**HAVA YOLU ŞİRKETLERİNE İLİŞKİN TWITTER PAYLAŞIMLARININ SENTİMENT ANALİZİ**

Ecem Bölük

Bursa Teknik Üniversitesi

**ÖZET**

Sosyal medya uygulamaların yaygınlaşmasıyla insanlar deneyimlerini bu uygulamalardan paylaşmaya başladılar. Özellikle Twitter uygulaması bu deneyimleri paylaşmak için en yaygın olarak kullanılan uygulamalardan biri olmaya başladı. Birçok şirket, kendisi ile ilgili atılan tweetleri analiz ederek müşteri memnuniyetlerini belirlemeye başladı. Bu projede doğal dil işleme ve derin öğrenme yöntemleri kullanılarak, tweetler üzerinde duygu tahmini yapılmış, sonuçlar nedenleri ile birlikte sunulmuştur.

**Anahtar Kelimeler:** sentiment analizi, kelime gömme, derin öğrenme, doğal dil işleme

**GİRİŞ**

Duygular, deneyimlerimizi belirlememizde etkili olan önemli unsurlardan biridir. Kendi duygularımız kendi deneyimlerimizin yanında bir başka kişinin deneyimlerini de etkileyebilir. Bu sebeple birçok şirket müşteri deneyimlerini doğru analiz edip, gerekli iyileştirmeleri yapmaları gerekir. Bu analizler için çeşitli sentiment analizi modelleri kurgulanmaktadır. Bu projede Twitter’daki verilerin çok fazla sayıda olması ve herkese açık olması sebebiyle hava yolu şirketlerine atılan tweetlerden yararlanarak bir sentiment analizi modeli geliştirildi.

Doğal dil işleme, doğal dillerin çözümlenerek bilgisayarda işlenebilir hale getirilmesi işlemidir. Dil çevirileri,

sentiment analiz ve kişisel asistan uygulamaları gibi birçok alanda kullanılmaktadır. Özellike sentiment analizi konusunda pek çok araştırma yapılmıştır.

Çoban (2016), Twitter’dan elde edilen Türkçe veriler üzerinde iki farklı duygu

analizi çalışması yapmıştır. Twitter’dan elde edilen verilerin duyguların otomatik olarak tespitinde makine öğrenmesi yöntemleri kullanılmıştır. Etiketleme yöntemi Multinom Basit Bayes yöntemi ile %92,5 başarı sağlanmıştır. Çalışmada Türkçe ve İngilizce verileri işleyebilen bir yazılım örneği geliştirilmiştir.

Kaplan (2016) çalışmasında, sosyal medya sitesi Twitter kullanıcılarının paylaştığı Tweetleri analiz etmiştir. Elde edilen verilerin duyguları dört farklı grup olarak sınıflandırılmıştır. Bu sınıflar ‘Şaşkınlık’, ‘Mutluluk’, ‘Kızgınlık’, ‘Üzüntü’dür.

Verilerin sınıflandırılması için yazım hatalarının giderilmesinde ‘Zemberek’ programı kullanılmıştır. Hataları giderilen veriler karar ağacı analizinde %85.372, bulanık kural öğrenme analizinde %83.608 başarı sonuçlarına ulaşmıştır.

Demirtaş (2018) çalışmasında, yenilenen sinir ağları ve kelime gömme yöntemlerini birleştirerek geliştirdiği modelde farklı aktivasyon fonksiyonlarının başarımlarını karşılaştırmıştır. Klasik sınıflandırma algoritmaları olarak destek vektör makineleri, çoklu nominal Naive Bayes, rastgele orman, lojistik regresyon ve k-en yakın komşu algoritmaları kullanılmış ve hem kendi aralarında hem de derin öğrenme yöntemi ile karşılaştırılmıştır. Türkçe ve İngilizce veri setlerinde derin öğrenme yöntemi bazı aktivasyon fonksiyonları ile klasik sınıflandırma yöntemlerine göre daha başarılı sonuçlara ulaşmıştır.

Bu projede Twitter API yöntemiyle tweetler toplanmış. Toplanan tweetler, doğal dil işleme yöntemleri ile vektörlere dönüştürülerek derin öğrenme algoritması olarak LSTM kullanılarak modellenmiştir.

**MATERYAL VE METOT**

**1. Veri Seti**

Veriler, Twitter üzerinde Twitter API’nin sağladığı REST-API ile toplanmıştır. Python programlama dilinde bu API’yi kolay kullanabilmek için tweepy adında bir kütüphane mevcut. Bu kütüphane ile bir web servis toolu yazmadan tweetlere kolayca erişim sağlanır. Toplanan tweetler bir csv dosyasında depolanır. Duygu işaretleme işlemleri için Doccano isimli işaretleme aracı kullanılmıştır.

**2. Verinin Hazırlanması**

Veriyi derin öğrenme algoritmalarında işleyebilmek için çeşitli aşamalardan geçirmek gerekir. Bu aşamalardan ilki tweetlerin temizlenmesidir. Tweetler içerisinde bulunan noktalama işaretleri, sayılar, URL, html tagleri ve emojiler temizlenmesi için metotlar yazıldı.

|  |
| --- |
| ﻿# Remove Punctuations  def remove\_punct(text):  text = "".join([char for char in text if char not in string.punctuation])  text = re.sub('[0-9]+', '', text)  return text  # Remove Number  def remove\_numbers(text):  text = ''.join([i for i in text if not i.isdigit()])  return text |

|  |
| --- |
| ﻿# Remove URL    def remove\_URL(text):  url = re.compile(r'https?://\S+|www\.\S+')  return url.sub(r'',text)  # Remove HTML Tags    def remove\_html(text):  html=re.compile(r'<.\*?>')  return html.sub(r'',text)  ﻿# Remove Emoji    def remove\_emoji(text):  emoji\_pattern = re.compile("["  u"\U0001F600-\U0001F64F" # emoticons  u"\U0001F300-\U0001F5FF" # symbols & pictographs  u"\U0001F680-\U0001F6FF" # transport & map symbols  u"\U0001F1E0-\U0001F1FF" # flags  u"\U00002702-\U000027B0"  u"\U000024C2-\U0001F251"  "]+", flags=re.UNICODE)  return emoji\_pattern.sub(r'', text) |

Bu işlemlerden sonra kelimelerin tüm hafleri küçük harfe çevrildi. Türkçe karakterlerin küçük harfe dönüşümü sırasında oluşabilecek problemleri engellemek için bir büyük harfle yazılmış Türkçe karakterlerini, küçük harfe çeviren bir metot geliştirildi.

|  |
| --- |
| ﻿def tr\_lower(self):  self = re.sub(r"İ", "i", self)  self = re.sub(r"I", "ı", self)  self = re.sub(r"Ç", "ç", self)  self = re.sub(r"Ş", "ş", self)  self = re.sub(r"Ü", "ü", self)  self = re.sub(r"Ö", "ö", self)  self = re.sub(r"Ğ", "ğ", self)  self = self.lower()  return self |

Sözlük oluşturulurken, Türkçe karakter kullanılmaması nedeniyle aynı olan 2 kelimenin sözlük içerisinde bulunmasını engellemek amacıyla deascify işlemi uygulandı. Bu işlem için, Emre Sevinç’in Github reposunda paylaştığı Türkçe Deascify kütüphanesi kullanıldı.

|  |
| --- |
| ﻿﻿from turkishdeasciifier.turkish.deasciifier import Deasciifier  for t in tweet:  deasciifier = Deasciifier(t)  t = deasciifier.convert\_to\_turkish()  tweet[counter] = t  counter = counter +1 |

**3. Kelimelerin Kodlanması**

Kelimelerin kodlanması sürecinde kelimelerin kullanım sıklıklarından yararlanarak her kelime kullanım sıklığının interger değerini alarak kodlanmıştır. Bunun için öncelikle bir sözlük oluşturma işlemi gerçekleştirilmiş. Ardında bu sözlüğe göre tweet içerisindeki her kelime kodlanmıştır.

|  |
| --- |
| ﻿# Create Vocab  ﻿from collections import Counter  counts = Counter(word\_list)  vocab = sorted(counts, key=counts.get, reverse=True)  ﻿vocab\_to\_int = {word: ii for ii, word in enumerate(vocab, 1)}  ﻿tweets\_ints = []  for tweet in tweet\_list:  tweets\_ints.append([vocab\_to\_int[word] if word in vocab\_to\_int else 0 for word in tweet.split()]) |

LSTM algoritmasını besleyebilmek adına tweetler için bir sabit uzunluk belirlenmesi gerekmektedir. Tweetlerin kısa olması nedeniyle en uzun tweeti baz alarak tüm tweetler bu uzunlukta vektörlere çevrildi. En uzun tweetin uzunluğundan daha az olan tweetlerin olmayan kelimeleri matris içerisinde 0 ile dolduruldu.

|  |
| --- |
| ﻿#Padding sequences  def pad\_features(tweets\_ints, seq\_length):    features = np.zeros((len(tweets\_ints), seq\_length), dtype=int)  for i, row in enumerate(tweets\_ints):  features[i, -len(row):] = np.array(row)[:seq\_length]    return features  seq\_length = len(max(tokenize\_list)) # max tweet length  features = pad\_features(tweets\_ints, seq\_length=seq\_length)  assert len(features)==len(tweets\_ints)  assert len(features[0])==seq\_length |

**4. Modelin Oluşturulması**

Kelimelerin LSTM modelinde işlenmek üzere hazırlanmasından sonra modelin eğitilmesi için %80’i eğitim verisi, %20 test verisi şeklinde ikiye bölündü. Daha sonra LSTM modelimi oluşturuldu.

İlk katman olarak gömme(Embedding) katmanını oluşturuldu. Bu katmanda ilk parametre sözlükteki kelime sayısını temsil ediyor. Kelimeler 0’dan max\_feature’a kadar numaralandırılmış olduğundan parametre kelime sayısının bir fazlasını alır.

İkinci parametre kelimelerin tutulacağı vektörü oluşturur. Bu parametre hyperparameter’dır. Son parametre ise tweetlerin uzunluklarını alır.

İkinci katman LSTM katmanıdır. Bu katmanda da 14 bellek birimlik bir LSTM katmanı oluşturuldu. Son katman ise çıkış katmanı olduğundan bir adet sinir ağı bulunuyor. Pozitif ve negatif sınıflandırması yapıldığından ikili bir çıkış elde edileceğinden dolayı sigmoid aktivasyon fonksiyonunu tercih edildi.

Optimizer olarak “adam” algoritmasını kullanıldı. İkili bir çıkış sonucu elde edilecek kayıp fonksiyonu için binary\_crossentropy kullanıldı. Metrikleri de standart doğruluk metriği olarak ayarlandı.

|  |
| --- |
| ﻿# Create Model  max\_feature = len(dp.vocab) #number of word  model = Sequential()  # Embedding layer  model.add(Embedding(max\_feature+1,14,input\_length=dp.seq\_length))  # Recurrent layer  model.add(LSTM(40))  # Output layer  model.add(Dense(1,activation='sigmoid'))  # Compile the model  model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy']) |

Modelin hazırlanmasından sonra eğitim aşamasınde eğitim verilerini kullanıldı. Eğitim verilerinin LSTM üzerinde gidip, geri dönmesi işlemini 10 kez tekrarlamasını(epochs) parametre olarak eklendi. Daha sonra modelin accuracy(doğruluk) değerlerini hesaplandı. Bu model **%79 doğruluk** oranıyla tahminler yapmaktadır.

|  |
| --- |
| ﻿model.fit(train\_x, train\_y.reshape(-1,1), epochs = 10)  score,acc = model.evaluate(test\_x, test\_y.reshape(-1,1))  print("acc: %.2f" % (acc)) |

**5. Başarı Sonuçları**

Toplam 3362 tweet verisinin 2689’u eğitim, 673’ü test verisi olarak kullanılmıştır. Oluşturulan model 673 test verisinin, 529’unu doğru tahmin, 144 tanesini yanlış tahmin etmiştir. 463 tanesini pozitif duygu, 210 tanesini negatif duygu olarak tahminde bulunmuştur. 673 test verisinin içinde 475 pozitif tweet, 198 negatif tweet bulunmaktadır. X ekseni gerçek değerleri, Y ekseni tahmini değerleri göstermektedir.

|  |  |
| --- | --- |
| 397(TP) | 66(FP) |
| 78(FN) | 132(TN) |

**SONUÇLAR**

Verilerin hazırlama yöntemleri ve oluşturulan model oldukça başarılı sonuçlar elde etmiştir. Fakat bu model farklı yöntemler kullanılarak geliştirilebilir. Öncelikle daha fazla tweet sayısı modelin daha doğru sonuçlar vermesini sağlayabilir. Deascify için hazır tool kullanmak yerine daha doğru sonuçlar almak için sıfırdan tool geliştirilebilir. LSTM modeli geliştirilebilir. Kısacası oluşturulan modelde ufak değişiklikler ile daha çok başarı elde edilebilir.

**KAYNAKLAR**

[1] Çoban, Ö., (2016), “Metin Sınıflandırma Teknikleri İle Türkçe Twitter Duygu Analizi”, Atatürk Üniversitesi Fen Bilimleri Enstitüsü Yayımlanmış Yüksek Lisans Tezi, Erzurum.

[2] Kaplan, B. A., (2016), “Twitter Üzerindeki Türkçe Mesajlarda Veri Madenciliğiyle Duygu Analizi”, Beykent Üniversitesi Fen Bilimleri Enstitüsü, Yayımlanmış Yüksek Lisans Tezi, İstanbul.

[3] Demirtaş, M. A., (2018), “Derin Öğrenme Yöntemleri ile Duygu Analizi ve Aktivasyon Fonksiyonlarının Karşılaştırılması”, Atatürk Üniversitesi Fen Bilimleri Enstitüsü Yayımlanmış Yüksek Lisans Tezi, Erzurum.

[4] https://github.com/emres/turkish-deasciifier

[5] https://medium.com/@lamiae.hana/a-step-by-step-guide-on-sentiment-analysis-with-rnn-and-lstm-3a293817e314

[6] https://www.liip.ch/en/blog/sentiment-detection-with-keras-word-embeddings-and-lstm-deep-learning-networks