poem-generation-using-lstm

October 27, 2023

[4]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
↪outside of the current session
```

```
/kaggle/input/poem-generation/poem.txt
```

[5]:
```
from keras.models import Sequential, load_model
from keras.layers import Dense, LSTM, Dropout, BatchNormalization, Embedding
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import spacy, random
from keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping
from keras.preprocessing.sequence import pad_sequences
```

[6]:
```
def read_file(filepath):
    with open(filepath,'r') as f:
```

```python
        text = f.read()

    return text
```

[7]:
```python
poem = read_file('../input/poem-generation/poem.txt')
```

[8]:
```python
nlp = spacy.load('en_core_web_sm')
nlp.max_length = 20000000
```

[9]:
```python
def clean_text(text):
    return [token.text.lower() for token in nlp(text) if token.text not in '\n\n \n\n\n --!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n']
```

[10]:
```python
tokens = clean_text(poem)
```

[11]:
```python
train_len = 25 + 1
text_sequences = []

for i in range(train_len,len(tokens)):
    text_sequences.append(tokens[i-train_len:i])
```

[12]:
```python
len(text_sequences)
```

[12]: 19047

[13]:
```python
tokenizer = Tokenizer()
tokenizer.fit_on_texts(text_sequences)
```

[14]:
```python
' '.join(text_sequences[random.randint(0,len(text_sequences))])
```

[14]: 'to know i was leaving forever the land of my soul amid struggle and fear my parents did pray to place courage to leave oer the'

[15]:
```python
sequences = tokenizer.texts_to_sequences(text_sequences)
```

[16]:
```python
np.shape(sequences)
```

[16]: (19047, 26)

[17]:
```python
sequences = np.array(sequences)
sequences.shape
```

[17]: (19047, 26)

[18]:
```python
X = sequences[:,:-1]
y = sequences[:,-1]
```

[19]:
```
vocabulary_size = len(tokenizer.word_counts)
vocabulary_size
```

[19]: 3750

[20]:
```
y = to_categorical(y,num_classes=vocabulary_size+1)
```

[21]:
```
y.shape
```

[21]: (19047, 3751)

[22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

[23]:
```
seq_len = X.shape[1]
```

[24]:
```
seq_len
```

[24]: 25

[25]:
```
def create_model(vocabulary_size,seq_len):
    model = Sequential()
    model.
 ↪add(Embedding(input_dim=vocabulary_size,output_dim=seq_len,input_length=seq_len))
    model.add(LSTM(units=150,return_sequences=True))
    model.add(LSTM(units=150,return_sequences=True))
    model.add(BatchNormalization())
    model.add(Dropout(0.25))
    model.add(LSTM(units=300))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))
    model.add(Dense(units=150,activation='relu'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))
    model.add(Dense(units=vocabulary_size,activation='softmax'))
    model.
 ↪compile(loss='categorical_crossentropy',optimizer='adam',metrics='accuracy')
    model.summary()
    return model
```

[26]:
```
model = create_model(vocabulary_size+1,seq_len)
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 25, 25)            93775
```

```
_________________________________________________________________
lstm (LSTM)                  (None, 25, 150)           105600
_________________________________________________________________
lstm_1 (LSTM)                (None, 25, 150)           180600
_________________________________________________________________
batch_normalization (BatchNo (None, 25, 150)           600
_________________________________________________________________
dropout (Dropout)            (None, 25, 150)           0
_________________________________________________________________
lstm_2 (LSTM)                (None, 300)               541200
_________________________________________________________________
batch_normalization_1 (Batch (None, 300)               1200
_________________________________________________________________
dropout_1 (Dropout)          (None, 300)               0
_________________________________________________________________
dense (Dense)                (None, 150)               45150
_________________________________________________________________
batch_normalization_2 (Batch (None, 150)               600
_________________________________________________________________
dropout_2 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3751)              566401
=================================================================
Total params: 1,535,126
Trainable params: 1,533,926
Non-trainable params: 1,200
_________________________________________________________________
```

[27]:

```
early_stopping =␣
 ↪EarlyStopping(monitor='val_accuracy',patience=300,verbose=1,mode='max',restore_best_weights
reduce_lr =␣
 ↪ReduceLROnPlateau(monitor='val_accuracy',patience=10,mode='max',factor=0.
 ↪1,min_lr=0.001,verbose=1)
model_checkpoint = ModelCheckpoint('checkpoint/
 ↪',monitor='val_accuracy',mode='max',save_best_only=True,save_weights_only=True,verbose=1)

r = model.fit(X_train,
          y_train,
          batch_size=128,
          epochs=100,
          verbose=2,
          validation_data=(X_test,y_test),
          callbacks=[model_checkpoint,reduce_lr,early_stopping])
```

```
Epoch 1/100
105/105 - 11s - loss: 8.1214 - accuracy: 0.0257 - val_loss: 7.8475 -
val_accuracy: 0.0626
```

```
Epoch 00001: val_accuracy improved from -inf to 0.06264, saving model to
checkpoint/
Epoch 2/100
105/105 - 2s - loss: 7.2853 - accuracy: 0.0566 - val_loss: 7.3154 -
val_accuracy: 0.0626

Epoch 00002: val_accuracy did not improve from 0.06264
Epoch 3/100
105/105 - 2s - loss: 6.5623 - accuracy: 0.0668 - val_loss: 7.0689 -
val_accuracy: 0.0616

Epoch 00003: val_accuracy did not improve from 0.06264
Epoch 4/100
105/105 - 2s - loss: 6.2629 - accuracy: 0.0718 - val_loss: 7.0816 -
val_accuracy: 0.0600

Epoch 00004: val_accuracy did not improve from 0.06264
Epoch 5/100
105/105 - 2s - loss: 6.0945 - accuracy: 0.0744 - val_loss: 7.1220 -
val_accuracy: 0.0632

Epoch 00005: val_accuracy improved from 0.06264 to 0.06317, saving model to
checkpoint/
Epoch 6/100
105/105 - 2s - loss: 5.9374 - accuracy: 0.0776 - val_loss: 7.2384 -
val_accuracy: 0.0483

Epoch 00006: val_accuracy did not improve from 0.06317
Epoch 7/100
105/105 - 2s - loss: 5.8074 - accuracy: 0.0797 - val_loss: 7.4156 -
val_accuracy: 0.0413

Epoch 00007: val_accuracy did not improve from 0.06317
Epoch 8/100
105/105 - 2s - loss: 5.6854 - accuracy: 0.0845 - val_loss: 7.4148 -
val_accuracy: 0.0518

Epoch 00008: val_accuracy did not improve from 0.06317
Epoch 9/100
105/105 - 2s - loss: 5.5588 - accuracy: 0.0873 - val_loss: 7.4079 -
val_accuracy: 0.0437

Epoch 00009: val_accuracy did not improve from 0.06317
Epoch 10/100
105/105 - 2s - loss: 5.4325 - accuracy: 0.0899 - val_loss: 7.6389 -
val_accuracy: 0.0485
```

```
Epoch 00010: val_accuracy did not improve from 0.06317
Epoch 11/100
105/105 - 2s - loss: 5.3100 - accuracy: 0.0913 - val_loss: 7.6609 -
val_accuracy: 0.0458

Epoch 00011: val_accuracy did not improve from 0.06317
Epoch 12/100
105/105 - 2s - loss: 5.1566 - accuracy: 0.0977 - val_loss: 7.6972 -
val_accuracy: 0.0444

Epoch 00012: val_accuracy did not improve from 0.06317
Epoch 13/100
105/105 - 2s - loss: 5.0183 - accuracy: 0.0991 - val_loss: 7.7441 -
val_accuracy: 0.0446

Epoch 00013: val_accuracy did not improve from 0.06317
Epoch 14/100
105/105 - 2s - loss: 4.8681 - accuracy: 0.1063 - val_loss: 8.2174 -
val_accuracy: 0.0247

Epoch 00014: val_accuracy did not improve from 0.06317
Epoch 15/100
105/105 - 2s - loss: 4.7023 - accuracy: 0.1132 - val_loss: 8.2135 -
val_accuracy: 0.0345

Epoch 00015: val_accuracy did not improve from 0.06317
Epoch 16/100
105/105 - 2s - loss: 4.5380 - accuracy: 0.1240 - val_loss: 7.9771 -
val_accuracy: 0.0486

Epoch 00016: val_accuracy did not improve from 0.06317
Epoch 17/100
105/105 - 2s - loss: 4.3553 - accuracy: 0.1358 - val_loss: 8.3590 -
val_accuracy: 0.0567

Epoch 00017: val_accuracy did not improve from 0.06317
Epoch 18/100
105/105 - 2s - loss: 4.1754 - accuracy: 0.1562 - val_loss: 8.3310 -
val_accuracy: 0.0541

Epoch 00018: val_accuracy did not improve from 0.06317
Epoch 19/100
105/105 - 2s - loss: 4.0012 - accuracy: 0.1708 - val_loss: 8.4359 -
val_accuracy: 0.0535

Epoch 00019: val_accuracy did not improve from 0.06317
Epoch 20/100
105/105 - 2s - loss: 3.8235 - accuracy: 0.1927 - val_loss: 8.5792 -
```

```
val_accuracy: 0.0434

Epoch 00020: val_accuracy did not improve from 0.06317
Epoch 21/100
105/105 - 2s - loss: 3.6447 - accuracy: 0.2131 - val_loss: 8.5657 -
val_accuracy: 0.0588

Epoch 00021: val_accuracy did not improve from 0.06317
Epoch 22/100
105/105 - 2s - loss: 3.5201 - accuracy: 0.2233 - val_loss: 8.6909 -
val_accuracy: 0.0504

Epoch 00022: val_accuracy did not improve from 0.06317
Epoch 23/100
105/105 - 2s - loss: 3.3386 - accuracy: 0.2520 - val_loss: 8.7751 -
val_accuracy: 0.0558

Epoch 00023: val_accuracy did not improve from 0.06317
Epoch 24/100
105/105 - 2s - loss: 3.2073 - accuracy: 0.2696 - val_loss: 8.9481 -
val_accuracy: 0.0499

Epoch 00024: val_accuracy did not improve from 0.06317
Epoch 25/100
105/105 - 2s - loss: 3.0806 - accuracy: 0.2908 - val_loss: 8.9068 -
val_accuracy: 0.0520

Epoch 00025: val_accuracy did not improve from 0.06317
Epoch 26/100
105/105 - 2s - loss: 2.9589 - accuracy: 0.3088 - val_loss: 9.3892 -
val_accuracy: 0.0576

Epoch 00026: val_accuracy did not improve from 0.06317
Epoch 27/100
105/105 - 2s - loss: 2.8157 - accuracy: 0.3326 - val_loss: 9.3600 -
val_accuracy: 0.0492

Epoch 00027: val_accuracy did not improve from 0.06317
Epoch 28/100
105/105 - 2s - loss: 2.7015 - accuracy: 0.3507 - val_loss: 9.5796 -
val_accuracy: 0.0507

Epoch 00028: val_accuracy did not improve from 0.06317
Epoch 29/100
105/105 - 2s - loss: 2.5920 - accuracy: 0.3711 - val_loss: 9.6639 -
val_accuracy: 0.0497

Epoch 00029: val_accuracy did not improve from 0.06317
```

```
Epoch 30/100
105/105 - 2s - loss: 2.5098 - accuracy: 0.3848 - val_loss: 9.6730 -
val_accuracy: 0.0455

Epoch 00030: val_accuracy did not improve from 0.06317
Epoch 31/100
105/105 - 2s - loss: 2.3893 - accuracy: 0.4050 - val_loss: 9.8309 -
val_accuracy: 0.0436

Epoch 00031: val_accuracy did not improve from 0.06317
Epoch 32/100
105/105 - 2s - loss: 2.2930 - accuracy: 0.4261 - val_loss: 10.1495 -
val_accuracy: 0.0507

Epoch 00032: val_accuracy did not improve from 0.06317
Epoch 33/100
105/105 - 2s - loss: 2.1979 - accuracy: 0.4462 - val_loss: 10.1489 -
val_accuracy: 0.0471

Epoch 00033: val_accuracy did not improve from 0.06317
Epoch 34/100
105/105 - 2s - loss: 2.1204 - accuracy: 0.4587 - val_loss: 10.1842 -
val_accuracy: 0.0465

Epoch 00034: val_accuracy did not improve from 0.06317
Epoch 35/100
105/105 - 2s - loss: 2.0260 - accuracy: 0.4778 - val_loss: 10.1095 -
val_accuracy: 0.0469

Epoch 00035: val_accuracy did not improve from 0.06317
Epoch 36/100
105/105 - 2s - loss: 1.9281 - accuracy: 0.4988 - val_loss: 10.6400 -
val_accuracy: 0.0492

Epoch 00036: val_accuracy did not improve from 0.06317
Epoch 37/100
105/105 - 2s - loss: 1.8600 - accuracy: 0.5095 - val_loss: 10.7543 -
val_accuracy: 0.0541

Epoch 00037: val_accuracy did not improve from 0.06317
Epoch 38/100
105/105 - 2s - loss: 1.7858 - accuracy: 0.5269 - val_loss: 10.7755 -
val_accuracy: 0.0542

Epoch 00038: val_accuracy did not improve from 0.06317
Epoch 39/100
105/105 - 2s - loss: 1.7111 - accuracy: 0.5515 - val_loss: 10.9005 -
val_accuracy: 0.0502
```

```
Epoch 00039: val_accuracy did not improve from 0.06317
Epoch 40/100
105/105 - 2s - loss: 1.6570 - accuracy: 0.5581 - val_loss: 10.9922 -
val_accuracy: 0.0516

Epoch 00040: val_accuracy did not improve from 0.06317
Epoch 41/100
105/105 - 2s - loss: 1.6121 - accuracy: 0.5663 - val_loss: 11.3927 -
val_accuracy: 0.0502

Epoch 00041: val_accuracy did not improve from 0.06317
Epoch 42/100
105/105 - 2s - loss: 1.5256 - accuracy: 0.5931 - val_loss: 11.5065 -
val_accuracy: 0.0467

Epoch 00042: val_accuracy did not improve from 0.06317
Epoch 43/100
105/105 - 2s - loss: 1.4841 - accuracy: 0.5959 - val_loss: 11.3217 -
val_accuracy: 0.0530

Epoch 00043: val_accuracy did not improve from 0.06317
Epoch 44/100
105/105 - 2s - loss: 1.4208 - accuracy: 0.6097 - val_loss: 11.5734 -
val_accuracy: 0.0486

Epoch 00044: val_accuracy did not improve from 0.06317
Epoch 45/100
105/105 - 2s - loss: 1.3590 - accuracy: 0.6238 - val_loss: 11.4980 -
val_accuracy: 0.0542

Epoch 00045: val_accuracy did not improve from 0.06317
Epoch 46/100
105/105 - 2s - loss: 1.3034 - accuracy: 0.6369 - val_loss: 11.7259 -
val_accuracy: 0.0537

Epoch 00046: val_accuracy did not improve from 0.06317
Epoch 47/100
105/105 - 2s - loss: 1.2721 - accuracy: 0.6444 - val_loss: 11.8098 -
val_accuracy: 0.0527

Epoch 00047: val_accuracy did not improve from 0.06317
Epoch 48/100
105/105 - 2s - loss: 1.2291 - accuracy: 0.6547 - val_loss: 11.8891 -
val_accuracy: 0.0539

Epoch 00048: val_accuracy did not improve from 0.06317
Epoch 49/100
```

```
105/105 - 2s - loss: 1.1719 - accuracy: 0.6717 - val_loss: 12.0269 -
val_accuracy: 0.0462

Epoch 00049: val_accuracy did not improve from 0.06317
Epoch 50/100
105/105 - 2s - loss: 1.1317 - accuracy: 0.6832 - val_loss: 12.3116 -
val_accuracy: 0.0451

Epoch 00050: val_accuracy did not improve from 0.06317
Epoch 51/100
105/105 - 2s - loss: 1.0984 - accuracy: 0.6905 - val_loss: 12.3608 -
val_accuracy: 0.0523

Epoch 00051: val_accuracy did not improve from 0.06317
Epoch 52/100
105/105 - 2s - loss: 1.0387 - accuracy: 0.7011 - val_loss: 12.4660 -
val_accuracy: 0.0490

Epoch 00052: val_accuracy did not improve from 0.06317
Epoch 53/100
105/105 - 2s - loss: 1.0307 - accuracy: 0.7053 - val_loss: 12.5156 -
val_accuracy: 0.0488

Epoch 00053: val_accuracy did not improve from 0.06317
Epoch 54/100
105/105 - 2s - loss: 0.9739 - accuracy: 0.7235 - val_loss: 12.6652 -
val_accuracy: 0.0474

Epoch 00054: val_accuracy did not improve from 0.06317
Epoch 55/100
105/105 - 2s - loss: 0.9446 - accuracy: 0.7267 - val_loss: 12.9489 -
val_accuracy: 0.0495

Epoch 00055: val_accuracy did not improve from 0.06317
Epoch 56/100
105/105 - 2s - loss: 0.9070 - accuracy: 0.7381 - val_loss: 12.8666 -
val_accuracy: 0.0493

Epoch 00056: val_accuracy did not improve from 0.06317
Epoch 57/100
105/105 - 2s - loss: 0.8997 - accuracy: 0.7358 - val_loss: 12.9107 -
val_accuracy: 0.0486

Epoch 00057: val_accuracy did not improve from 0.06317
Epoch 58/100
105/105 - 2s - loss: 0.8651 - accuracy: 0.7417 - val_loss: 13.1090 -
val_accuracy: 0.0474
```

```
Epoch 00058: val_accuracy did not improve from 0.06317
Epoch 59/100
105/105 - 2s - loss: 0.8572 - accuracy: 0.7536 - val_loss: 13.4113 -
val_accuracy: 0.0448

Epoch 00059: val_accuracy did not improve from 0.06317
Epoch 60/100
105/105 - 2s - loss: 0.8217 - accuracy: 0.7605 - val_loss: 13.0212 -
val_accuracy: 0.0516

Epoch 00060: val_accuracy did not improve from 0.06317
Epoch 61/100
105/105 - 2s - loss: 0.8293 - accuracy: 0.7562 - val_loss: 13.4311 -
val_accuracy: 0.0478

Epoch 00061: val_accuracy did not improve from 0.06317
Epoch 62/100
105/105 - 2s - loss: 0.7988 - accuracy: 0.7664 - val_loss: 13.4336 -
val_accuracy: 0.0499

Epoch 00062: val_accuracy did not improve from 0.06317
Epoch 63/100
105/105 - 2s - loss: 0.7295 - accuracy: 0.7890 - val_loss: 13.6459 -
val_accuracy: 0.0486

Epoch 00063: val_accuracy did not improve from 0.06317
Epoch 64/100
105/105 - 2s - loss: 0.7136 - accuracy: 0.7896 - val_loss: 13.6629 -
val_accuracy: 0.0497

Epoch 00064: val_accuracy did not improve from 0.06317
Epoch 65/100
105/105 - 2s - loss: 0.6975 - accuracy: 0.7947 - val_loss: 13.6383 -
val_accuracy: 0.0542

Epoch 00065: val_accuracy did not improve from 0.06317
Epoch 66/100
105/105 - 2s - loss: 0.6871 - accuracy: 0.8003 - val_loss: 13.9124 -
val_accuracy: 0.0469

Epoch 00066: val_accuracy did not improve from 0.06317
Epoch 67/100
105/105 - 2s - loss: 0.7424 - accuracy: 0.7843 - val_loss: 13.8446 -
val_accuracy: 0.0513

Epoch 00067: val_accuracy did not improve from 0.06317
Epoch 68/100
105/105 - 2s - loss: 0.6708 - accuracy: 0.8004 - val_loss: 14.0018 -
```

```
val_accuracy: 0.0479

Epoch 00068: val_accuracy did not improve from 0.06317
Epoch 69/100
105/105 - 2s - loss: 0.6999 - accuracy: 0.7952 - val_loss: 14.2046 -
val_accuracy: 0.0432

Epoch 00069: val_accuracy did not improve from 0.06317
Epoch 70/100
105/105 - 2s - loss: 0.6268 - accuracy: 0.8128 - val_loss: 14.2980 -
val_accuracy: 0.0476

Epoch 00070: val_accuracy did not improve from 0.06317
Epoch 71/100
105/105 - 2s - loss: 0.6343 - accuracy: 0.8176 - val_loss: 14.3163 -
val_accuracy: 0.0471

Epoch 00071: val_accuracy did not improve from 0.06317
Epoch 72/100
105/105 - 2s - loss: 0.5861 - accuracy: 0.8241 - val_loss: 14.1729 -
val_accuracy: 0.0541

Epoch 00072: val_accuracy did not improve from 0.06317
Epoch 73/100
105/105 - 2s - loss: 0.5630 - accuracy: 0.8294 - val_loss: 14.7152 -
val_accuracy: 0.0409

Epoch 00073: val_accuracy did not improve from 0.06317
Epoch 74/100
105/105 - 2s - loss: 0.5611 - accuracy: 0.8339 - val_loss: 14.5983 -
val_accuracy: 0.0430

Epoch 00074: val_accuracy did not improve from 0.06317
Epoch 75/100
105/105 - 2s - loss: 0.5572 - accuracy: 0.8354 - val_loss: 14.5968 -
val_accuracy: 0.0441

Epoch 00075: val_accuracy did not improve from 0.06317
Epoch 76/100
105/105 - 2s - loss: 0.5599 - accuracy: 0.8333 - val_loss: 14.6076 -
val_accuracy: 0.0488

Epoch 00076: val_accuracy did not improve from 0.06317
Epoch 77/100
105/105 - 2s - loss: 0.5930 - accuracy: 0.8239 - val_loss: 14.9697 -
val_accuracy: 0.0460

Epoch 00077: val_accuracy did not improve from 0.06317
```

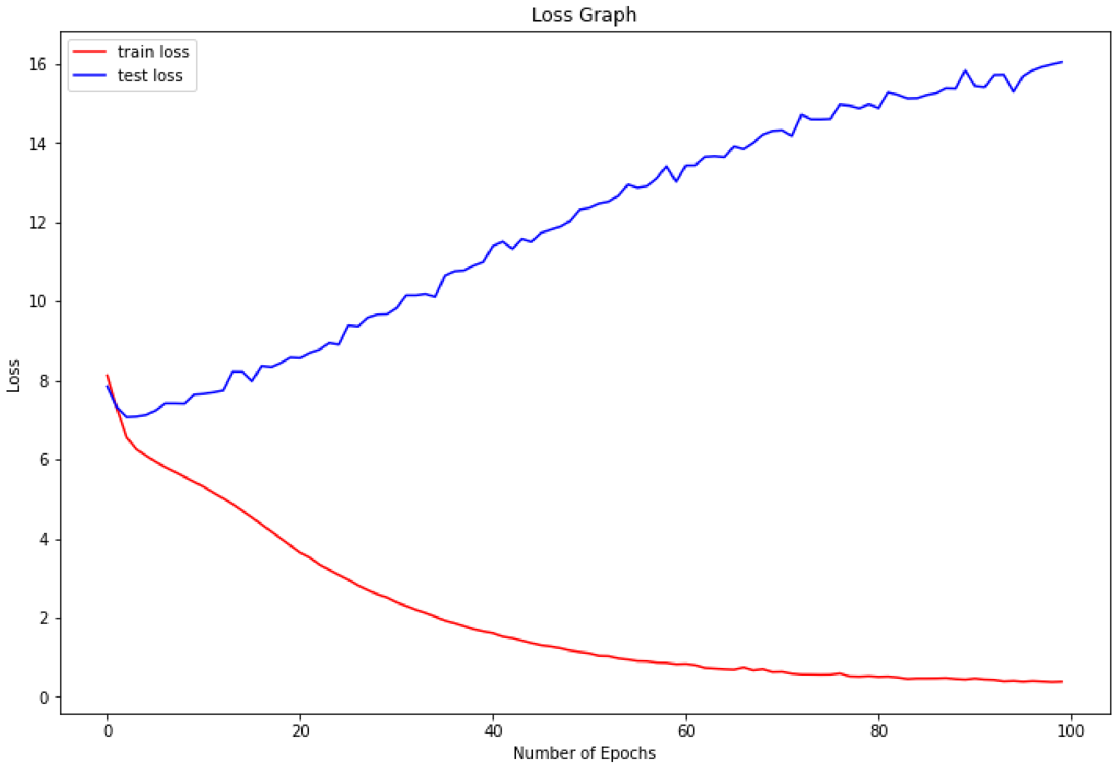
```
Epoch 78/100
105/105 - 2s - loss: 0.5143 - accuracy: 0.8474 - val_loss: 14.9392 -
val_accuracy: 0.0495

Epoch 00078: val_accuracy did not improve from 0.06317
Epoch 79/100
105/105 - 2s - loss: 0.5070 - accuracy: 0.8489 - val_loss: 14.8744 -
val_accuracy: 0.0486

Epoch 00079: val_accuracy did not improve from 0.06317
Epoch 80/100
105/105 - 2s - loss: 0.5202 - accuracy: 0.8440 - val_loss: 14.9757 -
val_accuracy: 0.0460

Epoch 00080: val_accuracy did not improve from 0.06317
Epoch 81/100
105/105 - 2s - loss: 0.5018 - accuracy: 0.8464 - val_loss: 14.8794 -
val_accuracy: 0.0458

Epoch 00081: val_accuracy did not improve from 0.06317
Epoch 82/100
105/105 - 2s - loss: 0.5092 - accuracy: 0.8510 - val_loss: 15.2785 -
val_accuracy: 0.0485

Epoch 00082: val_accuracy did not improve from 0.06317
Epoch 83/100
105/105 - 2s - loss: 0.4859 - accuracy: 0.8546 - val_loss: 15.2154 -
val_accuracy: 0.0458

Epoch 00083: val_accuracy did not improve from 0.06317
Epoch 84/100
105/105 - 2s - loss: 0.4484 - accuracy: 0.8672 - val_loss: 15.1226 -
val_accuracy: 0.0439

Epoch 00084: val_accuracy did not improve from 0.06317
Epoch 85/100
105/105 - 2s - loss: 0.4611 - accuracy: 0.8596 - val_loss: 15.1291 -
val_accuracy: 0.0453

Epoch 00085: val_accuracy did not improve from 0.06317
Epoch 86/100
105/105 - 2s - loss: 0.4611 - accuracy: 0.8601 - val_loss: 15.2090 -
val_accuracy: 0.0465

Epoch 00086: val_accuracy did not improve from 0.06317
Epoch 87/100
105/105 - 2s - loss: 0.4637 - accuracy: 0.8581 - val_loss: 15.2600 -
val_accuracy: 0.0474
```

```
Epoch 00087: val_accuracy did not improve from 0.06317
Epoch 88/100
105/105 - 2s - loss: 0.4718 - accuracy: 0.8585 - val_loss: 15.3824 -
val_accuracy: 0.0528

Epoch 00088: val_accuracy did not improve from 0.06317
Epoch 89/100
105/105 - 2s - loss: 0.4478 - accuracy: 0.8677 - val_loss: 15.3742 -
val_accuracy: 0.0493

Epoch 00089: val_accuracy did not improve from 0.06317
Epoch 90/100
105/105 - 2s - loss: 0.4336 - accuracy: 0.8707 - val_loss: 15.8349 -
val_accuracy: 0.0441

Epoch 00090: val_accuracy did not improve from 0.06317
Epoch 91/100
105/105 - 2s - loss: 0.4603 - accuracy: 0.8579 - val_loss: 15.4374 -
val_accuracy: 0.0502

Epoch 00091: val_accuracy did not improve from 0.06317
Epoch 92/100
105/105 - 2s - loss: 0.4354 - accuracy: 0.8645 - val_loss: 15.4088 -
val_accuracy: 0.0451

Epoch 00092: val_accuracy did not improve from 0.06317
Epoch 93/100
105/105 - 2s - loss: 0.4256 - accuracy: 0.8687 - val_loss: 15.7161 -
val_accuracy: 0.0451

Epoch 00093: val_accuracy did not improve from 0.06317
Epoch 94/100
105/105 - 2s - loss: 0.3905 - accuracy: 0.8799 - val_loss: 15.7224 -
val_accuracy: 0.0518

Epoch 00094: val_accuracy did not improve from 0.06317
Epoch 95/100
105/105 - 2s - loss: 0.4024 - accuracy: 0.8792 - val_loss: 15.2950 -
val_accuracy: 0.0485

Epoch 00095: val_accuracy did not improve from 0.06317
Epoch 96/100
105/105 - 2s - loss: 0.3821 - accuracy: 0.8855 - val_loss: 15.6855 -
val_accuracy: 0.0423

Epoch 00096: val_accuracy did not improve from 0.06317
Epoch 97/100
```

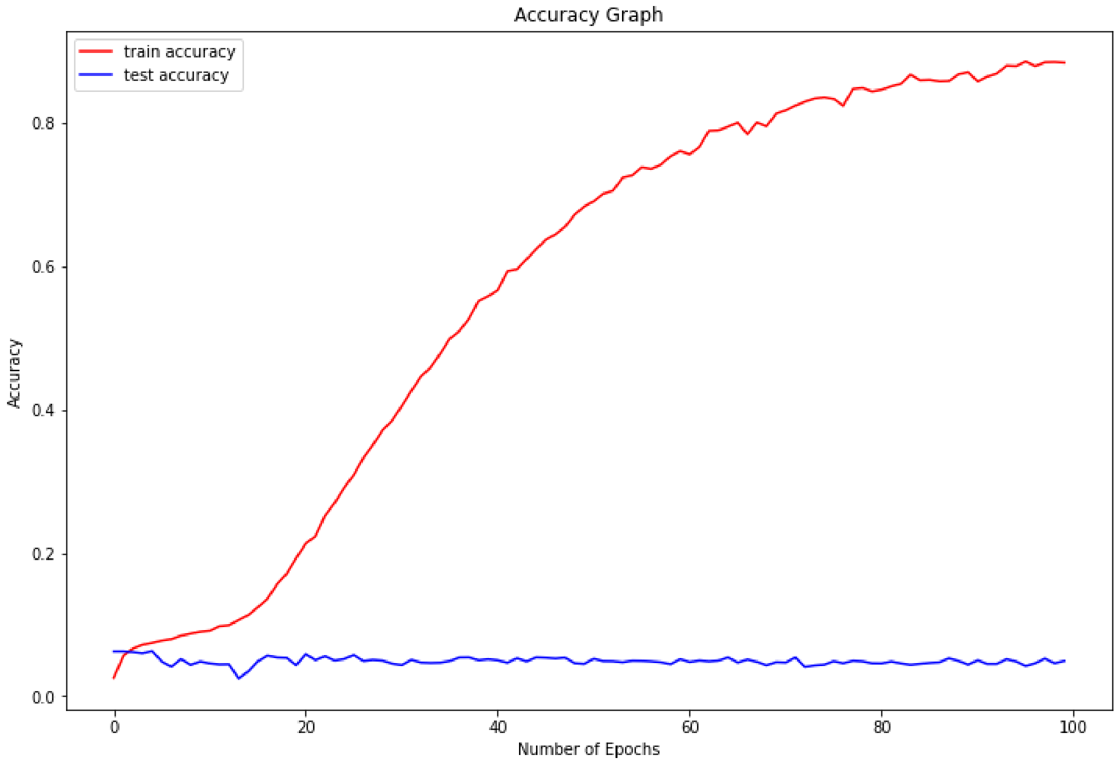
```
105/105 - 2s - loss: 0.3987 - accuracy: 0.8792 - val_loss: 15.8366 -
val_accuracy: 0.0460

Epoch 00097: val_accuracy did not improve from 0.06317
Epoch 98/100
105/105 - 2s - loss: 0.3858 - accuracy: 0.8845 - val_loss: 15.9275 -
val_accuracy: 0.0527

Epoch 00098: val_accuracy did not improve from 0.06317
Epoch 99/100
105/105 - 2s - loss: 0.3731 - accuracy: 0.8848 - val_loss: 15.9890 -
val_accuracy: 0.0460

Epoch 00099: val_accuracy did not improve from 0.06317
Epoch 100/100
105/105 - 2s - loss: 0.3821 - accuracy: 0.8840 - val_loss: 16.0463 -
val_accuracy: 0.0492

Epoch 00100: val_accuracy did not improve from 0.06317
```

[28]:

```
plt.figure(figsize=(12,8))
plt.plot(r.history['loss'],'r',label='train loss')
plt.plot(r.history['val_loss'],'b',label='test loss')
plt.xlabel('Number of Epochs')
plt.ylabel('Loss')
plt.title('Loss Graph')
plt.legend();
```



[29]:
```
plt.figure(figsize=(12,8))
plt.plot(r.history['accuracy'],'r',label='train accuracy')
plt.plot(r.history['val_accuracy'],'b',label='test accuracy')
plt.xlabel('Number of Epochs')
plt.ylabel('Accuracy')
plt.title('Accuracy Graph')
plt.legend();
```



[30]:
```
model.evaluate(X_test,y_test)
```

```
179/179 [==============================] - 1s 6ms/step - loss: 16.0463 -
accuracy: 0.0492
```

[30]:
```
[16.046314239501953, 0.04916885495185852]
```

[31]:
```
model.save('literature_generator.h5')
```

[32]:
```
lstm = load_model('literature_generator.h5')
lstm
```

[32]:
```
<keras.engine.sequential.Sequential at 0x7b1b642fdb50>
```

[43]:
```
def generate_text(model,tokenizer,seq_len,seed_text,num_words):
    output_text = []
    input_text = seed_text

    for i in range(num_words):
        encoded_text = tokenizer.texts_to_sequences([input_text])[0]
        padded_text = 
 ↪pad_sequences([encoded_text],maxlen=seq_len,truncating='pre')
        predictions = model.predict(padded_text,verbose=0)[0]
```

```
        pred_word = tokenizer.index_word[predictions.argmax()]
        input_text += ' ' + pred_word
        output_text.append(pred_word)

    return ' '.join(output_text)
```

[44]:
```
for i in range(10):
    random_seed_text = ' '.join(text_sequences[random.
 ↪randint(0,len(text_sequences))])
    print(generate_text(lstm,tokenizer,seq_len,random_seed_text,num_words=25))
```

```
i our bow fought for swell danced in a swell danced to unexpected to bowl of
warns to warns the punctual warns the moon men
and a flower flower a flower bell ringing revenge and longer of the sun time
sand away for your touch and not not moving n't
and in your war and captive slave for do do no do or desolate but good dear take
your chain they love more will no
home it is is the best it we can turn a trial why calls with the face is the
difference waiting of wrapped to old
my delight thou my souls shelter thou high high tower raise thou me heavenward
high power of my power riches i i not nor mans
true true lover no lover do but with do do but ever ever but with going with
home between warm but and no good living
prince edward edward waiting an a ball of a chisel one and marchin a pipers
flashed in the pipers of claim of a man too
will play the more rover no more sounds the harm eye and never alas i never was
take up and the boys grow it lie
a paradise to a paradise to the maid malone for me here though blood here they
casey and no course they their missed did they
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_17/2821180405.py in <module>
      1 for i in range(10):
----> 2     random_seed_text = ' '.join(text_sequences[random.
 ↪randint(0,len(text_sequences))])
      3     ␣
 ↪print(generate_text(lstm,tokenizer,seq_len,random_seed_text,num_words=25))

IndexError: list index out of range
```

[45]:
```
print(generate_text(lstm,tokenizer,seq_len,'once upon a time',num_words=50))
```

```
good wives jest of this brave at the wives fearless is fearless and fall are
fall how the red 's coat coat country within the town is gone is slavery for
irish first is the world far is drowsily difference gone of april every april
every world seems spite and
```

[46]:
```
print(generate_text(lstm,tokenizer,seq_len,'you belong',num_words=50))
```

```
good good jar grave hiccough like the envy and a man chieftains who o'er
laughters is sweet is is has even your face and much toward toward you
unsurpassed in the stars on the claiming song and the glimmering and the banks
and s was young foolish and i took
```

[47]:
```
print(generate_text(lstm,tokenizer,seq_len,'I wandered lonely as a␣
↪cloud',num_words=100))
```

```
at men and do if if danny found i call nt nt going the call of girls all the
barn of barn and shining spent in a shocking wet an shocking mad i was her was
tree and i was standing from the first was then she smiled and was the sky but
his weirs malone i was his salley lad and she was threw and she gear and was the
vision of she was gardens and rest i gear and her wheels the sweet of saw and
she can rose like the summer time was time grow and time
```

[ ]:
```

```