T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

DERİN ÖĞRENME İLE TWEET KONUSUNUN BULUNMASI

Bengi YÖRÜKOĞLU

Danışman

Dr. Öğr. Üyesi Burcu YILMAZ

Ocak, 2019
Gebze, KOCAELİ

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

DERİN ÖĞRENME İLE TWEET KONUSUNUN BULUNMASI

Bengi YÖRÜKOĞLU

Danışman
Dr. Öğr. Üyesi Burcu YILMAZ

Ocak, 2019
Gebze, KOCAELİ

Bu çalışma ..../..../200.. tarihinde aşağıdaki jüri tarafından Bilgisayar Mühendisliği Bölümü'nde Lisans Bitirme Projesi olarak kabul edilmiştir.

Bitirme Projesi Jürisi

Danışman Adı	Burcu YILMAZ	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

Jüri Adı	Murat Şeker	
Üniversite	Gebze Teknik Üniversitesi	
Fakülte	Mühendislik Fakültesi	

ÖNSÖZ

Projenin hazırlanmasında emeği geçenlere, raporun son halini almasında yol gösterici olan Sayın Dr. Öğr. Üyesi Burcu YILMAZ hocama ve bu çalışmayı destekleyen Gebze Teknik Üniversitesi'ne içten teşekkürlerimi sunarım.
Ayrıca eğitimim süresince bana her konuda tam destek veren aileme ve bana hayatlarıyla örnek olan tüm hocalarıma saygı ve sevgilerimi sunarım.

Ocak, 2019 Bengi Yörükoğlu

İÇİNDEKİLER

ŞEKİL LİSTESİ

TABLO LİSTESİ

KISALTMA LİSTESİ

LSTM	: Uzun Kısa Süreli Bellek (Long Short Term Memory)
RNN	: Tekrarlayan Sinir Ağı (Recurrent Neural Network)
Seq2seq	: Sekans Ağı (Sequence to Sequence)
UniLSTM	: Tek Yönlü Lstm Katmanı (Uni-directional LSTM layers)
Bi-LSTM	: Çift Yönlü Lstm Katmanı (Bi-directional LSTM layers)

ÖZET

Projenin amacı derin öğrenme yöntemi ile tweetin konusunun bulunmasıdır. Oluşturduğumuz veri setini sequence to sequence model ile eğiterek konunun tespit edilmesi amaçlanmıştır. Sekans modeli (Seq2Seq), sekansları bir alandan (tweet örnekleri) başka bir alandaki sekanslara (tweet konusu) dönüştürmek amacı ile kullanılır. Sekans modeli kodlayıcı ve kod çözücüye sahiptir. Kodlayıcı-kod çözücüye sahip sekans modeli, makine çevirisi gibi zorlu dizilim tahmin problemlerini ele almak için tekrarlayan sinir ağlarını kullanmak için bir model sağlamaktadır.

Projede eğitim için 15 farklı etikete sahip toplam 60.000 veri oluşturulmuştur. Bu 60.000 veri Fenerbahçe, Galatasaray, Beşiktaş, yapay zeka, uzay, edebiyat, hava durumu, hayvan sever, vegan olmak, sanat, sosyal medya, meditasyon, bitcoin, yeni yıl mesajım, moda konuları ile alakalı eşit sayıda verilerden oluşturulmuştur. Projede veriler öncelikle verilen tweetlerin konularını bulan LSTM katmanına dayanan, kelime tabanlı (kelimelerin vektörlere çevrilmesi), kodlayıcı ve kod-çözücü yapısı içeren bir sekans modeli ile eğitilmiştir. Daha sonra aynı yapı, karakterlerin vektörlere çevrilmesi ile eğitilmiştir. Son olarak da model çift yönlü kodlayıcı ve kod çözücü mimarisine dönüştürülmüştür ve çift yönlü kodlayıcı-kod çözücü mimarisi ile eğitilmiştir.

Farklı modellerle eğitim sonucunda en doğru sonuçlar çift yönlü kodlayıcı ve kod çözücü mimarisi ile elde edilmiştir. UniLstm ile eğitim sonucunda tüm test setinin sonucu olarak hep aynı sonuç (her tweet için aynı hastag tahmini) alınmıştır. Bi-Lstm katmanı ile oluşturulmuş encoder decoder ile eğitilen model ise 100 veriden oluşan test sonucunda hepsinde doğru hastag çıktısını vermiştir. Eğitime verilen veri seti için Bi-lstm modelinin Uni-lstm modeline göre çok daha başarılı sonuçlar verdiği gözlemlenmiştir.

SUMMARY

The aim of the project is to find the subject of Tweet with deep learning method. The solution includes determining the subject with sequence to sequence model. The sequence model (Seq2Seq) is used to convert sequences from one area (tweet samples) to another in the field (tweet subject). The sequence model has the encoder and decoder. The sequence model with the encoder-decoder provides a model for using recurrent neural networks to approach difficult sequence prediction problems, such as machine translation.

A total of 48,000 data were created with 15 different labels for training in the project. This 48.000 data is composed of an equal number of data related to Fenerbahce, Galatasaray, Besiktas, artificial intelligence, space, literature, weather, animal loving, being vegan, art, social media, meditation, bitcoin, new year message, fashion topics. The data in the project were firstly trained with a sequence model based on the LSTM layer, which finds the subjects of the given tweets, with a word-based (translation of words into vectors), coding and decoding structure. The same structure was then trained by translating the characters into vectors. Finally, the model has been transformed into a two-way encoder and decoder architecture and is trained with a bi-directional decoder architecture.

As a result of the training with different models, the most accurate results were obtained by the two-way encoder and decoder architecture. As a result of the whole test set with UniLstm, the same result (same patient estimate for each tweet) was taken as the result of the whole test set. The model, which was trained with encoder decoder which was formed with Bi-Lstm layer, gave the correct hastag output in all of the 100 test results. It is observed that the Bi-lstm model gives more successful results than the Uni-lstm model for the given data set.

1. GİRİŞ

Yaşadığımız bu dijital çağda, herkes tarafından kolayca ve rahatça ulaşılabilir büyük bir bilgi kaynağı vardır. Sosyal medya kullanımının patlaması ile birleştiğinde, hayatın her kesiminden insanlar, kamuya açık olarak okudukları bilgilerle ilgili görüş ve düşüncelerini dile getiriyorlar. Dolayısıyla bu görüşler ve yorumların analizinde güçlü bir ilgi vardır ve hangi konuların çevrimiçi olarak trend olduğunu anlamak ilgi çekicidir. Konu tespiti araştırması, 1998 yılında DARPA Translingual Information Detection, Extraction ve Summarization (TIDES) programı kapsamında Topic Detection and Tracking (TDT) kapsamında başlatılmıştır. Konu tespiti, bilgi içeriğine göre dokümanlardaki konuları otomatik olarak tanımlama yeteneğini ifade etmektedir.

Projede derin öğrenme yöntemiyle tweetin konusu tespit edilmeye ve tweetin o gün trend topic olup olmadığı belirlenmeye çalışılmıştır. Gözetimli öğrenme yöntemi kullanılmıştır. Gözetimli öğrenme, makinenin yeni verileri etiketlemek için daha önceden etiketlenmiş bir eğitim verisi grubundan öğrendiği görevini ifade etmektedir. Eğitim verileri, giriş vektörleri ve bunlara karşılık gelen etiketlerden oluşur.

Projede kullanılan sekans modeli ile iki tekrarlayan sinir ağı bir diziyi diğerine dönüştürmek için birlikte çalışır. Sıra ağı ya da seq2seq ağı encoder decoder ağına bir sıra kodlayıcı ve kod çözücü olarak adlandırılan iki RNN'den oluşan bir modeldir. Bu yöntemde, giriş dizisini sabit boyutsal bir vektörle eşleştirmek bir Uzun Kısa Dönemli Bellek (LSTM) kullanılır ve ardından hedef diziyi vektörden çözmek için başka bir LSTM kullanılır. LSTM, uzun süreli bağımlılıkları hatırlayabilen bir RNN türüdür ve bilgiyi sahip olduğu bellek sayesinde bağlam içerisinde değerlendirebilir. Dolayısı ile LSTM kullanımı ile başarının artacağı düşünülmüştür.

Sekans ağında kodlayıcı bir giriş dizisini okur sonucunda tek bir vektöre çevirir ve kod çözücü bu vektörü bir çıkış dizisi üretecek şekilde okur. Hem karakter seviyesindeki RNN öğreticilerinde kullanılan karakterlerin vektörlere çevrilmesi ile, hem de her bir kelimeyi tek-sıcak bir vektöre (one hot vector) çevrilmesi ile oluşturulan modelin eğitim sonucunda en iyi sonucu vermesi amaçlanmıştır.

2.METOT

Projede kullanılan model değişken uzunluklu bir giriş sırasını okuyan bir kodlayıcı ve değişken uzunluktaki bir çıktı sırasını öngören bir kod çözücüden oluşur. Seq2seq Encoder - Decoder ağı ile oluşturulan model eğitim için kullanılmıştır. Eğitim için kullanılmak üzere twitter'dan türkçe atılan 15 farklı kategoriye sahip toplam 60.000 veri oluşturulmuştur. Konu başlıkları tablo 1'de görülmektedir.

Hastag	Veri Adedi
Edebiyat	4000
Bitcoin	4000
Yapay Zeka	4000
Moda	4000
Hava Durumu	4000
Galatasaray	4000
Fenerbahçe	4000
Beşiktaş	4000
Sanat	4000
Yeni Yıl Mesajım	4000
Uzay	4000
Hayvansever	4000
Vegan Olmak	4000
Meditasyon	4000
Sosyal medya	4000

Tablo 1: Veri seti içeriği

Oluşturulan veriler olabildiğince gereksiz verilerden (emoji, link, ve konuyla alakasın gelen veriler) başarıyı olumsuz yönde etkilememesi için temizlenmiştir. Ayrıca araştırma sonucu tüm kategoriye ait verilerin eşit bir şekilde dağıtılmasının ve verilerin karıştırılması gerektiği görülmüştür ve karıştırma işlemi sağlanmıştır.[1]

Eğitim için verilerin doğru formatta hazırlanma aşaması gerçekleştirilmiştir. Veri setinde tweet örneğinin bulunduğu giriş dizisi, tweetin konusunun bulunduğu hastag dizisinden tab ile ayrılmıştır. Giriş dizisinin bulunduğu encoder_input_data ve hastag dizisinin bulunduğu decoder_input data '\t'' ye göre ayrıştırılarak ayrı arraylerde toplanmıştır ve içlerindeki unique karakterler Şekil 1'de gösterildiği gibi ayrı setlerde toplanmıştır.

```
# We use "tab" as the "start sequence" character
# for the targets, and "\n" as "end sequence" character.
target_text = '\t' + target_text + '\n'
input_texts.append(input_text)
target_texts.append(target_text)
for char in input_text:
    if char not in input_characters:
        input_characters.add(char)
for char in target_text:
    if char not in target_characters:
        target_characters.add(char)
```

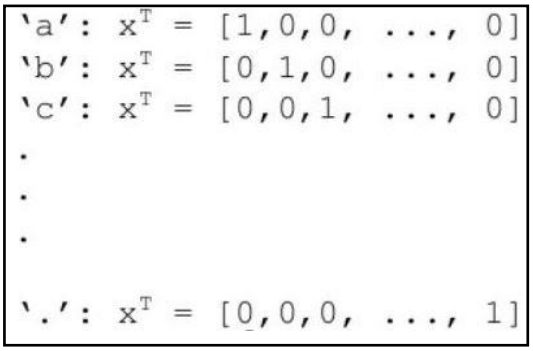
Şekil 1: Karakterleri ayrı setlerde toplama

Karakter düzeyinde bir model geliştirdiğimiz için, benzersiz karakterler (“a”, “b”, “c”, gibi) şekil 2’de gösterildiği gibi sayısal göstergelerle ilişkilendirilmiştir.

```
input_token_index = dict(
    [(char, i) for i, char in enumerate(input_characters)])
target_token_index = dict(
    [(char, i) for i, char in enumerate(target_characters)])
```

Şekil 2: Karakterleri indeksleme

Her decoder ve encoder girdileri için numpy kütüphanesinin zeros methodu ile her indeksine 0 atanmış birer array oluşturulmuştur, dosyamızdan gelen etiketli verinin input ve target kısmında rastlanan harflerin indekslerine 1 değeri atanmıştır. Oluşan array örneği şekil 3’te gösterilmiştir.



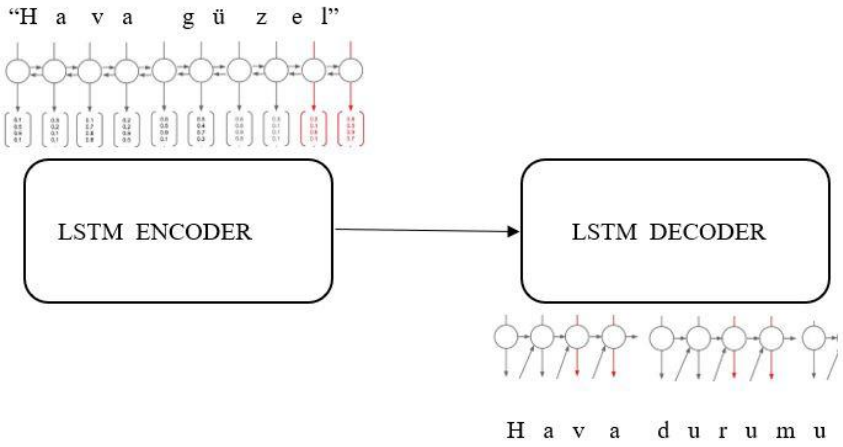
Şekil 3: One hot encode gösterimi

Aşağıda veri setindeki yapay zeka konulu bir tweet ve etiketi olan konusu örnek olarak gösterilmiştir.

Yapay zeka ile nadir hastalıkların teşhisi artık daha kolay.	#YapayZeka
Tweet örneği	Konusu

Eğitim için oluşturulan modelin çalışma mantığı [2] aşağıda anlatıldığı şekilde gerçekleşmektedir.

1. Tweet ve Hastag cümleleri için karakter vektörleri oluşturulmaktadır. Bunlar, kodlayıcı ve kod çözücünün girişleri olmaktadır. Tweet durumunu belirten hastag’in bir karakter vektörü, kayıp fonksiyonu için hedef veri olarak da kullanılacaktır.
2. Tweet cümle dizisi, dizi sonuna kadar karakter karakter kodlayıcıya gömülür.
3. Oluşan son encoder durumları (encoder_states) decoder’a başlangıç durumu olarak verilir.
4. Kod çözücü her adımda 3 girişe sahip olmaktadır. 2 kod çözücü durumuyla ve Hastag(konular) dizisinden karakter karakter beslenir.
5. Kod çözücünün her aşamasında kod çözücünün çıktısı, hedef verilerle karşılaştırılan softmax katmanına gönderilir.



Şekil 4 Karakter Tabanlı Encoder – Decoder Modeli

Model iki alt modele bölünmüştür: giriş tweet dizisinin sabit uzunluklu bir kodlamasını çıktılamaktan sorumlu kodlayıcı ve çıktı dizisini tahmin etmekten sorumlu kod çözücü. Şekil 5’te modelimizin ilk adımı olan encoder katmanı tanımlanmıştır.

```
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_state=True)
encoder_outputs, state_h, state_c = encoder(encoder_inputs)
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

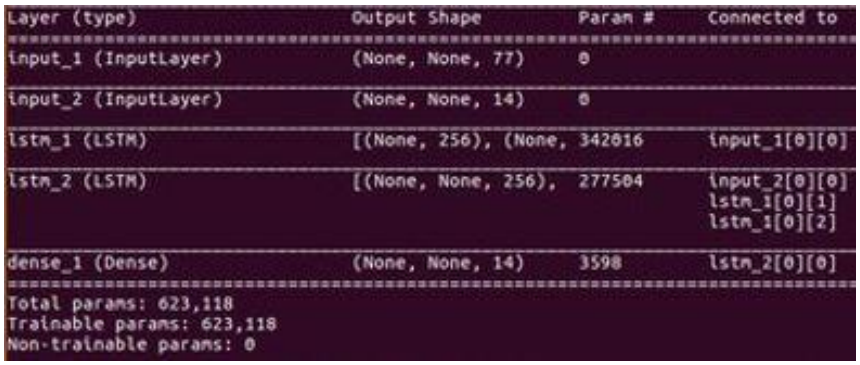
Şekil 5: Encoder Katmanı Tanımlaması

Kodlayıcıdaki LSTM katmanı, True olarak ayarlanan return_state argümanı ile tanımlanmıştır. LSTM katmanları tarafından döndürülen gizli durum çıktısını ve ayrıca katmandaki tüm hücreler için gizli ve hücre durumunu döndürmesi anlamına gelmektedir.

```
# Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None, num_decoder_tokens))
# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.
decoder_lstm = LSTM(latent_dim, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs,
                                     initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
```

Şekil 6: Decoder Katmanı Tanımlaması

Şekil 6’da modelimizin ikinci adımı olan decoder katmanı tanımlanmıştır. Kodlayıcıdan elde edilen son gizli ve hücre durumu kod çözücünün durumunu başlatmak için kullanılır. Her bir karakteri tahmin etmek için bir yoğun çıktı (Dense) katmanı kullanılır.



Layer (type)	Output Shape	Paran #	Connected to
input_1 (InputLayer)	(None, None, 77)	0	
input_2 (InputLayer)	(None, None, 14)	0	
lstm_1 (LSTM)	[(None, 256), (None,	342016	input_1[0][0]
lstm_2 (LSTM)	[(None, None, 256),	277504	input_2[0][0] lstm_1[0][1] lstm_1[0][2]
dense_1 (Dense)	(None, None, 14)	3598	lstm_2[0][0]

Total params: 623,118
Trainable params: 623,118
Non-trainable params: 0

Şekil 7: Katmanlar

Tek yönlü Lstm katmanıyla oluşturulmuş sekans yapısındaki katmanlar şekil 7'de gösterilmiştir.

```
# Run training
model.compile(optimizer='rmsprop', loss='categorical_crossentropy')
model.fit([encoder_input_data, decoder_input_data], decoder_target_data,
          batch_size=batch_size,
          epochs=epochs,
          validation_split=0.2)
```

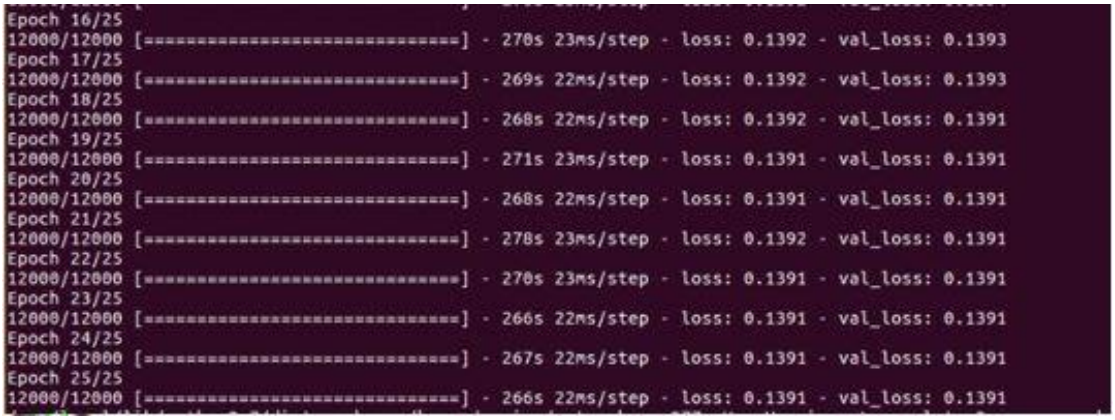
Şekil 8: Hiperparametreler

Model tanımlandıktan sonra modelin eğitimi için şekil 8'de görülmekte olan compile ve fit fonksiyonları çağrılmıştır. Çok sınıflı bir sorun üzerinde çalışıldığı için loss parametresinde categorical_crossentropy kullanılmıştır. Datayı eğitmek üzere kullanılacak fit fonksiyonunun parametrelerinde bulunan validation_split değeri 0.2 verilerek verimizin %80'inin eğitim için kullanılması, geri kalan %20'sinin test ederek hataları düzeltmesi amaçlanmıştır.

3.DENEYLER VE BULGULAR

Model ilk olarak 15.000 veriden oluşan veri kümesi ile epoch ve batch değerleri değiştirilerek farklı parametrelerle eğitilmiştir.
Epoch: Sistemin öğrenme aşamasında döngüye kaç sefer gireceğini gösteren değerdir.
Batch Size: Girdilerin parçalar halinde işlenmesi olarak ifade edilir.
Model epoch değeri 25 ve batch size değeri 120 olarak ayarlanarak 15.000 veri ile eğitilmiştir. Epoch 16 ve 25 aralığında loss ve validation loss değerleri şekil 9'da görülmektedir.

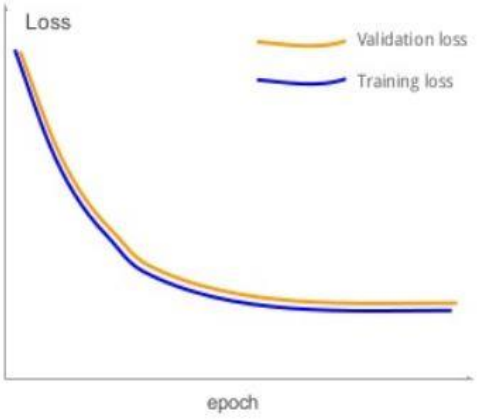
Batch size=120
Epochs=25
Num_samples=15000

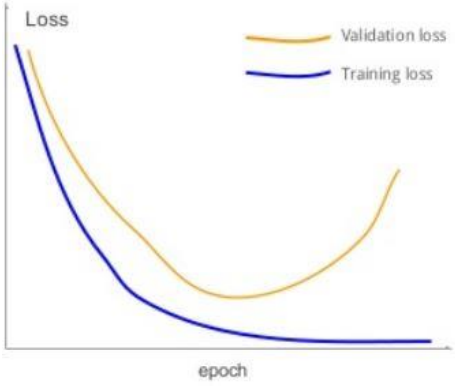


Şekil 9: Loss ve validation loss değerleri

İlerleme çubuğunda görülen loss değeri, her batch üzerindeki kaybın ortalamasıdır. Model eğitim sırasında sürekli adapte oluyor ve değişiyor, bu sayı gerçek loss değerini göstermez, sadece bir göstergedir. Validation loss, model tamamlandığında, yani her epoch tamamlandığında hesaplanır. Aynı verileri kullanıyor olsak bile, her iki değerin aynı çıkmama olasılığının olma sebebi budur.

- Loss değeri validation loss değerinden çok daha düşükse, bu durum modelin overfitting olduğu anlamına gelir.
- Loss / validasyon kaybı yaklaşık olarak eşitse, modelin underfitting olduğu anlamına gelir.[3]



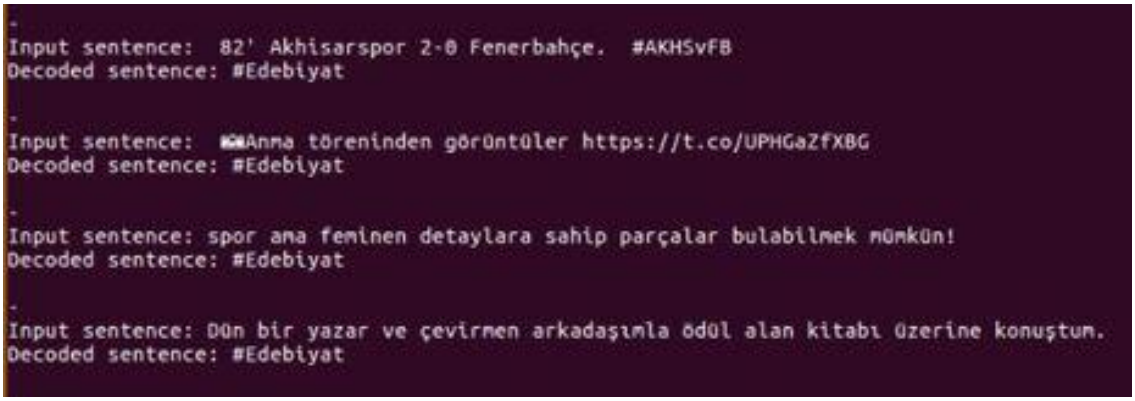


Şekil 10- Underfitting grafik gösterimi Şekil 11 – overfitting grafik gösterimi [4]

Derin öğrenmede düşük performansın nedeni ya veriye aşırı uyumu ya da yetersiz uyumudur. Şekil 10'da yetersiz uyum, şekil 11'de ise aşırı uyum grafikleri gösterilmektedir.[5]

- Aşırı uyum(overfitting), eğitim verilerini çok iyi modelleyen bir modele ilişkindir. Bu kavramlar yeni verilere uygulanamamaktadır ve modellerin genelleme yeteneğini olumsuz yönde etkilemektedir..
- Yetersiz uyum, eğitim verilerini modelleyemeyen veya yeni verilere genellenemeyecek bir modele işaret etmektedir. Yetersiz bir derin öğrenme modeli uygun bir model değildir ve eğitim verileri üzerinde düşük performansa sebep olmaktadır.

Eğitim sonucu elde edilen losss ve validation loss değerleri bu bilgilere göre değerlendirilerek en doğru epoch değeri ayarlanmaya çalışılmıştır.

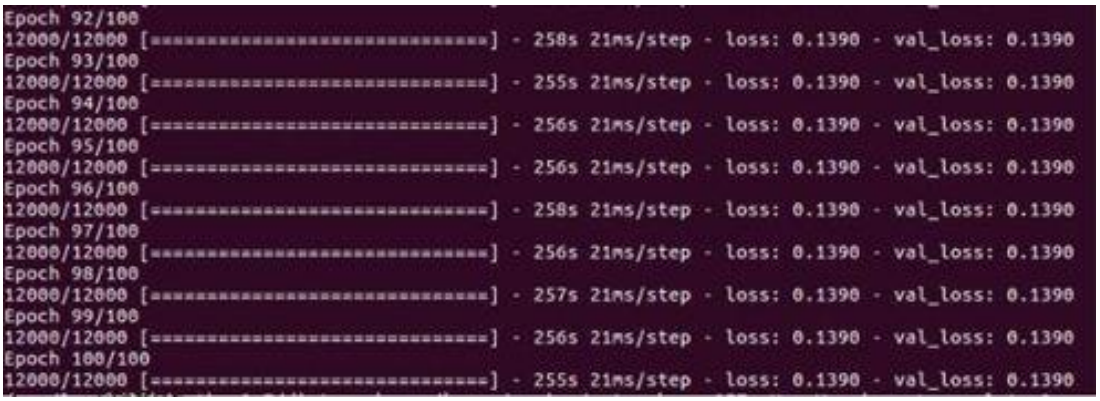


Şekil 12: Eğitim sonucu çıktı

Eğitim sonucu 100 adet test verisi verilmiştir ve tüm verilerin sonucu olarak aynı konu çıktısını vermiştir. Şekil 12'de futbol, moda, edebiyat tweet örneğinin sonuçları gözlemlenebilir.

Batch size=120 Epochs=100 Num_samples=15000

Batch size değeri 120, epoch değeri 100 olarak ayarlanarak 15.000 veri içeren data seti ile model eğitilmiştir.



Şekil 13: Loss ve validation loss değerleri

Epoch 92 ve 100 aralığında loss ve validation loss değerleri şekil 13'de görülmektedir.

Şekil 14: Eğitim sonucu çıktı

Eğitim sonucu 100 adet test verisi verilmiştir ve tüm verilerin sonucu olarak aynı konu çıktısını vermiştir. Şekil 14'de futbol, moda, edebiyat, hava durumu tweet örneğinin sonuçları gözlemlenebilir.

Görülüğü gibi epoch sayısının 4 katına çıkması loss değerlerinde ve sonuçlarında bir değişiklik yaratmamıştır ve her 2 durumda da her tweet örneğine aynı konu çıktısını vermiştir. Her tweet örneğine karşılık aynı konu çıktısını verdiği için başarı sağlanamamıştır.

Kodlayıcı ve kod çözücüye daha fazla Lstm katmanı eklenmesinin başarıya olan katkısı araştırılmıştır ve fazladan bir Lstm katmanı eklenerek yeni bir encoder yapısı tanımlanmıştır.[6]

```
# Define an input sequence and process it.
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = LSTM(latent_dim, return_sequences=True)
encoder_outputs, state_h, state_c = LSTM(latent_dim, return_state=True)(encoder(encoder_inputs))
# We discard `encoder_outputs` and only keep the states.
encoder_states = [state_h, state_c]
```

Şekil 15: Encodera 2.Lstm katmanı eklenmesi

Şekil 15'de görüldüğü üzere bu yapıda katman ekleme başka bir katmanı var olan bir katmanın çıktısını üstüne çağırarak yapılmıştır.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None, 89)	0	
lstm_1 (LSTM)	(None, None, 256)	354304	input_1[0][0]
input_2 (InputLayer)	(None, None, 17)	0	
lstm_2 (LSTM)	[(None, 256), (None,	525312	lstm_1[0][0]
lstm_3 (LSTM)	[(None, None, 256),	280576	input_2[0][0] lstm_2[0][1] lstm_2[0][2]
dense_1 (Dense)	(None, None, 17)	4369	lstm_3[0][0]

Total params: 1,164,561
Trainable params: 1,164,561
Non-trainable params: 0

Şekil 16: Katman gösterimi

Şekil 16'da bir katman daha eklendikten sonra oluşan katmanların yeni görüntüsü görülmektedir. Katman eklendikten sonra model önce epoch değeri 25, batch size değeri 120 ile eğitilmiştir. Daha sonra epoch değeri tekrar 4 katına çıkartılarak epoch 100 ve batch size 120 ile eğitilmiştir.

```
Epoch 20/25
12000/12000 [==============================] - 477s 40ms/step - loss: 0.1393 - val_loss: 0.1391
Epoch 21/25
12000/12000 [==============================] - 481s 40ms/step - loss: 0.1392 - val_loss: 0.1392
Epoch 22/25
12000/12000 [==============================] - 484s 40ms/step - loss: 0.1393 - val_loss: 0.1394
Epoch 23/25
12000/12000 [==============================] - 483s 40ms/step - loss: 0.1392 - val_loss: 0.1395
Epoch 24/25
12000/12000 [==============================] - 484s 40ms/step - loss: 0.1392 - val_loss: 0.1391
Epoch 25/25
12000/12000 [==============================] - 490s 41ms/step - loss: 0.1392 - val_loss: 0.1393
```

Şekil 17 - Lstm katmanı eklendikten sonra loss ve validation loss değerleri

Loss değerlerinde ve çıktı sonuçlarında bir önceki katman eklemediğimiz sonuçlar ile karşılaştırıldığında değişiklik gözlemlenmemiştir.

Modelde başarı arttırımı sağlanması için bidirectional Lstm kullanılmasına karar verilmiştir. Bu yapı ile yapılan önceki çalışmalar incelenmiştir ve bu ağların geçmişe ve gelecekteki bilgilere erişebildiği, bu nedenle çıktı tahmin ve dil çevirisinde başarıyla kullanıldığı görülmüştür. [7]

Yeni oluşturulan model çift yönlü kodlayıcı-kod çözücü mimarisine sahiptir. Kodlayıcı ve kod çözücü çift yönlü LSTM'lerdir. İleri kod çözücü, geri kodlayıcının son gizli durumu ile başlatılırken geri kod çözücü, ileri kodlayıcının son gizli durumu ile başlatılır. Modelin değişen tanımlaması şekil 18'de görülmektedir.

```
# encoder
encoder_inputs = Input(shape=(None, num_encoder_tokens))
encoder = Bidirectional(LSTM(latent_dim, return_state=True))
encoder_outputs, forward_h, forward_c, backward_h, backward_c = encoder(encoder_inputs)
state_h = Concatenate()([forward_h, backward_h])
state_c = Concatenate()([forward_c, backward_c])
encoder_states = [state_h, state_c]

# decoder
decoder_inputs = Input(shape=(None, num_decoder_tokens))
decoder_lstm = LSTM(latent_dim*2, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
decoder_dense = Dense(num_decoder_tokens, activation='softmax')
decoder_outputs = decoder_dense(decoder_outputs)
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
```

Şekil 18: Bidirectional tanımlaması

Oluşturulan bu model son halini alan 15 kategoriden oluşan data seti ile eğitilmiştir. Batch size değeri 120 ve epoch değeri 200 ile eğitildiğinde test setindeki tüm tweet örneklerinin konularını doğru tahmin etmiştir. Ancak şekil 19'da görülen loss ve validation loss değerlerine bakılınca modelin overfitting olduğu gözlemlenmektedir.

Batch size=120 Epochs=200 Num_samples=10000

```
Epoch 192/200
8000/8000 [==============================] - 327s 41ms/step - loss: 6.6021e-04 - val_loss: 0.1178
Epoch 193/200
8000/8000 [==============================] - 325s 41ms/step - loss: 9.9785e-04 - val_loss: 0.1239
Epoch 194/200
8000/8000 [==============================] - 327s 41ms/step - loss: 7.1657e-04 - val_loss: 0.1187
Epoch 195/200
8000/8000 [==============================] - 326s 41ms/step - loss: 0.0013 - val_loss: 0.1293
Epoch 196/200
8000/8000 [==============================] - 327s 41ms/step - loss: 9.4021e-04 - val_loss: 0.1161
Epoch 197/200
8000/8000 [==============================] - 325s 41ms/step - loss: 4.6581e-04 - val_loss: 0.1221
Epoch 198/200
8000/8000 [==============================] - 327s 41ms/step - loss: 0.0032 - val_loss: 0.1155
Epoch 199/200
8000/8000 [==============================] - 327s 41ms/step - loss: 5.4903e-04 - val_loss: 0.1208
Epoch 200/200
8000/8000 [==============================] - 327s 41ms/step - loss: 6.2206e-04 - val_loss: 0.1289
```

Şekil 19: 200 epoch sonucu loss ve validation loss değerleri

Overfitting sorununu çözmek için epoch sayısı daha az verilerek 2 eğitme daha yapılmıştır. Eğitimler batch size değerleri aynı kalmak üzere ilki 50 epoch ikincisi ise 75 epoch ile tamamlanmıştır.

Batch size=240
Epochs=50
Num_samples=10000

Model epoch değeri 50 ve batch size değeri 240 ile eğitildiğinde 15 farklı konu başlığından oluşan, 100 adet verinin bulunduğu test setindeki tweetlerin 67 tanesinin konusunu doğru bulmuştur. Sonuçlar incelendiğinde konu başlığının tweet içerinde geçtiği tüm örneklerin konusunu doğru bulduğu gözlemlenmiştir. Birbirleri ile alakalı vegan olmak ve hayvanseverlik konularına bazen yanlış konu ataması yaptığı görülmüştür. Tweet içerisinde kedi, köpek vb. kelimelerinin geçtiği tweetlerde hayvanseverlik konusunu bulması gerekirken hayvanseverlik ile yakından alakalı olan vegan olmak konu çıktısını verdiği gözlemlenmiştir. Eğitim sonucu %67 başarı oranı sağlanmıştır.

Batch size=240
Epochs=75
Num_samples=10000

Model epoch değeri 75 ve batch size değeri 240 ile eğitildiğinde 15 farklı konu başlığından oluşan, 100 adet verinin bulunduğu test setindeki tweetlerin 95 tanesinin konusunu doğru bulmuştur. Sonuçlar incelendiğinde konu başlığının tweet içerinde geçtiği tüm örneklerin konusunu doğru bulduğu gözlemlenmiştir. Epoch değeri 50 verildiğinde gözlemlenen yakın konu başlıklarından oluşan tweetlerin konuları yanlış bulması sorununun bu değerler ile eğitildiğinde düzeldiği görülmüştür. Eğitim sonucu %95 başarı oranı sağlanmıştır.

Epoch değerini 50'den 75'e çıkarmak başarı oranını arttırmıştır. Ancak 200 gibi daha yüksek bir değer ile eğitmek modelin overfitting olmasına neden olmuştur.

4. SONUÇLAR

Üzerinde çalıştığım projede tweetlerin konularını bulmakta en doğru sonuca ulaşmak için farklı modeler ile eğitim gerçeklenmiştir ve sonuçları karşılaştırılmıştır. Tek yönlü Lstmden oluşan encoder decoder yapısı, oluşturduğum data seti üzerinde eğitim sonucunda her tweet örneğine aynı konu çıktısını vermiştir. Epoch ve batch size değerleri değiştirilerek eğitildiğinde de aynı sorun ile karşılaşılmıştır ve testler üzerinde başarı sağlanamadığı görülmüştür.

Eğitimde başarı artışı sağlanması için modelde bidirectional Lstm kullanılmıştır ve bidirectional Lstm'in başarıda çok büyük etkisi olduğu gözlemlenmiştir. Tek yönlü Lstm ile eğitim sonucu test üzerinde hiçbir doğru sonuç alınamıyorken çift yönlü Lstm ile oluşturulan modelin eğitimi sonucu 50 epoch'ta %67 başarı sağlanmıştır. Konu başlığının tweet örneğinde geçtiği tüm testlere doğru sonuç vermiştir. En çok yanlış sonuç vegan olmak ve hayvansever konuları arasında olmuştur. Tweet örnekleri incelendiğinde iki konunun da ortak özelliği hayvanlarla ve hayvanlara olan sevgi ile ilgili örnekler olmasıdır. Konu çıktısı yanlış olmasına rağmen vermesi gereken doğru çıktıya yakın bir cevap verdiği gözlemlenmiştir.

Çift yönlü Lstm ile başarı sağlandığı görüldükten sonra epoch değerini 75'e yükselterek eğitim yapılmıştır. Bu kez 100 test örneğinin 95'inin konusunun doğru bulmuştur ve başarı oranını %95'e yükseldiği görülmüştür. 50 epoch değeri ile eğitim sonucu olan birbirine yakın olan yanlışı veren çıktı problemi de büyük oranda ortadan kalkmıştır.

Epoch değerinin sürekli olarak artması başarıyı da sürekli arttırmadığı, epoch değerine 200 verildiğinde görülmüştür. Loss ve validation loss değerlerine bakıldığında aşırı öğrenmeye işaret ettiği görülmüştür.

Çalışmanın test verisi üzerinde daha iyi çalışması amacıyla iyileştirmeler yapılabilir. Veri seti daha çok çeşitte konu ve tweetlerin bulunduğu bir veri seti ile değiştirilebilir. Model tahmin yapmak için save ve load fonksiyonlarının çağrımına uygun hale getirilerek daha fazla veri seti üzerinde eğitilebilir. Çeşitli ağlarda eğitim yapılarak daha uygun bir ağ bulunabilir veya girdiye daha temiz bir veri haline gelecek şekilde işlemler uygulanabilir. Sonraki çalışmalar için daha iyi sonuçların elde edilmesi mümkündür.

5.KAYNAKLAR

[1] Why your Neural Network is not working?
https://blog.slavv.com/37-reasons-why-your-neural-network-is-not-working-4020854bd607
[2] Neural Machine Translation—Using seq2seq with Keras.
https://towardsdatascience.com/neural-machine-translation-using-seq2seq-with-keras-c23540453c74
[3] Validation Loss vs. Training Loss
https://towardsdatascience.com/rnn-training-tips-and-tricks-2bf687e67527
[4] https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-training-upc-2016
[5] Yetersiz uyum ve aşırı uyum (Underfitting and overfitting)
https://veribilimcisi.com/2017/07/14/yetersiz-uyumunderfitting-ve-asiri-uyumoverfitting-nedir/
[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 1997.
[7] How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras
https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/