Proje Adı : RAG based Web Logs LLM

Yazar : Mehmet Özkaya

Tarih : 22.08.2024

İçindekiler:

* Proje Özeti ve Açıklaması
* Yöntemler
* Bulgular
* Tartışma
* Sonuç
* Kaynakça
* Proje Özeti ve Açıklaması:

Bu proje RAG sistemi kullanılarak getirilen verilerin bir büyük dil modeline entegre edilerek web server logları hakkında sorulan sorulara cevap verebilmesi adına oluşturulmuştur. Kullanıcı tarafından girilen sorgunun işlenmesi sonucunda veritabanı içerisinde bu sorguyla en alakalı kayıtlar bulunur ve büyük dil modeline verilerek cevap oluşturması sağlanır. Projenin amacı programlama dilleri ile sadece bilirkişiler tarafından gerçekleştirilebilecek bir işlemin, her türlü kullanıcı tarafından yalnızca bir sorgu ile gerçekleştirilebilmesidir.

* Yöntem:

1. Veri Önişleme ve Vektörleştirme:

Bu projenin veriseti access.log dosyası içerisinde bulunan ve Kaggle üzerinden temin edilmiş olan (link: https://www.kaggle.com/datasets/eliasdabbas/web-server-access-logs), İranlı bir e-commerce web sitesi log verileridir. Projede kullanılmak üzere log verileri uygun şekilde analiz edilmelidir. Her bir log dosya içerisinden okunmalı, her biri log pattern’ine uygun bir şekilde parçalanmalı ve isteğe göre sözlük içerisinde saklanmalıdır. Bu işlemler gerçekleştirilirken Python’ın en yaygın veri önişleme ve düzenleme kütüphanelerinden olan “re, Pandas, Numpy” kütüphaneleri kullanılabilir ve bu projede kullanılmıştır. Her bir log’un sözlük biçiminde saklanmasının sebebi istenilen özelliklerine ‘key’ler aracılığıyla erişiminin kolaylığıdır. Sözlükler oluşturulduktan sonra her bir log’a index numarası da eklenir. RAG temelli sistemlerin çalışabilmesi için gereken kavram bilgisinin eklenmesi ve sözlükte bulunan elemanların anlamlı bir şekilde birleştirilebilmesi gereklidir. Bu nedenle sözlük verileri kullanılarak her bir log verisi bir cümleye çevrilir ve sözlüğün ‘context’ key’ine atanır. Ardından bu cümleleri vektörleştirme aşamasına geçilir.

Vektörleştirme işleminin temel amacı makinenin anlayamayacağı ‘string’ veri türüyle depolanmış cümlelerin makine dilinin anlayabileceği numerik verilere çevrilmesidir. Bu işlemi gerçekleştirmek için bu projede Python ‘sentence\_transformers’ kütüphanesinden elde edilen ‘SentenceTransformer’ sınıfı kullanılmıştır. Bu sınıf cümleleri vektörlere dönüştürebilen daha önceden eğitilmiş dil modellerinin yüklenmesini kolaylaştırır. Bu projede vektörleştirme işlemini gerçekleştirmesi için ‘all-mpnet-base-v2’ modeli kullanılmıştır ve daha hzılı işlem yapabilemesi için GPU’ya taşınmıştır. Tüm cümleler vektörler haline getirilir ve yine her bir log sözlük verisinin ‘embedding’ key’ine verilir. Daha sonra ise tüm log’ların sahip olduğu ‘embedding’ verileri tek bir ‘embeddings’ değişkeninde toplanılır ve vektörleştirilmiş cümlelerimiz hazır hale gelir.

1. Vektör Veritabanı:

Bu projede vektörler haline getirilen cümleleri saklaması ve aramanın yapılabilmesi için bir vektör veritabanı olan FAISS kullanılmıştır. FAISS, Facebook tarafından üretilen ve vektör verilerinin depolanması, vektör benzerliklerinin hızlı ve optimal yöntemlerle araştırılması adına üretilmiş bir vektör veritabanıdır. Çok yüksek depolama alanına sahip verilerin depolanmasında ve benzer vektörlerin bulunmasında çok etkilidir.

FAISS vektör veritabanı, daha önceden oluşturmuş olduğunuz vektörleri alır ve bu vektörlerin boyutlarında index’ler oluştururak depolarlar. FAISS vektörlerin benzerliklerini birden fazla yöntemle bulabilir. En yaygın olarak kullanılan yöntemlerden bazıları ‘dot scores (noktasal çarpım puanları)’ aracılığıyla, ‘cosine similarity (kosinüs benzerliği)’ veya ‘Euclidian similarity (Öklid benzerliği)’dir. Kullanılan yönteme bağlı olarak yüksek boyutlu vektörler arasında hesaplamalar yapılır ve istenilen kadar benzer vektörlere ulaşılabilir. Bu projede FAISS kütüphanesiyle elde edilen benzer vektörlerin yanısıra ‘sentence\_transformers.util’ içerisinde bulunan ‘dot\_scores’ isimli fonksiyonla hesaplanan vektörler de vardır ve tüm vektörlerin aynı olduğu sonucuna varılmıştır.

1. RAG (Retrieve-Augmented Generation):

Proje Retrieval Augmental Generation (RAG) yöntemini içeren bir büyük dil modelinden oluşuyor. RAG temelli büyük dil modelleri belirli bir konu hakkında çok daha tutarlı ve sağlam yanıtlar üretebilirler. Bu modeller cevaplarını üretmeden önce belirli bir dış bilgi kaynağına başvururlar ve buradan elde ettikle veri ile cevaplarını üretirler. Temel olarak iki ayrı bileşenden oluşurlar ve bu iki bileşeni birbiriyle birleştirerek daha doğru ve kapsamlı cevaplar üretebilirler.

1. Retriever (Bilgi Getirici):

Büyük bir belge veya veri kümesinden (veritabanı, web sayfaları, PDF dosyaları vs.) ilgili bilgiyi bulup getirmekle yükümlüdür. Cevap üretmede kullanılacak olan veri vektör tabanlı bir veritabanında saklanabilir. Kullanıcının sorduğu sorguya en uygun ve en yakın sonuçları veritabanından seçer.

1. Generator (Cevap Üretme):

Bu bileşen Retriever bileşeninin veritabanından getirmiş olduğu verileri alır ve bir büyük dil modeli kullanarak cevap oluşturulmasını sağlar. Büyük dil modelleri olarak GPT, T5 ve BERT örnek olarak verilebilir. Veritabanından gelen verileri doğal dil cevapları haline getirir.

Oluşturduğumuz RAG model, FAISS vektör veritabanı içerisinde bulunan ve kullanıcının girmiş olduğu sorguya en yakın vektörleri getirecek, ardından da bu vektörlerin decode (tekrar yazıya çevirme) işlemlerinden sonra LLM modele vererek yanıtların oluşmasını sağlayacaktır.

1. Retrieve (Veri Çekme):

Veri çekme işlemi kullanıcının sorgusunu girmesinin ardından ‘embeddings’ değişkeni içerisinde sakladığımız veriyi oluşturan dil modeli ile bu sorgunun da vektörleştirme işleminin gerçekleşmesiyle başlar. Aynı dil modelini kullanmak önemlidir çünkü her bir dil modeli kendi yapısına uygun şekilde ‘embedding’ işlemini gerçekleştirir. Ardından FAISS vektör veritabanı tarafından oluşturulmuş ‘index’ içerisinde bu sorgu vektörüne en yakın istenilen sayıda vektör getirilir. Bu arama sonucunda vektörlerin birbirine olan benzerlik/uzaklıklarına ve bu vektörlerin indislerine erişilir. Decode aşamasına gerek kalmadan daha önceden oluşturmuş olduğumuz sözlük verisi içerisinde elde ettiğimiz indisler çekilir ve bunların ‘context’leri içerisinden ‘string’ veri tipli cümleler alınır. Böylece sorguya en uygun olan cümlelere bir veritabanı içerisinden erişilmiş olunur.

1. LLM (Büyük Dil Modeli):

Büyük dil modelleri daha önce çok büyük verisetleriyle eğitilmiş ve doğal dilden gelen soruları cevaplayabilmek için tasarlanmış olan yapay zeka modelleridir. LLM'ler, metin üretme, anlama, özetleme, çeviri, soru-cevap ve çok daha fazlası gibi çeşitli dil görevlerinde oldukça başarılıdır. Çok yüksek sayıda parametre uzun süren eğitim aşamasında en optimize şekilde ayarlanmış ve ardından gelen sorulara en iyi cevapları üretebilecek şekilde tasarlanmıştır.

Bu projede büyük dil modeli olarak Google’ın tasarlamış olduğu Google Gemma 7B Instruction-tuned modeli kullanılmıştır. 7B, 7 milyar parametresinin olduğu, Instruction-tuned (it) ise belirli bir görev veya bir dizi görev üzerinde daha iyi performans gösterecek şekilde ayarlanmış olduğunu ifade eder. Aynı modelin 2 milyar parametreli versiyonu denenmiş fakat cevapların yetersiz olduğu kanaatine varılmıştır. 7 milyar parametreli versiyon ise cevaplar istenilen seviyeye ulaşmış fakat maliyeti arttırmıştır. Yüksek parametreli dil modelleri GPU’lar üzerinde çalıştırılır ve çok yüksek miktarda hesaplama maaliyetlerine ihtiyaç duyarlar. Kullanılmadan önce GPU VRAM kapasitesi mutlaka kontrol edilmelidir. Aksi taktirde model çalışmayı durdurabilir ya da alakasız sonuçlar üretebilir. Hesaplama maliyetlerini azaltmak için bu projede büyük dil modelinin ‘bit’ sayısı azaltılmış ve ‘quantization\_config’ parametresi düzenlenmiştir.

Projedeki büyük dil modeli ‘transformers’ kütüphanesi kullanılarak ‘HuggingFace’ üzerinden indirilmiştir. Bu modele ait ‘tokenizer’ ise yine aynı şekilde temin edilmiştir.

1. Augmentation (Entagrasyon):

Entegrasyon aşaması Retrieve aşamasıyla Generation aşamasını birleştirir. Yüklemiş olduğumuz büyük dil modeline Retrieve aşamasında elde ettiğimiz kayıtlar verilir ve bu kayıtları inceleyen modelin bir yanıt oluşturması beklenir. Bu projede öncelikle kullanıcının sorgusu alınır, vektör haline getirilir, vektör veritabanındaki vektörlerle karşılaştırılır ve en yakın vektörlerin indislerine ve benzerliklerine ulaşılır. Ardından sözlük veritipinde depoladığımız log kayıtlarından belirlenen indis değerlerine sahip olanları tüm bilgileriyle çekilir ve büyük dil modeline sorguyla beraber verilir.

1. Cevap Üretme (Generation):

Büyük dil modelinin doğru cevaplar üretebilmesi için özel bir sorgu hazırlayıcı fonksiyon oluşturulmuştur ve bu fonksiyon Instruction-tuned dil modeline nasıl bir yanıt oluşturması gerektiğini, kullanması gereken kayıtları ve sorguyu ayarlar. Bu sorguyu ve kayıtları alan büyük dil modeli, yanıtları oluşturur.

* Bulgular:

Oluşturulan yanıtlar incelendiğinde büyük dil modelinin yanıtları üretmek ve getirilen veriyi incelemek için yeterli kapasiteye sahip olduğu sonucuna ulaşılmıştır. Eğer getirilen kayıtlar verilen sorgu ile en alakalı ve doğru kayıtlar ise büyük dil modeli istenilen şekilde doğru cevaplar üretmektedir. Zaman zaman veri getirme aşamasında sorgunun derinliğine bağlı yanlış veriler getirilebiliyor. Bunun sebeplerine aşağıdaki ‘Tartışma’ bölümünde bahsedilecektir.

* Tartışma:

Yanlış verilerin getirilmesinin sebeplerinden en önemlisinin log verilerinin ve bu verilerle oluşturulan ve benzerliklerine bakılan cümlelerin birbirine olan benzerliğidir. Çünkü tüm log verileri belirli bir formata sahip verilerdir ve yalnızca cümle içindeki birkaç kelime farklılık yaratır. Örneğin log kayıtlarının işlenmesi sırasında cümleler oluşturulurken belirli bir kalıba yalnızca farklı kelimeler koyulur. Bir PDF dosyasında yüzlerce farklı cümle bulunurken ve yapılan sorguya benzer cümleler yüksek doğruluk oranıyla bulunabilirken log verileri bu özelliği taşımamaktadır. Aynı tarayıcı ya da aynı işletim sistemi yüzlerce log verisinde bulunur. Ayrıca cümleler halinde değil de sözlük verisetine benzer şekilde de model denenmiş fakat cümlelerde başarının daha yüksek olduğu kanısına varılmıştır. LLM modelin getirilen sonuçlardaki yapıyı tam olarak algılayamaması da yanlış cevap üretmenin bir sebebi olabilir. Daha fazla parametreli LLM modeller, ‘embedding’ işlemini gerçekleştiren dil modellerinin çok daha kapsamlıları ve daha tutarlı ve anlaşılabilir bir veriseti kullanılırsa çok daha iyi sonuçlar üretebilir fakat hesaplama maliyeti artacaktır.

* Sonuç:

Sonuç olarak istenilen RAG temelli LLM model üretilebilmiş ve çıktı verebilecek şekilde oluşturulmuştur. Getirilen verilerin büyük çoğunluğu doğru olsa da girilen sorgunun derinliğine göre yanlış yapabilme ihtimali vardır. Tartışma kısmında bahsedilen sorunların üstesinden gelinebilirse çok daha iyi bir model oluşturulabilir.

* Kaynakça:
  + <https://chatgpt.com/>
  + <https://www.kaggle.com/datasets/eliasdabbas/web-server-access-logs>
  + https://www.youtube.com/watch?v=qN\_2fnOPY-M&t=14036s