Yıldız Teknik Üniversitesi

Elektrik Elektronik Fakültesi
Bilgisayar Mühendisliği Bölümü

Yapay Zeka Dersi

Dönem Projesi Raporu

20011024 – Sait Yalçın

20011901 – Muhammed Kayra Bulut

sait.yalcin@std.yildiz.edu.tr

kayra.bulut@std.yildiz.edu.tr

Dersin Yürütücüsü: Prof.Dr. Mehmet Fatih AMASYALI

1. CHATBOT NEDİR?

Chatbotlar, insan dilini analiz ederek ve uygun şekilde yanıtlayarak yazılı iletişimde bulunan bir tür yazılım uygulamasıdır. Genellikle müşteri desteği sağlamak, rezervasyonları yönetmek ve bilgi yaymak gibi çeşitli görevler için programlanırlar. İnsan dilini anlama ve yanıtlama yeteneği sayesinde, chatbotlar genellikle bir web sitesinde veya bir mesajlaşma platformunda kullanılarak insanların belirli sorgularını yanıtlar veya gereken işlemleri yürütürler.

1.1 Chatbotların Yapay Zeka İle İlgisi Nedir?

Chatbotlar ve yapay zeka (AI) sıkı sıkıya bağlıdır çünkü çoğu modern chatbot, doğal dil işleme (NLP), makine öğrenmesi ve derin öğrenme gibi AI teknolojilerini kullanır. Bu teknolojiler, chatbotların insan dilini anlamasına, kullanıcının girdisine göre yanıtlar oluşturmasına ve daha doğru ve özelleştirilmiş yanıtlar vermek için geçmiş etkileşimlerden öğrenmesine yardımcı olur. AI, bir chatbot'un kullanıcının niyetini anlamasına ve daha karmaşık sorguları yanıtlamasına olanak tanır. Öyleyse, yapay zeka, chatbotların daha insana benzer bir iletişim kurabilmesini ve kullanıcıların ihtiyaçlarına daha etkili bir şekilde hizmet edebilmesini sağlar.

1.2 Chatbotlar nasıl çalışır?

Chatbotlar, çeşitli yöntemlerle programlanırlar ve çalışırlar. Bir chatbot, bir insanla yazılı olarak sohbet edebilmek için özel bir dil öğrenir ve anlar. Bu dil, genellikle insan diline benzer bir dil olur, ancak bir chatbot tarafından anlaşılması için daha yapay bir dil olarak tasarlandığı da olabilir. Chatbotlar, insan dilini anladıklarında, bu dil üzerinden yapılan sorulara yanıt verirler. Chatbotların yanıtları, genellikle önceden programlanmış cevaplar veya veritabanından çekilen bilgilerden oluşur.

2. CHATBOT ÇEŞİTLERİ

Chatbotlar genellikle işlevlerine ve kullanılan teknolojilere göre sınıflandırılır. İşte en yaygın chatbot çeşitleri:

2.1 Kurallara Dayalı Chatbotlar: Bu chatbotlar önceden belirlenmiş kurallara göre çalışır. Sadece belirlenen senaryoları ve komutları anlarlar ve sınırlı yanıtlar verirler. Karmaşık soruları yanıtlamakta zorlanırlar.

2.2 Yapay Zeka Chatbotları: Bu chatbotlar, doğal dil işleme (NLP), makine öğrenmesi ve derin öğrenme gibi yapay zeka teknolojilerini kullanır. Kullanıcıların ihtiyaçlarını daha iyi anlamak ve onlara daha özel yanıtlar verebilmek için kullanıcıların önceki konuşmalarıyla kendini besleyebilir.

2.3 Hizmet Chatbotları: Bu chatbotlar, belirli bir hizmeti gerçekleştirmek için tasarlanmıştır. Örneğin, taksi çağırma, berber rezervasyonu yapma ya da yemek sipariş etme gibi.

2.4 Bilgi Chatbotları: Bu chatbotlar, belirli bir konuda bilgi vermek üzere tasarlanmıştır. Mesela, kullanıcıların kanser hakkındaki sorularını yanıtlamak gibi.

2.5 Sosyal Medya Chatbotları: Bu chatbotlar, sosyal medya platformlarındaki kullanıcı etkileşimlerini otomatikleştirmek için tasarlanmıştır. Örneğin, bir Instagram hesbındaki mesajları yanıtlama ya da Tinder hesabını yönetme gibi.

2.6 Sesli Chatbotlar: Bu chatbotlar, sesli girdi ve çıktıyı destekler. Örneğin, Apple'ın Siri'si ya da Google Asistan gibi.

3. CHATBOT PROJEMİZ

Projemizde transformer model kullanarak reddit üzerinden belli konu başlıklarının sayfalarından web scraping metodu ile çektiğimiz verilerle modelin eğitimin gerçekleştireceğiz. Projemizi gerçeklerken tensorflow, selenium, Numpy, beautifulsoup4, flatbuffers, keras gibi Python kütüphanelerini ve Python 3.10 versiyonunu kullandık.

Doysamız ve açıklamalarına kısaca değinirsek;

- **layers.py** – Transformer model için gereken sayısallaştırma metodlarını içerir.
- **utils.py** – Transformer model için gerekli olan tokenizer metodlarını ve diğer yardımcı metodları içerir.
- **export.py**– Modelin dosyadan checkpointleri çekip, ilklendirilmesini sağlayan init fonksiyonu ve girdi olarak verilen kelimelere çıktı üretilmesini sağlayan fonksiyonları içerir.
- **config.yml** – Çekilen verilerilerin sonucunda oluşan **train.from** ve **train.to** dosyalarını kullanarak, **training_data** dosyasını, tek sayıdaki satırlar soru, çift sayıdaki satırlar cevap olacak şekilde oluşturur.
- **main.py** – Çalıştırılma durumuna göre, **train-eval-test** modunda modelin çalıştırılmasını sağlar.

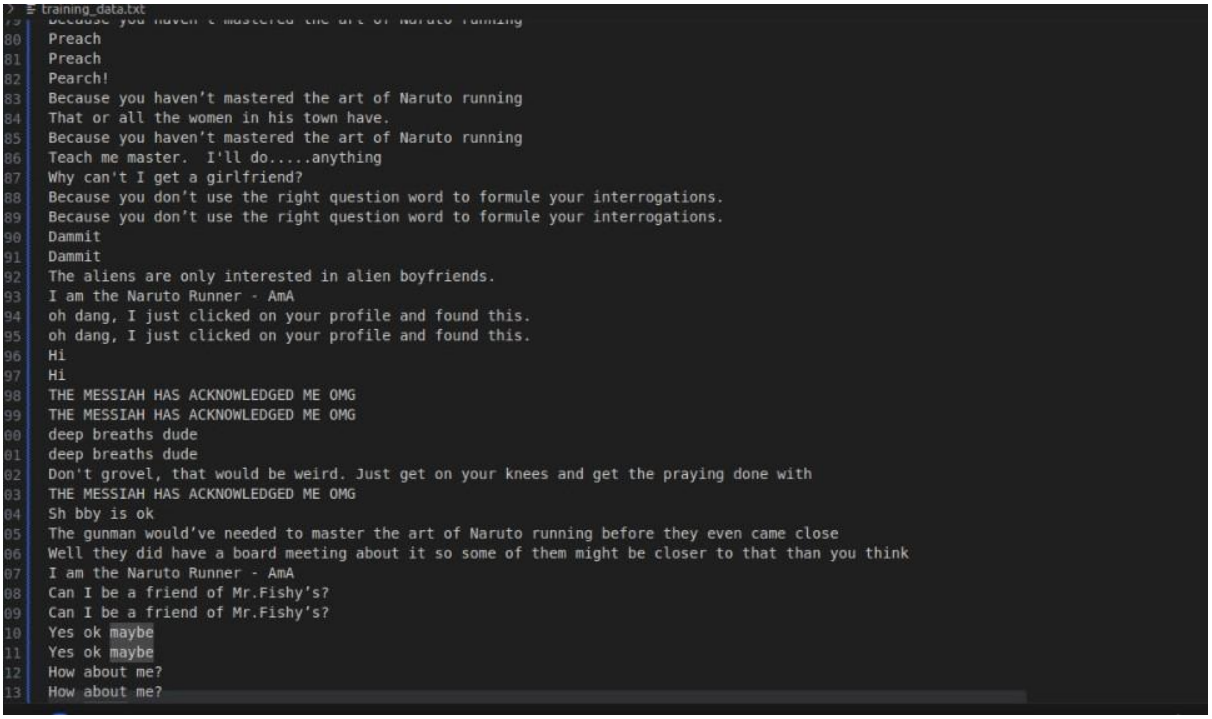
Projemizi kabaca özetlersek

1. Web scraping metodlarını hazırladık ve verileri toplayabileceğimiz konuları belirledik.
2. Web scraping metodlarımızı çalıştırarak verileri topladık.
3. Modelimizi kaydederek yeni sorgulara hazırladık.
4. Arayüz üzerinden yeni sorguları alarak eğitilen modelimizdeki oluşan tahmini cevapları ekrana gönderdik.

3.1 Veri Setini Oluşturma ve Yükleme

Öncelikle, içeriğini bizim oluşturduğumuz, temel soru ve cevapları oluşturmak için oluşan get_reddit_data.py isimli bir dosya oluşturduk. Sonrasında bu kod yardımıyla redditteki verileri soru-cevap olarak çektik. Bunları train.from ve train.to olarak kaydettikten sonra, dataloader.py dosyası yardımıyla da verileri tek ayıdaki satırlar soru, çift sayıdaki satırlar cevap olacak şekilde oluşturduk.

Burada bazı sorular kendini tekrar ediyor, bunun sebebine bakacak olursak, bir sorunun cevabına karşılık başka bir cevap olabilmesi. Verilerimizin bir kısmı da aşağıdaki fotoğrafta göründüğü gibidir.



3.2 Veri Setini Temizleme ve Ön İşlemden Geçirme

Metin verileriyle çalışırken makine öğrenimi veya derin öğrenme modeli yapmadan önce veriler üzerinde çeşitli ön işlemler gerçekleştirmemiz gerekir. Gereksinimlere bağlı olarak, verileri önceden işlemek için çeşitli işlemler uygulamamız gerekir.

Tokenization, metin verileri üzerinde yapabileceğiniz en temel ve ilk şeydir. Tokenization, tüm metni kelimeler gibi küçük parçalara ayırma işlemidir.

Burada kalıpları yineliyoruz, cümleyi tokenize ediyoruz ve kelime listesindeki her kelimeyi ekliyoruz. Ayrıca etiketlerimiz için bir sınıf listesi oluşturuyoruz.

Sonrasında verileri çeşitli ön işlemlerden geçiriyoruz. Bu da modelin başarısının artması için önemli bir etkiye sahip.

3.3 Modeli Oluşturma

Biz model olarak Transformer modeli kullandık. Transformer modeli, dikkat mekanizmalarını kullanarak karmaşık dil işleme görevlerini gerçekleştirebilen bir tür derin öğrenme modelidir.Transformer modelini eğitmek için **main.py** dosyası içinde, train fonksiyonunu kullandık. Kod, verilen girdi ve hedef değerleri üzerinde belirli bir sayıda epoch boyunca çalışacak şekilde tasarlanmıştır. Her bir epochta, kod öncelikle giriş ve hedef verilerinden maske oluşturur ve bu maskeleri Transformer modeline gönderir.

Modelden tahminler alındıktan sonra, kod hedef ve tahmin değerleri arasındaki loss'u hesaplar. Bu loss, modelin ne kadar iyi performans gösterdiğinin bir ölçütüdür; kayıp değeri düştükçe, modelin performansı artar.Daha sonra kod, bu kayıp değerini kullanarak modelin eğitilebilir parametrelerine göre gradyanları hesaplar ve bu gradyanları bir optimizör aracılığıyla uygular. Bu, modelin parametrelerini günceller ve modelin gelecekte daha düşük bir kayıp değeri vermesini sağlar.

Kod ayrıca kayıp ve doğruluk metriklerini hesaplar ve bu değerleri ekrana yazdırır. Bu, modelin eğitim süreci boyunca nasıl performans gösterdiğinin bir ölçüsüdür.Son olarak, belirli bir dönem sayısında, kod eğitim sürecinin bir checkpoint'ini kaydeder. Bu, eğitim sürecini durdurma ve daha sonra kaldığı yerden devam etme yeteneği sağlar.

3.4 Cevap Tahmini

Eğitilmiş modeli yükleyerek ve ardından bottan gelen yanıtı tahmin edecek bir grafik kullanıcı arayüzü kullanacağız. Bunun için yine **main.py** dosyası içindeki evaluate fonksiyonunu kullandık.

Bu fonksiyon kodu basit anlamda, eğitilen modeli kullanarak aldığı girdiye göre bir çıktı üretir. Kod öncelikle başlangıç ve bitiş belirteçlerini girdi cümlesine ekler ve girdiyi modelin beklediği şekle getirir. Ardından, çıktının başlaması gereken yer için bir başlangıç belirteci ekler.

Daha sonra kod, büyük bir for döngüsü içinde çalışır. Her döngüde, kod öncelikle girdi ve çıktı için maske oluşturur ve bu maskeleri modelimize gönderir. Modelden tahminler ve dikkat ağırlıkları alındıktan sonra, kod en son tahmini alır ve bu tahminin en olası sonucunu belirler. Eğer bu sonuç, çıktının bitişiyle eşleşirse, kod çıktı dizisini ve dikkat ağırlıklarını döndürür. Eğer sonuç bulunmazsa, kod tahmin edilen sonucu çıktı dizisine ekler ve döngü devam eder.

Modeli oluşturduğumuzda eğitim 100 **Epoch** sürdü. **Loss** 0.1425 ve **Accuracy** 0.3635'a eşit oldu.

4. PROGRAMI ÇALIŞTIRMA

Chatbotu çalıştırmak için öncelikle train sonucu oluşan checkpointlerin checkpoints klasör ismiyle ana dizinde bulunması gerekmektedir. *main.py* dosyası çalıştırılarak train işlemi gerçeklenebilir. Eğer hazır checkpoint kullanılmak istenirse,yaptığımız trainler sonucu oluşan checkpointlerin bulunduğu drive linki aşağıdadır;

https://drive.google.com/drive/folders/13XL-9g-tn-Qb4aoH1I_fdc2kHWVTe9hS?usp=sharing

Daha sonra app.py çalıştırılarak model için konfigürasyonların yüklenmesi gibi ilklendirme işlemleri yapılır ve server ayağa kaldırılır.

Bu işlemden sonra da *yzproje-ui* klasörü içinde konsol üzerinde *npm start* komutu çalıştırılarak arayüz de çalıştırılmış olur ve bu sayede uygulama kullanılmaya hazır hale gelir.

5. BAŞARIM DEĞERLENDİRİLMESİ

Dil modelimizi değerlendirmek için ilk olarak F1 score kullanmaya çalıştık. Ancak kullandığımız model transformer model olduğu için sürekli olarak farklı çıktılar sağladığı ve F1 score kullanabilmek için çıktıların ortak tokenları olması gerektiği için sayısal başarı elde etmeye çalıştığımızda *'unseen token'* hata metniyle bir hata aldık. Daha sonra yaptığımız araştırmalar sonucunda sayısal başarımı değerlendirebilmek için chatbot için BLEU score ile değerlendirmenin daha uygun olabileceğini ve bizim de BLEU score kullanabileceğimizi gördük ve sayısal başarıyı ölçebilmek için bunu kullandık. Ortalama BLEU score değerimiz aşağıda görüldüğü şekildedir;

Average BLEU *Score: 0.0019700372*

```
PS C:\Users\saity\Desktop\yzf1> & C:/Users/saity/AppData/Local/Microsoft/WindowsApps/python3.8.exe c:/Users/saity/Desktop
yzf1/main.py
C:\Users\saity\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.8_qbz5n2kfra8p0\LocalCache\local-packages\Python3
\site-packages\nltk\translate\bleu_score.py:552: UserWarning:
The hypothesis contains 0 counts of 2-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
Average BLEU Score: 0.0019700372
```

6. SONUÇ

Doğal dil işleme ve veri bilimi alanındaki projemizde, sohbet botlarının nasıl çalıştığını inceleyerek bu konudaki kavramları ve dinamikleri öğrenmeye çabaladık. Aynı zamanda, derin öğrenme tekniklerini Python programlama dili kullanarak bir sohbet botu üzerinde uyguladık. Proje boyunca, veri setimizi belirli iş gereksinimlerine uygun şekilde özelleştirebilmeyi başardık. Veri içeriğimizi, Yapay Zeka konseptine dayanarak hazırladık. Çok miktarda veriyi manuel olarak girmek oldukça zaman alıcı bir süreçti, Bundan dolayı verileri sosyal medyadan alarak işledik. Chatbot'ların yaygın bir şekilde kullanıldığı ve birçok işletmenin iş süreçlerinde chatbot uygulamalarını dahil etmeye başladığı bir dönemden geçiyoruz. Bundan dolayı proje konusu aslında popüler bir konuydu. Artık elimizde Yapay Zeka konusunda cevap alabileceğimiz bir sohbet botu var!!!

7. Kaynaklar

https://tr.wikipedia.org/wiki/Yapay_zek%C3%A2

https://www.cbot.ai/tr-blog/chatbot-konusuna-genel-bir-bakis/

https://www.smartmessage.com/tr/chatbot-nedir-nasil-calisir-avantajlari-nelerdir/

https://medium.com/@akanesen/chatbot-nedir-ve-%C3%A7e%C5%9Fitleri-nelerdir-1b1ca0321e65#:~:text=G%C3%BCn%C3%BCm%C3%BCzde%20kullan%C4%B1lan%20chatbotlar%20kendi%20i%C3%A7lerinde,ve%20DD%C4%B0%20teknolojisi%20kullanan%20chatbottur.

https://sites.google.com/view/mfatihamasyali/yapay-zeka