# Web Trafik Loglarına Dayalı Yapay Zeka Destekli Soru Cevap Sistemi Geliştirme

## Giriş

Web sitesinin kullanıcılar tarafında ziyaret edilmesi sonucu oluşan loglar kullanılarak yapay zeka destekli sohbet sistemi geliştirilmesine örnek olacak bir çalışma yürütülmüştür. Sistemde soru, cevap ve log verilerinin vektörlere dönüştürme yeteneklerine sahip bazı modeller kullanılmıştır. Bu modeller HuggingFace platformundan alınmıştır. Çalışmada kullanılan modeller, all-MiniLM-L6-v2, gpt2, bert-base-uncased, distilbert-base-uncased, t5-small, gpt-neo-1.3B ve roberta-base modelleri kullanılmıştır.

## Soru-Cevap Sistemleri

Sistemlerin tasarlanma sürecinde çalışılan ortamın donanım özelliklerinden dolayı modeller mümkün olduğunca farklı kombinasyonlarla denenmiştir. Colab ve Kaggle’da sistemlerin geliştirilmesine başlanıp modellerin çıktılarına daha rahat erişiminden dolayı local’de bulunan bilgisayarda geliştirilmeye devam edilip notebook çalışması uzun sürmesinden dolayı süreç, Kaggle’da tamamlanmıştır. Local’de RAM yetersizliği, düşük seviye sayılabilecek GPU ve orta seviye sayılabilecek CPU’nun süreci yavaş ilerletmesinden dolayı daha küçük çaplı modeller denenmeye çalışılmıştır.

### Verinin İşlenmesi

Web trafik logları, gpt-4o modeli yardımıyla chatgpt.com üzerinden 10.000 satırlık veri üretilmiştir. Veriler üretildiğinde IP Address, Timestamp, HTTP Method, URL, HTTP Version, Status Code, Response Size, Referer ve User Agent kolonları bulunmaktaydı. IP Address, konum ve cihaz bilgisini, Timestamp, web sitesine giriş isteğinin yapıldığı tarih ve saat bilgisini, HTTP Method, istek türünün (GET, POST, PUT, DELETE) bilgisini, URL, giriş yapılmak istenen web site adresinin bilgisini, HTTP Version, HTTP protokolünün sürüm bilgisini, Status Code, isteğin başarıyla tamamlanıp tamamlanmadığını sunucu tarafından isteğe verdiği yanıt ile belirten durum kodunun (200, 404 gibi) bilgisini, Response Size, Sunucu yanıt boyutunun (byte cinsinden) bilgisini, Referer, kullanıcının web sitesine hangi sayfadan geldiğinin bilgisini, User Agemt, kullanıcının kullandığı tarayıcı, işletim sistemi ve cihaz bilgilerini taşır. Yapılan denemeler ve istemde yer alan öneriler sonucunda HTTP Method, URL, HTTP Version, Status Code, Response Size ve User Agent kolonlarının kullanılmasına karar verilmiştir. Modellere girdi olarak verilecek veri kümesinin bir kısmı Tablo 1’de yer almaktadır.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| HTTP Method | URL | HTTP Version | Status Code | Response Size | User Agent |
| POST | /index.html | HTTP/1.1 | 404 | 1696 | Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like M... |
| POST | /index.html | HTTP/1.1 | 302 | 1785 | Mozilla/5.0 (Windows NT 10.0; Win64; x64) Appl... |
| POST | /user/profile | HTTP/1.1 | 200 | 339 | Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7... |
| POST | /user/profile | HTTP/1.1 | 200 | 1838 | Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:90.... |
| GET | /index.html | HTTP/1.1 | 404 | 509 | Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_15\_7... |

**Tablo 1:** Modellere girdi olarak verilecek veri kümesinin bir kısmı

Visual Studio Code üzerinde Jupyter Notebook’ta sistemler geliştirilmeye çalışılmıştır. Giriş kısmında bahsedilen text generation ve embedding modelleri, literatürdeki kullanım yaygınlığından ve çalışılan ortamdaki uygunluğundan dolayı seçilmiştir. Bu yüzden modelelrle ilgili detaylı bilgi HuggingFace platformundan bulunabilir. Retrieval-Augmented Generation (RAG) modelleri kurulumunda daha düzenli, temiz ve okunabilirliği fazla tutulması amacıyla Object Oriented Programming (OOP) tekniği temel alınmaya çalışılmıştır. LogProcessor adında sınıf oluşturulmuştur. Bu sınıfta literatürde de bilinen chunk\_size, overlap\_size, top\_k, tokenazation, padding, max\_new\_tokens ve num\_return\_sequences hiperparametrelerin bulunduğu ve ayarlanabildiği fonksiyonlar yer almaktadır.

Implemente edilen sınıfta bulunan ve hiperparametrelerin yer aldığı fonksiyonlardan bahsedilirse, chunk\_text, log\_to\_vector\_no\_logits, log\_to\_vector\_logits, log\_to\_vector\_logits, preprocess\_logs, create\_faiss\_index, retrieve\_logs\_no\_logits, retrieve\_logs\_logits, process\_no\_logits, process\_logits ve generate\_answer’dır. Kısaca bahsedilirse, chunk\_text, web sitesi log bilgisini chunklara bölmek için, log\_to\_vector fonksiyonları, log verilerimizi vektörlere dönüştürmek için, create\_faiss\_index, Facebook AI Similarity Search (FAISS) index oluşturmak için, preprocess\_logs, log verilerinin birleştirilmesi için, process fonksiyonları, daha önce vektörlerinin oluşturulma kontrolünün yapılması için ve generate\_answer, text generation modelinden cevabın jeneratif edilmesi için oluşturuldu. Bazı fonksiyonlarda yer alan logits ve no\_logits son eklerinden anlaşılacağı gibi bazı modellerin son katmanında aktivasyon fonksiyonu olmasından veya olmamasından dolayı farklı fonksiyonlar yazıldı. Kodun tamamına GitHub’dan erişilebilir.

Veri kümesinin iletilme, vektöre dönüştürülme, indexlenme ve sorgulanma gibi sıralı işlemlerinin implemente edilen sınıfta yapılması ile birnevi pipeline (processor\_pipeline) tasarlanmıştır (import edilen kütüphanelerdeki kullanılan fonksiyonlarda varolan pipeline’ların yanı sıra). Süreç, veriyi, veriyi vektörlere dönüştürecek modeli ve tokenizer’ı bu pipeline’a vermek bu işlemlerin yapılması için yeterli olacak şekilde tasarlandı. Veriyi vektörlere dönüştürme yeteneğine sahip modeller olarak daha önceden de bahsedilen kısıtlamalar ve önerilerin sonucunda all-MiniLM-L6-v2, bert-base-uncased, distilbert-base-uncased, t5-small, EleutherAI/gpt-neo-1.3B ve roberta-base seçilmiştir. Generator model olarak da gpt2 ve EleutherAI/gpt-neo-1.3B seçilmiştir. Böylece oluşturulan sistem kombinasyonları Tablo 2 ve Tablo 3’teki gibidir.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Generator:** | gpt2 (137M params) | | | | | |
| **Vectorize:** | all-MiniLM-L6-v2  (22.7 params) | bert-base-uncased  (110M params) | distilbert-base-uncased" (66M params) | t5-small"  (60M params) | EleutherAI/gpt-neo-1.3B  (1.3B params) | roberta-base  (125M params) |

Tablo 2: RAG soru-cevap sistemlerinin model kombinasyonları

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Generator:** | EleutherAI/gpt-neo-1.3B (1.3B params) | | | | | |
| **Vectorize:** | all-MiniLM-L6-v2  (22.7 params) | bert-base-uncased  (110M params) | distilbert-base-uncased" (66M params) | t5-small"  (60M params) | EleutherAI/gpt-neo-1.3B  (1.3B params) | roberta-base  (125M params) |

Tablo 3: RAG soru-cevap sistemlerinin model kombinasyonları

all-MiniLM-L6-v2, bert-base-uncased, distilbert-base-uncased ve t5-small modellerinin veriyi vektörlere dönüştürebilme yeteneği bulunduğundan ve EleutherAI/gpt-neo-1.3B ile roberta-base modelinin sonraki belirteci veya cümleyi tahnmin etme yeteneği olduğundan dolaylı olarak veriyi vektörlere dönüştürebildiğinden vektörleşme görevi için seçildiler. Generator olarak belirteci veya cümleyi tahnmin etme yeteneği olan EleutherAI/gpt-neo-1.3B ve gpt2 modelleri seçildi.

### Sistemlerin Performans ve Doğruluk Değerlendirmesi

Sistemlerde gpt2’nin generator olarak kullanıldığı notebook’ta pipeline bitme süresi en fazla yaklaşık 3 saat ile EleutherAI/gpt-neo-1.3B tutarken diğer vektörleştirme modelleri yaklaşık 15-30 dakika aralığında süreçlerini tamamlamışlardır. Generator olarak yanıt süresi ve vektör veritabanı arama süresinde kayda değer gecikmeye rastlanılmadı. Aynı sayılabilecek benzer sonuçlar EleutherAI/gpt-neo-1.3B modelinin generator olarak kullanıldığı notebook’ta da deneyimlenmiştir.

Sistemlerin geliştirilmesi sürecinde kullanılan kütüphanelerin torch tensorleriyle işlemlerin yürütülmesinin yanında notebook’ta numpy arraylerinin kullanılmasıyla CPU’da devam edilmiştir. Bir daha soru-cevap sistemleri geliştirilme ihtiyacında notebook’ta da torch tensorleriyle pipeline’ın ve diğer kodların torch tensorü türünde implemente edilmesi daha doğru olabileceği ve dolayısıyla GPU’da daha hızlı işlem yapılabileceği deneyimlenmiştir. Model boyutlarının en çok sıkıntı yarattığı durum yine EleutherAI/gpt-neo-1.3B modeli olabilirken bu modelin yer aldığı sistemlerde süre ve hafıza da yaşanmıştır. Vektörleştirilmiş veri çıktısının boyutu da doğal olarak daha büyük olmuştur. Model boyutlarının yer aldığı bilgiler Tablo 4’te yer almaktadır.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Modeller: | all-MiniLM-L6-v2  (22.7 params) | bert-base-uncased  (110M params) | distilbert-base-uncased" (67M params) | t5-small  (60.5M params) | EleutherAI/gpt-neo-1.3B  (1.37B params) | roberta-base  (125M params) | gpt2  (137M params) |
| Boyutlar (mb): | 87.3 | 420 | 256 | 232 | 5.058,56 | 478 | 525 |

Tablo 4: Model boyutları

Sistemlere,

* How many requests resulted in a 404 status code?
* What is the most common HTTP method?
* What is the most common status code?
* What is the most common user agent?
* Which User Agent was used most frequently?
* What is the average response size for requests to /user/profile?
* How many unique URLs were accessed?
* Which HTTP method was used most often for successful requests (status code 200)?

soruları sorulmuştur. Sorular, verilerin üretildiği sohbette ChatGPT tarafından oluşturulmuş uygunluğu belirlendikten sonra seçilmiştir. Sistemlerin verdiği cevapların doğruluğunun, alaka düzeyinin ve anlamlılığının değerlendirilmesi için insan değerlendirme ile test süreci oluşturulması en gerçekçi senaryolardan biri sayılabilir. Fakat bu çalışma daha çok gelişmelerin örneklenmesi amacı güttüğünden ve verinin sentetik bir şekilde oluşturulduğundan tarafımca analiz edilip yorumlanmıştır. Soruların RAG modelleri tarafından verilen cevapları tam bir şekilde GitHub üzerinden incelenebilir ve istek doğrultusunda farklı sorularla local bilgisayarda, Colab ve Kaggle’da (önerilir) test edilebilir. En benzer 5 veri, her 8 soruda vektörleştirme modelleri tarafından birkaç yanıt haricinde neredeyse aynı bulunmuş olup generator’ün yanıtları aşağıdaki gibi yer almaktadır. En benzer verilerin de yer aldığı notebook’un incelenmesi tavsiye edilir.

#### How many requests resulted in a 404 status code?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + 3.
* Vektörleştirme: bert-base-uncased, generator: gpt2
* 400 404 0 478 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/15E.
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + As of 12/07/10 a request containing information about a product or service status can lead to a 404 status code for all products or service services. Therefore, one can always obtain all the information about the product within the source and without any data
* Vektörleştirme: t5-small, generator: gpt2
  + 10. Request code: 0x8080. Request result code: 639. Response from the browser: Please respond using the "contact.response" message to your inquiry. After the response has been received, it must
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
* In the following question, you received two different email addresses for the same question. The replies corresponded to the same email addresses which are the same person (with different name) but which have different mail addresses, which is the error message:
* Vektörleştirme: roberta-base, generator: gpt2
  + No, this is an example of a page on the web that might have been returned by the browser.POST /com.googleapis.apis.mobile.test HTTP/1.1 200 200 yes yes yes no yes.

#### What is the most common HTTP method?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + HTTP/1.1 200 OK Yes OK Yes OK /about HTTP/1.1 403 633 Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML…
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + HTTP/1.1. The following list contains the most frequently used HTTP methods that return errors. HTTP/1.1 GET /User/Profile HTTP/1.1 404 3837 Mozilla/5.0 (iPhone; CPU…
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + It's the way Mozilla implements the web interface. When it changes the behaviour of a resource provider the Web interface will be changed. More information about this behavior is published elsewhere : https://developers.mozilla.org/en-US
* Vektörleştirme: t5-small, generator: gpt2
  + This parameter is an easy way to tell what HTTP method works. Get help with HTTP methods and how to make some queries. GET /request HTTP/1.1 200 OK Content-Type: application/json (charset=[…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + If you have three possible options to get an expected response, then all of these might work as well if you have only a few options and there are only a handful of arguments the most common one to use would look something like this.
* Vektörleştirme: roberta-base, generator: gpt2
  + This is the most common HTTP method.

#### What is the most common status code?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + The most common code is: "Content-Length": 1. HTTP status code is: 200 OK. I've read this site for years, but still haven't gotten past 100! Not only was my page never taken down…
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + 0, The most common. 1: The last non-zero number (such as 15). 2: The last non-zero integer in the sequence (such as 16 or 29…
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + HTTP/1.1 200 422 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/
* Vektörleştirme: t5-small, generator: gpt2
  + This parameter is an easy way to tell what HTTP method works. Get help with HTTP methods and how to make some queries. GET /request HTTP/1.1 200 OK Content-Type: application/json (charset=[…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + HTTP/1.1 503 2014 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/14.0 Mobile/…
* Vektörleştirme: roberta-base, generator: gpt2

#### There are a lot of different answers given here, but I would like to share my experience. Basically, after seeing this question, I've decided to go with no answer.

#### What is the most common user agent?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + The most common is Mozilla's Firefox. However, you can use other websites which show different versions of Firefox and add it to your existing page. If you use a third party search engine to find it, you must specify your version of Firefox with a…
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + If you think some software does better and others do better, then you are doing the right thing for many users. I have noticed that some software supports the most common user agent: Mozilla Firefox as default, as a default browser with…
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + The most commonly used user agent is Safari, or the basic browser window manager. Source: http://developer.samsungwedge.com/samsung.html <http://mobile.samsungwedge.com/samsung>.
* Vektörleştirme: t5-small, generator: gpt2
  + The most common user agent is: <http://www.google.com/view-ad-data?hl=en>. This is how you want to use this. Check the box which you want.After some…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + There are many common and quite useful agents for your business (sometimes called different users agents for some reasons). The results of this list are pretty good. In order to find the most common and useful users agents there is no one right answer.
* Vektörleştirme: roberta-base, generator: gpt2
  + Not much. Canonical says it has several extensions available for use on Macs. Many use this as the browser to hide any user agents from the browser or if it has access to the Internet for one or more purposes only. In particular

#### Which User Agent was used most frequently?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + “Microsoft Windows" instead of "Linux" for Google Chrome
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + https://gist.github.com/dodboggin/8164845dcd/file/7e78e29ca-5de-47ac-bb9b-3fc0e5de1
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + From our experience, this is mostly not the case: there is clearly a lot of work to be done before it is clear who would benefit the most from this. We would love to hear if this has anything to do with Mozilla's experience in choosing
* Vektörleştirme: t5-small, generator: gpt2
  + WebKit/537.36 was the most-used browser when the original request went through (see the "User Agent Used Most Frequently" chart for more information). Here's how users are usually able to use their browsers to find and delete…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + I haven't looked through the data to find out. PATI-X (I'm looking at you: <http://stackoverflow.com/questions/3414862>). GET /tags HTTP/1.
* Vektörleştirme: roberta-base, generator: gpt2
  + I chose the browser that most closely follows my needs and preferences. I was also curious about the other Mozilla-related things. You can check out what we have as an active topic on our list of frequently asked questions. What do please share…

#### What is the average response size for requests to /user/profile?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + An error notification notification, A complete list of error conditions is given below, Error Condition: None, Response Size: 16384 bytes, Description: A complete list of error conditions is given below. Error Condition: None
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + /users/{id}/{name} HTTP/1.1 500 4042 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X) AppleWebKit/605.1.15 (KHTML,
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + Response Length Type: response Description: GET /user/profile HTTP/1.1 200 422 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X) AppleWebKit/604.1.15 (KHTML
* Vektörleştirme: t5-small, generator: gpt2
  + 1.0ms (or 100 MB) for all requests. Example: The "total" type response is 200 bytes, where 2MB is a total number of response bytes. You can specify a "total number" of responses from the…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + Request size may very well be a determining factor in the response size. If a request is larger than or equal to your target user profile size, you should use a smaller response size. To ensure good response strength, check the client-side response size…
* Vektörleştirme: roberta-base, generator: gpt2
  + The average response size for requests to http://localhost:8080/webroot should be: HTTP/1.1 200 404 Not found HTTP/1.1 100 1 302 Not found HTTP/1.1 100 1 301 Not found…

#### How many unique URLs were accessed?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + To answer this question, we need to start by comparing two datasets using the common database "QA: The Numbers" to see how many unique URL's were accessed. As shown in the Figure, the number of unique URLs that were accessed were…
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + The numbers we used above are from the site's index.html. GET /api/1.0 557 2154 Firefox (1.1+) 3.0 Yes Yes Yes (default) yes yes Yes Yes 993 1790…
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + The best way to test out the data for your user's specific search query is to dig down into a JSON file, sort by author name, and then see which results you got. If there are no answers, that's because Google has a weird…
* Vektörleştirme: t5-small, generator: gpt2
  + 1. We can't check the same list by all of the URLs which are searched. 2. On the same day there were no search queries for "parsing" the content. (Example: 1).
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + A total of 21 unique URLs accessed by Yahoo alone.
* Vektörleştirme: roberta-base, generator: gpt2
  + Yes, each unique URL is unique, which makes the entire web inaccessible. That being said, there are two ways to get around this limitation: By using the URL string you are giving users access to the entire website, and by using their web browser…

#### Which HTTP method was used most often for successful requests (status code 200)?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt2
  + HTTP/1.1 504 2067 Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:90.0) Gecko/20100101 Firefox/90.0
* Vektörleştirme: bert-base-uncased, generator: gpt2
  + None.
* Vektörleştirme: distilbert-base-uncased, generator: gpt2
  + Some HTTP methods are useful for a lot of HTTP requests. Some options such as: GET responses from http://localhost HTTP/1.1 302 2016 Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6
* Vektörleştirme: t5-small, generator: gpt2
  + HTTP/1.1 302 3527 Mozilla/5.0 (iPhone; "iPad"; "iPhone"; "iPhoneOS"; "-webkit8;") AppleWebKit/621.10.4 (KHTML, like Gecko)
* Vektörleştirme: gpt-neo-1.3B, generator: gpt2
  + What type of requests?
* Vektörleştirme: roberta-base, generator: gpt2
  + HTTP/1.1 404 Not Found In 1.3 Content-Type: application/x-www-form-urlencoded Content-Length: 5078 Content-Class: application/x-www-form-urlencoded <object

#### How many requests resulted in a 404 status code?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + There was only one 404.
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + Only one. The problem is: there isn't a 404 status code. (The http 404 is the HTTP status code that is returned by the server when a requested URL can not be found.)
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + I think the number of 404 requests should be zero, but that's only a guess. I used the code below to count requests by URL and counted the number of 404 codes.
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + I was wondering how many GET's the /product filter is filtering, in order to get the current count.
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + 200 was the only response that was returned which was not due to server error.

Update: I got some better answers from other people in the answers below:

The request was NOT actually a 404, the status code was 5.

* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + 1.

#### What is the most common HTTP method?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + All. The most common HTTP method of course is GET. The server is telling the client that it needs some data from the request. Your browser is parsing the request and is sending the response.
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + GET, POST, PUT, DELETE, HEAD, OPTIONS, TRACE, TRACK, OPTIONS (HEAD and TRACE are also supported for OPTIONS responses)
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + PUT, DELETE. As you may have suspected, the HTTP methods by default are POST and GET, but the HTTP methods can be changed.
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + POST. I checked all the existing POST requests on this site: http://www.htc.org/htdocs/tools/request\_tools.php. One thing that stands out is POST requests with a 302:
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + GET. GET is the most common method (there are several other possible answers). Example: Using GET, GET http://www.test.com/products?product\_id=9008.
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + GET. The most common HTTP method is GET. HTTP/1.1 200 3075 Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:90.0). Geo/0.

#### What is the most common status code?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + 0 for all requests, but for some requests the status code can be different. Note: It is not possible for us to answer your question by saying "what is the most common status code" because we cannot know for a fact the…
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + 200 (OK) (or any other status code)
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + 302 on any HTTP status code, because the 302 response has already been removed: I can see how 302 is different from 302/301. But why does 302 code get removed when I access a link?
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + 302. I've seen this many different ways. Some people see it as an indication that the server is done processing the request. Other people think it's a redirect, which happens all the time. Others see it as an indication that the resource has…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + It depends... I'm thinking the answer will be different for different web servers (apache, tomcat, etc..). The most common status code is 500, which means that the request was not successful. The reason for this is that the…
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + 200 – ok, 301 - no content, 404 - no such resource, 500 - server error

#### What is the most common user agent?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + Google/Chrome/Safari, Google Chrome is used by most mobile users, Safari is used by some and Microsoft Edge is used by some too. You can use this chrome-detection script to detect the…
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + Android. "What is the most common user agent" is a very broad question, so let me try to help with a few other things, When a user is on your website, what user agent gets sent to…
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + Apple's UA is Safari, The most common user agent is Safari. Not because this is the most popular browser, but because it's the most common browser. As you know, in iOS Safari is the default browser…
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + Google Chrome, Chrome has a preference pane that shows the user agent in the "Advanced" tab. It can change the user agent to "Mozilla/5.0 (Windows NT 5.1; en…
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + The most common browser is Firefox. Browser: IE (Internet explorer), Safari, Google Chrome. OS: Mac OS X. Version: 14.0.5 (14A145). The most common browser is Firefox Browser.
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + It is Chrome.

#### Which User Agent was used most frequently?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + Mozilla/5.0 (X11; Linux i686; rv:2.0.1.2pre3-s7) Gecko/20100101 Firefox/3.6.8
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + It was my personal preference to use the first one that would match the request. Why: The user agent is a web-based tool that helps you identify the browser used to send the request. It's a bit tricky to get your head around…
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + It was "Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X)"
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + Mozilla/5.0 (iPhone; CPU iPhone OS 14\_6 like Mac OS X)
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + User Agent was used most frequently, it was a Chrome. When we switch to Opera, Safari or another browser, Chrome was the most preferred User Agent.
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + Chrome. It's not really a user agent problem, its a user's browser problem which causes issues like the current one.

#### What is the average response size for requests to /user/profile?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + I'd suspect that the max response size would be at least that seen here. However, /user/profile is a pretty large, complex page. Is there a simple way to find the average response time?
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + The average response size for requests to /user/profile is 2.4 KBytes, which is 5.5% of the 2.5 KBytes requested.
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + When I checked the Network tab for /user/profile, I saw that the average response size was 2KB, 2.1KB, 2.1KB, and 200 bytes.
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + According to the spec, the average response size is 0.7 KB. Is there a way to calculate the average response size?
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + 1,000 KB
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + The average response size for requests to /user/profile is 5 bytes. As of the time of this question the average response size is not tracked anywhere.

#### How many unique URLs were accessed?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + 1.6 billion.
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + The first query was found 0 times within one minute. The second query was found 0 times within one minute. The third query was found 0 times within one minute. The fourth query was found 0 times within one minute. The fifth query was found 0...
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + 6.
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + 4,086 unique requests.
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + It looks like the server returned all the links in the same order. However, there is a slight difference in the HTML source code. I found the problem by testing each URL individually. I tried to create a GET request to each URL in the
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + 0. You could try using curl\_multi instead of cURL. As for the "hundreds of thousands of unique URLs accessed": You're using curl\_multi to create multiple calls, which is a poor solution.

#### Which HTTP method was used most often for successful requests (status code 200)?

* Vektörleştirme: all-MiniLM-L6-v2, generator: gpt-neo-1.3B
  + GET.
* Vektörleştirme: bert-base-uncased, generator: gpt-neo-1.3B
  + POST. I've been browsing through the PHP logs and the only thing I see is POST, but I'm not sure if that makes any sense.
* Vektörleştirme: distilbert-base-uncased, generator: gpt-neo-1.3B
  + POST.
* Vektörleştirme: t5-small, generator: gpt-neo-1.3B
  + The GET method. The answer is simple, because the request URL and the request bodies are the same http://en.wikipedia.org/wiki/HTTP/1.0...
* Vektörleştirme: gpt-neo-1.3B, generator: gpt-neo-1.3B
  + Answer: POST, 302, 500... but most of time was 302.
* Vektörleştirme: roberta-base, generator: gpt-neo-1.3B
  + GET /payment HTTP/1.1 200 4983 Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:90.0) Gecko/20100101 Firefox/90.0. PUT /…

Modellerin bulunduğu sistemlere 8 farklı soru sorulup verilen cevaplar ChatGPT 4o modeliyle üretilen veri kümesi ile olabildiğince kontrol edildiğinde veri ile cevap verme gpt-neo-1.3B modelinin generator olduğu ve bert-base-uncased ile distilbert-base-uncased modellerinin vektörleştirme modeli olarak kullanıldığı sistemlerde bulunduğu söylenebilir. Onun haricindeki modellerde kafa karışıklığı yaşadığı görülmektedir. Hem generator hem de vektörleşme modeli olarak kullanılan gpt-neo-1.3B modelli sistemin yanında generator olarak gpt-neo-1.3B kullanılan sistemlerde internetten bilgilerden feyz alma eğilimi olduğu da görülmektedir.

Generator olarak kullanılan gpt2 ve gpt-neo-1.3B modelli sistemler pipeline süreci çok uzun sürdüğü için ayrı iki notebook’ta geliştirilip GitHub’a yüklenmiştir. outputs\_gpt2 ve outputs\_gptNemo isimli klasörlerde gerekli çıktılar, notebook’un bir kez çalıştırıldıktan sonra yer almaktadır. Böylece işlemler local’de de görülebilmektedir. Farklı sorular ile modellerin test edilmesi gereksiniminde son kısımdaki ‘Testing with quesitons’ kısmında ‘query’ değişkenine soruyu eşlemek yeterlidir. Testin, Kaggle üzerinde çalıştırılıp version oluşturulması pipeline süresi nedeniyle önerilmektedir. Eğer outputs oluşunca maksimum 30 saniye ile modeller tarafından yanıtlar alınabilmektedir. outputs\_gpt2 ve outputs\_gptNemo’ya ulaşmak için outputs\_links.txt’e ziyaret edip direkt Kaggle ve Colab ortamına (Local’de de istenirse yapılabilir.) hızlıca çekilebilir.