```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# from wordcloud import WordCloud

import nltk
nltk.download("punkt")
nltk.download("wordnet")
nltk.download("stopwords")

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence #unique id

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, Dropout, Embedding
import warnings
warnings.filterwarnings("ignore")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
df = pd.read_csv('/content/tweet_emotions.csv')
```

```
df.head()
```

	tweet_id	sentiment	content
0	1956967341	empty	@tiffanylue i know i was listenin to bad habi...
1	1956967666	sadness	Layin n bed with a headache ughhhh...waitin o...
2	1956967696	sadness	Funeral ceremony...gloomy friday...
3	1956967789	enthusiasm	wants to hang out with friends SOON!
4	1956968416	neutral	@dannycastillo We want to trade with someone w...

```
df.isnull().sum()
```

```
tweet_id     0
sentiment    0
content      0
dtype: int64
```

```
df['sentiment'].value_counts()
```

```
neutral       8638
worry         8459
happiness     5209
sadness       5165
love          3842
surprise      2187
fun           1776
relief        1526
hate          1323
empty          827
enthusiasm     759
boredom        179
anger          110
Name: sentiment, dtype: int64
```

```
len(df['sentiment'].value_counts())
```

```
13
```

```
def cleantext(text):
  tokens = word_tokenize(text.lower())
  ftoken = [t for t in tokens if(t.isalpha())]
  stop = stopwords.words("english")
```

```
    ctoken = [t for t in ftoken if(t not in stop)]
    lemma = WordNetLemmatizer()
    ltoken = [lemma.lemmatize(t) for t in ctoken]
    return " ".join(ltoken)


df['content']=df['content'].apply(cleantext)


sentlen = []

for sent in df["content"]:
  sentlen.append(len(word_tokenize(sent)))

df["SentLen"] = sentlen
df.head()
```

	tweet_id	sentiment	content	SentLen
0	1956967341	empty	tiffanylue know listenin bad habit earlier sta...	9
1	1956967666	sadness	layin n bed headache ughhhh waitin call	7
2	1956967696	sadness	funeral ceremony gloomy friday	4
3	1956967789	enthusiasm	want hang friend soon	4
4	1956968416	neutral	dannycastillo want trade someone houston ticke...	7

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
x = df['content']
y = le.fit_transform(df['sentiment'])


np.quantile(sentlen, 0.95)
```

```
14.0
```

```
max_len = np.quantile(sentlen, 0.95)


max(df['SentLen'])
```

```
25
```

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.30,random_state=1)


tok = Tokenizer(char_level=False, split=" ")
tok.fit_on_texts(xtrain)


vocab_len = len(tok.index_word)
vocab_len
```

```
29955
```

```
seqtrain = tok.texts_to_sequences(xtrain) #step1
seqtrain
```

```
[[9752, 417, 16, 264, 316],
 [4, 496, 3514, 9753, 9754, 88, 453, 26, 1355, 1539, 3123],
 [9755, 9756, 1540, 9757, 1474, 2219, 931, 9758, 46],
 [23, 6, 9759, 105, 256, 257, 286, 9760, 13],
 [785, 204, 353, 69, 3124, 163, 3, 16],
 [37, 176],
 [9761, 338, 144, 1475, 3515, 197, 412, 988],
 [631, 38, 173],
 [211, 473, 6426, 4966],
 [11, 231, 786, 407, 11, 4967, 9762, 26, 1356, 1, 9763, 9764, 1933, 1933],
 [9765, 167, 4, 300],
 [9766, 90, 655, 260, 418, 419, 4],
 [4968, 9767, 1289, 1243, 2819, 1619, 12],
 [508, 64, 157, 27],
 [1021],
 [153, 41, 4969],
 [32, 420, 44],
 [294],
 [508, 4970],
 [1541, 549, 71, 51],
 [1712, 1244, 2061, 1713, 370, 25, 273, 606, 293, 1357, 2820, 2377, 1542],
 [160, 4096, 509, 4971, 519, 9768, 326, 434, 598, 474, 3, 132],
 [556, 227, 787, 46, 267, 689],
 [86, 4972, 9769, 247],
```

```
        [9, 902, 233, 4973, 4097, 68],
        [802, 77, 154, 4098, 334, 989, 201, 156, 10],
        [1620, 9770, 9771, 14, 234, 1290, 823, 58, 903, 238, 4974],
        [79, 122, 21, 132, 932, 118],
        [15, 22, 4099, 9, 9772],
        [9773, 29, 155, 9, 9774, 885, 4975, 4976, 733, 42, 1476],
        [9775, 489, 284, 1],
        [1046, 677, 607, 2821, 271, 139],
        [42, 599, 413, 6427, 8],
        [37],
        [2062, 239, 4977, 9, 49, 53, 425],
        [9776, 228, 1815, 39, 497, 267, 3125, 3516],
        [177, 222, 158, 20, 168],
        [134, 4978, 145, 253, 223, 52],
        [170, 51, 375, 3126, 10, 339, 93, 2220, 845, 144, 1477],
        [154, 4, 902, 656, 41, 1358, 1816, 12],
        [16, 56, 5, 2378, 5, 157, 128],
        [9777, 265, 1934, 3127, 144, 146, 933, 1714, 51, 1, 378, 191],
        [9778, 586, 30, 145, 103],
        [17, 46, 9779, 17, 21, 371, 4979, 9780],
        [119, 18, 4100, 9, 283, 122, 35, 1715],
        [1047, 9781, 2, 56, 644, 2063],
        [15, 22, 1, 317],
        [9782, 33, 532],
        [111, 34, 2, 379, 9783, 79, 9784, 4980],
        [308, 303, 82],
        [1245, 271],
        [308, 241, 231, 788],
        [9785, 111, 34, 61, 6, 6428, 198],
        [1291, 6429, 9786, 28, 1543, 1292, 745, 4981, 336, 103],
        [9787, 9788, 2822, 3517, 236, 6430],
        [556, 72, 28, 318, 14, 16, 1817],
        [789, 2823, 2824, 1048, 600, 4101, 1716, 7, 525, 600, 4101, 9789, 17, 4, 139],
        [533, 150],
```

```
seqmattrain = sequence.pad_sequences(seqtrain, maxlen= int(max_len)) #step2
seqmattrain
```

```
array([[    0,     0,     0, ...,    16,   264,   316],
       [    0,     0,     0, ...,  1355,  1539,  3123],
       [    0,     0,     0, ...,   931,  9758,    46],
       ...,
       [    0,     0,     0, ...,  3464,    99,   322],
       [    0,     0,     0, ...,  6075,   390,   710],
       [    0,     0,     0, ...,    95,  1801, 29955]], dtype=int32)
```

```
seqtest = tok.texts_to_sequences(xtest)
seqmattest = sequence.pad_sequences(seqtest, maxlen=int(max_len))
```

```
rnn = Sequential()

rnn.add(Embedding(vocab_len+1,25, input_length=int(max_len), mask_zero=True))
rnn.add(SimpleRNN(units=32, activation="tanh"))
rnn.add(Dense(units=32, activation="relu"))
rnn.add(Dropout(0.2))

rnn.add(Dense(units=13, activation="softmax"))

rnn.compile(optimizer="adam", loss="sparse_categorical_crossentropy")

rnn.fit(seqmattrain, ytrain, batch_size=50, epochs=25)

ypred = rnn.predict(seqmattest)
```

```
Epoch 1/25
560/560 [==============================] - 12s 19ms/step - loss: 2.1997
Epoch 2/25
560/560 [==============================] - 11s 20ms/step - loss: 1.8659
Epoch 3/25
560/560 [==============================] - 11s 20ms/step - loss: 1.3168
Epoch 4/25
560/560 [==============================] - 11s 19ms/step - loss: 0.8749
Epoch 5/25
560/560 [==============================] - 11s 20ms/step - loss: 0.6261
Epoch 6/25
560/560 [==============================] - 11s 19ms/step - loss: 0.4809
Epoch 7/25
560/560 [==============================] - 11s 19ms/step - loss: 0.3875
Epoch 8/25
560/560 [==============================] - 11s 19ms/step - loss: 0.3223
Epoch 9/25
560/560 [==============================] - 11s 20ms/step - loss: 0.2905
Epoch 10/25
560/560 [==============================] - 11s 20ms/step - loss: 0.2513
```

```
Epoch 11/25
560/560 [==============================] - 11s 19ms/step - loss: 0.2309
Epoch 12/25
560/560 [==============================] - 11s 19ms/step - loss: 0.2070
Epoch 13/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1931
Epoch 14/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1790
Epoch 15/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1659
Epoch 16/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1557
Epoch 17/25
560/560 [==============================] - 10s 18ms/step - loss: 0.1481
Epoch 18/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1417
Epoch 19/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1354
Epoch 20/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1254
Epoch 21/25
560/560 [==============================] - 11s 20ms/step - loss: 0.1232
Epoch 22/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1199
Epoch 23/25
560/560 [==============================] - 10s 18ms/step - loss: 0.1171
Epoch 24/25
560/560 [==============================] - 10s 19ms/step - loss: 0.1126
Epoch 25/25
560/560 [==============================] - 11s 19ms/step - loss: 0.1048
375/375 [==============================] - 1s 2ms/step
```

```
pred = []
for i in ypred:
  pred.append(i.argmax())
pred
```

```
[10,
 12,
 8,
 12,
 10,
 8,
 9,
 4,
 10,
 7,
 10,
 0,
 4,
 8,
 8,
 8,
 7,
 10,
 5,
 5,
 5,
 12,
 6,
 12,
 10,
 10,
 8,
 10,
 4,
 7,
 10,
 4,
 12,
 7,
 9,
 12,
 9,
 5,
 8,
 12,
 7,
 8,
 4,
 8,
 5,
 2,
 7,
 7,
 7,
 10,
 2,
 5,
```

```
        12,
        7,
        8,
        5,
        8,
        8,
```

```
from sklearn.metrics import classification_report
print(classification_report(ytest,pred))
```

```
                   precision    recall  f1-score   support

               0       0.00      0.00      0.00        47
               1       0.03      0.04      0.03        51
               2       0.03      0.04      0.03       223
               3       0.05      0.06      0.05       235
               4       0.07      0.06      0.06       542
               5       0.20      0.21      0.21      1526
               6       0.14      0.13      0.13       381
               7       0.25      0.27      0.26      1167
               8       0.30      0.30      0.30      2598
               9       0.06      0.08      0.07       458
              10       0.20      0.20      0.20      1590
              11       0.07      0.07      0.07       635
              12       0.28      0.24      0.26      2547

        accuracy                           0.21     12000
       macro avg       0.13      0.13      0.13     12000
    weighted avg       0.22      0.21      0.21     12000
```