



T.C.

GEBZE TEKNİK ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

STAJ DEFTERİ

Öğrencinin;

Adı, Soyadı	Süleyman Gölbol
Numarası	1801042656
Bölümü	Bilgisayar Mühendisliği
Staj Yaptığı Yer	İnosens Bilişim Teknolojileri San. ve Tic. Ltd. Şti.
Staj Tarihleri	22/08/2022 – 19/09/2022

YAPILAN İŞİN;	
TARİHİ:	KAPSAMI:
22/08/2022	Eğitim
<p>İlk gün olarak staj süreci hakkında bilgiler alındı, ekipteki kişiler ile şirketin şu ana kadarki işleyişi öğrenildi.</p> <p>Günümüzde kullanılan teknolojiler hakkında fikirler konuşuldu.</p> <p>Özellikle yapay zeka (artificial intelligence) ve derin öğrenme (deep learning) alanlarında yapılan çalışmalarda büyük önemi olan tensorflow hakkında bilgiler edinildi.</p> <p>Tensorflow Google tarafından geliştirilen ve tamamen açık kaynaklı olan, makine öğrenmesi, derin öğrenme ve yapay zeka konusunda birçok araç ve kütühaneye sahip bir platformdur. Data-flow graph'ları kullanarak nümerik hesaplama yapmamızı sağlar.</p> <p>Diğer büyük avantajlarından bazıları;</p> <ol style="list-style-type: none">1) Veri görselleştirmesine sahip olması2) Javascript C#, Python, C++ gibi birçok dille uyumlu olması3) Donanımsal hızlandırıcı kütüphanelerindeki çalışma modellerine sahip olmasından dolayı GPU'da farklı dağıtım stratejileri kullanır ve bu memory allocation'ı düşürür. <p>Bunlar dışında TPU mimarisel kısıtlamasının modeli eğitmeye değil sadece çalıştırmaya izin vermesi ve birçok dependency'si olması (bağımlı olduğu destek dosyaları) TensorFlow'un dezavantajlarından bazılarıdır.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
23/08/2022	Eğitim
<p>Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda Natural Language Processing'i (NLP) öğrenmeye başladı.</p> <p>Natural Language Processing (Doğal Dil İşleme) Human-Computer Interaction'ı en iyi hale getirebilmek veya değişik dilleri konuşan kişiler arasında oluşan iletişimi duygu analizi, sestten yazıya çevirme, komut anlama, imlayı denetleme, kelime filtreleme gibi alanlarda çalışan bilim dalına denir.</p> <p>Bag of Words hakkında bilgiler öğrenildi. Bag of Words yazı modelleme için kullanılan bir tekniktir. Özellikle yazı verisinden feature extraction için kullanılır. Kelimelerin ne sıklıkla yazıda tekrarlandığını gramatik detay veya kelime sıralaması olmadan inceler. N-grams ise birden fazla kelimenin bir arada kullanılmasıyla anlam kazanan kelime grupları içindir.</p> <p>Named Entity Recognition(NER) konusu hakkında bilgiler öğrenildi. Türkçesi Varlık İsmi Tanıma olan NER'de amaç yazıdaki varlık isimlerini önceden belirlenmiş olan kişi, tarih, konum, değer, saat, kurum gibi kategorilere sınıflandırmaktır.</p> <p>Örneğin;</p> <p>Süleyman Gölbol registered to Gebze Technical University on 03/09/2018.</p> <p>KİŞİ KURUM TARİH</p> <p>TF-IDF (Term Frequency-Inverse Document Frequency) öğrenildi. TF-IDF (Türkçesi terim sıklığı-ters belge sıklığı) istatistiksel olarak bir dokümanda geçen kelimenin dokümanla ne kadar alakalı olduğunu, anahtar kelimenin geçme sıklığı bilgisini ve logaritma fonksiyonunu kullanarak öğrenmemizi sağlar.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

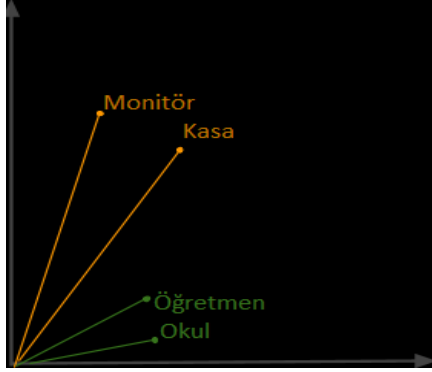
YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
24/08/2022	Eğitim

Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda Natural Language Processing'in önemli bir parçası olan Word Embedding'i (Kelime Gösterilimleri) öğrenmeye başladı.

Word Embedding, kelimeleri Deep Learning ortamında nümerik biçimde temsil etmeyi sağlar.

Word Embedding'in Bag of Words ve TF-IDF gibi metodlardan en büyük farkı sözdizimsel (sentaktik) ve anlamsal (semantik) yapıyı korumaya çalışmasıdır. Bunu yaparken oluşturulan matrisin sparse matrix (seyrek matris) olmamasını sağlar yoksa model training süresi uzar.

Bunu sağlamak için n boyutlu uzayda benzer veya alakalı anlamı olan kelimeler vektör uzayında yakında yerleştirilir.



Word2Vec 2 katmanlı neural network kullanarak Word embedding oluşturmamızı sağlayan metoddur. Aslında deep neural network olmamasına rağmen yazıyı deep neural network'lerin anlayabileceği hale getirir.

Word2Vec, CBOW (Continuous bag of words) ve Skip-gram modellerinin birlikte kullanılmasıyla oluşur. CBOW'da öncelikle kelimeler "One-Hot Encoding" yöntemiyle sayısal vektör haline getirilir.

KELİMELER: [BEN, ROBOT, DEĞİLİM]
BEN = 100 , ROBOT = 010, DEĞİLİM = 001

Sonrasında "ben" ve "değilim" kelimeleri input olarak alınıp aradaki kelime output olarak bulunmaya çalışılır.

Skip-gram'da ise ortadaki kelime input olarak alınıp onu çevreleyen kelimeler bulunmaya çalışılır.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
25/08/2022	Eğitim

Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda Amazon product reviewları ile ilgili Sentiment Analysis Problemine yönelik dataset incelemesi yaptı. Probleme alakalı olarak çözüm fikri önerilerinde bulundu.

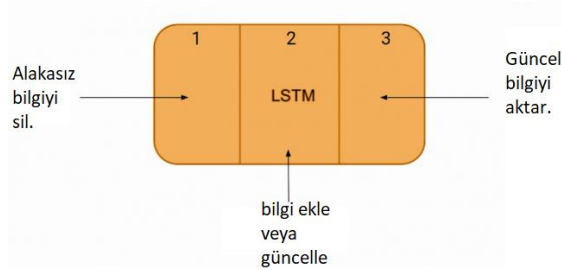
Sentiment Analysis (Duygu Analizi), Doğal Dil İşleme'de yazı gövdesindeki duyguyu tanımlaya yönelik bir işlemdir. Bu işlemle bir yazı parçasının pozitif, negatif, nötr olduğu anlaşılabilir. Duygu analizinde doğal dil işleme yöntemleriyle derin/makine öğrenimi yöntemleri bir arada kullanılır.

Duygu analizini yaparken farklı algoritmalarından yararlanılabilir.

RNN (Recurrent Neural Network, Özyinelemeli Sinir Ağı) kullanılıyorsa, cümledeki sözcüklerin kendisinden önce ve sonra gelen sözcükle bağlantılı olduğunu kabul edip onların bu ilişkisini kayıtlı tutar bu yüzden yazıları sınıflandırırken iyi sonuçlar elde edilir. Fakat Recurrent Neural Network uzun cümlelerde ve ifadelerdeki sonuçları başarılı değildir.

RNN'nin içinde bir ileri düzey bir algoritma yöntemi olan LSTM (Long Short Term Memory Network, Uzun Kısa Vadeli Hafıza Ağları), uzun vadeli dependency (bağımlılık) problemini azaltmak tasarlanmıştır.

Değerleri rastgele aralıklarla hatırlayan mimarisiyle, ilerleme kaydedilince saklanan değerler değiştirilmez.



Natural Language Processing'de Sentiment Analysis için kullanılabilecek diğer algoritmalar Çok Katmanlı Yapay Sinir Ağları (MLP), Bayes Algoritması, Merkez Tabanlı Sınıflayıcı, Destek Vektör Makineleri (SVM) gibi sınıflandırma algoritmalarıdır.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
26/08/2022	Eğitim, Kodlama

Bugün stajyer, mentoru tarafından verilen yönlendirmeye yapay zekanın önemli parçalarından biri olan ve Google Translate, Arama motorları gibi birçok alanda kullanılan NLP'deki Stemming (Gövdeleme) ve Lemmatization (Kök Bulma) hakkında öğrenmeye başladı.

Bu yöntemlerin ikisi de Word Embedding'den önce yapılabilen, kelimeyi eklerinden ayırıştıran preprocessing basamaklarıdır.

Stemming 'de amaç her bir kelimeden ekleri kaldırmaktır. Bunu yaparken ön tanımlı ekler kullanılır. Basitliğinden dolayı sıkça tercih edilir. Fakat sıklıkla eksik harfli köklere neden olabilmektedir.

Oysa lemmatization bükünlü (yapım ve çekimde sözcük kökleri değişikliğe uğrayan) kelimelerle de ilgilenir. Lemmatization'da amaç kelimeleri kök kelimeyi bulacak şekilde indirgemektir. Ön tanımlı ekler yerine kelimenin bulunduğu morfolojik konteksti göz önüne alır.

Stajyer bugün bunları öğrendikten sonra şirket projesinin veri setinde Python yardımıyla data cleansing yaptı.

ÖRNEK STEMMING KODU:

CODE	OUTPUT
<pre>from nltk.stem import PorterStemmer stemming = PorterStemmer() words = ["programmer", "programming", "studies", "corpora", "goes"] print("Printing the words with stemming:") for w in words: print(w, " : ", stemming.stem(w))</pre>	<pre>PS C:\Apparatus\Others\tr\Internship\codes\stem and lemma> python .\ Printing the words with stemming: programmer : programm programming : program studies : studi corpora : corpora goes : goe</pre>

ÖRNEK LEMMATIZATION KODU:

CODE	OUTPUT
<pre>import nltk from nltk.stem import WordNetLemmatizer nltk.download('omw-1.4') nltk.download('wordnet') lemmatizer = WordNetLemmatizer() words = ["programmer", "programming", "studies", "corpora", "goes"] print("Printing the words with lemmatization:") for w in words: print(w, " : ", lemmatizer.lemmatize(w))</pre>	<pre>Printing the words with lemmatization: programmer : programmer programming : programming studies : study corpora : corpus goes : go</pre>

NLTK kütüphanesinin içindeki PorterStemmer gibi algoritma classları stemmingi, WordNetLemmatizer gibi classlar ise Lemmatization'ı sağlar.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
29/08/2022	Eğitim ve Yazılım Kodlama

Bugün stajyer, mentoru tarafından verilen yönlendirmeye POS Tagging (Part of Speech, Sözcük Türü Etiketleme)'yi öğrendi ve bunu önceden öğrendiği tekniklere birleştirip verisetine kodla uyguladı. Özellikle kelimeleri köke indirirken farklı kelime türleri bilgisayarın çözümünü yanıtılabilmektedir.

Örneğin 'purchase' kelimesi 'satın almak' anlamına gelmektedir. 'Purchased' kelimesi ise farklı sözcük türleriyle farklı anlamlara gelebilmektedir. Kelimenin geçtiği cümleye göre geçmiş zaman içinde olan fiil kullanımı olabilmektedir veya 'satın alınmış' sıfat anlamı da olabilmektedir. Özellikle sıfat, isim, fiil ve zarf olan sözcükleri ayırmak doğal dil işlemede önemli bir yer kaplamaktadır.

```
# Create pos tagger function to use in Lemmatization
def part_of_speech(word):
    tag = nltk.pos_tag([word])[0][1].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.ADV)
```

Kelimeler sözcük türüne göre etiketlere ayrılır ve bu değer return edilir.

```
# Lemmatizing sentence by using part_of_speech() tagging function
text = [lem.lemmatize(word,
                      part_of_speech(word))
         for word in text if word not in stop_words]
text = " ".join(text)
```

Lemmatization kullanılırken stop wordler (Stop-word: bilgisayar için gereksiz olacak kelimeler) arasında olmayan kelimeler sırayla part of speech'le gruplara ayrılır ve lemmatize fonksiyonu çağrılır. Buna göre kelimeler köke indirgenir.

```
reviewText[0] = "Purchased some of the good products"
print(reviewText[0], "\n", handled_text(reviewText[0]))

Purchased some of the good products
purchase good product
```

Görüldüğü gibi kelimeler bu şekilde kök haline getiriliyor ve stopwordler listeden çıkarılıyorlar.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
31/08/2022	Yazılım Geliştirme
<p>Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda verisetine NLP teknikleri uygulayıp sonrasında Keras, Tensorflow kütüphaneleri kullanılacak uygulamayı geliştirmeye başladı.</p> <p>Önceki günde öğrenilen Part-of-Speech ve Lemmatizer koda eklendi. Sonrasında, elde edilen indirgenmiş sözcüklerin bulunduğu cümleler ayrıştırıldı ve her biri kendi türüne göre gruplara eklendi. (Örneğin 5,4,3,2,1 isimli 5 grup var ve her grup için farklı yazı içerikleri var.)</p> <p>Deep Learning'e hazırlık için tüm gruptaki eleman sayıları en küçük eleman sayılı gruba indirgendi.</p> <pre># Train and test docs train_docs = reviews1[:-testRating1] + reviews2[:-testRating2] + reviews3[:-testRating3] + reviews4[:-testRating4] + reviews5[:-testRating5] test_docs = reviews1[-testRating1:] + reviews2[-testRating2:] + reviews3[-testRating3:] + reviews4[-testRating4:] + reviews5[-testRating5:] print("Length of train_docs: ", len(train_docs), " Length of test_docs: ", len(test_docs))</pre> <pre>trainLabels = np.concatenate((np.zeros(totalReviewsForEachRating-testRating1), np.ones(totalReviewsForEachRating-testRating2), np.ones(totalReviewsForEachRating-testRating3) * 2, np.ones(totalReviewsForEachRating-testRating4) * 3, np.ones(totalReviewsForEachRating-testRating5) * 4), testLabels = np.concatenate((np.zeros(testRating1), np.ones(testRating2), np.ones(testRating3) * 2, np.ones(testRating4) * 3, np.ones(testRating5) * 4)) print("Length of trainLabels: ", len(trainLabels), " Length of testLabels: ", len(testLabels))</pre> <p>Veri sınıflarındaki küçük eleman sayısı 80 olduğu için her bir grupta test için %10 ayrıldı. 5 grup için $80 \times 5 = 400$ uzunluğuna geldi.</p> <pre>Total test reviews for each rating: (80/10) 8 Length of train_docs: 360 Length of test_docs: 40 Length of trainLabels: 360 Length of testLabels: 40</pre> <p>Sonrasında metinler, fit_on_texts() ve Tokenizer() fonksiyonlarının yardımıyla tokenlara ayrıldı. texts_to_sequences() fonksiyonu yardımıyla da metindeki tokenlar vektörel hale getirildi.</p> <p>En büyük cümle bulundu ve pad_sequences() fonksiyonu yardımıyla her bir cümlenin boyutu buna eşitlendi. (Kalanlar 0 ile dolduruldu. Böylece her vektör eşit boyutlu hale geldi.)</p> <pre>[[21 26 4 ... 0 0 0]</pre> <pre># Encode training data sentences into sequences train_sequences = tokenizer.texts_to_sequences(train_docs) # Pad the training sequences train_padded = pad_sequences(train_sequences, padding='post', truncating='post', maxlen=max_training_sequence_length)</pre> <p>Aynı işlem training data dışında test data içinde uygulandı. Böylece Tensorflow ve Keras'la eğitim için gerekli yapı oluştu.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
01/09/2022	Eğitim, Yazılım Geliştirme
<p>Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda NLP uygulamasında Keras ve Tensorflow kütüphanelerini kullanmaya başladı. Bunun yanı sıra gerekli optimizasyon için overfitting, dropout, batch normalization gibi gerekli yapıları öğrendi.</p> <p>Overfitting (aşırı-uyma), eğitim verisi (training data) üzerinden en alt kırılıma kadar çalışıp, sonuçları ezberlemesi ve sadece o veriler üzerinde başarı elde edebilmesidir. Eğitim verisi ile kurulan model, test verisi (test data) üzerinde denince sonuçlar training data'ya göre çok düşük çıkar. Bunu önlemek için training başlamadan ve veri seti sıralı parçalara ayrılmadan önce veri seti karıştırılmalıdır. Engellemenin diğer yolları, değişkenleri azaltma, regülerizasyon (düzenleştirme), daha fazla veri ekleme veya training-test data arası hata farkı belli seviyeye ulaşınca durdurma işlemi yapmaktır.</p> <p>Dropout(bırakma), training işlemi sırasında overfitting'i engellemek için ortaya konmuş bir yapıdır. Bunu sağlarken rastgele seçiler bazı neural unit'leri göz ardı eder.</p> <p>Batch Normalization (Yığın Normalleştirme) ise internal covariate shift'i (iç değişken kaymasını) engellemek amacıyla ortaya konmuş bir yapıdır. İç değişken kayması; training sırasında her katmanın hatasını düzeltmek için ayrı ayrı hareket etmeye başlayıp değişime yol açmasıdır. Giriş katmanlarındaki bu kaymalar sebebiyle hesaplama süresi artar.</p> <p>Normalizasyon işlemi, tüm inputların dağılım ortalamasını sıfır, standart sapmasını 1 olacak hale getirmektir. Böylece değerler -1 ile 1 arasına sıkıştırılmış olur. Batch normalization ise bundan diğer katmanların yararlanabilmesi için katmanlar arasına uygulanmasıyla oluşan, eşzamanlı öğrenimi sağlayan işlemdir.</p> <p>Aktivasyon fonksiyonu ise, yapay bir nörondaki, girdilere dayalı bir çıkış sağlayan ve modellerin karmaşıklık seviyesini artıran fonksiyondur. Bu sayede YSA modelleri veri setleri üzerindeki karmaşık yapıları daha iyi bir şekilde öğrenerek daha iyi sonuçlar verir. Daha karmaşık problemlere çözüm bulabilmek için doğrusal olmayan aktivasyon fonksiyonları daha başarılıdır.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:

TARİHİ:

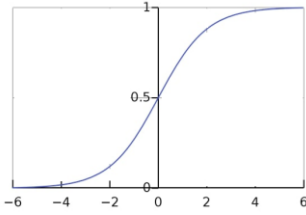
02/09/2022

KAPSAMI:

Eğitim, Yazılım Geliştirme

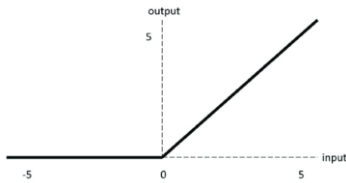
Bugün stajyer, mentoru tarafından verilen yönlendirme doğrultusunda NLP uygulamasında Keras ve Tensorflow kütüphanelerini kullanarak geliştirilen uygulamaya devam etti. Bunu yaparken yeni kavramlar öğrenildi.

Aktivasyon fonksiyon çeşitlerinden Softmax, Relu ve Sigmoid aktivasyon fonksiyonları arasındaki fark kavrandı. Sigmoid fonksiyonu 0 ile 1 arası değer üretir. Fonksiyonu, smooth bir yapıya sahip olduğu için küçük değişiklikler dahi fonksiyonda görülebilir.

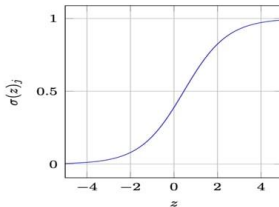


Fakat kaybolan Gradyan problemi isimli dezavantajı vardır. x değerleri ilerlerken uç noktalarda fark 0'a yakınsamaktadır. Bir yerden sonra y değerindeki fark görülmemeye başlar bu da yapay sinir ağları eğitirken çıkan bir sorundur.

Relu (Doğrusal doğrultulmuş birimler) fonksiyonu çıktı değerleri $[0, +\infty]$ arasına sahiptir. 0'dan büyük olan inputlarda sabit türev değerine sahip olduğu için ağın hızlı eğitilmesinde etkisi vardır.



Softmax aktivasyon fonksiyonu ise her inputun bir sınıfa ait olmasını gösteren çıktıları $[0, 1]$ değer arasına alıp gösterir. Bu da olasılık dağılımını sağlar. Sigmoid'den tek farkı elde edilen çıktıların 1'e kadar olan toplamalarının normalize edilmesidir.

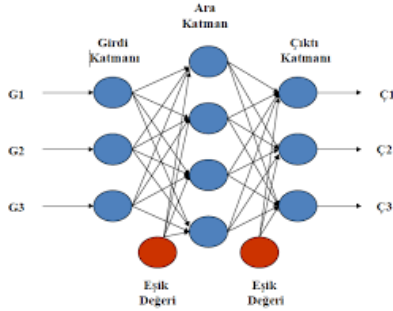
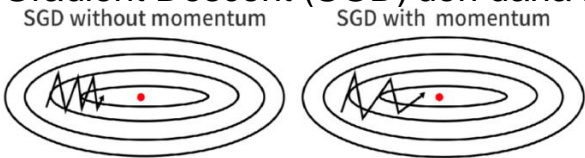


Stajımı bu firmada yaptım.

Staj Yapanın İmzası

Staj Yeri Yetkilisinin

Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
05/09/2022	Eğitim, Yazılım Geliştirme
<p>Bugün stajyer tarafından yapay zeka ile sınıflandırma yazılımı üzerinde araştırmalar yapıldı. Yeni kavramlar öğrenildi ve entegre edildi.</p> <p>Mini-batch kavramı parametre güncellemesinin gerçekleştiği ağa verilen alt örneklerin sayısıdır. Toplu boyut için iyi bir varsayılan değer 32 olabilir. Verileri aynı anda işlemek zaman ve bellek için sıkıntılı olacağı için backpropagation işlemiyle ağ üstünde geri dönük olarak gradient descent hesaplaması yapılmaktadır. Birden fazla girdinin parçalar halinde işlenmesine “mini-batch” denmektedir. Epoch sayısı ise aynı anda eğitime katılmayan verilerin parçalar halinde eğitimde yer almasıyla, başarıya göre de backpropagation ile ağırlıklar güncellenmesini belirler. İlk epoch'larda başarımlar düşük olacak, epoch sayısı arttıkça başarımlar da artacaktır.</p> <p>Multi-Layer Perceptron(MLP), bir giriş katmanı, hidden layer olarak adlandırılan bir ya da daha fazla LTU katmanı ve bir çıkış katmanından oluşur. Çıkış katmanı hariç diğer katmanlar yanlılık nöronu içerir ve diğer katmanlara tamamen bağlıdır.</p>  <p>Adaptif momentum anlamına gelen Adam optimizasyon algoritması öğrenildi ve kullanıldı. Burada parametrelerin her birinin öğrenme oranlarının yanı sıra momentum değişikliklerini de önbellekte saklar. Bu da RMSprop ile ve momentumu birleştirir. Adaptif algoritmalar hız açısından; konveks kayıp fonksiyonları altında lineer sınıflandırıcıların ayırt edici öğreniminde çok etkili bir yaklaşım olan Stochastic Gradient Descent (SGD)'den daha iyi performans gösterir.</p> 	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
06/09/2022	Yazılım Geliştirme
<p>Bugün yazılım geliştirme sırasında framework ops EagerTensor hatasıyla karşılaşıldı. Çözüm için Adam optimizer'ının decay değeri 1e-6'ya ayarlandı. Bunun dışında DataFrame ile alakalı değer DataFrame'deki parçanın kopyası olarak ayarlanmaya çalışıldığı için problem çıktı. Bu da chained_assignment değerini deaktif hale getirilerek düzeltildi.</p> <pre> x_train, x_test = train_test_split(dataset, test_size=0.2, random_state=111) from sklearn.utils import class_weight class_weights = list(class_weight.compute_class_weight(class_weight='balanced', classes=np.unique(dataset['overall']), y=dataset['overall'])) # class_weights = dict(zip(np.unique(dataset['overall']), class_weights)) print(dataset['overall'].value_counts()) </pre> <p>Verileri test ve train olmak üzere 2'ye ayırırken test boyutu %20'ye ayarlandı. Bunu yaparken verisetinin overall sütunundaki değerler unique hale getirildi ve sonrasında fonksiyon ile değerler değişkene aktarıldı. Sort() fonksiyonu ile sıralı hale çevirildi. Fakat verisetindeki overall sütunu sadece float değerler içerdiği için Tensör parçalarından çevirirken hatalar oluştu. Sorunlar her bir tipin explicit cast dönüştürülmesiyle düzeltildi.</p> <pre> dataset_train = tf.data.Dataset.from_tensor_slices((x_train['reviewText'].values, x_train['overall'].values)) dataset_test = tf.data.Dataset.from_tensor_slices((x_test['reviewText'].values, x_test['overall'].values)) </pre> <p>Implicit cast kullanılmadı çünkü bu compiler'ın fraksiyonel kısımla alakalı uyarı vermesine neden olurdu. Explicit cast durumun farkında olduğumuzu göstermenin bir yoludur.</p> <pre> table = tf.lookup.StaticHashTable(initializer=tf.lookup.KeyValueTensorInitializer(keys=tf.constant(['1.0', '2.0', '3.0', '4.0', '5.0']), # keys=tf.constant([1.0, 2.0, 3.0, 4.0, 5.0]), values=tf.constant([1, 2, 3, 4, 5]),), default_value=tf.constant(-1), name="target_encoding") </pre> <p>Anahtar ve değerlerin encode edilmesi için StaticHashTable fonksiyonu kullanıldı fakat 3d üstü değerler desteklenmediği için DenseHashTable denendi. Daha kötü olduğu için statige geri dönüldü.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
07/09/2022	Yazılım Geliştirme, Hata Analizi
<p>Bugün mentör, stajyer ile neden çıkan sonuçların accuracy (test veri doğruluğu) değerlerinin beklenenden daha düşük olduğu hakkında konuştu. 4 ana problem üzerinde duruldu.</p> <ol style="list-style-type: none"> 1. veri sayısının, deep learning için çok az olması 2. modelde çok fazla node'dan direkt olarak sınıf sayısı kadar node'a inmenin overfitting durumuna neden olabilmesi. (Az veri için fazla node çok yüksek kalıyor.) 3. Katman (layer)lardaki filtre sayısı ve baştan sona kaç katmanda inildiği. (Örneğin 512 node'dan 5 node'a birden inmek overfittinge neden olur) 4. Koddaki 500 sequence length'in (cümledeki kelime sayısının) çok fazla olması durumu. (Normalde dataset'teki veri sayıları en uzun kelimeli cümleye göre ayarlanır. Fakat küçük bir veride, bir cümledeki kelime sayısı 500'se one_hot encoding gibi işlemler çok fazla gereksiz veriye yol açar. Hem süre hem de overfitting bakımından zarar oluşur. Bu yüzden uzun cümlelerde cümleyi kırpma, kısa cümlelerde cümlelerin sayısal vektörüne 'padding' ekleyip uzatma işlemi yapıldı.) <p>Stajyer sorunların çözümü için kodlarda düzenleme yaptı. Fakat değişimler istenilen sonucu vermedi. Bunun üzerine multiclass classification'daki class(sınıf) sayısı 5'ten 2'ye düşürüldü. Softmax yerine sigmoid, 'sparse_categorical_crossentropy' yerine de 'binary_crossentropy' kullanıldı. Bu şekilde küçük veri için orta-seviyeli bir accuracy değerine ulaşıldı.</p> <p>4 farklı deep-learning metodu test edildi.</p> <ol style="list-style-type: none"> 1- Word Embedding + LSTM 2- Vanilla Deep Neural Network 3- Word Embedding + Deep Neural Network 4- Word Embedding + Convolutional Neural Network <p>Vanilla DNN'ler en kötü sonucu verdi. Diğer üçünde ise sonuçlar her testte farklı sonuç veriyordu fakat genel anlamda en iyisi LSTM + Word Embedding çıktı. Fakat diğerlerine göre en uzun süren de buydu.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
08/09/2022	Yazılım Geliştirme, Hata Analizi

Bugün stajyer accuracy değerlerinin yüksekliğini artırmak için gereken adımları mentör ve ekiple konuştu. Lemmatization için farklı bir kütüphanenin kullanılmasına karar verildi. Önceden NLTK'nin WordNet corpus'u ve WordNetLemmatizer fonksiyonları kullanılıyordu.

Fakat WordNet kitaplığı yeterince kök çözümülemiyordu. Kod parçası:

```
'''Wordnet lemmatizer'''
# Create pos tagger function to use in lemmatization
def part_of_speech(word):
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {"J": wordnet.ADJ,
                "N": wordnet.NOUN,
                "V": wordnet.VERB,
                "R": wordnet.ADV}
    return tag_dict.get(tag, wordnet.ADV)

# For every review, apply data cleaning and preprocess
def handled_text(text):
    text = str(text).lower()
    text = text.split()
    lem = WordNetLemmatizer()

    # Lemmatizing sentence by using part_of_speech()
    text = [lem.lemmatize(word,
                           part_of_speech(word))
            for word in text]
    text = " ".join(text)
```

Onun yerine spacy'i kullanmaya karar verildi.

```
%pip install https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.0.0/en_core_web_sm-3.0.0.tar.gz --user
```

Spacy'de "en_model incompatible" ve "model symlinks are not supported" hataları düzeltildi. İzin problemleri --user parametresi ile yüklendi. en_core_web_lg kütüphanesi indirilecekti fakat bilgisayar RAM'i yüklemeye yetmediği için ve Colab'de de yavaş yüklendiği için en_core_web_sm'ye geçilmesine karar verildi. Kodun bir parçası:

```
import spacy
lem = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
def handled_text(text):
    text = str(text).lower()
    text = lem(text)
    # convert spacy.tokens.doc.Doc to str
    text = " ".join([token.lemma_ for token in text])

    return text
```

ASIL CUMLE

Purchased this for my device, it worked as advertised. You can never have too much phone memory, since I download a lot of stuff this was a no brainer for me.

Text after spacy: purchase this for my device , it work as advertise . you can never have too much phone memory , since I download a lot of stuff this be a no brainer for I .

Text after wordnet: purchase this for my device, it work as advertised. you can never have too much phone memory, since i download a lot of stuff this be a no brainer for me.

Görüldüğü gibi spacy kelime köklerini çok başarılı şekilde buluyor.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
09/09/2022	Yazılım Geliştirme
<p>Bugün mentör stajyere sadece Embedding gibi layer fonksiyonları yerine, embedding'i daha iyi bir şekilde sağlamak için word2vec'in hazır pretrained modellerini araştırmasını ve incelemesini istedi. Stajyer ilk olarak GoogleNews-vectors-negative300.bin.gz isimli 3.5 Gb'lık modeli indirdi ve pythonla yükledi. Fakat dosyanın çok büyük olması sebebiyle algorithmada nasıl daha verimli bir şekilde, rami doldurmadan yüklenebileceği tartışıldı.</p> <pre>import gzip import shutil from gensim.models import KeyedVectors # unzip the bin.gz file GoogleNews.bin.gz and load it with word2vec with gzip.open('GoogleNews-vectors-negative300.bin.gz', 'rb') as f_in: with open('GoogleNews-vectors-negative300.bin', 'wb') as f_out: shutil.copyfileobj(f_in, f_out)</pre> <p>Gzip ve shutil'in birlikte kullanılmasıyla dosyanın sıkıştırma arşivinden çıkarken aynı anda yüklenmeyeceği, böylece rami hafifleteceğine karar verildi. Ayrıca arşivden çıkarken hata oluşması durumunda dosyaların kapatılacağından ve boş hafıza kaplamaması açısından with keywordüyle kullanılmasına karar verildi.</p> <pre>from gensim.models import KeyedVectors model = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin', mmap='r') model.most_similar(positive=['woman', 'king'], negative=['man'], topn=5)</pre> <pre>[('queen', 0.7118193507194519), ('monarch', 0.6189674139022827), ('princess', 0.5902431011199951), ('crown_prince', 0.5499460697174072), ('prince', 0.5377321839332581)]</pre> <p>KeyedVectors, 2 katmanlı neural network kullanarak Word2Vec WordEmbedding'i CBOW (Continuous bag of words) ve Skip-gram modellerinin birlikte kullanılmasıyla modeli oluşturuyor. Kelimelerin sayısal vektör haline getirilmesiyle vektörsel olarak karşılaştırmayı sağladı. Woman+King – man ='in 71% oranla 'queen' olduğunu buldu.</p> <p>Sonrasında bu metodun keras ve tensorflow'la nasıl birlikte kullanılabileceği tartışıldı. Ayrıca hala yavaş olmasından dolayı bunun dışında başka hangi pretrained corpusların kullanılabileceği konuşuldu.</p>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
12/09/2022	Yazılım Geliştirme
<p>Bugün, önceki gün kullanılan model corpus'u(kitaplığı)nın yüklenmesini hızlandırmak için başka bir yöntem olan mmap (Bellek Eşlemeli Dosyalar) fonksiyonu konuşuldu.</p> <p>Memory Mapped File ile çalışırken açılan dosya byte array gibi kullanılır. Yani dosya RAM'e sayfalar şeklinde yüklenir. Eğer mmap ile bir dosya açılmış ise, RAM üzerinde yapılan değişiklikler alt fonksiyonlardan birisi çağırılınca kadar diske aktarılmaz.</p> <pre>import mmap from zlib import Adler32 as compute_crc n_chunk = 1024**2 crc = 0 with open("fileno") as f: mm = mmap.mmap(f.fileno(), 0, prot=mmap.PROT_READ, flags=mmap.MAP_SHARED) while True: buf = mm.read(n_chunk) if not buf: break crc = compute_crc(buf, crc)</pre> <p>Fakat fonksiyonu kullanırken oluşan sıkıntılar yüzünden farklı bir veri modeline geçilmeye karar verildi.</p> <p>https://tfhub.dev/google/Wiki-words-250/2 TfHub'ın İngilizce Wikipedia corpus'undan token'lara dayalı eğitilmiş text embeddinglerden oluşan bir modeldir. Ayrıca boyutunun 900 mb olması modelin yüklenmesini hızlandırıyor. Model için ayrı bir avantaj ise modeli indirmeye gerek kalmaması. Paylaşılabilir kısaltılmış kod:</p> <pre>import tensorflow as tf import keras import tensorflow_hub as hub hub_layer = hub.KerasLayer("https://tfhub.dev/google/Wiki-words-250/2", input_shape=[], dtype=tf.string)</pre> <p>Bu Word2Vec modelini kullanmadan önce veriden gereksiz sütunlar ve "rating" satırındaki metni 2li sınıflandırmaya indirmek için 2.0, 3.0, 4.0 rate'leri çıkarıldı. Geriye 1.0 ve 5.0 kaldı. Sonrasında lemmatizer yardımıyla kelimeler kök hale getirildi. Başka bir fonksiyonla model verideki tokenlara uygulandı ve her token sayısal değere dönüştürüldü. Sonrasında tüm cümlelerin aynı boyutlu hale getirilmesi için uzun cümleler 50 kelimeye sınırlandırıldı, kısalar ise sona 0 ekleyerek 50'ye çıkarıldı. Bundan sonra preprocessing yapıldı.</p> <pre>def getTrainAndTestData(dataset): ''' Split the data to train and test''' # 90% of the dataset train = dataset.sample(frac=0.9, random_state=100) # 10% of the dataset test = dataset.drop(train.index) return train, test</pre> <p>Veriler random state 100 olacak şekilde %90-%10 olarak dağıtıldı.</p>	
Stajımı bu firmada yaptım.	Staj Yeri Yetkilisinin
Staj Yapanın İmzası	Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
13/09/2022	Yazılım Geliştirme
<p>Bugün TfHub text embedding modelinin çalışma süresini düşürmek için çalışmalar yapıldı. Aktivasyon fonksiyonları değiştirildi, Embedding, LSTM gibi farklı katman fonksiyonları kullanıldı. Epoch(devir) sayısı değiştirildi. Son katmandaki (layerdaki) sınıf sayısı 1'den 2'ye çıkarıldı. Bunun yapılmasının nedeni her ne kadar son katmandaki node sayısı 1 olunca zaten 2'li sınıflandırsa da labellar 2li one-hot encoding'e göre yapıldığı için node sayısı 2'ye çıkarıldı. Sigmoid node sayısı 1'ken daha iyi çalışıyor fakat softmax lojistik fonksiyonun çoklu boyutlara genelleştirilmesi olduğu için 2 node'da daha iyi çalışıyor. Bu nedenle sigmoid'den softmax'e geçiş yapıldı.</p> <pre> model.add(hub_layer) model.add(keras.layers.Dense(32, activation='relu')) model.add(keras.layers.Dense(32, activation='tanh')) model.add(keras.layers.Dense(2, activation='softmax')) model.summary()</pre> <p>Modeli compile ederken son katmandaki node 1 olmadığı için loss fonksiyonunu 'binary_crossentropy' yerine 'categorical_crossentropy' olarak ayarlandı. İstenilen hıza ulaşıldı. Model train datalarına göre fit() edilip test datalarına göre evaluate() edilince %95 doğruluk oranına ulaşıldı.</p> <pre> Fitting Train Data Epoch 1/5 - 7s 42ms/step - loss: 0.2857 - accuracy: 0.9392 Epoch 2/5 - 5s 41ms/step - loss: 0.2269 - accuracy: 0.9400 Epoch 3/5 - 5s 44ms/step - loss: 0.2247 - accuracy: 0.9400 Epoch 4/5 - 6s 49ms/step - loss: 0.2245 - accuracy: 0.9400 Epoch 5/5 - 6s 50ms/step - loss: 0.2227 - accuracy: 0.9400 Model: "sequential_39" Layer (type) Output Shape Param # ===== lstm_36 (LSTM) (None, 32) 36224 dense_36 (Dense) (None, 2) 66 ===== Total params: 36,290 Trainable params: 36,290 Non-trainable params: 0 Evaluating Test Data 13/13 - 1s - loss: 0.1872 - accuracy: 0.9543 Test score: 0.18716658651828766 Test Accuracy: 0.9543269276618958</pre>	
Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:

TARİHİ:

14/09/2022

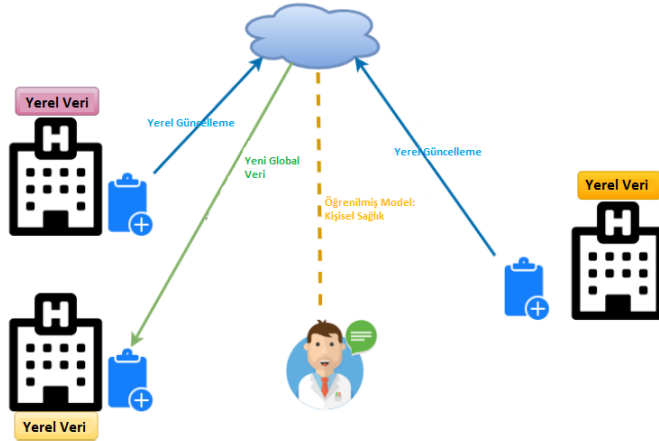
KAPSAMI:

Eğitim

Bugün stajyer, mentoru tarafından verilen yönlendirme ve paylaşılan makaleler doğrultusunda Federated Learning (Federe Öğrenme) hakkında bilgiler öğrenmeye başladı.

Federe öğrenme istatistiksel modelleri uzak cihazlarda veya diğer uygulamalarla etkileşimde olmayan yalıtılmış veri merkezlerinde (örneğin mobil cihaz veya hastaneler) eğitip bunu yaparken lokalize eden, bunu yaparken anonimleşmiş biçimde tutup sunucular üzerinde eğiten makine öğrenmesi tekniğidir.

Normalde klasik makine öğrenmesi modelleri training verilerinin veri merkezinde toplanmasını gerektirir. Ama federe öğrenme cihazdaki training verilerini kullanır ve bu verileri cihazda tutan ortak bir modeli öğrenmeyi sağlar. Verilerin bulutta saklanması ihtiyacı ortadan kalkmış olur. Hastane Örneği:



Federated Learning'in Hastane Sisteminde Kullanılması

Yerel hedef verisini ise bu formülle bulabiliriz. M toplam cihaz sayısını, F_k ise K'nıncı cihaz için yerel hedef fonksiyonu olduğunu gösterir. Aynı zamanda yerel verideki deneysel riski de gösterir. P_k ise kullanıcı tarafından belirlenir ve her cihazdaki göreceli etkiyi gösterir. Doğal yerleştirme verisinde p_k değeri $1/n$ 'e veya n_k/n 'e eşittir.

$$\min_w F(w), \text{ where } F(w) := \sum_{k=1}^m p_k F_k(w).$$

Stajımı bu firmada yaptım.
Staj Yapanın İmzası

Staj Yeri Yetkilisinin
Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
15/09/2022	Eğitim, Yazılım Geliştirme

Bugün stajyer federe öğrenmenin önemli yaklaşımlarından olan, model güncellemede kullanılan Stochastic gradient descent (SDG) kavramını öğrendi.

Olasılıksal Gradyan Azalma anlamına gelen SDG, rastgele alınan değişkenlerden başlayarak global minimum değerine loss fonksiyonlarının türevini alarak ve parametreler için rastgele değerler toplayan bir sınıflandırma yaklaşımıdır. Lineer sınıflandırıcıları ayırt etmede kullanılır.

Federe öğrenmede her çalışan güncel modeli kaynaktan alır, güncel hale getirir (örneğin SDG'yi yerel veri setinde kullanarak), en son da hesapladığı yerel güncellemeyi kaynağa geri bırakır. Kaynak toplayıcı yerel veri güncellemelerini yeni geliştirilmiş global model için kullanır. Hem hesaplama için gereken güç paylaştırılmış olur hem de kullanıcının gizlilik riski azalmış olur. En son da yerel kullanıcılar global modeli kaynaktan indirir ve global eğitim tamamlanana kadar sonraki yerel güncellemeyi hesaplamaya devam ederler.

Öğleden sonra stajyer, ekibe kullanıcı eklenince güncelleme yapabilen, tüm eklenen kullanıcıların bilgilerini alabilen discord botu geliştirmeye başladı.

Discordun varsayılan intenti ile başlandı fakat botta hata verdiği için nextcord'un intentsine geçildi. Botu tetikleyecek karakter ! olarak belirlendi.

```

5 #client = discord.Client()
6
7 intents = discord.Intents.default()
8 intents = nextcord.Intents.all()
9 intents.members = True
10
11 client = commands.Bot("!", intents=intents)
12

```

Client isimli botun global değişken yerine ctx adlı context fonksiyonu kullanmasına karar verildi. Ayıklayıcılar guild'den alındı. En sonsa token ile bot çalıştırıldı.

```

@client.command()
async def getmember(ctx, member:str):
    print([i.name + '#' +
           i.discriminator
           for i in ctx.message.guild.members
           if member in i.name+'#'+i.discriminator])
client.run("OTE1MjE0MzMTIzODc1MzU5.YaYVxQ.gZ2HX2WPydqhPe7ZBSwI-N54n-k")

```

Bot başarılı bir biçimde güncel kullanıcıları aldı.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
16/09/2022	Yazılım Geliştirme, Eğitim

Bugün discord botunun geliştirilmesi için çalışmalar yapıldı. Bottaki eksik botun sadece çalıştırıldığı zamanki kullanıcıları almasıydı fakat istenilen şey bir kullanıcı ekipten çıkınca veya yeni bir kullanıcı ekibe girince bunu güncellemesiydi.

```
@client.event
async def on_ready():
    #guild = client.get_guild(915220129111998524)
    while(True):
        channel=client.get_channel(915220129111998527)
        memberList=[]
        member=client.get_all_members()
        for i in member:
            memberList.append(i)
        print(i)
        sleep(2)
    # member2 = guild.get_member_named("User#Discrim")
    member2 = commands.Converter.MemberConverter().convert("User")
```

While döngüsü içine almak işe yaramadı. İstenilen şey client get_all_members()'ı çağırınca yeni kullanıcılar eklendiyse güncelleme yapılmasıydı ama yapılmadı. Botun ilk oluşturulduğu clients = commands.Bot('!', intents) while döngüsünün içine alınmak istendi fakat @client.event'in içinde client bir daha oluşturulmaya izin verilmediği için hata verdi.

```
@bot.command()
@commands.is_owner()
async def reload(ctx, extension):
    bot.reload_extension(f"cogs.{extension}")
    embed = discord.Embed(title='Reload', description=f'{extension} successfully reloaded')
    await ctx.send(embed=embed)
```

En son adminin eklentiye bot kapanmadan yenilemesine izin verilecek biçimde Embed() gömme işlemi yapılmasına izin verildi. Await global düzeyde kullanılamadığı için fonksiyonda kullanıldı böylece send işleminin tamamlanması beklenmiş oldu.

Stajyer ayrıca makalelere devam etti ve heterojen veriyi federated learning ile nasıl modelleyebileceğini öğrendi. MOCHA isimli optimizasyon framework'ü bağlantılı fakat ayrılmış olan farklı cihazdaki modelleri çoklu görevli öğrenme ile kişiselleştirme yapabilir. Başka bir yaklaşımsa Bayes ağını öğrenme sırasında yıldız topolojisi ile modellemektir. Yıldız topolojisi tüm sistemleri merkezi bir düğüme bağlanılarak ağ hata olasılığını azaltır. Fakat bunun sorunu geniş federe ağlarda genelleştirme yapmanın bayesle çok ağır olmasıdır. Çünkü yıldız topolojisinde merkez işleyişine bağımlılık sistemi çok yüksektir.

Stajımı bu firmada yaptım. Staj Yapanın İmzası	Staj Yeri Yetkilisinin Adı, Soyadı, İmzası, Firma Kaşesi

YAPILAN İŞİN:	
TARİHİ:	KAPSAMI:
19/09/2022	Eğitim, Dokümantasyon Oluşturma

Bugün stajyer dağıtık optimizasyonda en sık kullanılan yaklaşımlardan bulk senkron yaklaşımı and asenkron yaklaşımları öğrendi.

Federe öğrenmenin (FL) Derin öğrenme modellerine uygulanmasıyla ham client verisine erişim olmadan çalışması kullanıcı gizliliğini korumak için büyük fayda sağlar.

Horizontal FL ve Vertical FL kavramları öğrenildi.

- HFL sistemlerde tüm öğrenen clientlar global modeli kendi yerel veri setleriyle eğitirler. Aynı veri özelliklerini kullanmaları nedeniyle clientlar, lineer regrasyon veya SVM gibi aynı yapay zekâ modellerini yerelde eğitmede kullanabilirler. Güvenliği/gizliliği artırmak için yerel güncellemeyi Secure Multiparty Computation, Differential Privacy / K- anonymity veya Homomorphic Encryption şifreleme mekanizmalarıyla maskeleyerek kullanırlar.
- VFL sistemlerde ise çeşitli özellikleri gruplayarak eğitimdeki veri kaybı bulunur. Böylece 2 oluşumdan da veri içeren model oluşturmuş oluruz. Örneğin IoT uygulamalarında VFL, akıllı bir şehirdeki oluşumlar arasında paylaşılan öğrenme modeli olabilir.

Teknik	Avantajı
Mini-Batch Gradient Descent	Verimlilik Sık Model Güncellemesi
Ada-Grad	Hız ve güvenilirlik
RMSProp	Az RAM gerektirmesi
Fed Adagrad	Uyum Yeteneği
FedYogi	Uyum Yeteneği
FedAdam	Yüksek Doğruluk

FL'yi uygulayan TensorFlow Federated'in Python'da kullanımı öğrenildi. FedAdaGrad, FedYogi ve FedAdams optimizasyon yöntemlerini kullanır ve yüksek doğruluk oranına sahiptir.

```
import tensorflow_federated as tff

iterative_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=lambda: tf.keras.optimizers.SGD(learning_rate=0.02),
    server_optimizer_fn=lambda: tf.keras.optimizers.SGD(learning_rate=1.00))
```

Stajımı bu firmada yaptım.
Staj Yapanın İmzası

Staj Yeri Yetkilisinin
Adı, Soyadı, İmzası, Firma Kaşesi