**Web Trafik Loglarına Dayalı Yapay Zekâ Destekli Soru-Cevap Sistemi Geliştirme Projesi Raporu**



Betül Nur YILDIRIM

Proje Teslim Tarihi: 22.08.2024

İÇİNDEKİLER

1. Tanıtım…………………………………………………………………………………………….3
   1. Proje Amacı…………………………………………………………………………………3
   2. Problem Tanımı……………………………………………………………………………3
   3. Kullanılan Teknolojiler…………………………………………………………………..3
2. Veri Seti…………………………………………………………………………………………..4
3. Veriler Üzerinde İşlem……………………………………………………………………….5
4. Veri Analizi………………………………………………………………………………………5
5. LLM ……….………………………………………………………………………………………7
   1. Chunking/ Ayırma……………………………………………………………………….7
   2. Vektörleştirme/ Vektör Database……………………………………..……………8
   3. RAG Mimarisi…………………………………………………………………………….10

**1.1 Proje Amacı**

Bu projede, bir web sitesine ait trafik loglarını kullanarak yapay zekâ destekli bir soru-cevap sistemi geliştirimek amaçlanmıştır. İstenen, web trafik verilerinden elde edilen bilgileri kullanarak kullanıcıların sorularına doğru ve etkili yanıtlar sunmaktır. Bu sistem, web sitesi trafiği hakkında bilgi almak ve kullanıcı davranışlarını analiz etmek için kullanılabilir.

**1.2 Problem Tanımı**

Bir web sitesinin günlük trafiği, çeşitli kullanıcı etkileşimleri vb. olaylar gibi çok sayıda veri içerir. Ancak bu büyük veri yığınları arasında anlamlı sonuçlar çıkarmak ve bu verileri yorumlayarak sorulara yanıt vermek karmaşıktır. Bu projede, bu verilerden anlamlı sonuçlar çıkararak kullanıcılardan gelen sorulara anlamlı yanıtlar vermeyi amaçlayan bir sistem geliştirilmeye çalışılmıştır.

* 1. **Kullanılan Teknolojiler**

**Python**: Projenin geliştirilmesinde ana programlama dili olarak kullanılmıştır. Veri setinin oluşturulmasından metin üretimine kadar birçok işlem Python dili ile gerçekleştirilmiştir.

**Faker Kütüphanesi**: Sentetik veri oluşturmak için kullanılmıştır. Özellikle sahte IP adresleri, zaman damgaları ve URL'ler gibi log verisi için gerekli bilgileri üretir.

**Pandas**: Web trafik loglarının işlenmesi ve analiz edilmesi amacıyla kullanılmıştır. Log verileri DataFrame formatına dönüştürülerek veri temizleme ve analiz işlemleri gerçekleştirilmiştir.

**LangChain**: Büyük dil modelleriyle çalışırken verileri daha küçük parçalara bölmek için kullanılmıştır. CharacterTextSplitter sınıfı ile log verileri satır bazında ayrılmıştır.

**SentenceTransformer**: Metinlerin vektöre dönüştürülmesinde kullanılan bir dil modelidir. all-mpnet-base-v2 modeli, metin benzerliklerini ölçmek için kullanılmıştır.

**FAISS**: Vektörlerin hızlı ve etkili bir şekilde saklanması ve arama yapılması için kullanılmıştır. L2 (Öklidyen) mesafesiyle vektörler arasındaki benzerlikleri ölçen bir indeks yapısı oluşturulmuştur.

**Hugging Face Transformers**: BART ve T5 modellerini içeren bu kütüphane, metin üretiminde ve dil modeli uygulamalarında kullanılmıştır. T5Tokenizer ve T5ForConditionalGeneration sınıfları ile log verilerinden detaylı metinler üretilmiştir.

**RAG (Retrieval-Augmented Generation)**: Bilgi alma ve metin üretimini birleştiren bir model türüdür. Bu projede, Retrieval aşaması ile veriler arandı ve Augmented Generation aşamasında bu veriler kullanılarak daha bilgilendirici metinler üretilmeye çalışıldı.

**2. Veri Seti**

Bu proje için uygun bir veri seti internet üzerinden araştırılmasına rağmen bulunamamıştır. Bu nedenle, Python dili kullanılarak sentetik bir Trafik Log Veri Seti oluşturulmuş ve bu veri seti, proje üzerinde çalışan ancak doğru kaynak bulamayan kişilerin de kullanabilmesi için Kaggle’da paylaşılmıştır.

(https://www.kaggle.com/datasets/betulbny/web-traffic-log)

Python kodları ile Apache web trafik log dosyası oluşturulmuştur. Bu dosya, çeşitli HTTP isteklerini ve durum kodlarını simüle eden veriler içerir. Faker kütüphanesi, sahte IP adresleri, zaman damgaları, URL gibi ilgili bilgileri oluşturmak için kullanılmıştır.

**Log Formatı:** Log dosyasındaki her satırın, Apache log formatına uygun şekilde üretilmesi sağlanmıştır. Bu formatta IP adresi, zaman damgası, HTTP metodu, URL, HTTP durum kodu ve veri boyutu gibi bilgiler bulunur. Log dosyasındaki değişkenler şu şekildedir:

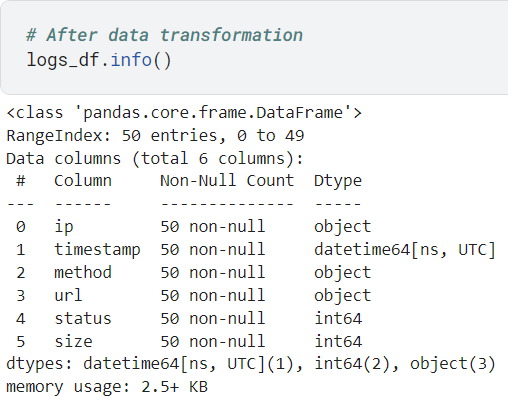
* **IP**: İsteği gönderen cihazın IP adresi.
* **Timestamp**: İsteğin gönderildiği tarih ve saat.
* **Method**: HTTP isteği yöntemi (örneğin, GET, POST).
* **URL**: İsteğin yapıldığı URL yolu.
* **Status**: HTTP yanıt durumu kodu.
* **Size**: Yanıtın byte cinsinden boyutu.

123.45.67.89 - - [10/Aug/2023:15:35:18 +0000] "GET /home HTTP/1.1" 200 1024

**3. Veriler Üzerinde İşlem**

Log verilerinin işlenmesi ve analiz edilmesi, ham veri formatından daha kolay anlaşılabilen bir formata dönüştürülmelidir. Bu projede, Python'da Pandas kütüphanesi kullanılarak web trafik loglarının DataFrame formatına dönüştürülmesi sağlanmıştır.

Veri temizleme işleminden önce bütün değişkenler/ özellikler “object” türündeydi. Ancak “timestamp”, “status” ve “size” değişkenlerinin veri tiplerinin farklı olması gerektiği için bunları değiştirildi.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

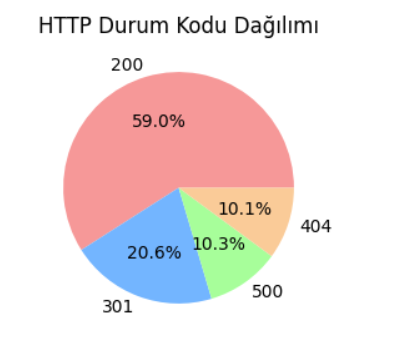
Açıklama otomatik olarak oluşturuldu

**4. Veri Analizi**

PUT ve DELETE yöntemlerinin kullanım sıklığının diğerlerinden fazla olması, kullanıcıların daha çok veriyi güncellediği ve sildiği bir kullanım modelinin mevcut olduğunu göstermektedir.

metin, ekran görüntüsü, dikdörtgen, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu



Verilen HTTP durum kodları dağılımı, web sunucusunun performansı hakkında bazı önemli bilgiler sunmaktadır. En yüksek orana sahip olan 200 OK durum kodu, sistemin büyük bir kısmının başarılı bir şekilde yanıt verdiğini ve kullanıcı isteklerinin çoğunun sorunsuz bir şekilde karşılandığını gösteriyor. 301 Moved Permanently kodunun yüksek sayısı, birçok kaynağın kalıcı olarak taşındığını veya yönlendirildiğini gösterir. 500 Internal Server Error kodları, sunucu tarafında bazı teknik sorunların yaşandığını ve bu sorunların çözülmesi gerektiğini işaret eder. Son olarak, 404 Not Found kodu, kullanıcıların aradıkları kaynakların bulunamadığını gösterir. Genel olarak, sistemin çoğu isteğe başarılı şekilde yanıt verdiği, ancak yönlendirme ve hata yönetimi konularında iyileştirmeler yapılması gerektiği anlaşılmaktadır.

metin, yazı tipi, ekran görüntüsü, tasarım içeren bir resim

Açıklama otomatik olarak oluşturuldu

En yüksek ziyaret sayısına sahip /products/item?id=1 ve /search?q=test URL'leri, kullanıcıların ürün detaylarını ve arama sonuçlarını sıklıkla görüntülediğini gösteriyor. Diğer yüksek ziyaret edilen sayfalar olan /about ve /products sayfaları, kullanıcıların hakkında bilgi almak ve ürünleri incelemek için sıkça bu sayfalara başvurduğunu gösteriyor.

metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, ekran görüntüsü, diyagram içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada ise yıllara göre sitedeki trafiğin büyüklüğü gösterilmiştir. Grafiğe bakıldığında yıllar arasında bir değişim yokmuş gibi görünmesinin sebebi bu verilerin gerçekliği yansıtmamasından kaynaklanıyor. Gerçek bir veri üzerinde çalışılmış olunsaydı zamanla birlikte trafik büyüklüğünün de arttığı görünürdü.

1. **LLM**

Öncelikle data frame halinde olan veri csv formatına çevrilerek üzerinde işlem yapma kolaylaştırıldı.

* 1. **Chunking/ Ayırma kısmı:**

Büyük dil modelleriyle (LLM'ler) çalışmak, özellikle bellek yönetimi ve modelin ince ayarı(fine tuning) konusunda zorluklar teşkil eder. Bu zorlukları hafifletebilecek tekniklerden biri ise, girdileri daha küçük ve yönetilebilir parçalara bölme stratejisi olan Chunking işlemidir. LLM’de chunking işlemi hem girdi hem çıktılara uygulanabilir. Bu projedeki chunking’in amacı ise LLM’i veriler ile beslemeden önce küçük parçalara ayırmaktır. Böylece dökümanı daha kısa bölümlere ayırarak modelin her bir parçayı bağımsız olarak işlemesi sağlanır. Bu da modelin daha hızlı ve daha verimli çalışmasına olanak tanır. Örneğin, bir soru-cevap sistemi geliştirilirken, modelin her bir chunk üzerinde ayrı ayrı cevaplar üretmesi daha etkin olabilir.

Bu işlem için ise Langchain framework’u kullanıldı. Normalde String sınıfından Split metodu da kullanılarak chunklar’a ayrılabilirdi ekte görüldüğü gibi. Ancak bu tür projeler için bir for döngüsü yazmak zahmetli ve hatalara açık olabilir. Bunun yerine LangChain, manuel çabaya gerek bırakmadan, çeşitli metin bölücü sınıflar sunar. Bu projede kullanılan sınıf ise CharacterTextSplitter sınıfıdır. Bütün log’ları satır olarak ayrılması planlandığı için bu sınıfı kullanmak yeterli ancak daha farklı çeşitlerde ayrılmalar gerçekleştirilmek istenseydi bunun yerine RecursiveCharacterTextSplitter sınıfının metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldukullanımı tercih edilebilirdi.

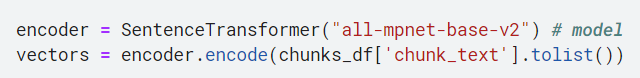
metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

* 1. **Vektörleştirme /Vektör Database:**

Metinlerin, görsellerin, seslerin ve daha birçok veri türünün makine öğrenmesi ve derin öğrenme algoritmaları için anlaşılabilir olması için sayısal bir şekilde ifade edilmesi gerekir. Daha önce chunking işlemi ile bölünmüş chunks\_df adlı dataframe’nin chunk\_text sütunundaki değerler kullanılan modelin yardımı ile vektöre dönüştürülürler.

Bu projede SentenceTransformer kütüphanesi tarafından kullanılan all-mpnet-base-v2 isimli bir dil modeli tercih edildi. OpenAI, modellerinin ücretli olması sebebi ile kullanılmadı. Bunun yerine Hugging Face de kullanılabilirdi. Bu vektörler daha sonra metinlerin benzerliğini ölçmek için kullanılacak.



Sonrasında vektör dizisi bir indeks yapısına eklendi. FAISS kütüphanesi ile bir L2 (Öklidyen) mesafesine dayalı düz bir indeks oluşturur. Bu indeks, dim boyutlu (dim = 768) vektörleri saklayacak şekilde tasarlanmıştır. Aşağıda verilen kod, dim boyutlu vektörleri içeren bir indeks oluşturuyor. Bu indeks, vektörler arasındaki benzerlikleri L2 mesafesine göre ölçmek için kullanılır ve hızlı arama yapılmasını sağlar.

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Aşağıdaki kod parçasında bir arama sorgusu verildi. Bu sorgu, sistemdeki log girdilerinde "/contact" URL'si için DELETE isteği içeren girdilerin bulunmasını istiyor. Burada daha önce kullanılan SentenceTransformer modeli kullanılarak bu sorgu sayısal bir vektöre dönüştürüldü.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

yazı tipi, metin, grafik, beyaz içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu kodda ise FAISS kullanılarak bir arama işlemi gerçekleştiriliyor. Burada svec ise arama yapmak için kullanılan vektör( Yukarıda oluşturulan). “K” parametresi ise kaç tane yakın komşunun döndürülmesi istendiğini belirtir. K = 2’nin anlamı “svec’e benzer 2 vektörü döndür.” demektir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Arama sorgusunun yanıtı olarak daha önce k parametresi 2 olarak verildiği için indeks numaraları 3679 ve 6392 olan iki "/contact" URL'si için DELETE isteği içeren log satırları getirilmiştir.

Text Generation Kısmı:

metin, yazı tipi, ekran görüntüsü, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Metin üretimi(generation) için Hugging Face’nin BART modeli kullannıldı. Daha sonra da metinleri modelin anlayabileceği formata dönüştürmek için modelin tokenizeri kullanıldı.

**5.3 RAG Mimarisi**

RAG (Retrieval-Augmented Generation), dil modeli çıktılarının kalitesini artırmak için bilgi retrieval (bilgi alma) ve metin üretim tekniklerini birleştiren bir model türüdür. RAG, genellikle iki ana bileşenden oluşur:

1. **Retrieval (Bilgi Alma):**
   * **Açıklama:** Bilgi alma aşaması, ilgili verileri veya belgeleri büyük bir veri havuzundan seçmek için kullanılır. Bu, modelin bilgi tabanından bilgi çekerek metin üretimini zenginleştirmesine olanak tanır.
   * **İşlevi:** Kullanıcının sorgusuyla veya metinle ilgili olan belgeleri arar ve getirir. Bu belgeler, modelin daha bilgilendirici ve doğru yanıtlar üretmesini sağlar.
2. **Augmented Generation (Geliştirilmiş Üretim):**
   * **Açıklama:** Üretilmiş metinlerin kalitesini artırmak için bilgi alma aşamasında elde edilen verileri kullanarak metin üretir. Bu aşama, dil modelinin elde edilen bilgileri entegre ederek daha kapsamlı ve doğru yanıtlar oluşturmasına yardımcı olur.
   * **İşlevi:** Bilgi alma aşamasından elde edilen verileri kullanarak daha bilgilendirici ve tutarlı metinler üretir. Bu, genellikle bir dil modeli tarafından gerçekleştirilir ve metni geliştirir.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Retrieve:** Retrieve metodu istenen arama sorgusuna en yakın vektörleri bulmak için bir bilgi alma (retrieval) işlemi gerçekleştirir. Yukarıda yaptığımız vektörleştirme ve benzerini bulma işleminini aynısıdır.

**Generation:** Aşağıdaki kod, `t5-small` modelini kullanarak log verilerinden detaylı metinler üretir. `T5Tokenizer` ve `T5ForConditionalGeneration` sınıfları ile model yüklenir ve `generate` fonksiyonu, log verisine dayanarak yeni bir metin oluşturur. Fonksiyon, log girişini bir "prompt" olarak kullanarak metin üretir ve sonuç olarak ayrıntılı bir yanıt sağlar.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu