None

with open(input_file_path, 'r') as infile, open(output_file_path, 'w') as outfile:
    for line in infile:
        formatted_line = parse_log_line(line)
        if formatted_line:
            outfile.write(formatted_line + '\n')
```

output.txt

```
IP Address: ::1, Timestamp: 12/Aug/2024:09:34:16 +0300, Page: /
IP Address: ::1, Timestamp: 12/Aug/2024:09:39:32 +0300, Page: /
IP Address: ::1, Timestamp: 12/Aug/2024:09:39:32 +0300, Page: /favicon.ico
IP Address: ::1, Timestamp: 12/Aug/2024:09:39:53 +0300, Page: -
IP Address: ::1, Timestamp: 12/Aug/2024:09:45:25 +0300, Page: /about
IP Address: ::1, Timestamp: 12/Aug/2024:09:45:45 +0300, Page: -
IP Address: ::1, Timestamp: 12/Aug/2024:09:48:28 +0300, Page: /contact
IP Address: ::1, Timestamp: 12/Aug/2024:09:48:52 +0300, Page: /services
IP Address: ::1, Timestamp: 12/Aug/2024:09:48:58 +0300, Page: /nonexistentpage
IP Address: ::1, Timestamp: 12/Aug/2024:09:49:14 +0300, Page: /submit
IP Address: ::1, Timestamp: 12/Aug/2024:09:49:21 +0300, Page: /update
IP Address: ::1, Timestamp: 12/Aug/2024:09:49:27 +0300, Page: /delete
IP Address: ::1, Timestamp: 12/Aug/2024:09:49:36 +0300, Page: /protected
IP Address: ::1, Timestamp: 12/Aug/2024:09:49:36 +0300, Page: /
IP Address: ::1, Timestamp: 12/Aug/2024:09:50:07 +0300, Page: /protected
IP Address: ::1, Timestamp: 12/Aug/2024:09:50:07 +0300, Page: /
IP Address: ::1, Timestamp: 12/Aug/2024:09:50:16 +0300, Page: /search?q=keyword
IP Address: ::1, Timestamp: 12/Aug/2024:09:50:59 +0300, Page: /search?q=MELIHSINAN
IP Address: ::1, Timestamp: 12/Aug/2024:09:51:54 +0300, Page: /update
IP Address: ::1, Timestamp: 12/Aug/2024:09:52:19 +0300, Page: /submit
```

4. Vektörizasyon:

Bu aşamada önce .txt dosyasındaki veriyi string biçiminde bir listeye yerleştiren load_data methodunu oluşturdum ki FAISS kütüphanesi ile token'leri oluşturabileyim. Ardından bu verileri TF-IDF ile her bir verinin tüm dosya içerisinde ağırlığına bağlı vektörler oluşturdum.

Vektörlerin oluşup oluşmadığını görmek için display_vectors methodunu oluşturdum, bu methodu aslında RAG modelini yaparken aklıma geldi. İlk başlarda sadece "vektörler oluşturuldu" gibi bir print ile devam ediyordum yani aslında kodun execute edip etmemesine bakıyordum sonra gerçekten vektörlerin oluşup oluşmadığını yazdırmak istedim, aslında bir kere vektörleri teyit ettikten sonra kodun içerisinde yer almasına gerek yoktu ama saklıyorum. Daha sonra, vektörlerin oluştuğunu teyit edip oluşturulan vektörler için bir dizin ve boyut oluşturdum. Bunun sayesinde ileride model arama yaptığı zaman en uygun log satırlarına erişebilecek.

5. Model Entegrasyonu:

RAG modelini tamamlaması için T5 modelini seçtiniz. T5 modelini seçmemin en büyük 2 sebebi API key kullanımı ve segmentasyon hataları oldu. GPT API key kullanmaya çalışırken sürekli kota (quota) hatası aldım, bunun için yeni hesap oluşturdum ve forumlarda araştırdırdım ama yine de önüne geçemedim ardından API key kullanmadan gpt-2 modellerini kullanmaya çalıştım. Bu sırada da sürekli segmentasyon hatası almaya başladım. Bu hatayı gidermek için modeli hem CPU hem de GPU üzerinden çalıştırmaya çalıştım üstüne de modelin oluşturacağı cevaptaki maximum uzunluğu olabildiğince kısalttım ama segmentasyon hatasının önüne geçemedim. Bu sorunları araştırdıkça üzerinden kalkamayınca artık T5 modelini kullanmaya karar verdim.

6. Anahtar Kelime Çıkarımı ve Log Alma

Artık kullanıcan soru almaya yaklaşırken, kullanıcı sorgusundan "which", "log", "has" gibi stopwords kullanarak modelin sorgusu esnasında en anlamlı log'ları bulmasını hedefledim. Sistemin ilk aşamalarında sadece bu 3 stopword vardı ve sonlara doğru olabildiğince farklı stcpword ekledim.Sonra da bu kelimeleri tek bir string'e aktardım.
Son olarak, kullanıcının sorusuna göre en alakalı günlük satırını bulacak bir retrieve_similar_logs methodu yazdım. Bu method FAISS ile anahtar sözcükler için logların vektörlerinde en alakadar ya da ağırlığı en çok olan vektörü/vektörleri getiriyor.

7. Model Yanıtını Oluşturma

Kullanıcı sorgusuna göre en alakalı günlük satırlarını çektikten sonra T5 ile anlamlı cevaplar oluşturmaya çalıştım.
Önce T5 ile bağlam oluşturdum, ilgili loglar bulunursa bu logların bir özetini yapıp kullanıcıya tam olarak neyin gösterileceğine dair bağlam oluşturdum. Hazırlanan metni T5 modeline uygun bir biçimde model girdisi oluşturup T5 modeli daha sonra girdi üzerinde arama yaparak kullanıcıya sorunun cevabı olarak sunulacak en olası yanıtı vermesini sağlamaya çalıştım.

Karşılaştığım Zorluklar:

Apache2 tarafından oluşturlan access log'ları FAISS veritabanına vektörize ederken özel karakterleri elemek.

İlk aşamlarda FAISS yerine pinecone kullanmaya çalıştım. Hem site üzerinden işlediği için daha temiz bir yapıya sahip olduğunu düşündüm ama yine API key üzerinden ilerlediği için bir türlü öbür tarafta vektörleri oluşturamadım.

GPT modelini kurarken segmentasyon hatalarını alınca olabildiğince modeli daha küçük hale getirmeye çalıştım. İlk başlarda gpt-3.5 ,3.5-turbo, 2 gibi modelleri kullanmaya çalıştım ama başaramadım.

Bunların dışında hiç deneyimimin olmadığı bir alan olduğu için vaktimin çoğu dokümanları okumakla geçti.

İyileştirme Önerileri:

1. Logların Yüklenmesi ve Vektörizasyon: Oluşturuduğum log dosyasını bir .txt dosyasına çevirdiğim için daha büyü verilerde sorun yaşabileceğimi düşünüyorum ve buna bağlı FAISS üzerinden vektörlerin oluşturulmasında da bir sorun yaratabileceğini düşünüyorum.

2. Anahtar Kelime Çıkarma: extract_keywords methodunda kullandığım stopword listesine oldukça kelime ekledim ama sorguların doğal dildeki varyasyonlarına karşı sistemi o kadar geliştiremedi. Bazı durumlarda, önemli bilgiler de stopword listesi nedeniyle elenebilir, bu da doğruluğu olumsuz etkileyebilir.Vektörleştirme ve Benzer Logların Bulunması: retrieve_similar_logs fonksiyonu, anahtar kelimelerle ilgili logları başarılı bir şekilde buluyor. Ancak, sorguların karmaşıklığı arttıkça, FAISS tabanında bulunan logların gerçek anlamdaki ilgisi sorgulanabilir. Bu durum, TF-IDF modelinin vektörize edilmiş sonuçlarının yeterince anlamlı olup olmadığına bağlıdır.

3. Modelin Cevap Kalites: T5 Kısa ve özlü yanıtlar üretebiliyorum ama daha uzun ve karmaşık sorgularda modeling performansı yeterli olmayabilir bunun için GPT gibi bir model kullanmam daha iyi olabilirdi. Model daha GPT gibi bir modele göre daha küçük olduğu için daha hızlı sonuçlar saplıyor ancak yanıtların doğruluğunu ve kapsamını bir o kadar da sınırlı. Bunun yanında daha büyük bir T5 modeli (örneğin t5-large) doğruluğu artırabilir ancak işlem süresi ve bellek kullanımı artardı.