Belirli bir alanda uzmanlaşmış dil modelinin servis halinde sunulması

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Muhammed Burak Karahan

ORCID 0009-0003-2627-9687

Proje Danışmanı: Dr. Öğr. Üyesi Serpil Yılmaz

Ocak 2024

Belirli bir alanda uzmanlaşmış dil modelinin servis halinde sunulması

Öz

Bu proje, matematik ve fizik gibi temel bilimlerle ilgili veri setleri kullanılarak eğitilen uzmanlık dil modellerinin oluşturulması ve bu modellerin API servisi haline getirilerek farklı programlarda kullanılabilecek özelleştirilmiş GPT benzeri projelerin geliştirilmesi üzerine odaklanmaktadır. ArtifactAI/arxiv-math-instruct-50k ve ArtifactAI/arxiv-physics-instruct-tune-30k modelleri örnek olarak gösterilerek, bu modellerin özelleştirilmiş yapay zekaların nasıl oluşturulabileceğini, şirketlerin bilgileriyle kendi çözümlerini üreten GPT benzeri projelerin nasıl gerçekleştirilebileceğini açıklamaktadır.

Proje, matematik ve fizik alanlarında uzmanlık kazanmış dil modelleri olan ArtifactAI/arxiv-math-instruct-50k ve ArtifactAI/arxiv-physics-instruct-tune-30k modellerini kullanarak özelleştirilmiş bir dil modeli oluşturma sürecini detaylı bir şekilde ele alır.

Oluşturulan özelleştirilmiş dil modeli, bir REST API servisi haline getirilir. Bu adım, dil modelini kaydedip yerel dosya haline getirme sürecini açıklar. Ayrıca, bu dil modelini kullanarak Flutter ile geliştirilmiş mobil veya masaüstü uygulamalardan soru-cevap işlemlerini gerçekleştirebilecek GPT benzeri projelerin nasıl geliştirilebileceğini gösterir.

Proje, oluşturulan özelleştirilmiş dil modelin bir API servisi olarak sunulması ve bu modelin kullanımının nasıl gerçekleştirileceği üzerinde durur. Flutter ile geliştirilen

uygulama örneği üzerinden, mobil veya masaüstü uygulamalardan soru sorma ve cevaplama süreçlerinin nasıl gerçekleştirilebileceğini pratiğe dönük bir şekilde gösterir.

Proje, geliştirilen projelerin avantajları ve dezavantajları üzerinde değerlendirme yapar. Ayrıca, bu özelleştirilmiş GPT projelerinin gelecekteki potansiyelini ve bu alanda yapılacak olası çalışmalara yönelik önerileri içerir.

Bu proje, matematik ve fizik uzmanlık dil modelleri üzerinden özelleştirilmiş yapay zeka projelerinin oluşturulması sürecini ayrıntılı bir şekilde açıklar. Oluşturulan projelerin şirketlerin spesifik ihtiyaçlarına uygun çözümler sunabileceği ve mobil/ masaüstü uygulamalarda etkili bir şekilde kullanılabileceği gösterilir.

Anahtar Sözcükler: Yapay Zeka Modelleri, Matematik ve Fizik Eğitimi, Özelleştirilmiş Dil Modelleri, Şirket Bilgileri ile Eğitim, GPT Benzeri Projeler, Amazon Q Hizmeti, REST API, DRF, Django

Publishing language model specialized in certain areas as a service

Abstract

This project focuses on creating expertise language models trained on datasets related to fundamental sciences, particularly in mathematics and physics. The aim is to develop customized GPT-like projects by turning these models into API services that can be utilized across different programs. Using the ArtifactAI/arxiv-math-instruct-50k and ArtifactAI/arxiv-physics-instruct-tune-30k models as examples, the project illustrates how these models can be employed to create customized artificial intelligence (AI) solutions, generating proprietary answers tailored to specific company needs.

The project delves into the detailed process of creating a customized language model using expertise language models specialized in mathematics and physics, such as ArtifactAI/arxiv-math-instruct-50k and ArtifactAI/arxiv-physics-instruct-tune-30k. The resulting customized language model is transformed into a REST API service. This step explains the process of saving the language model and converting it into a local file. Additionally, it demonstrates how GPT-like projects, allowing question-answer interactions, can be developed using this language model in Flutter-based mobile or desktop applications.

The project emphasizes the deployment of the created customized language model as a REST API service and elucidates how to implement its usage. Through an example application developed with Flutter, it practically showcases the question-answering process from mobile or desktop applications. Evaluation of the advantages and

disadvantages of the developed projects is conducted in the project. Furthermore, the project provides recommendations for future studies and explores the potential of these customized GPT projects in the future.

This project meticulously explains the process of creating customized artificial intelligence projects using expertise language models in mathematics and physics. It demonstrates that the developed projects can offer tailored solutions to specific company requirements, proving effective in mobile and desktop applications.

Keywords: Artificial Intelligence Models, Mathematics and Physics Education, Customized Language Models, Training with Company Information, GPT-Like Projects, Amazon Q Service, Django, DRF, REST API

Teşekkür

Proje çalışmasına katkılarından dolayı Eray Özer'e (eray.ozer@ffreality.com) teşekkür ederim.

İçindekiler

Şekiller Listesi

Tablolar Listesi

Kısaltmalar Listesi

AI	Artificial intelligence
CRUD	Create Read Update Delete
CSRF	Cross Site Request Forgery
DRF	Django Rest Framework
FBE	Fen Bilimleri Enstitüsü
GPT	Generative Pre-trained Transformer
HTTP	Hypertext Transfer Protocol
IKCU	İzmir Kâtip Çelebi Üniversitesi
JSON	JavaScript Object Notation
LLM	Large Language Model
MVC	Model View Controller
MVT	Model View Template
ORM	Object-Relational Mapper
ORCID	Open Researcher and Contributor ID
REST	Representational State Transfer
UI	User Interface
URL	Uniform Resource Locator
VENV	Virtual Environment
VS	Visual Studio
XSS	Cross-site scripting attack
YZ	Yapay Zeka

BÖLÜM 1

Yapay Zeka

Bu bölümde HuggingFace üzerindeki ArtifactAI/arxiv-math-instruct-50k (Matthew Kenney, 2022) adlı bir belirli bir veri kümesi kullanılmıştır. Belirli bir alanda eğitilmiş yapay zekaya örnek olarak, önceden eğitilmiş GPT-2 üzerinde fine-tune[1] edilmiş olan Sharathhebbar24/math_gpt2 (Sharath S Hebbar, 2024) projesi kullanılmıştır. Böylece GPT-2 modeli matematik için daha spesifik sorulara cevap verebilir durumda olmuştur. Bu projede modelin eğitim sürecinden çok daha önce eğitilmiş ve/veya fine-tune edilmiş modelleri çıktı alarak nasıl servis olarak sunulacağı üzerinde durulmuştur.

1.1 LLM Nedir?

LLM (Large Language Model), büyük dil modeli anlamına gelir ve genellikle doğal dil işleme (NLP) görevlerinde kullanılan büyük boyutlu dil modellerini ifade eder. GPT-2 (Generative Pre-trained Transformer 2), OpenAI tarafından geliştirilen bir LLM örneğidir. Bu model, büyük bir dil modeli olup çeşitli dil görevlerini gerçekleştirebilir.

1.2 GPT-2 Nedir?

GPT-2 (Generative Pre-trained Transformer 2), OpenAI tarafından geliştirilen bir yapay zeka dil modelidir. GPT-2, büyük miktarda metin verisi üzerinde eğitilmiş ve genel dil anlama ve metin üretme görevlerinde etkileyici bir performans sergileyebilen bir modeldir.

---

[1] Fine-tune, önceden eğitilmiş bir makine öğrenimi modelinin belirli bir görev veya veri kümesi üzerinde daha spesifik hale getirilmesi anlamına gelir. Bu süreç, genellikle modelin başlangıç ağırlıklarını (weights) koruyarak, özel bir görev veya veri kümesine daha iyi uymasını sağlamak için modelin daha fazla eğitim almasını içerir. (Ian Goodfellow and Yoshua Bengio and Aaron Courville, 2016)

GPT-2'nin temelinde "Transformer" adı verilen bir mimari bulunmaktadır. Bu mimari, özellikle dil işleme görevlerinde başarılı olan bir derin öğrenme modelidir. Transformer mimarisi, metin verilerini anlamak ve üretmek için öğrenilmiş temsilleri kullanır.

GPT-2'nin önemli özelliklerinden biri, büyük bir dil modeli olmasıdır. Model, 1.5 milyar parametre içermektedir, bu da onu o dönemdeki en büyük dil modellerinden biri yapmaktadır. Bu büyük boyuttaki model, genel dil anlama görevlerinde ve çeşitli metin tabanlı uygulamalarda etkileyici sonuçlar elde etmeye olanak tanır.

GPT-2'nin önceden eğitilmiş bir dil modeli olması, belirli görevler veya uygulamalar için özelleştirilebilmesine olanak tanır. Bu, GPT-2'nin çeşitli metin tabanlı görevlerde ve uygulamalarda kullanılabilmesini sağlar, örneğin metin üretimi, dil anlama, çeviri ve daha birçok alan.

1.2.1 GPT-2 Çalışma Mantığı

- *Transformer Mimarisi*: GPT-2, Transformer adı verilen bir mimariyi kullanır. Transformer, özellikle dikkat mekanizmasıyla bilinen bir mimaridir. Bu, modelin belirli kelimelere daha fazla dikkat vermesine ve bağlamı daha iyi anlamasına olanak tanır.

- *Önceden Eğitilmiş Model:* GPT-2, büyük bir dil modeli olarak önceden eğitilmiştir. Geniş bir metin veri kümesinde (örneğin, internet üzerindeki milyonlarca web sayfası) eğitilir. Bu sayede genel dil bilgisi kazanır.

- *Tek Adımlı Tahmin*: GPT-2'nin temel özelliği, bir kelimeyi tahmin etme yeteneğidir. Model, verilen bir kelimenin ardından gelebilecek en olası kelimeleri sıralar.

- *Çoklu Görev Yeteneği*: GPT-2, çeşitli dil görevlerini tek bir modelde birleştirir. Örneğin, metin tamamlama, çeviri, soru-cevap gibi görevlerde kullanılabilir.

- *Transfer Öğrenme*: GPT-2, önceden eğitildikten sonra belirli bir görev veya veri kümesine uyarlanabilir. Bu, fine-tuning olarak adlandırılır ve modelin belirli bir uygulamada daha spesifik olmasını sağlar.

1.2.2 GPT-2 Görselleştirilmesi

OpenAI, GPT-2 modelini kötüye kullanım endişeleri nedeniyle tam olarak yayınlamamış olsa da, orijinal GPT'nin boyutuna eşdeğer, ancak yeni, daha büyük bir veri seti üzerinde eğitilmiş, daha küçük bir versiyonunu yayınladı (117 M parametre). Büyük model kadar güçlü olmasa da, bu daha küçük versiyonun da belirli dil üretme yetenekleri bulunmaktadır. Görselleştirme kullanarak bu modeli daha iyi anlayabilir miyiz?

Görselleştirme için Jesse Vig'in Apache-2.0 Lisanslı bertviz[2] kütüphanesi incelenebilir.

GPT-2'nin bir cümle tamamlama kabiliyetini inceleyecek olursak, örneğin prompt olarak

> "On the string of one kite we saw Mother's new gown! Her gown with the dots that are pink, white and…"

Cümlesi verildiğinde, GPT-2 bize şu cevabı vermiştir,

> "Her gown with the dots that are pink, white and **blue.**"

Orijinal metin kırmızıdır, en azından bunun sadece ezberlemek olmadığını biliniyor, Peki GPT-2 renk seçmeyi nasıl biliyor? Virgülle ayrılmış listeleri tanıyan aşağıdaki dikkat modeli nedeniyle olabilir:

[2] https://github.com/jessevig/bertviz

Şekil 1.1 GPT-2 Görselleştirme

Model *virgül* ve *ve*'den sonraki kelimeye karar vermek için dikkati listedeki önceki öğelere (pembe ve beyaz) odaklıyor. Önceki öğelerin türüyle eşleşen bir kelimeyi, bu durumda bir rengi seçmeyi biliyor.

Bir deneme olarak, GPT-2'nin "<isim> kim?" isteminden metin oluşturmasını deneniyor. Bu özel istek genellikle modeli kısa bir biyografi yazmaya teşvik eder; bunun nedeni muhtemelen Web'deki makalelerde yazar biyografisi için ortak bir önsöz olmasıdır.

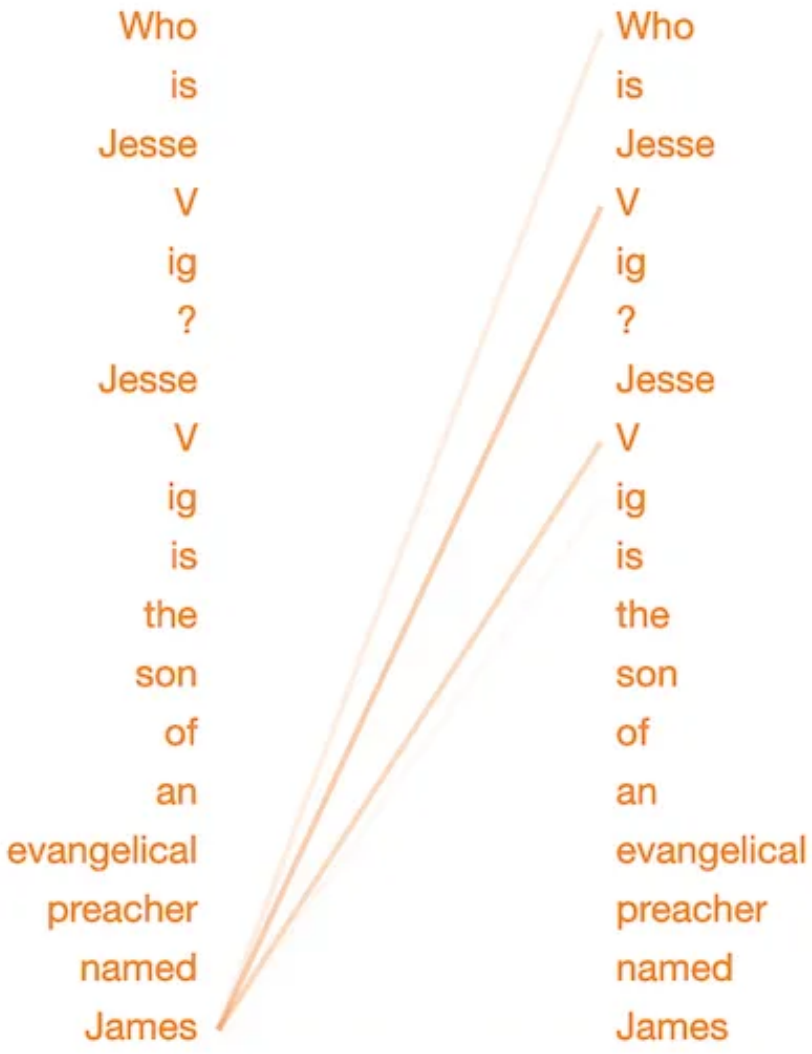
İşte “Jesse Vig Kimdir?” istemi için oluşturulan birkaç biyografi:

> “Jesse Vig is a social media marketing expert and former social media marketing manager. He is the co-founder and CEO of VigMedia.com and recently the founder and CEO of VigMedia.com.”

Biraz tekrarlı ama hikayeyi beklenmedik bir şekilde kişiselleştirme konusunda iyi bir iş çıkardı. Bir tane daha örnek denenmiştir:

> “Jesse Vig is the son of an evangelical preacher named James Vig. He moved to the United States in 1964 and became a preacher at the University of Michigan, where he taught for 18 years until his death in October 2007.”

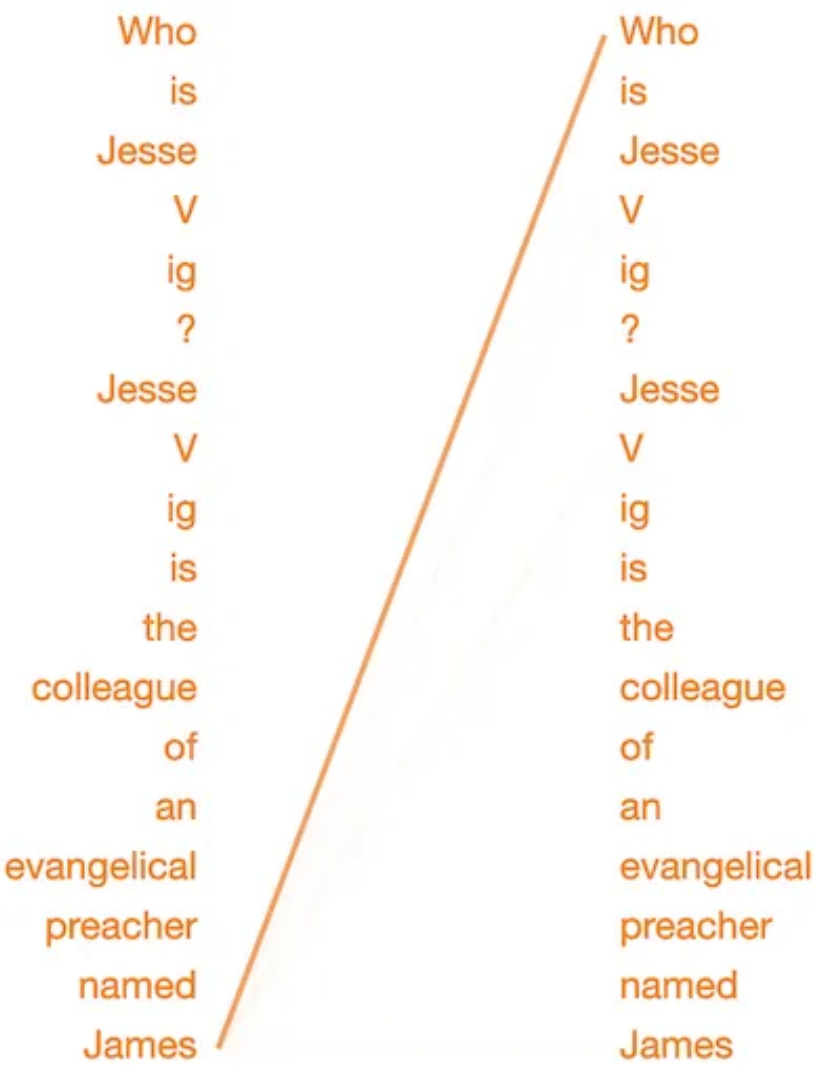
Bu son örnekte GPT-2, ikinci kişiliğimin babasının da aynı soyadına sahip olduğunu bilecek kadar akıllı. GPT-2 bu soyadını seçerken dikkatini nereye yoğunlaştığını anlamak için aşağıdaki şekile bakıldı:



Şekil 1.2: GPT-2 Görselleştirme 2

James'ten sonra tahmin edilecek kelimeyi seçerken, bu desen dikkati soyadın önceki geçişlerine odaklıyor. (Unutulmamalıdır ki, model içinde "Vig" kelimesi sık kullanılmayan bir kelime olduğu için "V" ve "ig" şeklinde kelime parçalarına ayrılmıştır.) Bu dikkat deseninin aile isimleri arasındaki ilişkileri belirleme konusunda uzmanlaştığı görünüyor. Bunun test etmek için metni hafifçe değiştirildi:

"Jesse Vig is the **colleague** of an evangelical preacher named James…"



Şekil 1.3 - GPT-2 Görselleştirme 3

Artık James sadece bir meslektaş olduğundan, bu dikkat modeli soyadımı neredeyse tamamen görmezden geliyor.

GPT-2'nin bir isimle ilişkilendirilen algılanan etnik kökene ve cinsiyete dayalı biyografiler oluşturduğu görülmektedir. Modelin hangi önyargıları kodlayabileceğini görmek için daha fazla çalışmaya ihtiyaç vardır; Bu konu hakkında daha fazlasını buradan[3] okuyabilirsiniz. (Jesse Vig, 2019)

1.3 Özelleştirilmiş Veri ve Fine-Tune

Fine-tune işlemi, önceden geniş bir veri kümesi üzerinde eğitilmiş bir modelin, belirli bir uygulama veya alan için daha spesifik bilgiler öğrenmesini sağlamak amacıyla kullanılır. Bu, daha az veriyle daha spesifik bir görevde daha iyi performans elde etmek için kullanışlı olabilir.

Örneğin, GPT-2 gibi büyük dil modelleri genel dil anlama görevlerinde geniş bir veri kümesi üzerinde eğitilebilir. Ancak, bu modeli belirli bir konuda daha iyi performans göstermesi için fine-tune etmek istiyorsanız, örneğin tıbbi metinlerdeki hastalık teşhisi gibi, modeli bu özel konuya uygun hale getirebilirsiniz. Fine-tune sürecinde, modelin öğrenme oranı, epoch sayısı ve diğer hiperparametreler genellikle ayarlanarak, modelin hedef görevde daha iyi performans elde etmesi hedeflenir. (Ian Goodfellow and Yoshua Bengio and Aaron Courville, 2016)

1.3.1 Veri Kümesini İnceleme

Veri kümesi, eğitim için kullanılacak soru-cevap çiftlerini içerir. Her bir örnek, bir soru ve buna ait bir cevaptan oluşur. Toplamda 50,488 soru-cevap çiftinden oluşur.

[3] http://www.fatml.org/

Tablo 1.1: Veri seti örnek 5 soru cevap

Soru	Cevap
What structure is classified as a definite lie algebra?	A definite Lie algebra is a Lie algebra equipped with an inner product that is positive definite. Such an algebra is called a positive definite or a definite Lie algebra.
What type of laser is used to study infrared?	The type of laser used to study infrared is a tunable infrared laser. This type of laser emits light in the infrared spectrum and its wavelength can be adjusted or "tuned" to match the desired wavelength range for the study.
What type of detector detects photon arrival?	A photodetector or a photon detector is a type of detector that detects the arrival of individual photons or light particles. Examples of photodetectors include photomultiplier tubes, avalanche photodiodes, and single photon detectors.
Can a qw be made shapeless?	No, a qw cannot be made shapeless. "qw" is a combination of letters, and letters by their nature have a shape.
Which of the following is the only finite width of quark gluon plasma bags?	The only finite width of quark gluon plasma bags is the coherence length, which refers to the distance over which the coherence of the collective motion persists.

1.3.2 Veri Hazırlama

Tokenleme (Tokenize), Soruları ve cevapları modele vermek için, metinleri tokenlara ayırmak gerekebilir. GPT-2 modeli genellikle kelime veya alt kelime seviyesinde çalışır. Input-Output Formatı, GPT-2 modeli, belirli bir metni alıp ardından bir metin üretir. Bu nedenle, eğitim veri setinizi bu giriş-çıkış formatına uygun hale getirmemiz gerekecektir.

1.3.3 Model Seçimi ve İnce Ayar (fine-tune)

Projede GPT-2 modeli kullanılmıştır, Hugging Face Transformers modeli indirilmesi ve kurmanız gerekecektir ardından veri kümesi ve modeli kullanarak eğitim işlemine başlanabilir. Modelinizi daha önce eğitilmiş bir GPT-2 modeliyle başlanılabilir ve ardından belirtilen veri kümesi üzerinde ince ayar yapılabilir. Fakat bu projede fine-tune işlemini de yapmayıp, doğrudan daha önce fine-tune edilmiş model kullanılmıştır. Bunun nedeni modelin eğitimi ve ince ayar kısmına değil servis haline getirilmesine odaklanılmıştır. (Amazon Web Services, 2024)

1.4 Kullanım senaryoları

Büyük dil modelleri, geniş bir dil anlama yeteneğine sahip olmalarıyla bilinir. Ancak, özelleştirilmiş büyük dil modelleri elde etmek için fine-tune işlemi, belirli şirket veya endüstri ihtiyaçlarına daha iyi uyan modeller oluşturabilir. Bu bölümde, şirketlerin kendi iç veri setleri üzerinde fine-tune edilmiş büyük dil modellerini nasıl kullanabilecekleri üzerinde durulacaktır.

1.4.1 Örnek kullanım alanları

- *İnsan Kaynakları ve İş Performansı*: Şirketler, kendi iç İK veri setleriyle fine-tune edilmiş büyük dil modellerini kullanarak, işe alım süreçlerini optimize edebilir ve performans değerlendirmelerini daha etkili bir şekilde gerçekleştirebilir. Bu modeller, özlük bilgilerini analiz ederek eğitilebilir ve işgücü planlamasına katkıda bulunabilir.

- *Mühendislik Dokümanları ve Proje Yönetimi:* Mühendislik departmanları, kendi projeleri ve bakım süreçleri üzerine fine-tune edilmiş modellerle, mühendislik dökümanlarını otomatik olarak anlamlandırabilir. Bu sayede, proje yönetimi ve bakım süreçleri daha etkili hale getirilebilir.

- *Üretim ve Bakım Metrikleri:* Üretim ve bakım süreçleri ile ilgili veri setleri üzerinde fine-tune edilen modeller, şirketlere üretim verimliliği, makine bakım süreçleri ve operasyonel mükemmelliği artırmak için değerli iç görüler sağlayabilir.

- *Müşteri Hizmetleri ve Sorun Çözme:* Şirketler, müşteri şikayetleri ve destek talepleriyle ilgili verilerle fine-tune edilmiş büyük dil modellerini kullanarak, müşteri hizmetleri süreçlerini optimize edebilir ve hızlı sorun çözme süreçleri geliştirebilir.

- *Finansal Analiz ve Tahmin:* Finans sektöründeki şirketler, kendi finansal verileri üzerinde fine-tune edilmiş modellerle, finansal analizleri ve tahminleri daha doğru bir şekilde gerçekleştirebilir. Bu, risk yönetimi ve stratejik planlama süreçlerini güçlendirebilir.

Kendi iç veri setleri üzerinde fine-tune edilmiş büyük dil modelleri, şirketlere özelleştirilmiş ve daha etkili çözümler sunma potansiyeli taşır. Ancak, bu süreçte veri gizliliği ve etik sorunlarına dikkat edilmelidir. Ayrıca, doğru hiperparametre ayarları ve eğitim stratejileri kullanılarak, bu modellerin yüksek performans elde etmesi sağlanabilir.

1.5 Sektörde kullanım örneği: Amazon Q

Amazon Q, geniş bir işlev yelpazesi ile kullanıcılara çeşitli konularda yardımcı olan bir konuşma tabanlı yapay zeka asistanıdır. Bu aracın web tabanlı arayüzü, çalışanların daha verimli bir şekilde çalışmalarına olanak tanırken, bir dizi görevi yerine getirmek için entegre veri ve bilgileri kullanır.

1.5.1 Amazon Q’nun Özellikleri

- *Soru Cevaplama ve Bilgi Oluşturma*: Şirketler, kendi iç İK veri setleriyle fine-tune edilmiş büyük dil modellerini kullanarak, işe alım süreçlerini optimize edebilir ve performans değerlendirmelerini daha etkili bir şekilde gerçekleştirebilir. Bu modeller, özlük bilgilerini analiz ederek eğitilebilir ve işgücü planlamasına katkıda bulunabilir.

- *Veri Entegrasyonu:* Mühendislik departmanları, kendi projeleri ve bakım süreçleri üzerine fine-tune edilmiş modellerle, mühendislik dökümanlarını otomatik olarak anlamlandırabilir. Bu sayede, proje yönetimi ve bakım süreçleri daha etkili hale getirilebilir.

- *Gelişmiş Soru Anlama Yeteneği*: Amazon Q, takip sorularını anlama yeteneğine sahiptir. Bu, bir soruya gelen takip sorularına mantıklı yanıtlar verebilme kabiliyetini içerir.

- *Görevlerin Tamamlanması*: Amazon Q, veri analizi, rapor özeti oluşturma, içerik üretme gibi çeşitli görevleri yerine getirebilir, bu da çalışanların daha etkili bir şekilde işlerini yapmalarına yardımcı olur.

- *Referans ve Kaynak Bildirimi*: Amazon Q, yanıtlarını oluştururken kullandığı kaynaklara referans ve atıflar sağlayarak kullanıcı güvenini artırır.

1.5.2 Fine-Tune edilmiş GPT ve Amazon Q’nun karşılaştırılması

Amazon Q'nun özellikleri, bir dereceye kadar, GPT-2 veya benzeri büyük dil modellerini fine-tune etmiş özel bir dil modeline benzer. Ancak, Amazon Q'nun spesifik bir iş uygulamasına yönelik olarak tasarlanmış olması, şirket içi veri sistemleriyle entegrasyonu ve referans bildirimi gibi özellikleri, genel bir dil modelinden farklılık gösterir. Özellikle iş süreçlerine entegre edilmiş olması, özelleştirilmiş çözümler sunma yeteneğini artırır.

Amazon Q'nun açıkça belirtilen avantajları, kullanıcı dostu bir arayüz, geniş bir işlev yelpazesi, veri entegrasyonu ve referans bildirimi gibi özelliklerle çalışanların iş süreçlerini daha etkili bir şekilde yönetmelerine yardımcı olabilir.

1.6 Geliştirme Ortamı

Modelin tokenize edilmesi ve .pt dosya formatında kaydedilmesi için, Amazon SageMaker[4] kullanılmıştır. SageMaker üzerinde ml.p3.2xlarge instance tipi kullanılmıştır. 1 Adet Nvidia Tesla V100 16 GB VRAM, 8 Çekirdek işlemci 61 GB RAM özelliklerindeki bir sanal bilgisayar kullanılmıştır. Jupyter Lab üzerinde kernel olarak PyTorch 2.0.0, Python 3.10 GPU Optimized kullanılmıştır.

1.7 Kodlar

1.7.1 Eğitilmiş Modeli Yükleme ve Dışarı aktarma

Amazon SageMaker'da PyTorch ile bir JupyterLab notebook'u oluşturuldu, gerekli Python paketleri aşağıdaki komutla yüklendi.

```
% !pip install transformers torch
Requirement already satisfied: transformers in /opt/conda/lib/python3.10/site-
packages (4.37.2)
Requirement already satisfied: filelock in /opt/conda/lib/python3.10/site-packages
(from transformers) (3.12.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /opt/conda/lib/
python3.10/site-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.10/site-
packages (from transformers) (1.23.5)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.10/site-
packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/lib/python3.10/site-
packages (from transformers) (5.4.1)
Requirement already satisfied: regex!=2019.12.17 in /opt/conda/lib/python3.10/
site-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-packages
(from transformers) (2.28.2)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /opt/conda/lib/
python3.10/site-packages (from transformers) (0.15.1)
```

[4] Amazon SageMaker, Amazon Web Services (AWS) tarafından sunulan bir hizmettir ve endüstri standardında makine öğrenimi modelleri oluşturmak, eğitmek ve dağıtmak için kullanılır. Bu hizmet, kullanıcıların makine öğrenimi projelerini daha hızlı ve daha etkili bir şekilde geliştirmelerine olanak tanır.

```
Requirement already satisfied: safetensors>=0.4.1 in /opt/conda/lib/python3.10/
site-packages (from transformers) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in /opt/conda/lib/python3.10/site-
packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec>=2023.5.0 in /opt/conda/lib/python3.10/site-
packages (from huggingface-hub<1.0,>=0.19.3->transformers) (2023.5.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /opt/conda/lib/
python3.10/site-packages (from huggingface-hub<1.0,>=0.19.3->transformers)
(4.5.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/conda/lib/
python3.10/site-packages (from requests->transformers) (3.1.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.10/
site-packages (from requests->transformers) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.10/site-
packages (from requests->transformers) (2023.5.7)
WARNING: Running pip as the 'root' user can result in broken permissions and
conflicting behaviour with the system package manager. It is recommended to use
a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: There was an error checking the latest version of pip.
```

Ardından gerekli kütüphaneler içeri aktarıldı.

```
% from transformers import AutoTokenizer, AutoModelForCausalLM
% import torch
```

Ardından önceden eğitilmiş modelin ismi girildi, model ve tokenizer değişkenlerini hazırlandı

Çıktı verecek fonksiyon hazırlandı.

```
def generate_text(prompt):
    inputs = tokenizer.encode(prompt, return_tensors='pt')
    outputs = model.generate(inputs, max_length=64,
    pad_token_id=tokenizer.eos_token_id)
    generated = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return generated[:generated.rfind(".")+1]
```

Bir örnek denendi.

```
% prompt = "Which transition occurs in a quartic potential barrier?"
% res = generate_text(prompt)
% res

'Which transition occurs in a quartic potential barrier?\n\nThe transition occurs in a
quartic potential barrier when the energy of the electrons in the system is equal to
the energy of the electrons in the system.'
```

Torch modeli ve tokenizer'i kaydetmek için aşağıdaki kodlar yazıldı.

```
# Tokenize the prompt
inputs = tokenizer.encode(prompt, return_tensors='pt')

# Run the forward pass
with torch.no_grad():
    outputs = model(inputs)

# Save the model to a .pt file
torch.save(model.state_dict(), "math_gpt2.pt")

# Save the tokenizer configuration to a file (optional)
tokenizer.save_pretrained("math_gpt2_tokenizer")
```

Bu kod klasörde math_gpt2.pt dosyası ve math_gpt2_tokenizer klasörü oluşturacaktır. Oluşturulan klasörün içerisinde 5 adet dosya bulunmalıdır bunlar, merges.txt, special_tokens_map.json, tokenizer_config.json, tokenizer.json ve vocab.json.

Bu dosya ve klasörü bölüm 2'de oluşturulacak REST API sunucumuzun model klasörüne kopyalanacaktır. Daha sonra bu 6 dosya ile modelimizi yükleyip cevap alınabilecektir.

BÖLÜM 2

Django ile RESTful API oluşturma

Bu bölümde projenin yapay zeka modelinin .pt dosya uzantısı şeklinde kaydedilmiş yerel dosyalarla yüklenip REST API olarak sunulması anlatılacaktır.

2.1 Arka-yüz çalışma senaryosu

Proje'de *REST API*[5] oluşturmak için *Django*[6] ve üzerine eklenecek *Django Rest Framework*[7] (bundan sonra *DRF* olarak anılacak) kullanılacaktır. *DRF* yardımıyla bir *POST*[8] *endpoint*'i (bundan sonra *uç-nokta* olarak anılacaktır) oluşturulup, buraya *JSON* tipinde bir istek atıp *prompt* isimli değerimizi sormak istediğimiz soru olarak göndereğiz, model'e verdiğimiz girdi olarak bu *prompt*'u kullanıp, çıktıyı da başarıyla cevap alabildiysek HTTP 200 status koduyla *JSON* tipinde alacağız.

2.2 Django nedir?

Django, Python tabanlı bir web uygulama çerçevesidir ve genel mantığı, etkili ve hızlı web uygulamaları geliştirmeyi kolaylaştırmak üzerine odaklanmıştır. Django'nun genel mantığı şu temel prensipler üzerine kurulmuştur:

---

[5] REST API, "Representational State Transfer" kısaltmasıyla bilinen bir mimari stildir ve genellikle web tabanlı servislerin iletişimini kurmak için kullanılır. REST, ölçeklenebilir, hafif ve basit bir yapı sunar. Temelinde kaynaklar (resources) üzerindeki işlemleri (GET, POST, PUT, DELETE vb.) tanımlayan bir yaklaşımı benimser. Bir RESTful API (Application Programming Interface), bu mimari stili takip eden ve genellikle HTTP protokolünü kullanarak kaynaklar üzerinde çeşitli operasyonları gerçekleştiren bir web servisidir.

[6] Django, Python programlama dilinde yazılmış açık kaynaklı bir web uygulama çerçevesidir. Web uygulamaları ve web siteleri geliştirmek için kullanılır, https://www.djangoproject.com

[7] Django Rest Framework (DRF), Django ile web tabanlı RESTful API'lar oluşturmak için kullanılan güçlü bir üçüncü taraf kütüphanedir. Django'nun sağladığı güçlü özellikleri ve ORM sistemini temel alarak, RESTful API geliştirmeyi kolaylaştıran bir dizi araç ve görünüm (view) sağlar.

[8] Bir POST endpoint'i, bir web uygulamasındaki belirli bir URL (Uniform Resource Locator) veya route (yol) üzerindeki bir noktadır. Bu endpoint, HTTP POST istekleri aldığında belirli bir işlemi gerçekleştiren bir web servisini temsil eder.

2.2.1 Model-View-Template (MVT) Mimarisi

Django, genellikle Model-View-Controller (MVC) olarak adlandırılan bir mimariyi benimser, ancak bu modeli kendi şeklinde uygular ve "Model-View-Template (MVT)" olarak adlandırır. Bu, uygulamanın veritabanı modellerini (Model), kullanıcı arayüzünü (View), ve iş mantığıyla ilgili işlemleri (Template) içerir.

2.2.2 Nesne-İlişkisel Haritalama (ORM)

Django, bir Nesne-İlişkisel Haritalama (ORM) sistemini içerir. Bu, veritabanındaki tabloları Python sınıfları olarak temsil etmenizi sağlar. Veritabanı işlemleri için SQL sorguları yazmak yerine, Python kodu kullanabilirsiniz.

2.2.3 Şablon Motoru

Django'nun kendi şablon motoru vardır. Bu, HTML sayfalarını oluşturmak ve dinamik içerik eklemek için kullanılır. Şablonlar, veritabanından alınan verileri görüntülemek ve kullanıcılara içerik sunmak için kullanılır.

2.2.4 URL Yönlendirme Sistemi

Django, URL yönlendirme sistemine sahiptir. Bu, kullanıcıların belirli URL'leri ziyaret ettiğinde hangi görünümlerin çalıştığını belirlemenizi sağlar. URL'leri belirli bir görünümle eşleştirmek için düzenli ifadeler kullanabilirsiniz.

2.2.5 Admin Paneli

Django, otomatik olarak oluşturulan bir yönetici paneli (admin panel) içerir. Bu panel, veritabanındaki verileri görüntülemek, eklemek, güncellemek ve silmek gibi yönetim görevlerini kolaylaştırır.

2.2.6 Güvenlik Özellikleri

Django, güvenlik konusuna büyük önem verir. Cross-Site Scripting (XSS[9]) ve Cross-Site Request Forgery (CSRF[10]) saldırılarına karşı otomatik koruma sağlar. Ayrıca, şifreleme ve oturum yönetimi gibi güvenlik önlemlerini içerir.

2.2.7 Form ve Doğrulama

Django, web formları oluşturmayı ve işlemeyi kolaylaştıran bir form API'si içerir. Ayrıca, kullanıcı kimlik doğrulaması ve yetkilendirme işlemlerini sağlayan bir sistem de içerir.

2.2.8 MVC Kavramını Benimseme

MVT mimarisine rağmen, Django, kullanıcı arayüzü, iş mantığı ve veritabanı etkileşimini birbirinden ayırmayı ve modüler bir yapı oluşturmayı hedefler. Django, bu temel prensipleri kullanarak hızlı ve güvenli web uygulamaları geliştirmeyi sağlayan bir çerçevedir. Bu prensipler, Django'nun kullanıcı dostu, modüler ve ölçeklenebilir bir çerçeve olmasına katkıda bulunur. (Django Software Foundation ve bireysel katkı sağlayıcılar, 2024)

Bu projede, hızlı bir şekilde oluşturma ve yapay zeka sistemlerinin genellikle Python üzerinde koşuyor olmasından dolayı kullanılmıştır.

2.3 Django Rest Framework nedir?

Django Rest Framework (DRF), Django ile birlikte kullanılan bir RESTful web servisi geliştirme çerçevesidir. DRF, Django'nun güçlü yapısını kullanarak, web servisleri ve API'lar oluşturmak için ekstra özellikler ekler.

---

[9] XSS, web uygulamalarında yaygın olarak görülen bir güvenlik açığıdır. Bu saldırıda, kötü niyetli bir kullanıcı tarafından gönderilen veya enjekte edilen kodlar, bir web sitesindeki kullanıcıların tarayıcılarında çalıştırılır. Bu, kullanıcıların tarayıcıları üzerinden çeşitli zararlı eylemler gerçekleştirmek amacıyla kötü niyetli kodları çalıştırma olasılığını içerir. (OWASP, 2024)

[10] CSRF, bir kullanıcının tarayıcısında oturum açıkken, kullanıcının bilgisi veya izni olmadan kötü niyetli bir saldırgan tarafından oluşturulan bir isteğin gönderilmesini içerir. Bu saldırı, kullanıcının kimlik doğrulamasını kullanarak, yanıltıcı bir şekilde kullanıcı adına işlem yapma yeteneğini hedefler. (OWASP, 2024)

2.3.1 Serileştirici (Serializer)

DRF, Django modellerini ve sorgu kümelerini JSON veya diğer formatlara dönüştürmek için kullanılan Serializerlar aracılığıyla veri serileştirme ve ayrıştırma işlemlerini kolaylaştırır. Bu, veritabanındaki bilgileri kullanıcı dostu bir formata dönüştürmeyi sağlar.

2.3.2 Görünümler (Viewset)

Django Rest Framework, Django'nun view sistemini genişleterek API'lar için özel viewset'ler ve görünümler sunar. Viewset'ler, CRUD (Create, Read, Update, Delete) operasyonlarını işleyen bir tür Controller olarak düşünülebilir. DRF, önceden tanımlanmış genel görünümler de sağlar.

2.3.3 URL Yönlendirilmesi

Django Rest Framework, Django'nun URL yönlendirmesi sistemini kullanarak, API'larınızın belirli URL'lerle eşleştirilmesini sağlar. URL konfigürasyonu, hangi viewset ve görünümlerin hangi URL'leri işleyeceğini belirler.

2.3.4 Kimlik Doğrulamalar (Autentication) ve İzinler (Permissions)

Django Rest Framework, API'larınızın güvenliğini sağlamak için kimlik doğrulama (authentication) ve izin (permission) sistemlerini içerir. Bu, hangi kullanıcıların API'ya erişebileceğini ve hangi işlemleri gerçekleştirebileceğini kontrol etmenizi sağlar.

2.3.5 Sayfalama ve Filtreleme

DRF, büyük veri kümeleri ile çalışırken sayfalama özelliği sunar. Ayrıca, veri kümelerini belirli kriterlere göre filtreleme yeteneği de vardır.

2.3.6 Yönlendiriciler (Router)

Django Rest Framework, API'larınızın URL yapılarını otomatik olarak oluşturmanıza yardımcı olan router'ları içerir. Bu, URL konfigürasyonunu daha modüler ve otomatik hale getirir.

2.3.7 Kimlik Doğrulama Sınıfları (Authentication Classes) ve İzin Sınıfları (Permission Classes)

DRF, kullanıcı kimlik doğrulama ve izinleri özelleştirmenize olanak tanıyan çeşitli sınıflar içerir. Bu sayede güvenlik gereksinimlerinize göre özelleştirmeler yapabilirsiniz.

2.3.8 İç İçe Serileştiriciler (Nested Serializers)

DRF, iç içe geçmiş modeller veya ilişkili modellerle çalışmayı kolaylaştıran iç içe geçmiş (nested) serializer'ları destekler.

Django Rest Framework, RESTful API'lar oluşturmak için Django'nun gücünü kullanır ve bu API'ları geliştirmek, bakımını yapmak ve genişletmek için çeşitli özellikler ve araçlar sunar. Bu, geliştiricilere hızlı ve güçlü bir API geliştirme deneyimi sağlar. (Encode OSS Ltd, 2024)

2.4 Geliştirme ortamı

Arka-uç'ta[11], *Python 3.9.6* kullanılmıştır, doğrudan sistem genelinde değil de, paket

yönetimini kolaylaştırmak amacıyla Python ortamı (Python3'ün kendi *venv*[12] modülü) kullanılacaktır. Yerel olarak kullanılan bilgisayar, *macOS 14.2.1 (23C71)* işletim sistemli bir bilgisayardır. 16 GB RAM ve *Apple Silicon M2* işlemcisi bulunmaktadır. Metin editörü olarak *VS Code*[13] kullanılmaktadır. Projeyi başlatmak

[11] Arka-uç (Back-end) bir bilgisayar uygulamasının veya web sitesinin, kullanıcının görmeyeceği, genellikle sunucu tarafında çalışan ve uygulamanın mantıksal işlemlerini gerçekleştiren kısmını ifade eder. Back-end, uygulamanın veritabanı yönetimi, iş mantığı, kullanıcı doğrulama ve diğer çeşitli görevleri ele alır.

[12] python3 -m venv venv komutuyla rahatlıkla ulaşılabilir

[13] Visual Studio Code, çok popüler metin editörü

ve çeşitli işlemler için doğrudan *Zsh* (Z Shell)[14], terminali kullanılmıştır. Projede sunucu, Wi-Fi üzerinden yerel IPi üzerinden ağa sunulacaktır.

2.5 Arka-yüz kodları

Öncelikle kodlarımızı yazacağımız klasörde bir terminal açılmış ve aşağıdaki komut çalıştırılarak kurulu Python'nın sistemimizde hangi versiyon olduğu kontrol edilmiştir.

```
% python3 —version
Python 3.9.6
```

Ardından bir Python ortamı oluşturulmuştur.

```
% python3 -m venv venv
```

Oluşturulan ortam, mevcut terminalde aktif edilmiştir

```
% source venv/bin/activate
```

Bu komutu çalıştırdıktan sonra terminalin aktif satırında (venv) görüyor olmalıyız. Örnek çıktımız aşağıdaki şekilde görülmüştür.

```
(venv) marlonjd@MarlonJD-MacBook-Pro django_llm %
```

Django projesi oluşturmak için yeni ortamımıza pip yardımıyla aşağıdaki komutla Django kurulmuştur.

```
% pip install Django
```

Bulunan klasöre Django projesi şu komutla eklenmiştir.

```
% django-admin startproject project .
```

[14] Bash'a benzer bir Unix kabuğu

Ardından projenin daha derli toplu durması amacıyla gereksiz dosyaların Git'e[15] eklenmemesi için bir **.gitignore** dosyası oluşturulmuştur

```
% touch .gitignore
```

Oluşturulan dosyaya metin editörümüzden girip, aşağıdaki satırlar eklenmiştir.

```
# Created by https://www.toptal.com/developers/gitignore/api/macos,django
# Edit at https://www.toptal.com/developers/gitignore?templates=macos,django

### Django ###
*.log
*.pot
*.pyc
__pycache__/
local_settings.py
db.sqlite3
db.sqlite3-journal
media

# If your build process includes running collectstatic, then you probably don't need
or want to include staticfiles/
# in your Git repository. Update and uncomment the following line accordingly.
# <django-project-name>/staticfiles/

### Django.Python Stack ###
# Byte-compiled / optimized / DLL files
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
.eggs/
lib/
lib64/
parts/
sdist/
var/
wheels/
share/python-wheels/
*.egg-info/
.installed.cfg
*.egg
```

[15] Git, bir versiyon kontrol sistemi (VCS) olarak bilinen açık kaynaklı bir yazılımdır. Git, yazılım geliştirme projelerinde kaynak kodu ve dosyaların sürümlerini takip etmek, yönetmek ve işbirliği yapmak için kullanılır.

```
MANIFEST

# PyInstaller
#  Usually these files are written by a python script from a template
#  before PyInstaller builds the exe, so as to inject date/other infos into it.
*.manifest
*.spec

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.nox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
*.py,cover
.hypothesis/
.pytest_cache/
cover/

# Translations
*.mo

# Django stuff:

# Flask stuff:
instance/
.webassets-cache

# Scrapy stuff:
.scrapy

# Sphinx documentation
docs/_build/

# PyBuilder
.pybuilder/
target/

# Jupyter Notebook
.ipynb_checkpoints

# IPython
profile_default/
ipython_config.py

# pyenv
#   For a library or package, you might want to ignore these files since the code is
#   intended to run in multiple environments; otherwise, check them in:
# .python-version
```

```
# pipenv
#   According to pypa/pipenv#598, it is recommended to include Pipfile.lock in version control.
#   However, in case of collaboration, if having platform-specific dependencies or dependencies
#   having no cross-platform support, pipenv may install dependencies that don't work, or not
#   install all needed dependencies.
#Pipfile.lock

# poetry
#   Similar to Pipfile.lock, it is generally recommended to include poetry.lock in version control.
#   This is especially recommended for binary packages to ensure reproducibility, and is more
#   commonly ignored for libraries.
#   https://python-poetry.org/docs/basic-usage/#commit-your-poetrylock-file-to-version-control
#poetry.lock

# pdm
#   Similar to Pipfile.lock, it is generally recommended to include pdm.lock in version control.
#pdm.lock
#   pdm stores project-wide configurations in .pdm.toml, but it is recommended to not include it
#   in version control.
#   https://pdm.fming.dev/#use-with-ide
.pdm.toml

# PEP 582; used by e.g. github.com/David-OConnor/pyflow and github.com/pdm-project/pdm
__pypackages__/

# Celery stuff
celerybeat-schedule
celerybeat.pid

# SageMath parsed files
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# mkdocs documentation
```

```
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
#  JetBrains specific template is maintained in a separate JetBrains.gitignore that
can
#  be found at https://github.com/github/gitignore/blob/main/Global/
JetBrains.gitignore
#  and can be added to the global gitignore or merged into this file.  For a more
nuclear
#  option (not recommended) you can uncomment the following to ignore the entire
idea folder.
#.idea/

### macOS ###
# General
.DS_Store
.AppleDouble
.LSOverride

# Icon must end with two \r
Icon


# Thumbnails
._*

# Files that might appear in the root of a volume
.DocumentRevisions-V100
.fseventsd
.Spotlight-V100
.TemporaryItems
.Trashes
.VolumeIcon.icns
.com.apple.timemachine.donotpresent

# Directories potentially created on remote AFP share
.AppleDB
.AppleDesktop
Network Trash Folder
Temporary Items
.apdisk

### macOS Patch ###
# iCloud generated files
```

```
*.icloud

# End of https://www.toptal.com/developers/gitignore/api/macos,django
```

Dosya kaydedilmiş, git üzerinde oluşturulan tüm dosyalar versiyon kontrole aşağıdaki komut ile eklenmiştir.

```
% git add -A
```

Git status ile eklenen dosyaların doğrulu kontrol edilmiştir, çalıştırılan komut ve cevabı aşağıdadır.

```
# Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   manage.py
        new file:   project/__init__.py
        new file:   project/asgi.py
        new file:   project/settings.py
        new file:   project/urls.py
        new file:   project/wsgi.py
```

Ardından aşağıdaki komut ile versiyon takip sistemine işlenmiştir.

```
% git commit -m "first commit”
```

Oluşturulmuş boş Django projesinde veritabanını oluşturmak için aşağıdaki komut çalıştırılmıştır ve çıktısı şöyle olmuştur

```
% python manage.py migrate

Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

Klasörde db.sqlite3 oluştuğu görüşmüştür ve veritabanı oluştuktan sonra projede kullanılacak python paketlerini eklendi.

```
% touch requirements.txt
```

Ardından requirements.txt dosyasına şu satırlar eklendi.

```
Django>=4.0.0
djangorestframework>=3.11.0

# for model
torch
transformers
```

Bu dosyayı kullanarak gerekli paketler ortama aşağıdaki komutla yüklendi.

```
% pip install -r requirements.txt
```

Proje klasöründe project/settings.py'e girildi ve INSTALLED_APPS liste değişkenine şu satırlar eklendi.

```
'rest_framework',
‘api.apps.ApiConfig'
```

Örnek görünüş şu şekilde olmalı, projenin değişmiş kısmı aşağıdaki şekilde göründüğü teyit edildi.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'api.apps.ApiConfig'
]
```

Aynı dosyada sayfanın en altına şu satırlar eklendi, bu satırlar rest_framework paketi için gerekli izinlerle alakalıdır.

```
REST_FRAMEWORK = {
    # Use Django's standard `django.contrib.auth` permissions,
    # or allow read-only access for unauthenticated users.
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.DjangoModelPermissionsOrAnonReadOnly'
    ]
}
```

Ardından yeni bir Django uygulama klasörü aşağıdaki komut ile çalıştırıldı.

```
% python manage.py startapp api
```

urls.py dosyası oluşturuldu ve içeriği aşağıdaki gibi yapıldı.

```
% touch api/urls.py

from rest_framework import routers
from django.urls import path, include
from . import views

router = routers.DefaultRouter()

urlpatterns = [
    path('', include(router.urls)),
    path('answerPromt', views.AnswerPromt.as_view(), name='answerPromt'),
]
```

Views.py dosyası oluşturuldu ve içeriği aşağıdaki gibi yapıldı.

```
% touch api/views.py
from rest_framework.response import Response
from rest_framework.views import APIView
from rest_framework import permissions
from project.settings import BASE_DIR

# PyTorch model related imports
import torch
from transformers import GPT2LMHeadModel, GPT2Tokenizer


# Rest Framework Permissions
class IsLoggedInUserOrAdmin(permissions.BasePermission):
    """
    Permission class that allow user or admin to access
    """
    def has_permission(self, request, view):
        return (request.user.is_superuser or request.user.is_staff
                or request.user)

    def has_object_permission(self, request, view, obj):
        return (request.user or request.user.is_superuser
                or request.user.is_staff)


class AnswerPromt(APIView):
    permission_classes = [
        IsLoggedInUserOrAdmin,
    ]

    def post(self, request, *args, **kwargs):
        print(request.data)
```

```
        # parse json
        dataJson = request.data
        print(dataJson);

        try:
            prompt = dataJson['prompt']
        except:
            prompt = None
            return Response(data={"status": "error", "message": "prompt not found"})

        if prompt:
            # Model and tokenizer path
            model_path = f"{BASE_DIR}/model/math_gpt2.pt"
            tokenizer_path = f"{BASE_DIR}/model/math_gpt2_tokenizer"

            print(model_path)

            # Load tokenizer
            tokenizer = GPT2Tokenizer.from_pretrained(tokenizer_path)

            # Create model
            model = GPT2LMHeadModel.from_pretrained("gpt2")  # Model's
architecture should be GPT-2
            model.load_state_dict(torch.load(model_path))

            inputs = tokenizer.encode(prompt, return_tensors='pt')
            outputs = model.generate(inputs, max_length=64,
pad_token_id=tokenizer.eos_token_id)
            generated = tokenizer.decode(outputs[0], skip_special_tokens=True)
            return Response(data={"status": "success", "generated":
generated[:generated.rfind(".")+1]})
```

Sunucunun başlatılacağı yerel bilgisayarın IP'si öğrenildi.

```
% ipconfig getifaddr en0
192.168.0.124
```

IP Adresi, project.settings'de ALLOWED_HOSTS değişkenine eklendi.

```
ALLOWED_HOSTS = ['192.168.0.124']
```

Önceden oluşturulan math_gpt2.pt dosyası ve math_gpt2_tokenizer klasörü model klasörüne kopyalandı. Sunucu başlatıldı.

```
% python manage.py runserver 0:8000
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
February 01, 2024 - 06:49:48
Django version 4.2.9, using settings 'project.settings'
Starting development server at http://0.0.0.0:8000/
```

Quit the server with CONTROL-C.

Sunucu sorunsuz başlatıldı, API uç-noktası test edildi.

```
% curl --location --request POST 'http://192.168.0.124:8000/api/answerPromt' \
--header 'Content-Type: application/json' \
--header 'Accept: */*' \
--header 'Host: 192.168.0.124:8000' \
--header 'Connection: keep-alive' \
--data-raw '{
    "prompt": "Hey you, what is math"
}'
{"status":"success","generated":"Hey you, what is math?\n\nThe math of math is a complex subject that involves many different concepts and concepts. It involves the study of mathematical concepts, equations, and equations, and the use of mathematical tools and techniques to solve problems.”}
```

BÖLÜM 3

Flutter ile ön-yüz oluşturma

Bu bölümde projenin yapay zeka modelinin .pt dosya uzantısı şeklinde kaydedilmiş yerel dosyalarla yüklendi, REST API uç-noktasında girilen promtun cevabını döndüren servis kullanılarak bir arayüz yapıldı, kullanıcıdan bir metin girişi alındı ve servisten dönen cevap gösterildi. Projede *Flutter* kullanılacak, *http* paketiyle istek atılacak, proje macOS üzerinde masaüstü uygulama olarak test edilecek, Flutter'ın cross-platform[16] olması sebebiyle, projenin kodlarında platform özelinde izinler ayarlanarak android, iOS, Windows, Linux/GNU üzerinde de çalıştırılabilir.

3.1 Ön-yüz[17] çalışma senaryosu

Proje'de arayüz oluşturmak için *Flutter*[18] ve üzerine eklenecek paket olarak http[19] kullanılacaktır. Daha önceki bölümde *DRF* yardımıyla bir *POST uç noktası* oluşturuldu, buraya *JSON* tipinde bir istek atıp *prompt* isimli değerimizi sormak istediğimiz soru olarak gönderildi, servisten cevap olarak, çıktıyı da başarıyla cevap alabilecek HTTP 200 status koduyla *JSON* tipinde aldık. Bu JSON cevabını da arayüzümüzde gösterdik.

---

[16] Cross-platform terimi, bir yazılım veya uygulamanın farklı işletim sistemlerinde (örneğin, Windows, macOS, Linux) veya farklı cihazlarda (örneğin, bilgisayarlar, telefonlar, tabletler) çalışabilir olma yeteneğini ifade eder. Bu, yazılım geliştiricilerin aynı kod tabanını kullanarak farklı platformlara uygulama geliştirmelerini sağlar.

[17] Front-end veya ön-yüz, bir yazılım uygulamasının kullanıcı ile etkileşimde bulunduğu kısmı ifade eder. Kullanıcının gördüğü ve etkileşimde bulunduğu arayüz, düğmeler, formlar, grafikler, metinler ve diğer görsel öğeler genellikle front-end kısmında yer alır. Bu, bir web sitesinde tarayıcıda görünen bir sayfa olabilir veya bir masaüstü uygulamasının kullanıcı arayüzü olabilir.

[18] Flutter, Google tarafından geliştirilen bir açık kaynaklı bir kullanıcı arayüzü (UI) yazılım geliştirme kitidir. Flutter, özellikle mobil uygulama geliştirmek için tasarlanmış olup, Android ve iOS platformlarında kullanılabilecek şekilde cross-platform özelliklere sahiptir. Flutter'ın avantajlarından biri, aynı kod tabanını kullanarak hem Android hem de iOS platformlarına yönelik uygulamalar geliştirebilmenizi sağlamasıdır. Ayrıca, web ve masaüstü uygulamalar için de kullanılabilecek potansiyele sahiptir. (The Flutter Authors, 2014)

[19] http, bir dart, flutter paketidir, pub.dev üzerinde bulunabilir (the Dart project authors, 2014)

3.2 Ön-yüz görsel planı

Tablo 1.1: Arayüz şablonu

Metin giriş alanı	
	Buton
Cevap	

3.3 Flutter Nedir?

Flutter, Google ve bireysel geliştiriciler tarafından geliştirilen bir açık kaynaklı kullanıcı arayüzü[20] (UI) yazılım geliştirme kitidir. Dart[21] adlı bir programlama dilini kullanır ve özellikle mobil uygulamaların hızlı ve etkili bir şekilde geliştirilmesini amaçlar. Flutter, aynı kod tabanını kullanarak hem Android hem de iOS platformlarına yönelik uygulamalar geliştirmenizi sağlar. Ayrıca, web ve masaüstü uygulamaları için de kullanılabilecek potansiyele sahiptir.

3.3.1 Flutter'ın avantajları

- *Cross-Platform Geliştirme:* Flutter, tek bir kod tabanı ile hem Android hem de iOS uygulamaları geliştirmenizi sağlar, bu da zaman ve kaynak tasarrufu sağlar.
- *Hızlı Geliştirme:* Hot Reload özelliği sayesinde, kod değişikliklerini anında görebilir ve test edebilirsiniz, bu da geliştirme sürecini hızlandırır.
- *Güçlü Widget sistemi:* Flutter, zengin ve özelleştirilebilir bir widget sistemi içerir. Bu, kullanıcı arayüzünü hızlı bir şekilde oluşturmanıza ve özelleştirmenize olanak tanır.

[20] Kullanıcı arayüzü, bir kullanıcının bir bilgisayar programı, web sitesi, mobil uygulama veya diğer bir tür yazılım ile etkileşimde bulunduğu grafiksel ve görsel bir bileşendir.

[21] Dart, Google tarafından geliştirilen açık kaynaklı bir programlama dilidir. Dart'ın odak noktası, özellikle web ve mobil uygulama geliştirmek için kullanılmasıdır. Dart, genellikle Flutter framework'ü ile birlikte kullanılarak mobil uygulamaların geliştirilmesinde tercih edilen bir dil haline gelmiştir. (the Dart project authors, 2014)

- *Performans:* Flutter, native22 derlenmiş uygulamalar gibi yüksek performans sunar, çünkü uygulama, cihazın donanımına doğrudan erişim sağlamak için derlenmiş bir kod kullanır.

- *Güçlü Topluluk Desteği*: Flutter, büyük ve aktif bir topluluğa sahiptir. Bu topluluk, dokümantasyon, eğitim kaynakları ve sorun giderme konularında yardımcı olabilir.

3.3.2 Flutter'ın dezavantajları

- *Boyut:* Flutter uygulamalarının APK veya IPA dosya boyutu, bazı durumlarda diğer çerçevelere kıyasla daha büyük olabilir.

- *Optimizasyon Sorunları:* Flutter, her platformdaki performansı maksimize etmek adına bazı ek optimizasyonlara ihtiyaç duyabilir.

- *Dil Tercihi:* Dart programlama dili, geliştiriciler arasında popülerliği daha düşük olan dillerden biridir. Bu nedenle, Flutter öğrenmek isteyen geliştiriciler, yeni bir dil öğrenmek zorunda kalabilirler.

3.3.3 Flutter'ın yapısı

Flutter, Widget adı verilen özelleştirilebilir yapı taşları üzerine kuruludur. Her şey bir Widget'tır ve bu Widget'lar hiyerarşik bir şekilde birleştirilerek uygulama arayüzü oluşturulur. Flutter uygulamaları, birinci sınıf olarak derlenmiş ve native performansı sağlayan bir runtime üzerinde çalışır. Flutter, Dart adlı bir programlama dilini kullanır. Dart, kullanımı kolay, modern ve nesne yönelimli bir dil olarak tasarlanmıştır. Flutter ile birlikte Dart kullanmak, uygulamaların performansını ve geliştirme sürecini artırmayı amaçlar. (The Flutter Authors, 2014)

---

22 Native, bir platformun orijinal veya yerel yapısına uygun olarak geliştirilmiş anlamına gelir. Örneğin, Android işletim sistemine özgü bir uygulama "native" olarak adlandırılır çünkü Android'e özgü API'ları ve özellikleri kullanır.

3.4 Geliştirme ortamı

Ön-yüz'de, *Flutter 3.16.9* kullanılmıştır, tasarımsal olarak material design 3 tasarım dili kullanıldı. Android SDK version 34.0.0, Xcode 15.2, Android Studio 2023.1, Yerel olarak kullanılan bilgisayar, *macOS 14.2.1 (23C71)* işletim sistemli bir bilgisayardır. 16 GB RAM ve *Apple Silicon M2* işlemcisi bulunmaktadır. Metin editörü olarak *VS Code* kullanılmaktadır. Projeyi başlatmak ve çeşitli işlemler için doğrudan *Zsh* (Z Shell), terminali kullanılmıştır. Projeyi çalıştırmak için emülator ya da simulator kullanılmamış doğrudan yazıldığı bilgisayarda macOS uygulaması olarak çalıştırıldı ve test edildi.

3.5 Ön-yüz kodları

Aşağıdaki komutu çalıştırarak Flutter sistemde doğru kurulmuş ve çalışıyor mu kontrol edildi.

```
% flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.16.9, on macOS 14.2.1 23C71 darwin-arm64, locale
    tr-TR)
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
[✓] Xcode - develop for iOS and macOS (Xcode 15.2)
[✓] Chrome - develop for the web
[✓] Android Studio (version 2023.1)
[✓] VS Code (version 1.85.2)
[✓] Connected device (2 available)
[✓] Network resources

• No issues found!
```

Projeyi oluşturmak istenen klasöre gidip aşağıdaki komut çalıştırıldı ve çıktı olarak aşağıdaki sonuç alındı.

```
% flutter create flutter_llm
Signing iOS app for device deployment using developer identity: "Apple
Development: Burak Karahan"
Creating project flutter_llm...
Resolving dependencies in flutter_llm... (1.1s)
Got dependencies in flutter_llm.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
```

```
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider:
https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd flutter_llm
  $ flutter run

Your application code is in flutter_llm/lib/main.dart.
```

pubspec.yaml dosyasındaki dependecies kısmına http: ^1.2.0 paketi eklendi.

```
dependencies:
  cupertino_icons: ^1.0.2
  flutter:
    sdk: flutter
  http: ^1.2.0
```

Aşağıdaki komut çalıştırılarak paketler çekildi.

```
% flutter pub get
Resolving dependencies...
  flutter_lints 2.0.3 (3.0.1 available)
  lints 2.1.1 (3.0.0 available)
  matcher 0.12.16 (0.12.16+1 available)
  material_color_utilities 0.5.0 (0.8.0 available)
  meta 1.10.0 (1.11.0 available)
  path 1.8.3 (1.9.0 available)
  test_api 0.6.1 (0.7.0 available)
  web 0.3.0 (0.4.2 available)
Got dependencies!
8 packages have newer versions incompatible with dependency constraints.
Try `flutter pub outdated` for more information.
```

lib/main.dart dosyası aşağıdaki şekilde değiştirildi.

```
import 'dart:convert';

import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // TRY THIS: Try running your application with "flutter run". You'll see
        // the application has a purple toolbar. Then, without quitting the app,
        // try changing the seedColor in the colorScheme below to Colors.green
        // and then invoke "hot reload" (save your changes or press the "hot
        // reload" button in a Flutter-supported IDE, or press "r" if you used
        // the command line to start the app).
        //
        // Notice that the counter didn't reset back to zero; the application
        // state is not lost during the reload. To reset the state, use hot
        // restart instead.
        //
        // This works for code too, not just values: Most code changes can be
        // tested with just a hot reload.
        colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),
        useMaterial3: true,
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  bool loading = false;
  String? result;
  final TextEditingController _controller = TextEditingController();

  // Http Request to the API
  // http://192.168.0.124:8000/api/answerPromt
  // body: {
  //  "prompt": "text editing field"
  // }
  Future<void> makeRequest2API() async {
    setState(() {
      loading = true;
      result = "Loading...";
    });
    try {
      final response = await http
          .post(Uri.parse("http://192.168.0.124:8000/api/answerPromt"),
              headers: <String, String>{
                "Content-Type": "application/json; charset=UTF-8",
              },
              body: json.encode(<String, String>{
                "prompt": _controller.text.trim(),
              }))
          .timeout(
```

```
        const Duration(seconds: 5),
        onTimeout: () {
          setState(() {
            loading = false;
          });
          return http.Response('Error: timeout', 408);
        },
      );

      if (response.statusCode == 200) {
        print("Response: ${response.body}");
        // Answer is {"status":"success","generated":"fawefawe)\n\n|system|>\nYou
are a intelligent chatbot and expertise in Mathematics.</s>\n<|user|>\nWhat is the
purpose of the kondo-mills theory?."}
        final Map<String, dynamic> answer = json.decode(response.body);
        print("Answer: ${answer['generated']}");
        setState(() {
          loading = false;
          result = answer['generated'];
        });
      } else {
        setState(() {
          loading = false;
        });
        result = "Failed to load answer";
      }
    } on Exception catch (e) {
      print(e);
      setState(() {
        loading = false;
        result = "Failed to load answer";
      });
    }
  }

  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // TRY THIS: Try changing the color here to a specific color (to
        // Colors.amber, perhaps?) and trigger a hot reload to see the AppBar
        // change color while the other colors stay the same.
        backgroundColor: Theme.of(context).colorScheme.inversePrimary,
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: const Text("Math GPT"),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
```

```
// arranges them vertically. By default, it sizes itself to fit its
// children horizontally, and tries to be as tall as its parent.
//
// Column has various properties to control how it sizes itself and
// how it positions its children. Here we use mainAxisAlignment to
// center the children vertically; the main axis here is the vertical
// axis because Columns are vertical (the cross axis would be
// horizontal).
//
// TRY THIS: Invoke "debug painting" (choose the "Toggle Debug Paint"
// action in the IDE, or press "p" in the console), to see the
// wireframe for each widget.
mainAxisAlignment: MainAxisAlignment.start,
children: <Widget>[
  const SizedBox(
    height: 16,
  ),
  // Text Edit Field
  Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    child: TextField(
      controller: _controller,
      decoration: const InputDecoration(
        border: OutlineInputBorder(),
        labelText: 'Enter your prompt here',
      ),
    ),
  ),
  const SizedBox(
    height: 16,
  ),
  // Button
  Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    child: Row(
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        const Flexible(
          flex: 4,
          child: SizedBox(),
        ),
        Expanded(
          flex: 2,
          child: ElevatedButton(
            onPressed: loading ? null : makeRequest2API,
            child: const Text('Submit'),
          ),
        ),
      ],
    ),
  ),
  const SizedBox(
    height: 16,
  ),
  // Result
  Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16.0),
    child: Column(
```

```
                mainAxisAlignment: MainAxisAlignment.start,
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text(
                    'Result: ',
                    style: Theme.of(context).textTheme.labelLarge,
                  ),
                  Text(
                    result ?? "",
                    style: Theme.of(context).textTheme.bodyLarge,
                  ),
                ],
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

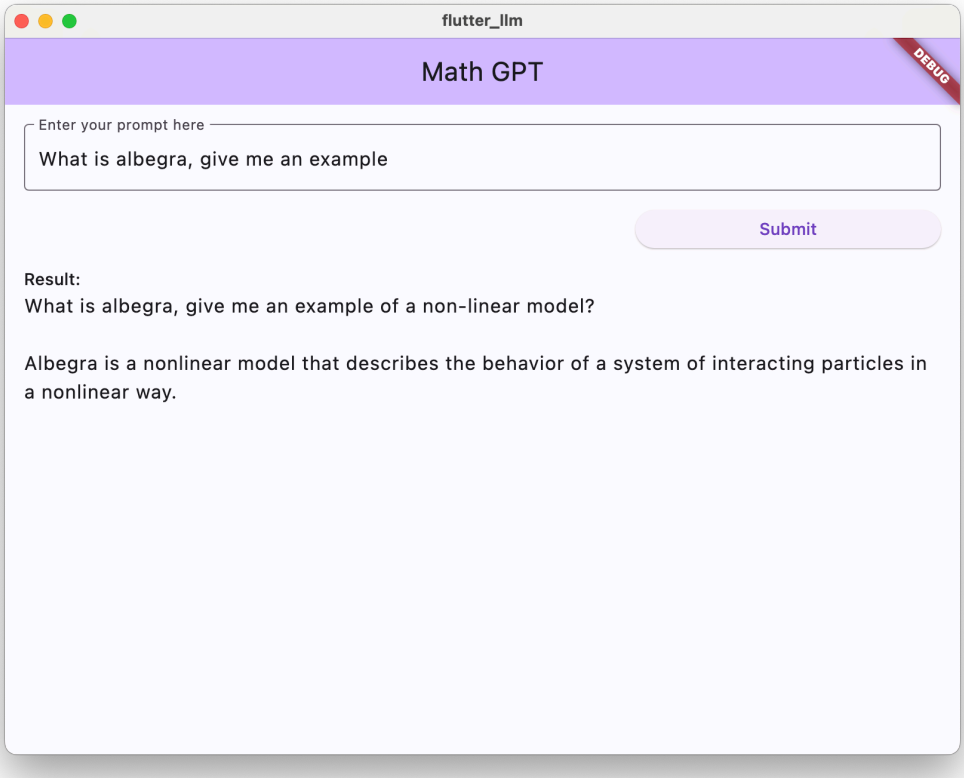
Ardından proje VS Code'da F5'e basarak ya da aşağıdaki komutla macOS'da mevcut bilgisayar üzerinde çalıştırıldı. Bu kısımda dilenilen her platform kullanılabilir.

```
% flutter run
```

Projenin macOS üzerinde internet erişimi iznine ihtiyacı vardır. Bu yüzden *macOS/Runner/Release.entitlements* dosyası aşağıdaki şekilde değiştirildi.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>com.apple.security.app-sandbox</key>
    <true/>
    <key>com.apple.security.network.client</key>
    <true/>
    <key>com.apple.security.network.server</key>
    <true/>
</dict>
</plist>
```

Proje tekrar çalıştırıldığında bir test metni yazıp cevap beklendi. Sonuç aşağıdaki şekilde oluşmuştur.



Şekil 1.1: Flutter Uygulaması

Kaynaklar

Amazon Web Services (2024). *Amazon Q (Preview)*. https://aws.amazon.com/tr/q/

Amazon Web Services (2024). *Amazon SageMaker*. https://aws.amazon.com/tr/sagemaker/faqs/?nc=sn&loc=4

Dart project authors (2014) - Dart Programming Language. https://dart.dev/overview

Dart project authors (2014) - http pub.dev package. https://pub.dev/packages/http

Django Software Foundation (2005) - Django. https://www.djangoproject.com/start/overview/

Django Software Foundation (2005) - Django. https://www.djangoproject.com/start/overview/

Encode OSS Ltd (2024) - Django Rest Framework. https://www.django-rest-framework.org/

Flutter Authors (2014) - Flutter. https://docs.flutter.dev/

Ian Goodfellow, Yoshua Bengio & Aaron Courville (2016). *Deep learning*. Mit Press.

Jesse Vig (2019) - GPT-2: Understanding Language Generation through Visualization. https://towardsdatascience.com/openai-gpt-2-understanding-language-generation-through-visualization-8252f683b2f8

Matthew Kenney (2022) - ArtifactAI/arxiv-math-instruct-50k Dataset. https://huggingface.co/datasets/ArtifactAI/arxiv-math-instruct-50k

KirstenS (2024) - OWASP Cross Site Request Forgery (CSRF). https://owasp.org/www-community/attacks/csrf

KirstenS (2024) - OWASP Cross Site Scripting (XSS). https://owasp.org/www-community/attacks/xss/

Sharath S Hebbar (2024) - Sharathhebbar24/math_gpt2. https://huggingface.co/Sharathhebbar24/math_gpt2

Ekler

Proje kaynak kodları: https://github.com/MarlonJD/masterScienceProject