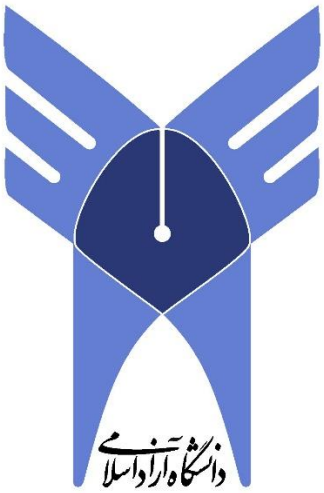

موضوع : تشخیص پیام های تبلیغاتی

ساناز الهیاری و مسعود جانفشان

استاد راهنما : دکتر میلاد سلطانی

دانشکده فنی و مهندسی، گروه کامپیوتر

هدف طراحی برنامه ایست که پیام های تبلیغاتی را از پیام های غیر تبلیغاتی تشخیص دهد

برای این برنامه از 4 مدل میتوان استفاده کرد

با تست های گرفته شده از 4 مدل گزارش زیر بدست آمده :

	accuracy	precision	recall	f1-score
MultinomialNB Model	0.962332	1.000000	0.720000	0.837209
Custom-Vec-Embedding Model	0.981166	0.970803	0.886667	0.926829
Bidirectional-LSTM Model	0.975785	0.936170	0.880000	0.907216
USE-Transfer learning Model	0.981166	0.944828	0.913333	0.928814

در چهار مدل بالا همگی از ساختار شبکه ی پرسپترون پیروی میکنند و تفاوت در ابعاد لایه ها و تعداد نورون های استفاده شده (معماری شبکه) در این مدل هاست

در جدول بالا از چهار مولفه اصلی برای مقایسه و ارزیابی مدل ها استفاده شده :

accuracy : میزان دقت هر مدل در تشخیص درست نمونه ها به نسبت کل داده ها (مثبت – مثبت و منفی – منفی)

Precision : درصد کل نمونه های واقعی مثبت / تعداد نمونه های تشخیص داده مثبت

Recall : برعکس Precision ارزیابی میکند (درصد تعداد نمونه های تشخیص داده مثبت / درصد کل نمونه های واقعی مثبت)

F1-score : در نهایت این مدل ارزیابی هم precision و هم recall را در نظر میگیرد و فرمول آن به شکل زیر است (جامع ترین معیار برای ارزیابی) F1-score = 2x (Precision*Recall)/(Precision + Recall)

❋در نهایت با محاسبه این چهار معیار برای هر مدل گزارشی مانند جدول بالا تهیه شده است ❋

الگوریتم مدل Custom-vec-embedding

این مدل از لایه های مختلف با ساختار پرسپترونی تشکیل شده :

– لایه TextVectorization برای تبدیل متن خام به یک فرمت عددی مناسب برای ورود به شبکه عصبی استفاده می‌شود. این لایه متن را توکنیزه کرده، آن را استاندارد می‌کند (با تبدیل به حروف کوچک و حذف علائم نگارش) و دنباله‌هایی از اعداد صحیح که کلمات را نمایندگی می‌کنند را خروجی می‌دهد.

– لایه Embedding مسئول یادگیری نمایندگی‌های معنایی برای کلمات در دنباله‌های ورودی است. این لایه دنباله‌های صحیح شده از TextVectorization را می‌گیرد و هر کلمه را به یک فضای برداری چگال با اندازه ثابت (در اینجا ۱۲۸) نگاشت می‌کند.

❋ دو لایه بالا بر اساس تعداد کلمات یکتا در دیتا ست ورودی و متوسط تعداد کلمات در هر پیام تنظیم میشوند

- معماری مدل شامل یک لایه ورودی، لایه TextVectorization، لایه Embedding، لایه GlobalAveragePooling1D، لایه Flatten، یک لایه چگال با فعال‌سازی ReLU و یک لایه چگال نهایی با فعال‌سازی سیگموئید برای دسته‌بندی دودویی (اسپم یا نه اسپم) است.

توضیح GlobalAveragePooling1D :

لایه GlobalAveragePooling1D یک لایه پردازش میانگین سراسری است که بر روی یک محور اعمال می‌شود. در اینجا، ما از GlobalAveragePooling1D بر روی محور طول دنباله‌های ورودی (axis=1) استفاده می‌کنیم. این لایه به ازای هر نمونه (سطر) از ورودی، میانگین مقادیر در هر بعد از دنباله را محاسبه می‌کند.

برای درک بهتر، فرض کنید که دنباله‌های ورودی شامل تعدادی بردار هستند. برای هر بردار، GlobalAveragePooling1D مقدار میانگین اعداد آن بردار را محاسبه می‌کند. این مقدار میانگین به جای یک بردار چگال، یک عدد است

مزیت اصلی از استفاده از این لایه این است که باعث کاهش ابعاد داده می‌شود. به جای این که با یک بردار چگال برای هر دنباله کار کنیم، میانگین گیری از تمام ابعاد بردارها باعث ایجاد یک بردار با ابعاد کمتر می‌شود. این کاهش ابعاد می‌تواند کمک کننده باشد تا مدل سریع‌تر آموزش ببیند و از بیش‌برازش (overfitting) جلوگیری کند.

– در نهایت لایه اخر قرار دارد ک یک نورون تعریف شده و با تابع فعالساز sigmoid تنظیم شده است (خروجی بازه ی 0 تا 1 می دهد)

از بین مدل ها Custom-Vec-Embedding را انتخاب می کنیم چون بهترین کارایی و دقت را دارد

در کد کتابخانه های زیر استفاده شده :

Numpy

Pandas

Tensorflow & Keras

sklearn

برای شروع کار یک دیتا ست به فرمت csv (به نام train) داریم که حاوی حدودا پنج هزار نمونه پیام است که برای آموزش مدل تهیه شده است

با استفاده از دستور read_csv در کتابخانه pandas فایلی که به عنوان آموزش انتخاب کردیم (train) را میخوانیم و در متغیری به اسم df میریزیم (آدرس فایل را میدهیم)

```
df = pd.read_csv("D:/Machine L projects/train.csv",encoding='latin-1')
```

برای پیش پردازش فایل ورودی ، ستون های اضافه را حذف کرده و نام ستون هایی که حاوی اطلاعات مورد نیاز برنامه هستند را به text و lable عوض میکنیم (با استفاده از دستور rename)

```
df = df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
df = df.rename(columns={'v1':'label','v2':'Text'})
```

سپس ستونی ایجاد میکنیم که پیام هایی که تبلیغاتی هستن را یک و غیر تبلیغاتی را صفر قرار میدهد

```
df['label_enc'] = df['label'].map({'ham':0,'spam':1})
```

ستون text حاوی پیام ها و ستون lable_enc که حاوی صفر و یک است را به آرایه numpy تبدیل میکنیم که به فرمت مطلوب برای ورود به مدل ، تبدیل میشه و به ترتیب در متغیر x و y میریزیم

```
X, y = np.asanyarray(df['Text']), np.asanyarray(df['label_enc'])
```

با دستور dataframe جدولی درست میکنیم با دو ستون

یک ستون با نام text که حاوی پیام ها است و ستون بعدی با نام lable که حاوی صفر و یک است

```
new_df = pd.DataFrame({'Text': X, 'label': y})
```

حال دو متغیر جدید به نام‌های X_train و y_train تعریف می‌شوند که به ترتیب برای نگهداری داده‌های متنی new_df و برچسب‌های مربوط به آن‌ها استفاده می‌شوند

```
X_train, y_train = new_df['Text'], new_df['label']
```

کلمات هر پیام را با استفاده از تابع split جدا کرده سپس تعداد کلمات را بدست می آورد و با تابع sum مجموع تعداد کلمات همه ی پیام ها را محاسبه میکند سپس تقسیم بر تعداد پیام ها میکند آن را رند میکند تا میانگین کلماتی که در پیام ها وجود دارد را بدست اورد و در متغیر avg_word_len میریزد

```
avg_words_len = round(sum([len(i.split()) for i in df['Text']])/len(df['Text']))
```

با استفاده از تابع set لیستی از کلمات یکتا درست می کنیم و تعداد کلمات را در متغیر total_words_length می ریزیم

```
s = set()
for sent in df['Text']:
    for word in sent.split():
        s.add(word)
total_words_length=len(s)
```

در اینجا تابع کمکی به اسم fit_model برای اموزش مدل طراحی شده ک پارمترهای ورودی آن ورودی و خروجی (برچسب های اموزش) یعنی X_train و y_train هستش و در پارامتر epochs تعداد تکرار اموزش را مشخص میکنیم

```
def fit_model(model, epochs, X_train=X_train, y_train=y_train,):
    '''
    fit the model with given epochs, train
    and test data
    '''
    history = model.fit(X_train,
                        y_train,
                        epochs=epochs,
                        )
    return history
```

متغیرMAXTOKENS تعیین‌کننده حداکثر تعداد کلمات مورد پذیرش است که مقدار کلمات یکتا را به آن میدهیم و OUTPUTLEN مقدار طول خروجی پیش‌بینی شده مفید است که میانگین کلمات را در آن مقدار دهی می کنیم.

```
MAXTOKENS=total_words_length
OUTPUTLEN=avg_words_len
```

پیش پردازش داده های متنی :

از کلاس TextVectorization در TensorFlow کتابخانه Keras برای ایجاد یک بردار از داده های متنی استفاده می کنیم که پیام ها به فرمتی تغییر کنند که الگوریتم‌های یادگیری عمیق بتوانند با آنها کار کنند(تبدیل به داده های عددی) که پارامتر های آن به صورت زیر است :

از max_tokens برای تعداد بیشینه توکن‌ها (کلمات) که باید برای واژه‌نامه بردارسازی در نظر گرفته شود استفاده میشود

با استفاده از پارامتر standardize='lower_and_strip_punctuation' همه متن‌ها پیش از بردارسازی یکدست می‌شوند(تمام حروف به حروف کوچک تبدیل شده و علائم نگارشی حذف می‌شوند).

پارامتر output_mode نحوه خروجی دادن داده‌های بردارسازی شده را مشخص می‌کند. که در اینجا int به این معناست که خروجی برای هر کلمه عدد صحیح منحصر به فردی خواهد بود

پارامتر output_sequence_length طول ثابت خروجی هر بردار را تنظیم می‌کند. اگر متن طولانی‌تر از OUTPUTLEN باشد، برش خواهد خورد و اگر کوتاه‌تر باشد، با صفر پر خواهد شد تا به این طول برسد

در نهایت text_vec.adapt(X_train) روی مجموعه داده آموزشی X_train عمل می‌کند تا لایه بردارسازی را برای تبدیل دقیق و کارآمد داده‌های متنی آینده آماده سازد. بعد از اجرای این متد، لایه TextVectorization می‌تواند متون جدید را با استفاده از همین واژه‌نامه بردارسازی کند، که به آن کمک می‌کند هر کلمه‌ای را به معادل عددی‌اش (که یک اندیس است) تبدیل کند.

```
text_vec = TextVectorization(
    max_tokens=MAXTOKENS,
    standardize='lower_and_strip_punctuation',
    output_mode='int',
    output_sequence_length=OUTPUTLEN
)
text_vec.adapt(X_train)
```

لایه‌های Embedding معمولاً به کار می‌روند تا داده‌های متنی که به صورت اندیس‌های عددی هستند، را به بردارهای چند بعدی تبدیل کنند. این می‌تواند به مدل کمک کند تا مفاهیم پیچیده و رابطه‌های معنایی بین کلمات را بهتر یاد بگیرد

با استفاده از این لایه ارزش ویژگی ها را بدست می آوریم که ببینیم کدام ویژگی ها برای تشخیص تبلیغاتی بودن یا نبود مناسب است

اول یک نمونه جدید از کلاس Embedding را ایجاد می‌کنیم و نتیجه را در یک متغیر به نام embedding_layer قرار می‌دهیم

پارامتر input_dim مشخص می‌کند که تعداد کل کلماتی که لایه Embedding باید پشتیبانی کند چقدر است. در این مورد، به تعداد مشخص شده توسط متغیر MAXTOKENS (کلمات یکتا) اشاره دارد

output_dim تعداد بعد برداری است که هر توکن به آن تبدیل می‌شود. هر کلمه در نمایش برداری به یک بردار 128 بعدی تبدیل خواهد شد.

embeddings_initializer تعیین می‌کند که وزن‌های اولیه لایه Embedding چگونه مقداردهی شوند. uniform بدین معنی است که وزن‌ها از یک توزیع یکنواخت در یک بازه مشخص، به طور تصادفی انتخاب می‌شوند.

input_length طول دنباله‌های ورودی را مشخص می‌کند که باید برابر با output_sequence_length باشد تا بتوان آن‌ها را پردازش کرد

```
embedding_layer = layers.Embedding(
    input_dim=MAXTOKENS,
    output_dim=128,
    embeddings_initializer='uniform',
    input_length=OUTPUTLEN )
```

حال یک مدل از شبکه عصبی به نام Custom-Vec-Embedding را با استفاده از کتابخانه Keras در TensorFlow ایجاد می‌کنیم

ابتدا یک لایه ورودی تعریف می‌کنیم shape=(1,) نشان می‌دهد که هر ورودی دقیقاً یک متن است و dtype=tf.string مشخص می‌کند که نوع داده‌های ورودی، رشته‌های متنی هستند.

text_vec که قبلاً تعریف شده بود را روی لایه ورودی اعمال می‌کند تا ورودی از نوع متنی به اندیس‌های عددی تبدیل شود.

سپس vec_layer بر روی خروجی لایه بردارسازی اعمال می‌شود. این خروجی حاوی اندیس‌های عددی است که حالا به بردارهای پیچیده‌تر تبدیل می‌شوند.

و با عبور بردار های بدست آمده از لایه Global Average Pooling 1D ابعاد بردار ها را کاهش میدهیم(میانگین بردارها را در هر اندیس محاسبه می کنیم) و در متغیر x میریزیم سپس لایه Flatten آن را از ابعاد دوبعدی به یک بردار یک بعدی تبدیل می‌کند تا بتواند به عنوان ورودی به لایه‌های بعدی انتقال داده شود متغیر x آپدیت میشود و در آخر از یک لایه تمام اتصال(layers.Dense) با 32 نورون و تابع فعال‌سازی ReLU عبور می کند

لایه خروجی، که تمام اتصال است، با یک نورون معرفی می‌شود و از تابع فعال‌سازی سیگموید استفاده می کنیم که برای سنجش خروجی‌های دودویی به کار می‌رود خروجی بین صفر و یک است اگر به صفر نزدیک تر باشد صفر در غیر این صورت یک بدست می آید.

```
input_layer = layers.Input(shape=(1,), dtype=tf.string)
vec_layer = text_vec(input_layer)
embedding_layer_model = embedding_layer(vec_layer)
x = layers.GlobalAveragePooling1D()(embedding_layer_model)
x = layers.Flatten()(x)
x = layers.Dense(32, activation='relu')(x)
output_layer = layers.Dense(1, activation='sigmoid')(x)
```

در نهایت یک مدل Keras جدید با ورودی و خروجی تعیین شده‌ای ایجاد می‌کنیم

و مدل را کامپایل می کنیم این کار محیط مدل را برای آموزش آماده می‌کند.

پارامتر `optimizer` تعیین می‌کند که چه الگوریتم بهینه‌سازی برای تغییر وزن‌ها و بیاس‌ها در فرایند آموزش به کاربرده شود.

loss تابع زیانی است که در طی آموزش سعی می‌شود کمینه شود(میزان عدم تطابق بین برچسب‌های پیش‌بینی شده توسط مدل و برچسب‌های واقعی)

label_smoothing=0.5 یک تکنیک است که به هدف بهبود روند تعمیم‌پذیری مدل، سخت‌گیری مدل نسبت به برچسب‌های داده آموزشی را کاهش می‌دهد.

در پایان ، مدل با استفاده از داده‌های آموزشی X_train و برچسب‌های y_train برای تعداد 5 دوره (epoch) آموزش داده می‌شود. نتایج آموزش در متغیر history_1 ذخیره می‌شوند

```
model_1 = keras.Model(input_layer, output_layer)

model_1.compile(optimizer='adam', loss=keras.losses.BinaryCrossentropy(
    label_smoothing=0.5))

history_1 =model_1.fit(X_train, y_train, epochs=5)
```

تا اینجا به کد آموزش های لازم را داده ایم و آماده دریافت ورودی است

از کاربر میخواهیم تصمیم بگیرد که تعدادی پیام را در قالب فایل به عنوان ورودی میدهد (ds) یا یک پیام را(ex)

اگر کاربر ex را انتخاب کرد وارد تابع user_exaple() میشویم و اگر dsرا انتخاب کرد وارد تابع dataset_result() میشویم و اگر غیر از این دو را وارد کرد دوباره سوال را از او بپرسد

```
while True  :
    us = input("Do you want the results of your dataset or do you have another
example? [ex/ds]")
    if us=="ex" :
        user_example()
        break

    elif us=="ds" :
        dataset_result()
        break
    else:
        print("Please enter either ex for an example or ds for a dataset.")
```

تابع user_example()

در اینجا، z به عنوان ورودی از کاربر گرفته شده و سپس به یک لیست از متن‌ها تبدیل شده است. سپس از این لیست برای ساخت دیتافریم استفاده می‌شود و در نهایت مدل بر روی داده‌های تست پیش‌بینی می‌شود.

در نتیجه، پس از اجرای این خط کد، تمام داده‌ها در Z_test نگه داشته می‌شود و قابل استفاده برای ارزیابی مدل یا پیش‌بینی در آینده است.

پیام را وارد مدل ساخته شده می کنیم اگر پیام تبلیغاتی باشد خروجی spam در غیر این صورت ham است

```
def user_example():
    z=(input("Mesal khod ra vared konid :"))
    text_list = [z]
    df_user = pd.DataFrame({'Text': text_list})
    Z_test = df_user['Text']
    p = model_1.predict(Z_test)
    if np.any(np.round(p)==1):
         print('spam')
    else:
         print ('ham')
```

تابع dataset_result()

از کاربر می خواهیم آدرس فایل مورد نیاز را وارد کند و با استفاده از دستور read آن را میخوانیم

برای پیش پردازش فایل ورودی ، ستون های اضافه را حذف کرده و نام ستونی را که حاوی اطلاعات مورد نیاز برنامه است را به text عوض میکنیم (با استفاده از دستور rename)

ستون text حاوی پیام ها است را به آرایه numpy تبدیل میکنیم که به فرمت مطلوب برای ورود به مدل ، تبدیل میشود و در متغیر u میریزیم سپس با دستور dataframe جدولی درست میکنیم با یک ستون به نام text که حاوی پیام های مورد نظر است

متغیر جدیدی به نام X_test تعریف می‌شوند که برای نگهداری داده‌های متنی new_df2 استفاده میشود

پیام ها را وارد مدل ساخته شده می کنیم اگر پیام تبلیغاتی باشد خروجی یک در غیر این صورت صفر است

```
def dataset_result():
    df_2 = pd.read_csv(input("Adress file ra vared konid :"),encoding='latin-1')
    #D:/Machine L projects/test.csv

    df_2 = df_2.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
    df_2 = df_2.rename(columns={'v2':'Text'})

    U = np.asanyarray(df_2['Text'])
    new_df_2 = pd.DataFrame({'Text': U})
    X_test = new_df_2['Text']

    predictions = model_1.predict(X_test)

    result_df = pd.DataFrame({
        'Text': df_2['Text'],
        'Prediction': np.round(predictions).flatten()
    })
    result_df['Prediction'] = result_df['Prediction'].replace({0: 'Not Spam ', 1:
'Spam'})

    result_df.to_csv('output.csv', index=False)
```

در نهایت در ذخیره سورس فایلی به فرمت csv ایجاد میشود که شامل دو ستون Text و Prediction (پیش بینی) است و اگر خروجی مدل 0 باشد Not spam و اگر 1 باید Spam به ما نمایش داده میشود (درستون prediction)

منابع

geeksforgeeks.org

kaggle.com

chatgpt.open.ai

stackoverflow.com