

**عنوان پروژه:**

Sentiment Analysis

**استاد محترم:**

دکتر جعفر رزم‌آرا

**تهیه کننده:**

امید نجاتی

986321027

بهار 1402



1. **صورت مسئله پروژه**

تعیین قطبیت متون، فرآیند تحلیل نظرات، کاربران است که از مستندات موجود در یک موضوع خاص استخراج می شود (نظرات یک فیلم، توییت های توییتر، نظرات سایت آمازون و ...). هدف این پروژه، تعیین قطبیت (مثبت، منفی و خنثی) متون انگلیسی است.

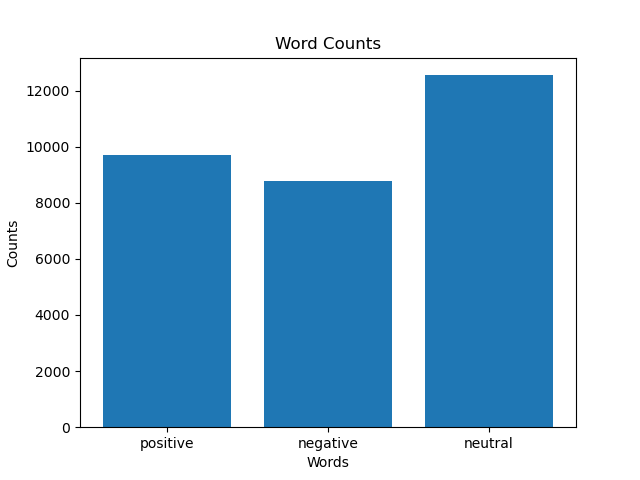
1. **الگوریتم و نحوه حل مسئله**

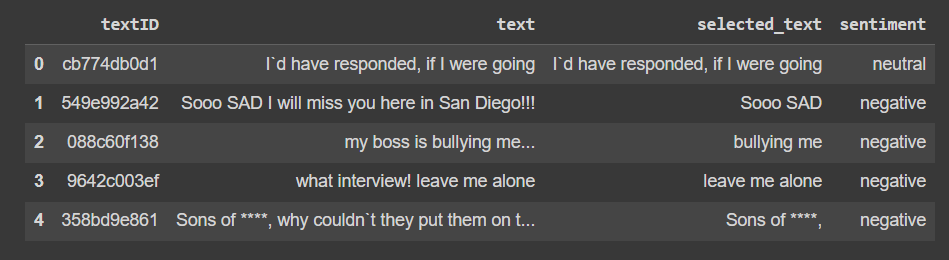
روش حل مسئله مورد استفاده در این کد را می توان به عنوان یک رویکرد یادگیری نظارت شده توصیف کرد، که در آن یک مدل یادگیری ماشینی بر روی داده های برچسب گذاری شده (بازبینی هایی که به عنوان مثبت یا منفی برچسب گذاری شده اند) آموزش داده می شود تا احساسات نظرات جدید و دیده نشده را پیش بینی کند. تکنیک‌های NLP مورد استفاده در این کد، مانند پیش‌پردازش داده‌ها و استخراج ویژگی‌ها، رویکردهای رایجی هستند که برای آماده‌سازی داده‌های متنی برای وظایف یادگیری ماشین استفاده می‌شوند.

1. **دیتاست**

با استفاده از web crawler از توییتر گردآوری شده است و پیش پردازش هایی بر روی این داده ها انجام شده و سپس در فایل به فرمت csv ذخیره شده است.

دیتاست جمع آوری شده شامل 31015 نظر می باشد. این دیتاست شامل متن نظر، برچسب های مثبت، منفی یا خنثی می باشد. برچسب های این داده ها با استفاده از امتیازات ثبت شده همراه نظرات استخراج شده است، داده‌های معرفی شده غیر متعادل بوده است و داده ها با برچسب مثبت مقداری بیشتر از منفی است و برچسب خنثی از مثبت هم بیشتر است. توزیع داده ها به صورت زیر است:





1. **کار های انجام شده**

4-1. پیش‌پردازش داده:

گام اول، پیش‌پردازش داده است. این شامل بارگیری مجموعه داده در حافظه، انتخاب ستون‌های مربوطه و تبدیل متغیرهای طبقه‌بندی شده به مقادیر عددی است. در این کد، مجموعه داده از یک فایل CSV با استفاده از کتابخانه Pandas بارگیری می‌شود. ستون‌های مربوط به تحلیل احساس (متن و احساس) انتخاب می‌شوند و برچسب‌های احساسی با استفاده از یک دیکشنری به صورت رشته‌ای (منفی=0، بی‌طرف=1، مثبت=2) تبدیل می‌شوند.

sentiment\_dict = {'negative': 0, 'neutral': 1, 'positive': 2}

tweet\_df['sentiment'] = tweet\_df['sentiment'].map(sentiment\_dict)

sentiment\_dict

{'negative': 0, 'neutral': 1, 'positive': 2}

tweet\_df["sentiment"].value\_counts()

1 12547

2 9685

0 8782

4-2. پیش‌پردازش متن:

4-2-1. توکن‌بندی داده‌های متنی:

داده‌های متنی در ستون 'text' از جدول داده به کمک کلاس Tokenizer از کتابخانه TensorFlow توکن‌بندی می‌شوند. توکن‌بندی‌کننده (Tokenizer) هر کلمه را در متن به یک مقدار عددی تبدیل می‌کند. حداکثر تعداد کلماتی که نگه داشته می‌شود را 5000 تعیین می‌کند و با استفاده از متد fit\_on\_texts()، توکن‌بندی‌کننده بر روی داده‌های متنی تنظیم می‌شود. توکن‌بندی‌کننده یک شاخص عددی منحصربفرد به هر کلمه بر اساس فراوانی تکرار آن در داده‌های متنی اختصاص می‌دهد.

4-2-2. پرکردن داده‌های متنی:

بعد از توکن‌بندی، داده‌های متنی برای اطمینان از اینکه تمام دنباله‌ها دارای طول یکسان باشند، پر می‌شوند. برای پرکردن دنباله‌ها، از تابع pad\_sequences() کتابخانه TensorFlow استفاده می‌شود. encoded\_docs که از مرحله قبل به دست آمده است، به عنوان ورودی به تابع pad\_sequences() داده می‌شود و پارامتر maxlen به مقدار 200 تنظیم می‌شود که نشان دهنده آن است که تمام دنباله‌ها به طول 200 پر یا قطع می‌شوند.

# Tokenize text data

tweet = tweet\_df.text.values

tokenizer = Tokenizer(num\_words=5000)

tokenizer.fit\_on\_texts(tweet.astype('str'))

vocab\_size = len(tokenizer.word\_index) + 1

encoded\_docs = tokenizer.texts\_to\_sequences(tweet.astype('str'))

padded\_sequence = pad\_sequences(encoded\_docs, maxlen=200)

print(tokenizer.word\_index)

{'i': 1, 'to': 2, 'the': 3, 'a': 4, 'my': 5, 'it': 6, 'you': 7, 'and': 8, 'is': 9, 'in': 10, 's': 11,…}

print(tweet[0])

print(encoded\_docs[0])

I`d have responded, if I were going

[1, 162, 19, 71, 1, 151, 49]

print(padded\_sequence[0])

[ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 162 19 71 1

151 49]

4-3. تقسیم داده:

داده به دو مجموعه آموزش و آزمون با استفاده از تابع train\_test\_split() از کتابخانه scikit-learn تقسیم می‌شود. دنباله‌های پرکرده (padded\_sequence) و برچسب‌های احساس (tweet\_df['sentiment']) به دو مجموعه آموزش و آزمون تقسیم می‌شوند، با نسبت آزمون 20٪ و یک حالت تصادفی (random state) با مقدار 42. مجموعه آموزش به train\_x و train\_y اختصاص داده می‌شود و مجموعه آزمون به test\_x و test\_y اختصاص داده می‌شود.

# Split data into training and testing sets

train\_x, test\_x, train\_y, test\_y = train\_test\_split(padded\_sequence, tweet\_df['sentiment'], test\_size=0.2, random\_state=42)

**5. تعریف مدل:**

معماری مدل LSTM با استفاده از مدل Sequential از Keras تعریف می‌شود. مدل شامل لایه Embedding، لایه SpatialDropout1D، لایه LSTM با 128 واحد و لایه Dense با 3 واحد است. لایه Embedding هر کلمه در دنباله ورودی را به یک فضای برداری بُعد بالا نگاشت می‌دهد. لایه SpatialDropout1D به صورت تصادفی بخش‌های کامل ویژگی‌های 1D را در لایه Embedding حذف می‌کند تا از اورفیتینگ جلوگیری شود. لایه LSTM نوعی از شبکه عصبی بازگشتی است که به ویژه برای داده‌های دنباله‌ای کارآمد است. لایه Dense برچسب احساس پیش‌بینی شده را برای هر ورودی خروجی می‌دهد.

# Define the model architecture

embedding\_vector\_length = 32

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_vector\_length, input\_length=200))

model.add(SpatialDropout1D(0.4))

model.add(LSTM(128, dropout=0.4, recurrent\_dropout=0.4))

model.add(Dense(3, activation='softmax'))

بطور کلی، این کد ساختار مدل تحلیل احساسات با استفاده از روش‌های یادگیری عمیق را تعریف می‌کند. از مدل Sequential در TensorFlow استفاده شده است که امکان اضافه کردن لایه‌ها به صورت پشت سر هم را فراهم می‌کند.

ساختار مدل شامل چند لایه است:

5-1. لایه Embedding: این لایه کلمات را به بردارهای چگال با اندازه ثابت تبدیل می‌کند. با نمایش کلمات در یک فضای برداری پیوسته، معنای معنایی کلمات را به خوبی درک می‌کند.

5-2. لایه SpatialDropout1D: این لایه به صورت تصادفی عناصر را از لایه قبلی حذف می‌کند تا از بیش‌برازش جلوگیری کند. با کاهش وابستگی به ویژگی‌های خاص، به تعمیم‌پذیری مدل کمک می‌کند.

5-3. لایه LSTM: لایه LSTM یک نوع لایه شبکه عصبی بازگشتی (RNN) است که به خوبی داده‌های دنباله‌ای را پردازش می‌کند. این لایه وابستگی‌های زمانی در متن را ضبط می‌کند و به درک زمینه کمک می‌کند.

5-4. لایه Dense: این لایه یک لایه کاملاً متصل است که عملیات طبقه‌بندی را انجام می‌دهد. ویژگی‌هایی که توسط لایه‌های قبلی یادگرفته شده‌اند را به کلاس‌های خروجی نگاشت می‌دهد. در این حالت، ۳ نرون که مربوط به کلاس‌های احساسی منفی، خنثی و مثبت هستند، وجود دارد. تابع فعال‌سازی softmax برای به دست آوردن احتمال برای هر کلاس استفاده می‌شود.

این ساختار مدل به مدل امکان یادگیری نمایش‌های معنادار از کلمات، درک اطلاعات زمینه‌ای و طبقه‌بندی احساسات داده‌های متنی را می‌دهد.

**6. کامپایل مدل:**

مدل با استفاده از متد compile() کامپایل می‌شود. این متد شامل تابع خطا، بهینه‌ساز و معیارهای استفاده شده در طول آموزش و ارزیابی است. در این کد، تابع خطا sparse\_categorical\_crossentropy استفاده می‌شود که برای مسائل طبقه‌بندی چند دسته‌ای مانند تحلیل احساس مناسب است. بهینه‌ساز استفاده شده Adam است که یک بهینه‌ساز محبوب برای مدل‌های یادگیری عمیق است با نرخ یادگیری 0.0001. معیارهای استفاده شده برای ارزیابی مدل، دقت است که نسبت نمونه‌های صحیحاً دسته‌بندی شده را مشخص می‌کند.

# Define the model architecture

embedding\_vector\_length = 32

model = Sequential()

model.add(Embedding(vocab\_size, embedding\_vector\_length, input\_length=200))

model.add(SpatialDropout1D(0.4))

model.add(LSTM(128, dropout=0.4, recurrent\_dropout=0.4))

model.add(Dense(3, activation='softmax'))

# Define the optimizer with a lower learning rate

optimizer = Adam(learning\_rate=0.0001)

# Compile the model with the new optimizer

model.compile(loss='sparse\_categorical\_crossentropy', optimizer=optimizer, metrics=['accuracy'])

**7. توقف زودهنگام:**

برای جلوگیری از اورفیت و بهبود توانایی تعمیم مدل، از توقف زودهنگام به عنوان یک بازخوانی در طول آموزش استفاده می‌شود. بازخوانی EarlyStopping() هنگام آموزش ضرر اعتبارسنجی را نظارت می‌کند و در صورت بهبود نداشتن ضرر اعتبارسنجی برای تعداد مشخصی از دوره‌ها (در این مورد 3) آموزش را متوقف می‌کند.

# Define early stopping callback

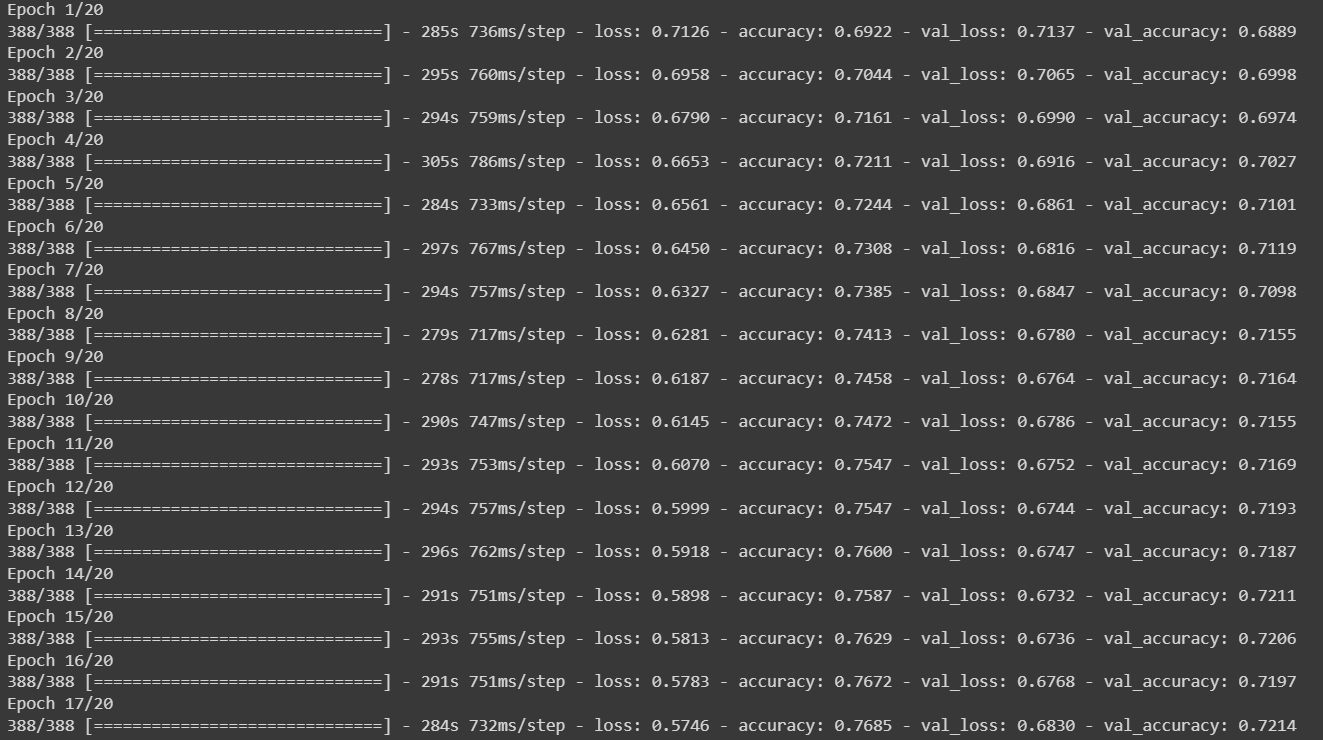
earlystop = EarlyStopping(monitor='val\_loss', min\_delta=0, patience=3, verbose=0, mode='auto')

**8. آموزش مدل:**

مدل با استفاده از متد fit() بر روی داده‌های آموزش آموزش داده می‌شود. در طول آموزش، مدل سعی می‌کند تا تابع خطا را کمینه کند، با تنظیم وزن‌های مدل با استفاده از پس‌انتشار خطا. اندازه دسته تنظیم شده برابر با 64 است، به این معنی که مدل هر بار بر روی 64 نمونه آموزش می‌بیند. تعداد دوره‌ها تنظیم شده برابر با 20 است که به این معنی است که مدل بر روی تمام مجموعه آموزش 20 بار آموزش دیده است.

# Fit the model

history = model.fit(train\_x, train\_y, validation\_data=(test\_x, test\_y), epochs=20, batch\_size=64, callbacks=[earlystop])



**9. ارزیابی مدل:**

بعد از آموزش مدل، با استفاده از متد evaluate() بر روی داده‌های آزمون، مدل ارزیابی می‌شود. این متد شامل داده‌های متنی پدینگ شده و برچسب‌های احساس مربوطه است و ضرر و دقت مدل را برمی‌گرداند. دقت نسبت نمونه‌های درستاً دسته‌بندی شده در مجموعه آزمون را نشان می‌دهد.

# Evaluate the model

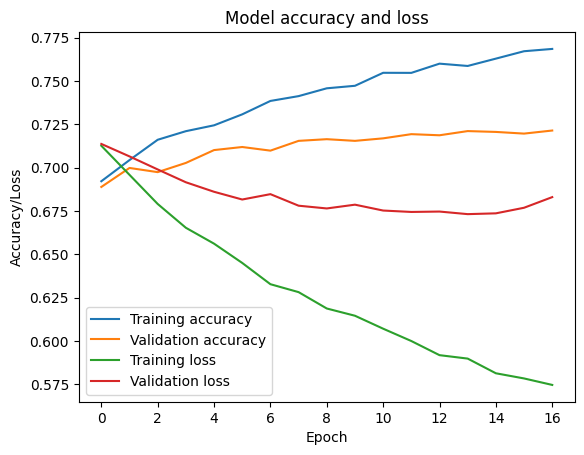
loss, accuracy = model.evaluate(test\_x, test\_y, verbose=0)

print('Accuracy: {:.2f}%'.format(accuracy\*100))

Accuracy: 72.14%

**10. نمودار کشیدن منحنی‌های دقت و ضرر:**

برای نمایش نحوه عملکرد مدل در طول آموزش، منحنی‌های دقت و ضرر با استفاده از شیء history که توسط متد fit() برگشت داده می‌شود، کشیده می‌شوند. دقت آموزش و اعتبارسنجی بر روی یک نمودار و ضرر آموزش و اعتبارسنجی بر روی یک نمودار دیگر نشان داده می‌شوند. این نمودارها به ما اجازه می‌دهند ببینیم چگونه دقت و ضرر در طول آموزش تغییر کرده‌اند و آیا مدل بیش‌برازش یا کم‌برازش است.



**11. پیش‌بینی:**

در نهایت، مدل برای پیش‌بینی داده‌های جدید و ناشناخته استفاده می‌شود. تابع predict\_sentiment() با دریافت مدل آموزش‌دیده، توکن‌ساز و متن ورودی، برچسب احساسی پیش‌بینی شده را به عنوان یک رشته (منفی=0، بی‌طرف=1، مثبت=2) برمی‌گرداند. سپس از این تابع برای دریافت متن ورودی از کاربر و پیش‌بینی برچسب احساس آن استفاده می‌شود. با پیش‌بینی احساس داده‌های جدید، می‌توانیم با استفاده از مدل تصمیماتی آگاهانه بر اساس احساس متن بگیریم، مانند اینکه آیا به یک شکایت مشتری پاسخ دهیم یا یک نقد مثبت را در رسانه‌های اجتماعی تبلیغ کنیم.

# Define function to predict sentiment of input text

def predict\_sentiment(model, tokenizer, input\_text):

    # Tokenize and pad the input text

    encoded\_text = tokenizer.texts\_to\_sequences([input\_text])

    padded\_text = pad\_sequences(encoded\_text, maxlen=200)

    # Predict the sentiment using the trained model

    sentiment\_prediction = model.predict(padded\_text)

    # Return the predicted sentiment label

    sentiment\_dict = {0: 'negative', 1: 'neutral', 2: 'positive'}

    predicted\_sentiment = sentiment\_dict[np.argmax(sentiment\_prediction)]

    return predicted\_sentiment

# Take input text from user

while True:

    input\_text = input("Enter the text to predict the sentiment: ")

    if input\_text == "":

        break

    # Predict the sentiment label of the input text

    predicted\_sentiment = predict\_sentiment(model, tokenizer, input\_text)

    # Print the predicted sentiment label

    print("Predicted sentiment: ", predicted\_sentiment)

Enter the text to predict the sentiment: Poor quality material inside. I've never had or have foot odor except with these shoes. Bought a pair for my Mom, same thing. Avias used to be great shoes. Would not buy again.

1/1 [==============================] - 0s 322ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: The shoes did not have any insoles. I received a pair of shoes that were not even completely put together.

1/1 [==============================] - 0s 49ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: I felt like this product was not made in a very sturdy way. I would in no way be able to recommend them.

1/1 [==============================] - 0s 99ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: Look nice but that is it. Everything else failed.

1/1 [==============================] - 0s 87ms/step

Predicted sentiment: neutral

Enter the text to predict the sentiment: The fit isn’t right at all and uncomfortable so I am returning. Bought usual size in sneaker.

1/1 [==============================] - 0s 56ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: Beautiful shows but be advised they run small. I wear 6.5 in all my other shoes but when I tried these, my ties hit the end. Unfortunately had to return.

1/1 [==============================] - 0s 49ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: i hate you, but i like you

1/1 [==============================] - 0s 58ms/step

Predicted sentiment: negative

Enter the text to predict the sentiment: i love you, but i don't like you

1/1 [==============================] - 0s 78ms/step

Predicted sentiment: positive

**12. کد و زبان برنامه نویسی:**

این سیستم با پایتون 3.9 پیاد سازی و اجرا شده است. دو فایل در خروجی پایتون وجود دارد که یکی به صورت script پایتون و دیگری به صورت یک فایل با پسوند .ipynb که در jupyter notebook یا google\_colab قابل اجرا است.

برای اجرای این برنامه نیاز است که موارد زیر نصب باشد:

Pip install pandas

Pip install matplotlib

Pip install numpy

Pip install tensorflow

Pip install sklearn

Pip install google.colab

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense, Dropout, Embedding, SpatialDropout1D

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.optimizers import Adam

from sklearn.model\_selection import train\_test\_split

from google.colab import drive

**12. منابع:**

<https://monkeylearn.com/sentiment-analysis/>

<https://www.qualtrics.com/experience-management/research/sentiment-analysis/>

<https://www.w3schools.com/python/python_ml_getting_started.asp>

<https://chat.openai.com/>

با تشکر فراوان از جناب آقای دکتر جعفر رزم‌آرا

که مرا در این پروژه راهنمایی کردند