KOCAELİ ÜNİVERSİTESİ
Iphone 11 Ürününe Yapılan Yorumların Duygu Analizi

Ramazan KAPLANER

Bilişim Sistemleri Mühendisliği
Kocaeli Üniversitesi

181307024@kocaeli.edu.tr

Özet

Bu çalışmada, iphone 11 ürününe yapılan yorumların duygu analizini gerçekleştirmek için metin işleme tekniklerini ve yapay zekanın temellerini kullanarak, yapılan yorumların duygusal olarak ifade etmek için derin öğrenmeye dayalı modeller oluşturup bu modelleri başarıya ulaştırmak bu projenin amacıdır.

1.Giriş

İnternet ve sosyal medyanın giderek büyümesiyle markaların müşteri geri bildirimlerine erişimleri her zamankinden daha hızlı ve kolay olmaktadır. Ürün incelemeleri, sosyal medya gönderileri, forumlar, blog yazıları, anket ve chatbot verileri gibi kullanıcılar tarafından oluşturulan içerikler şirketler için müşterilerinin ürün ve hizmetleri hakkındaki görüşlerine erişmede bir altın madenidir.

Birçok kurum bu altın madenini tek bir kanalda toplamak ve analiz etmek için ölçeklenebilir, uygun maliyetli çözümler aramaktadır. Duygu analizi markaların stratejik, hızlı ve daha bilinçli pazarlama ve ürün geliştirme kararları vermelerine yardımcı olur.

Duygu analizi temel olarak bir metin işleme işlemi olup verilen metinin duygusal olarak ifade etmek istediği sınıfı belirlemeyi amaçlar. Duygu analizinin ilk çalışmaları duygusal kutupsallık olarak geçmekte olup verilen metni olumlu olumsuz ve nötr olarak sınıflandırmayı amaçlamaktadır.

2. Veri Setinin Toplanması

Duygu analizinin makine öğrenmesi modelleriyle tanınmasını amaçlayan çalışmalarda kullanılmak üzere internet ortamında açık erişime sahip birçok veri seti bulunmaktadır. Bu projede Veri setini kendimiz oluşturmak için yorumbudur.com sitesi üzerinden BeautifulSoup ve Selenium kütüphaneleri ile web scraping yaparak verileri indexler halinde indirme işlemi yapılmıstır.

2.1 Veri Seti

Projeye başlamadan önce uygun veri setinin belirlenmesi önemli bir rol almaktadır. Aynı şekilde veri setini toplamak için kullanılacak araçlar da önceden belirlenmesi önemlidir. Saha araştırmasından ve kaynak fazlalığından dolayı projede selenium ve beautifulsoup kütüphaneleri kullanmayı uygun buldum.

3. Yazılım Mimarisi

Veri madenciliği modellerini oluşturmak için birçok yazılım dili vardır ve bunların en popüleri çok fazla makine öğrenimi kütüphanesi barındırması ve açık kaynak kodlu olması sebebiyle Python'dur.

3.1 PyCharm

PyCharm, çapraz platform bir Python geliştirme ortamı'dir. Kod analizleri, grafiksel hata ayıklamacısı, versiyon kontrol sistemi ile entegre ve Django ile Python web geliştirmeleri yapılmasını sağlamaktadır.

PhpCharm kullanmamın sebebi içinde barındırdığı

araçlar ve kolay kod analizleri ile birlikte hızlı ve ayakları yere basar bir geliştirme yapmamıza olanak sağlaması.

3.2 Web scraping

Projede web scraping yapmak için selenium ve beautifulsoup kütüphaneleri kullanılmıştır. Beautiful Soup, HTML ve XML belgelerini ayrıştırmak için Selenium ise Tarayıcıya özgü bir sürücü aracılığıyla iletişim kurar ve onu kontrol etmemizi sağlıyor.

3.2.1 BeautifulSoup

Beautiful Soup, HTML ve XML belgelerini ayrıştırmak için bir Python paketidir. HTML'den veri ayıklamak için kullanılabilen ayrıştırılmış sayfalar için bir ayrıştırma ağacı oluşturur ve bu, web kazıma için yararlıdır.

3.2.2 Selenium

Selenium, tarayıcı otomasyonunu desteklemeyi amaçlayan bir dizi araç ve kitaplık için açık kaynaklı bir şemsiye projedir. Test komut dosyası dili öğrenmeye gerek kalmadan işlevsel testler yazmak için bir oynatma aracı sağlar.

3.2.3 NLTK

Natural Language Toolkit veya daha yaygın olarak NLTK, Python programlama dilinde yazılmış İngilizce için sembolik ve istatistiksel doğal dil işlemeye yönelik bir kitaplık ve program paketidir.

4. Model Açıklamaları

4.1 Yapay Sinir Ağları

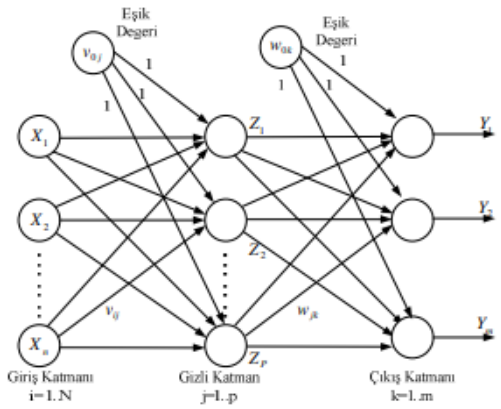
Yapay sinir ağları, biyolojik sinir ağlarından esinlenilerek ortaya çıkarılan ve biyolojik sinir ağlarına benzer bazı performans özellikleri içeren bir bilgi işleme sistemidir (Fausett,1994:3). Basit bir şekilde insan beyninin çalışma şeklini taklit eden YSA'lar veriden öğrenebilme, genelleme yapabilme, sınırsız sayıda değişkenle çalışabilme vb. birçok önemli özelliğe sahiptir.

Yapay Sinir Ağlarında Öğrenme Süreci;

Genel anlamda, bir sinir ağı birden fazla nörondan oluşur. Burada her nöron, bazı girdilere dayalı bir çıktıya ulaşmak için bir aktivasyon fonksiyonu ile doğrusal bir fonksiyonu hesaplar. Hesaplamada kullanılan aktivasyon fonksiyonu doğrusallığı bozmak için tasarlanmıştır. Hesaplanan çıktı, önem seviyesini temsil eden bir ağırlığa bağlıdır ve sonraki katmanda hesaplamalar için kullanılır. Bu hesaplamalar, nihai bir çıktıya ulaşılana kadar ağın tüm mimarisi boyunca gerçekleştirilir. Hesaplanan çıktılar, daha sonra hesaplama sürecini yeniden başlatmak ve ağın parametrelerini güncellemek için geri yayılımla kullanılırlar.

4.1.1 Çok katmanlı ileri beslemeli yapay sinir ağları (MLP)

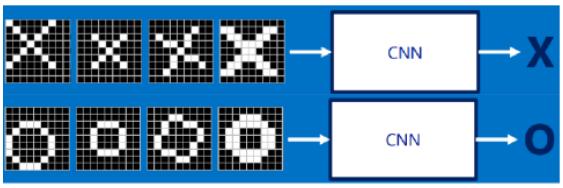
MLP ağlarında nöronlar katmanlar şeklinde organize edilmiştir. MLP'de ilk katman girdi katmanıdır. Girdi katmanı, çözülmesi istenilen probleme ilişkin bilgilerin YSA'ya alınmasını sağlar. Diğer katman ise ağ içerisinde işlenen bilginin dışarıya iletildiği çıktı katmanıdır. Girdi ve çıktı katmanlarının arasında yer alan katmana ise gizli katman adı verilir. MLP ağlarında birden fazla gizli katman da bulunabilir. Şekil 2, tipik bir MLP ağının yapısını göstermektedir.



Şekil1: Çok katmanlı ileri beslemeli yapay sinir ağı

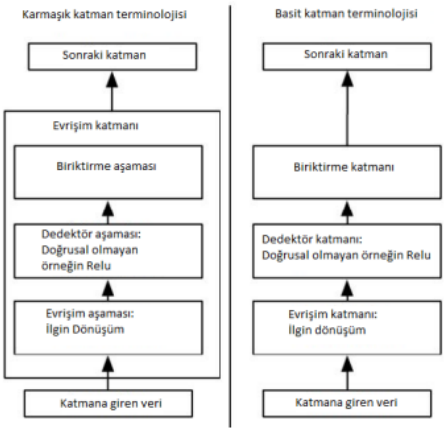
4.2 Derin Öğrenme

Evrişimli sinir ağları, sıradan sinir ağlarına çok benzer, eğitimleri sırasında bilgi öğrenebilen ağırlıkları ve bias değerleri olan nöronlardan oluşurlar. Her nöron bazı girdiler alır ve matematiksel işlemler gerçekleştirir. Tüm ağ, tek bir farklılaştırılabilir puanlama fonksiyonunu ifade eder: bir uçtaki (giriş) görüntü piksellerinin içeriğinden diğer uçtaki (çıktı) karşılık gelen sınıfı veya sonucu tanımlayan puanlayan bir ağ modelidir.



Şekil4.2: CNN girişi ve çıkışı

Bir CNN, bu katmanların her birinin fonksiyonlar yoluyla bir aktivasyon hacmini yenisine dönüştürdüğü bir katman dizisine sahip olmasıyla karakterize edilir ve bu katmanlardan birkaçı söz konusu katmanlar içindeki ve arasındaki yapılandırma parametrelerini değiştirerek kullanıldığında derin öğrenme gerçekleşir.



Şekil 4.3: Basit katmanlar ve terminolojisi

4.2.1 Evrişim Katmanı

Evrişimli süreç için her filtre boyut olarak küçüktür (genişlik ve yükseklik olarak), hatta giriş boyutunun (görüntü) toplam derinliği boyunca uzanır. Örneğin, bir ConvNet'in ilk katmanındaki tipik bir filtre 5x5x3 boyutunda olabilir (yani, 5 piksel genişliğinde ve yüksekliğinde ve renk kanalları nedeniyle 3 katman derinliğinde - RGB). İleri yayılım sırasında, her bir filtre, herhangi bir pozisyondaki filtre girişleri ile giriş arasındaki noktaları hesaplamak için giriş hacminin genişliği ve yüksekliği (eşit derinlik) boyunca kaydırılır.

4.2.2 Aktivasyon Katmanı

Bir sinir ağındaki tüm katmanların doğrusal olmaması nedeniyle, sinir ağındaki nöronların her birinin değerlerini hesapladıktan sonra, bu değerler bir aktivasyon fonksiyonundan geçirilir. Yapay bir sinir ağı temelde matrislerin çarpılması ve eklenmesinden oluşur. Sadece bu doğrusal hesaplamaları kullanıyor olsaydık, onları birbirinin üzerine istifleyebilirdik ve bu çok derin bir ağ olmazdı. Bu nedenle, doğrusal olmayan aktivasyon fonksiyonları genellikle ağın her katmanında kullanılır.

4.2.3 Havuzlama Katmanı

Havuzlama katmanı (çözünürlük azaltma katmanı olarak da bilinir), alıcı alanında daha az değer üreten bir dizi değeri birleştirir veya gruplandırır. Alıcı alanınızın boyutuna (örn. 2 x 2) ve Şekil 4.14'te gösterildiği gibi gruplama işlemine göre yapılandırılabilir. Tipik olarak havuzlama işlemi (max-pooling veya averagepooling), üst üste binmeyen belirli adımlarla kaydırılan bloklarda meydana gelir

4.2.3 Tam Bağlı (Fully-Connected) Katman

Veriler evrişim katmanlarından geçtikten sonra, sonunda yeterince küçük bir özellik haritası oluşur ve içerik tek boyutlu bir vektöre sıkıştırılır. Bu noktaya kadar, veriler sınıf tanımada belirli bir kesinlik derecesi için zaten yeterlidir. Bununla birlikte, karmaşıklık ve hassasiyet açısından daha iyi bir sonuç elde etmek için, modelin sonunda, çıkış nöronunun tüm nöronlara bağlandığı çok katmanlı sinir ağlarına benzer şekilde, tamamen bağlı bir nöral katman kullanılacaktır. Bağlantıların girdisi ve ağırlığı, geri yayılım yöntemi kullanılarak güncellenir.

4.2.4 Adam Optimizasyonu

Adam optimize edici (Adaptive Moment Estimation) derin sinir ağı eğitimi için özel olarak tasarlanmış uyarlanabilir bir öğrenme hızı optimizasyon algoritmasıdır

4.2.5 Softmax Fonksiyonu

Bu fonksiyon bir sinir ağının son katmanında kullanılan bir aktivasyon fonksiyonudur. Bu fonksiyon tam bağlı bir sinir ağının çıktısının olasılık dağılımını vermez ancak bu fonksiyonu kullanmak sinir ağının çıkışının olasılık dağılımını hesaplamaya

imkan tanır. Bu fonksiyon, her bir nöronun değerinin [0,1] arasına çekilmesini sağlayarak, nöronların sonuçlarında bir normalizasyon gerçekleştirir ve girdi görüntüsünün hangi sınıfa ait olma olasılığının belirlenmesine izin verir

5. Kod Açıklamaları

5.1 Web scraping ile verilerin indirilmesi

Gerekli kütüphaneler projeye ekliyoruz

```
import bs4
import requests
from selenium import webdriver
import os
import time
```

Chrome driverin yolunu verip çalıştırıyoruz.

```
chromePath = r'C:\Users\ramaz\Desktop\driver\chromedriver
driver = webdriver.Chrome(chromePath)

#one_way
search_URL="https://www.google.com/search?q=nature+traffi
#no_parking
```

Driverın get komutu ile verileri indirileceği adrese gitmesini sağlıyoruz

```
driver.get(search_URL)
```

Yorumbudur dan yorum çekeceğimiz için scroll önemli bir hale geliyor. Burada scrooll u aşağı doğru hareket ettirerek kodumuzun verilerin hepsini taramasını sağlıyoruz.

```
for timer in range(0,3000):
    driver.execute_script("window.scrollTo
    y += 900
    time.sleep(0.5)
```

İmport ettiğimiz bs4 kullanarak sayfaları ayrıştırıyoruz.

```
page_html = driver.page_source
pageSoup = bs4.BeautifulSoup(page_html, 'html.parser')
containers = pageSoup.findAll('div', {'class': "isv-r PNCib MSM1fd BUooTd"})
```

Ayrıştırdığımız sayfaların class bilgilerinin alıp for döngüsüne sokuyoruz.

```
for yorum in yorumlar:
    kelime=yorum.find("p",attrs
    print(kelime.text.strip())
    templist.append(kelime.text
```

Daha sonra diziye attığımız yorum bilgilerinin pandas kütüphanesi sayesinde dataframe çeviriyoruz ve csv formatı olarak yazdırıyoruz.

```
df=pd.DataFrame(templist)
df.to_csv('table2.csv', encoding="utf-8")
```

5.2 Veri Ön İşleme

Csv formatı olarak yazdırdığımız verisetini bazı ön işlemlerden geçirip temizlememiz gerekiyor.

İlk olarak veri temizleme kütüphanelerini ekliyoruz.

```
import pandas as pd
from nltk import pos_tag
from nltk.probability import FreqDist
from nltk.corpus import treebank
import nltk
nltk.download("popular")
import string
#!pip install emoji
import emoji as emj
import re
```

Csv formatındaki veri setimizi pandas kütüphanesi yardımı ile projemize dahil ediyoruz ve kolonumuza yorum ismini veriyoruz.

```
df = pd.read_csv('/content/drive/MyDrive/dataset/orjinal.csv'
turkce_sutun_isimleri = ['yorum']
df.columns = turkce_sutun_isimleri
df.head()
```

Daha sonra ön işlemede yorumları kelimelere parçalayacağımız için öncesinde birleştirmek için fonksiyon yazıyoruz.

```
def kelime_birlestir(*edatlari_temizle):
    for a in edatlari_temizle:
        kelime=" ".join(a)
    return kelime
```

Ön işlemeyi yapmaya başlıyoruz öncelikle nltk kütüphanelerini import ediyoruz ve for döngüsü oluşturuyoruz birinci adamı bütün yorumları küçük harfe çeviriyoruz.

```
from nltk.corpus import stopwords
from nltk import word_tokenize

TemizMetin=[]
pattern = r'[0-9]'

edatlar = stopwords.words('turkish')
for yorum in df.yorum:
  kucukharf=str(yorum).lower()
```

Regular Expressions uygulayarak küçük harfe çevirdiğimiz metinden sayıları kaldırıyoruz.

```
sayikaldir = re.sub(pattern, '', kucukharf)
```

Sırayla string modülünü kullanarak metinden noktalama işaretlerini çıkartıyoruz.

```
noktalama = sayikaldir.translate
```

```
(str.maketrans('','',string.punctuation))
```

Regular Expressions uygulayarak emojileri metinden kaldırıyoruz.

```
emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"
                           u"\U0001F300-\U0001F5FF"
```

```
emoji_kaldir=emoji_pattern.sub(r'', noktalama)
```

En son edatları metin den çıkartarak temizlenmiş kelimeleri birleştirme fonksiyonunu ile birleştirip diziye atıyoruz.

```
edatlari_temizle = [kelime for kelime
a=kelime_birlestir(edatlari_temizle)
TemizMetin.append(a)
```

Daha sonra diziye attığımız temizlenmiş yorum bilgilerini pandas kütüphanesi sayesinde dataframe çeviriyoruz ve csv formatı olarak yazdırıyoruz.

```
yeni_df=pd.DataFrame(TemizMetin)
yeni_df.to_csv('Temiz_Metin.csv',encoding="utf-8")
```

Eğitim verilerinin oluşacağı fonksiyon

```
def set_train_data(train_data_dir, num_classes,
                   resize_col=32, resize_row=32,
                   test_size=0.2, random_state=1):
```

Fonksiyon parametreleri;

- **num_classes:** Veri setindeki sınıf(label) sayısı
- **train_data_dir:** Eğitim verilerin bulunduğu konum
- **resize_row, resize_col:** Tüm verilerin aynı boyutta olmasını sağlamak amacıyla hepsi yeniden boyutlandırılacaktır.
- **test_size**: Validasyon verisi bölme oranı.
- **random_state**: Fonksiyon bölme işlemini gerçekleştirirken verileri veri seti içerisinden rastgele seçecektir.

Veri matrislerinin ve etiketlerin tutulması için boş bir dizi açtım.

```
data = []
labels = []
train_files = os.listdir(train_data_dir)
```

Her bir dosya yolunu dolaşacak döngümüzde dosya yolunun adı aynı zamanda sınıf numaramız olduğundan dosyadan tüm verilerin yolunu alıyoruz.

```
train_files = os.listdir(train_data_dir)
for classes in train_files:

    classname = str(classes)
    path = os.path.join(train_data_dir, classname)
    images = os.listdir(path)
```

Bütün verileri dolaşacak döngümüzde veri okunarak bir diziye aktarılıyor ve yeniden boyutlandırılıyor daha sonra ilgili dizilere ekleniyor.

```
for image in images:

   image_path = os.path.join(path, image)
   image_array = cv2.imread(image_path)
   image_array = cv2.resize(image_array, (resize_row, resize_col))
   data.append(image_array)
   labels.append(classname)
```

Dizilerin numpy dizilerine dönüştürüyoruz

```
data = np.array(data)
labels = np.array(labels)
```

Verilerinin train ve validation verisi olarak bölünmesi işlemi

```
x_train, x_val, y_train, y_val =
    train_test_split(
    data, labels, test_size=test_size,
    random_state=random_state)
```

Sınıfların kategorik matris hale getirilmesi

```
y_train = to_categorical(y_train, num_classes)
y_val = to_categorical(y_val, num_classes)
```

Test verilerinin oluşacağı fonksiyon

```
def set_train_data(train_data_dir, num_classes,
                   resize_col=32, resize_row=32,
                   test_size=0.2, random_state=1):
```

Fonksiyon parametreleri şunlardır;

- **test_data_dir:** Test verilerinin bulunduğu konum
- **train_info_dir:** Test verileriyle ilgili bilgilerin bulunduğu csv dosyasının konumu

Csv dosyasının okunması

```
y_test = pd.read_csv(test_info_dir)
```

Dosya yollarının ve sınıf bilgilerinin csv den okunması

```
images = y_test['filename'].values
```

```
y_test = y_test["class"].values
```

4.3 Model İşlemleri

4.3.1 CNN Modeli

Gerekli modülleri yüklenmesi

```
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization, Activation
from tensorflow.keras.models import Sequential
import os
```

Modeli oluşturuyoruz

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Embedding(45400,400,input_length=max_length))
model.add(tf.keras.layers.Conv1D(16,10,activation="relu"))
model.add(tf.keras.layers.MaxPooling1D(5))
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Conv1D(32,7,activation="relu"))
model.add(tf.keras.layers.MaxPooling1D(5))
model.add(tf.keras.layers.Dropout(0.4))
model.add(tf.keras.layers.Conv1D(64,5,activation="relu"))
model.add(tf.keras.layers.MaxPooling1D(2))
model.add(tf.keras.layers.Dropout(0.2))
```

Modeli oluşturup derledikten sonra detaylarını görmek için

```
model = build_model((32,32,3),13)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 activation (Activation)     (None, 32, 32, 32)        0

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )
```

Model eğitim

```
Epoch 1/5
299/299 [==============================] - 15s 20ms/step - loss: 0.7901 - accuracy: 0.7224
Epoch 2/5
299/299 [==============================] - 5s 18ms/step - loss: 0.6169 - accuracy: 0.7689
Epoch 3/5
299/299 [==============================] - 5s 18ms/step - loss: 0.3355 - accuracy: 0.8886
Epoch 4/5
299/299 [==============================] - 5s 18ms/step - loss: 0.2547 - accuracy: 0.9148
Epoch 5/5
299/299 [==============================] - 6s 19ms/step - loss: 0.2088 - accuracy: 0.9329
```

```
model.compile(optimizer = "adam",loss="categorical_
history=model.fit(X_train,y_train,epochs=5,validati
```

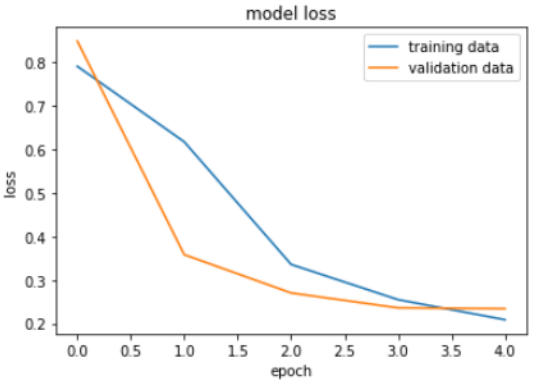
Fonksiyon parametreleri şu şekilde;

- **model**: Eğitilecek model
- **x_train, y_train**: Eğitim verileri
- **x_val, y_val**: Validasyon verileri
- **batch_size**: Her bir iterasyonda alınacak veri sayısı
- **epochs:** Eğitim iterasyon sayısı

Metrikler;

	precision	recall	f1-score	support
Negatif	0.94	0.94	0.94	216
Neutar	0.95	0.97	0.96	749
Positive	0.84	0.69	0.76	97
accuracy			0.94	1062
macro avg	0.91	0.87	0.89	1062
weighted avg	0.94	0.94	0.94	1062

Epoch/Loss oranı grafiği



4.3.2 RNN Modeli

Gerekli modülleri yüklenmesi

```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense,Dropout,SimpleRNN,
from tensorflow import keras
```

Modeli oluşturuyoruz

```
model = Sequential()

model.add(SimpleRNN(32,input_shape = (1,features.shape[1]), activation=


model.add(Dense(32, kernel_initializer='uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(64, kernel_initializer='uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(128, kernel_initializer='uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(256, kernel_initializer='uniform', activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(3, activation='softmax'))
```

Modeli oluşturup derledikten sonra detaylarını görmek için

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 32)                170336

 dense (Dense)               (None, 32)                1056

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 64)                2112

 dropout_1 (Dropout)         (None, 64)                0
```

Model eğitim

```
history=model.fit(trainX, trainY,validation_split=0.2 ,epochs=10,batc
```

```
Epoch 1/10
/usr/local/lib/python3.8/dist-packages/tensorflow/python/util/dispatc
  return dispatch_target(*args, **kwargs)
14/14 [==============================] - 5s 67ms/step - loss: 1.0404
Epoch 2/10
14/14 [==============================] - 0s 24ms/step - loss: 0.8290
Epoch 3/10
14/14 [==============================] - 0s 22ms/step - loss: 0.7537
Epoch 4/10
14/14 [==============================] - 0s 22ms/step - loss: 0.7081
Epoch 5/10
14/14 [==============================] - 0s 23ms/step - loss: 0.5673
Epoch 6/10
14/14 [==============================] - 0s 26ms/step - loss: 0.4130
```

```
model.compile(optimizer = "adam",loss="categorical_
history=model.fit(X_train,y_train,epochs=5,validati
```

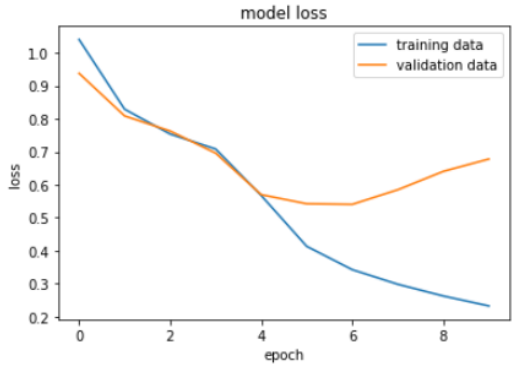
Fonksiyon parametreleri şu şekilde;

- **model**: Eğitilecek model
- **x_train, y_train**: Eğitim verileri
- **x_val, y_val**: Validasyon verileri
- **batch_size**: Her bir iterasyonda alınacak veri sayısı
- **epochs:** Eğitim iterasyon sayısı

Metrikler;

	precision	recall	f1-score	support
Negatif	0.00	0.00	0.00	203
Neutar	0.62	0.86	0.72	402
Positive	0.92	0.95	0.94	1519
accuracy			0.84	2124
macro avg	0.51	0.61	0.55	2124
eighted avg	0.78	0.84	0.81	2124

Epoch/Loss oranı grafiği



CHARLEVEL Doğruluk Oranı: 0.80

Derin Öğrenme Modelleri

CNN Doğruluk Oranı: 0.87

RNN Doğruluk Oranı: 0.61

6. Karşılaşılan Sorunlar

Projeye başlarken önce Colab üzerinden geliştirmeye başlamıştım ama selenium webdriver modülünü colab a eklerken path ve izin hataları ile karşılaştım. Bende projemi yerelde çalıştırma kararı aldım ve pycharm kullandım.

Verileri selenium ve bs4 kullanarak çekerken scroll'u aşağı indiren kodum çok hızlı çalışıyordu ve veri çekeceğim sitede veriler yüklenmeden scroll aşağı inmiş oluyordu araştırmalarım sonucu kodun üstünde değişiklik yaparak scroll'un daha yavaş bir şekilde aşağıya inmesi sağlandı.

7. Sonuçlar

Yorumbudur.com sitesinden iphone 11 ile ilgili 10.600 yorum web scraping yöntemiyle çekildi ve ön işlemeleri yapılarak işlenmeye hazır hale getirildi.

Makine Öğrenmesi Modellerinde;

Logistic Regression,

Count Vectors Doğruluk Oranı: 0.88

CHARLEVEL Doğruluk Oranı: 0.82

Naive Bayes

Count Vectors Doğruluk Oranı: 0.73

CHARLEVEL Doğruluk Oranı: 0.71

Randomforest

Count Vectors Doğruluk Oranı: 0.82

8.Kaynaklar

[1]

https://medium.com/geekculture/text-preprocessing-how-to-handle-emoji-emoticon-641bbfa6e9e7

[2]
https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/

[3]

https://medium.com/bili%C5%9Fim-hareketi/veri-bilimi-i%CC%87%C3%A7in-temel-python-k%C3%BCt%C3%BCphaneleri-2-pandas-dcc12ae01b7d

[4]

https://towardsdatascience.com/nlp-preprocessing-with-nltk-3c04ee00edc0

[5]

https://dergipark.org.tr/en/download/article-file/1440576

[6]

https://ybsansiklopedi.com/wp-content/uploads/2016/09/duygu_analizi.pdf

[7] https://www.nltk.org/

[8]

https://www.sinanerdinc.com/python-re-modulu

[9]

https://stackoverflow.com/questions/36620025/pass-array-as-argument-in-python

[10]

https://www.datasciencearth.com/dogal-dil-isleme1-5-veri-on-isleme-1/