```
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]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
df = pd.read_csv("/content/apple-twitter-sentiment-texts.csv")
```

```
df
```

	text	sentiment
0	Wow. Yall needa step it up @Apple RT @heynyla:...	-1
1	What Happened To Apple Inc? http://t.co/FJEX...	0
2	Thank u @apple I can now compile all of the pi...	1
3	The oddly uplifting story of the Apple co-foun...	0
4	@apple can i exchange my iphone for a differen...	0
...	...	...
1625	Those** PICK UP THE SLACK YOU FUCK BOYS @Apple	-1
1626	Finally got my iPhone 6 in the mail and it com...	-1

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

```
 0    801
-1    686
 1    143
Name: sentiment, dtype: int64
```

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

```
text         0
sentiment    0
dtype: int64
```

```
# Map tweets sentiment
df['sentiment'] = df['sentiment'].map({-1.0:'Negative', 0.0:'Neutral', 1.0:'Positive'})
```

```
df
```

	text	sentiment
0	Wow. Yall needa step it up @Apple RT @heynyla:...	Negative
1	What Happened To Apple Inc? http://t.co/FJEX...	Neutral
2	Thank u @apple I can now compile all of the pi...	Positive
3	The oddly uplifting story of the Apple co-foun...	Neutral
4	@apple can i exchange my iphone for a differen...	Neutral
...	...	...
1625	Those** PICK UP THE SLACK YOU FUCK BOYS @Apple	Negative
1626	Finally got my iPhone 6 in the mail and it com...	Negative

```
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["clean_text"]=df["text"].apply(cleantext)


from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df["sentiment"] = le.fit_transform(df["sentiment"])


df
```

	text	sentiment	clean_text
0	Wow. Yall needa step it up @Apple RT @heynyla:...	0	wow yall needa step apple rt heynyla music sna...
1	What Happened To Apple Inc? http://t.co/FJEX...	1	happened apple inc http aapl apple moneypress ...
2	Thank u @apple I can now compile all of the pi...	2	thank u apple compile pic communicate one plac...
3	The oddly uplifting story of the Apple co-foun...	1	oddly uplifting story apple sold stake aapl aa...

```
x = df["clean_text"]
y = df["sentiment"]


from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.3, random_state=1)


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

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

	text	sentiment	clean_text	SentLen
			wow yall needa	

```
max(sentlen)
```

```
20
```

	What Happened To		happened apple inc

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

```
14.0
```

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

```
tok = Tokenizer(char_level=False, split=" ")
#char_level if True, every character will be treated as a token.
```

```
tok.fit_on_texts(xtrain)
tok.index_word
```

```
 708: 'mine',
 709: 'tomorrow',
 710: 'local',
 711: 'giving',
 712: 'bread',
 713: 'plugged',
 714: 'lack',
 715: 'integration',
 716: 'sure',
 717: 'att',
 718: 'facing',
 719: 'huge',
 720: 'letting',
 721: 'forget',
 722: 'press',
 723: 'normal',
 724: 'ask',
 725: 'plug',
 726: 'faggot',
 727: 'class',
 728: 'buffett',
 729: 'radbarakat',
 730: 'rejected',
 731: 'australia',
 732: 'approved',
 733: 'predicted',
 734: 'truly',
 735: 'badservice',
 736: 'fragile',
 737: 'lightning',
 738: 'renoxalex',
 739: 'tho',
 740: 'fitness',
 741: 'ive',
 742: 'little',
 743: 'deserve',
 744: 'cloud',
 745: 'gold',
 746: 'explain',
 747: 'held',
 748: 'backup',
 749: 'hitting',
 750: 'happen',
 751: 'developer',
 752: 'hundred',
 753: 'turkey',
 754: 'talent',
 755: 'dropped',
 756: 'dying',
 757: 'timcook',
 758: 'therealjonyive',
 759: 'quality',
 760: 'bell',
 761: 'sake',
 762: 'name',
 763: 'hosting',
 764: 'programming',
 765: 'developing',
 766: 'lason'
```

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

```
2980
```

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

```
      [72, 294, 294, 1, 2],
      [5,
       2298,
       233,
       68,
       1046,
       585,
       337,
       1047,
       1048,
       482,
       1049,
       14,
       39,
       102,
       1,
       1050],
      [1105, 1, 2299, 2300, 59, 2301, 13, 662, 2302, 2303],
      [4, 36, 66, 771, 1],
      [243, 1, 117, 672, 113, 1090, 3, 2],
      [30, 335, 137, 1, 2, 2],
      [1, 162, 815, 514, 816, 193, 194, 3, 2],
      [673, 1, 987, 580, 10, 2304],
      [1, 20, 2305, 400, 2306, 17, 1106],
      [124, 7, 52, 526, 8, 25, 1, 2],
      [92, 1, 14, 279, 231, 2, 2],
      [782, 590, 22, 53, 127, 44, 1],
      [2, 14, 87, 346, 1, 12, 2307, 441, 1, 2],
      [1, 51, 15, 2308],
      [2309, 2310, 162, 1, 3, 24, 2, 3],
      [233, 1025, 818, 960, 2311, 583, 443, 703, 1107, 1, 4, 2312, 572],
      [114, 838, 75, 485, 1, 62, 2, 18, 2313],
      [964, 1, 43, 126, 2314, 198, 2315, 2316, 2317],
      [5, 96, 1, 209, 146, 147, 210, 17, 3, 3, 2],
      [1, 60, 90, 22, 37, 132],
      [368, 369, 370, 371, 372, 1, 1, 39, 177, 19, 2],
      [673, 1, 2318],
      [1, 180, 576, 608, 69, 2319, 88, 2320, 21, 150],
      [27, 164, 125, 1, 648, 206, 447, 234, 2],
      [174, 1, 2321, 2322, 1108, 1109, 2],
      [2323, 140, 97, 669, 427, 674, 328, 1],
      [368, 369, 370, 371, 372, 1, 1, 39, 444, 19, 2],
      [261, 44, 8, 188, 1, 46, 623, 104, 143, 483, 2324, 1110, 404],
      [5, 2325, 83, 1, 6, 756, 22, 72, 422, 84],
      [1, 2326, 70, 437, 1006, 1111, 490, 1112, 6, 1066, 1112, 675, 160],
      [428, 638, 3, 1113, 138, 230, 975, 2],
      [1114, 1, 1115, 1, 2],
      [957, 102, 25, 2327, 760, 5, 2328, 39, 325, 8, 71, 129, 324, 1, 2],
      [74, 16, 1, 2329, 618, 2330, 131],
      [5, 1116, 1, 136, 676, 2331, 2, 2332],
      [397, 7, 1],
      [1, 9, 64, 170, 347, 38, 312, 4, 359, 3, 3, 2],
      [463, 2333, 1117, 1, 2334, 64, 2335, 3, 3, 2],
      [729, 1, 1, 126, 128, 63],
      [368, 369, 370, 371, 372, 1, 1, 39, 313, 19, 2],
      [5, 44, 1, 48, 1118, 54],
      [5, 27, 164, 125, 1, 648, 206, 447, 234, 2],
      [1, 645, 677, 23, 16, 388, 310, 3, 3, 2],
      [5, 10, 314, 425, 186, 89, 799, 7, 10, 1],
```

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

```
array([[   0,    0,    0, ...,    1,    2,    3],
       [   0,    0,    0, ..., 1183,    1,    2],
       [   0,    0,    0, ...,    1,   77,  690],
       ...,
       [   0,    0,    0, ...,    3,    3,    2],
       [   0,    5, 2977, ...,    4,   34,    2],
       [   0,    0,    0, ...,    1,   72, 2980]], dtype=int32)
```

+ Code + Text

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

```
seqmattest
```

```
array([[   0,    0,    0, ...,  212,  791,    1],
       [   0,    0,    0, ...,   38,  232, 1473],
       [   5, 1973,  268, ...,    1,  478,    2],
       ...,
       [   0,    0,    0, ...,  223, 1504,    2],
```

```
            [   0,    0,    0, ...,    1,   26,    2],
            [   0,    0,    0, ...,  480,  180,    2]], dtype=int32)
```

```
vocab_len
```

```
     2980
```

```
rnn = Sequential()
rnn.add(Embedding(vocab_len+1,15, 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=6, activation="softmax"))
rnn.compile(optimizer="adam", loss="sparse_categorical_crossentropy",metrics=['accuracy'])
history=rnn.fit(seqmattrain, ytrain, validation_split=0.2, epochs=10)
```

```
     Epoch 1/10
     29/29 [==============================] - 3s 20ms/step - loss: 1.5488 - accuracy: 0.3728 - val_loss: 1.1267 - val_accuracy: 0.4061
     Epoch 2/10
     29/29 [==============================] - 0s 9ms/step - loss: 1.0375 - accuracy: 0.4726 - val_loss: 0.9244 - val_accuracy: 0.6900
     Epoch 3/10
     29/29 [==============================] - 0s 9ms/step - loss: 0.9470 - accuracy: 0.5450 - val_loss: 0.8764 - val_accuracy: 0.5502
     Epoch 4/10
     29/29 [==============================] - 0s 9ms/step - loss: 0.8365 - accuracy: 0.6732 - val_loss: 0.8456 - val_accuracy: 0.6201
     Epoch 5/10
     29/29 [==============================] - 0s 8ms/step - loss: 0.6587 - accuracy: 0.7807 - val_loss: 0.7971 - val_accuracy: 0.6594
     Epoch 6/10
     29/29 [==============================] - 0s 10ms/step - loss: 0.4402 - accuracy: 0.8629 - val_loss: 0.9497 - val_accuracy: 0.6332
     Epoch 7/10
     29/29 [==============================] - 0s 8ms/step - loss: 0.2895 - accuracy: 0.9024 - val_loss: 1.0566 - val_accuracy: 0.6376
     Epoch 8/10
     29/29 [==============================] - 0s 10ms/step - loss: 0.2026 - accuracy: 0.9386 - val_loss: 1.2043 - val_accuracy: 0.5939
     Epoch 9/10
     29/29 [==============================] - 0s 9ms/step - loss: 0.1329 - accuracy: 0.9649 - val_loss: 1.3864 - val_accuracy: 0.5721
     Epoch 10/10
     29/29 [==============================] - 0s 8ms/step - loss: 0.0781 - accuracy: 0.9825 - val_loss: 1.4493 - val_accuracy: 0.5808
```

```
seqmattest[0]
```

```
     array([  0,   0,   0,   0,   0,   0,   0,   0,   5,   7,  89, 212, 791,
             1], dtype=int32)
```

```
yprob=rnn.predict(seqmattest)
yprob[0]
```

```
     16/16 [==============================] - 1s 4ms/step
     array([7.7642131e-01, 1.6481873e-02, 2.0482288e-01, 9.7057933e-04,
            8.7147945e-04, 4.3189302e-04], dtype=float32)
```

```
ypred=yprob.argmax(axis=1)
ypred
```

```
     array([0, 1, 1, 1, 0, 2, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 2, 1, 2, 1, 1,
            1, 1, 0, 1, 1, 0, 1, 1, 2, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
            1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1,
            1, 1, 2, 2, 1, 1, 2, 0, 0, 0, 2, 0, 0, 1, 1, 0, 1, 2, 0, 1, 1, 1,
            1, 1, 1, 1, 0, 1, 2, 2, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 2, 0, 1,
            1, 0, 0, 1, 0, 1, 2, 2, 1, 1, 2, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
            1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 2, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
            0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1,
            0, 1, 2, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
            2, 0, 0, 1, 1, 1, 1, 0, 2, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0,
            1, 0, 1, 2, 1, 1, 1, 1, 2, 1, 1, 1, 0, 1, 2, 0, 1, 1, 0, 0, 0, 2,
            1, 0, 2, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1,
            0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
            0, 1, 1, 1, 0, 1, 0, 0, 2, 2, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0,
            2, 1, 1, 1, 1, 0, 1, 1, 1, 0, 2, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1,
            0, 0, 2, 1, 1, 1, 1, 0, 1, 0, 2, 1, 0, 1, 0, 1, 2, 1, 2, 1, 0, 1,
            2, 1, 1, 1, 0, 1, 0, 1, 1, 2, 1, 1, 2, 0, 1, 0, 0, 0, 1, 1, 2, 1,
            2, 0, 1, 1, 1, 0, 1, 0, 2, 1, 0, 2, 0, 0, 0, 2, 1, 1, 1, 1, 1, 0,
            1, 0, 0, 1, 0, 0, 2, 0, 1, 1, 1, 1, 0, 1, 2, 0, 0, 1, 1, 2, 0, 1,
            1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 2, 0, 1, 0, 0, 1,
            0, 1, 2, 2, 0, 1, 0, 0, 0, 1, 1, 2, 0, 0, 2, 1, 1, 1, 0, 0, 0, 1,
            0, 2, 1, 0, 1])
```

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

	precision	recall	f1-score	support
0	0.67	0.62	0.65	189
1	0.78	0.79	0.79	259
2	0.20	0.24	0.22	41
accuracy			0.68	489
macro avg	0.55	0.55	0.55	489
weighted avg	0.69	0.68	0.68	489

✓ 0s completed at 2:28 PM