Authors: Suraj Janakiraman, Dmitrii Obideiko

Date: December 4, 2022

Course: CS 4395.001- Human Language Technologies

Assignment: Text Classification

Test Classification Data

For our test classification data, we used women’s clothing E-Commerce Reviews.

This data contains:

- Clothing ID
- Age
- Title
- Review Text
- Rating
- Recommended IND
- Positive Feedback Count
- Division Name
- Department Name
- Class Name

We are planning to use this data to predict ratings using review text.

To download the data go this website: https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews

```
import pandas as pd
df=pd.read_csv('Womens Clothing E-Commerce Reviews.csv')
```

```
df=df.dropna()
```

```
df['ReviewText']=df['Review Text']
```

```
df
```

```
df
```

	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count
2	2	1077	60	Some major design flaws	I had such high hopes for this dress	3	0	0

```
df['Rating']=df['Rating'].astype("category")
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras import layers, models
from sklearn.preprocessing import LabelEncoder
import pickle
import numpy as np
import pandas as pd



# set seed for reproducibility
np.random.seed(1234)
```

```
#split the dataframe into train and test
i=np.random.rand(len(df)) <0.8
train = df[i]
test  = df[~i]
print("Train data size: ",train.shape)
print("Test data size: ",test.shape)
```

```
     Train data size:  (15753, 12)
     Test data size:  (3909, 12)
```

```
test
```

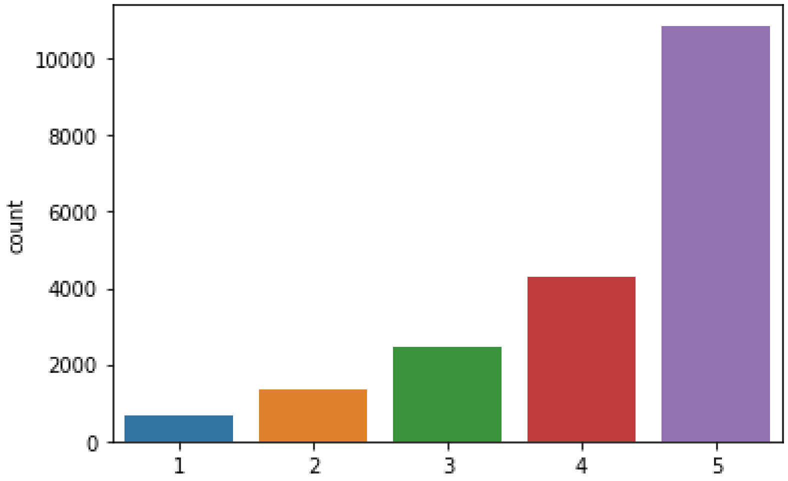
	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positiv Feedbac Coun
9	9	1077	34	Such a fun dress!	I'm 5"5' and 125 lbs. i ordered the s petite t...	5	1	0
10	10	1077	53	Dress looks like it's made of cheap material	Dress runs small esp where the zipper area run...	3	0	14
12	12	1095	53	Perfect!!!	More and more i find myself reliant on the rev...	5	1	2
22	22	1077	31	Not what it looks like	First of all, this is not pullover styling. th...	2	0	7
27	27	1003	31	Loved, but returned	The colors weren't what i expected either.	4	1	0

Create the Graphs

```
import seaborn as sb
```

```
import seaborn as sns
sns.countplot(x=df["Rating"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f09b896fdc0>
```



2. Sequential Learning via Keras

```
# set up X and Y
num_labels = 2
vocab_size = 25000
batch_size = 100

# fit the tokenizer on the training data
tokenizer = Tokenizer(num_words=vocab_size)


tokenizer.fit_on_texts(train.ReviewText)

x_train = tokenizer.texts_to_matrix(train.ReviewText, mode='tfidf')
x_test = tokenizer.texts_to_matrix(test.ReviewText, mode='tfidf')

encoder = LabelEncoder()
encoder.fit(train.Rating)
y_train = encoder.transform(train.Rating)


y_test = encoder.transform(test.Rating)

# check shape
print("train shapes:", x_train.shape, y_train.shape)
print("test shapes:", x_test.shape, y_test.shape)
print("test first five labels:", y_test[:5])
```

```
    train shapes: (15753, 25000) (15753,)
    test shapes: (3909, 25000) (3909,)
    test first five labels: [4 2 4 1 3]
```

```
#fit the model
model=models.Sequential()
```

```
model.add(layers.Dense(32, input_dim=vocab_size, kernel_initializer='normal', activati
model.add(layers.Dense(1, kernel_initializer='normal', activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history= model.fit(x_train, y_train, batch_size=batch_size, epochs=30, verbose=1, vali
```

```
Epoch 1/30
142/142 [==============================] - 4s 22ms/step - loss: -76.8452 - accu
Epoch 2/30
142/142 [==============================] - 3s 19ms/step - loss: -625.3054 - acc
Epoch 3/30
142/142 [==============================] - 3s 19ms/step - loss: -1854.4543 - ac
Epoch 4/30
142/142 [==============================] - 3s 19ms/step - loss: -3733.6201 - ac
Epoch 5/30
142/142 [==============================] - 3s 20ms/step - loss: -6210.4067 - ac
Epoch 6/30
142/142 [==============================] - 3s 19ms/step - loss: -9252.7832 - ac
Epoch 7/30
142/142 [==============================] - 3s 21ms/step - loss: -12833.1787 - a
Epoch 8/30
142/142 [==============================] - 3s 20ms/step - loss: -16925.3828 - a
Epoch 9/30
142/142 [==============================] - 3s 20ms/step - loss: -21509.3730 - a
Epoch 10/30
142/142 [==============================] - 3s 20ms/step - loss: -26557.4688 - a
Epoch 11/30
142/142 [==============================] - 3s 19ms/step - loss: -32051.8945 - a
Epoch 12/30
142/142 [==============================] - 3s 19ms/step - loss: -37977.6680 - a
Epoch 13/30
142/142 [==============================] - 3s 19ms/step - loss: -44317.5391 - a
Epoch 14/30
142/142 [==============================] - 3s 20ms/step - loss: -51057.6992 - a
Epoch 15/30
142/142 [==============================] - 3s 19ms/step - loss: -58183.7422 - a
Epoch 16/30
142/142 [==============================] - 3s 19ms/step - loss: -65688.2344 - a
Epoch 17/30
142/142 [==============================] - 3s 19ms/step - loss: -73556.9766 - a
Epoch 18/30
142/142 [==============================] - 3s 19ms/step - loss: -81781.1328 - a
Epoch 19/30
142/142 [==============================] - 3s 19ms/step - loss: -90354.1797 - a
Epoch 20/30
142/142 [==============================] - 3s 19ms/step - loss: -99263.4531 - a
Epoch 21/30
142/142 [==============================] - 3s 19ms/step - loss: -108507.9922 -
Epoch 22/30
142/142 [==============================] - 3s 20ms/step - loss: -118080.4219 -
Epoch 23/30
142/142 [==============================] - 3s 19ms/step - loss: -127971.3125 -
Epoch 24/30
142/142 [==============================] - 3s 19ms/step - loss: -138177.7188 -
```

```
Epoch 25/30
142/142 [==============================] - 3s 20ms/step - loss: -148692.1250 -
Epoch 26/30
142/142 [==============================] - 3s 20ms/step - loss: -159512.0000 -
Epoch 27/30
142/142 [==============================] - 3s 22ms/step - loss: -170630.4531 -
Epoch 28/30
142/142 [==============================] - 3s 20ms/step - loss: -182049.3906 -
Epoch 29/30
```

```
#evaluate
score=model.evaluate(x_test, y_test, batch_size=batch_size, verbose=1)
print('Accuracy: ', score[1])
```

```
40/40 [==============================] - 0s 8ms/step - loss: -213223.8125 - accu
Accuracy:  0.06625735759735107
```

```
print(score)
```

```
[-213223.8125, 0.06625735759735107]
```

```
# get predictions so we can calculate more metrics
pred=model.predict(x_test)
pred_labels=[1 if p>0.5 else 0 for p in pred]
```

```
123/123 [==============================] - 1s 4ms/step
```

```
pred[:10]
```

```
array([[1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.],
       [1.]], dtype=float32)
```

```
pred_labels[:10]
```

```
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
print('accuracy score: ', accuracy_score(y_test, pred_labels))
print('precision score: ', precision_score(y_test, pred_labels, average='micro'))
print('recall score: ', recall_score(y_test, pred_labels, average='micro'))
print('f1 score: ', f1_score(y_test, pred_labels, average='micro'))
```

```
accuracy score:  0.06625735482220517
precision score:  0.06625735482220517
recall score:  0.06625735482220517
f1 score:  0.06625735482220517
```

Performace

Performace seems to be low as we received an accurace score of .066. Therefore, we cannot use Sequential Learning via Keras to predict soemthing accuratly with the data that we chose.

3. Recurrent Neural Network (RNN)

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()

# RNN
import tensorflow as tf
from tensorflow.keras import datasets, layers, models, preprocessing

max_features = 10000
maxlen = 500
batch_size = 32

trainText = vectorizer.fit_transform(train["Review Text"]).todense()
testText =  vectorizer.fit_transform(test["Review Text"]).todense()

train_data = preprocessing.sequence.pad_sequences(trainText, maxlen=maxlen)
test_data=preprocessing.sequence.pad_sequences(testText, maxlen=maxlen)

train_data.shape

model = models.Sequential()
model.add(layers.Embedding(max_features, 128, input_length=maxlen))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.MaxPooling1D(5))
model.add(layers.Conv1D(32, 7, activation='relu'))
model.add(layers.GlobalMaxPooling1D())
model.add(layers.Dense(1))

model.summary()


# compile

model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=1e-4),  # set learning rate
              loss='binary_crossentropy',
              metrics=['accuracy'])


# train

history = model.fit(train_data,
                    train_labels,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)

from sklearn.metrics import classification_report

pred = model.predict(test_data)
pred = [1.0 if p>= 0.5 else 0.0 for p in pred]
print(classification_report(test_labels, pred))
```

Performace

We did run out of ram and therefore could not see the results of RNN. It is an indacator of bad performance. Therefore, we cannot use this neural network to make any predictions with our data.

Double-click (or enter) to edit