```
"""
LSTMによる文字レベルのテキスト生成：
ニーチェの著作をデータベースとしてLSTMモデルに学習させ、
任意に抽出された連続的なN個の文字からなる文字列を与えることで、
付いてくる文字列を予測させる
"""

import os
import keras
import sys
import numpy as np
import random
from keras.models import Sequential
from keras.layers import LSTM, Dense
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint, EarlyStopping
import matplotlib.pyplot as plt

dir_path = "/content/drive/My Drive/Python/LSTM_words_generator/"
os.chdir(dir_path)

#  最初のテキストファイルのダウンロードと解析
file_path = keras.utils.get_file('nietzsche.txt', origin='https://s3.amazonaws.com/text-datasets/nietzsche.txt')
text = open(file_path).read().lower()
print('Corpus length:', len(text))
print('Number of text:', len(text))

#  文字のシーケンスのベクトル化
maxlen = 60
step = 3
sentences = []
next_chars = []

for i in range(0, len(text) - maxlen, step):
```

```
for i in range(0, len(text) - maxlen, step):
    sentences.append(text[i: i + maxlen])
    next_chars.append(text[i + maxlen])

print('Number of sequences:', len(sentences))

chars = sorted(list(set(text)))
print('Unique characters:', len(chars))

char_indices = dict((char, chars.index(char)) for char in chars)

print('Vectorization...')

x = np.zeros((len(sentences), maxlen, len(chars)), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
    for j, char in enumerate(sentence):
        x[i, j, char_indices[char]] = 1
        y[i, char_indices[next_chars[i]]] = 1

#  次の文字を予測する単層LSTMモデルとコンパイル設定
rand_num = list(range(len(sentences)))
random.shuffle(rand_num)
split_point = len(sentences) // 10 * 7
x_train = x[rand_num[:split_point]]
y_train = y[rand_num[:split_point]]
x_test = x[rand_num[split_point:]]
y_test = y[rand_num[split_point:]]
print('shape of x_train:', x_train.shape)
print('shape of y_train:', y_train.shape)
print('shape of x_test:', x_test.shape)
print('shape of y_test:', y_test.shape)

model = Sequential()
model.add(LSTM(1024, dropout=0.1, recurrent_dropout=0.1, input_shape=(maxlen, len(chars))))
model.add(Dense(len(chars), activation='softmax'))

optimizer = RMSprop(lr=0.01)
```

```
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
model.summary()

#  モデルの予測に基づいて次の文字をサンプリング関数
def sample(preds, temp=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temp
    preds = np.exp(preds)
    preds /= np.sum(preds)
    probs = np.random.multinomial(1, preds, 1)
    return np.argmax(probs)

#  テキスト生成ループ
save_path = 'weights_max.hdf5'
checkpoint = ModelCheckpoint(save_path, monitor='val_accuracy', verbose=1, ¥
    save_best_only=True, mode='auto')
earlystopping = EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=10,
                              mode='auto', verbose=1)
hist = model.fit(x_train, y_train, batch_size=256, epochs=100, validation_split=0.1, callbacks=[checkpoint, earlystopping])

model.load_weights('weights_max.hdf5')
score = model.evaluate(x_test, y_test, verbose=1)
print('loss=', score[0], 'accuracy=', score[1])

plt.plot(hist.history["loss"])
plt.plot(hist.history["val_loss"])
plt.title("Loss")
plt.legend(["train", "test"], loc = "upper left")
plt.show()

plt.plot(hist.history["accuracy"])
plt.plot(hist.history["val_accuracy"])
plt.title("Accuracy")
plt.legend(["train", "test"], loc = "upper left")
plt.show()

model.save("LSTM_words_generator.h5")
```

```
start_index = np.random.randint(0, len(text) - maxlen)
generated_text = text[start_index: start_index + maxlen]
print('---Generating with seed:"' + generated_text + '"')

for temp in [0.2, 0.5, 1.0, 1.2]:
    print('------ temperature:', temp)
    for i in range(400):
        sampled_text = np.zeros((1, maxlen, len(chars)), dtype=np.bool)
        for i, char in enumerate(generated_text):
            sampled_text[0, i, chars.index(char)] = 1

        preds = model.predict(sampled_text, verbose=0)[0]
        next_index = sample(preds, temp)
        next_char = chars[next_index]

        generated_text += next_char
        generated_text = generated_text[1:]

        sys.stdout.write(next_char)
        sys.stdout.flush()
    print('¥n')
```

```
Corpus length: 600893
Number of text: 600893
Number of sequences: 200278
Unique characters: 57
Vectorization...
shape of x_train: (140189, 60, 57)
shape of y_train: (140189, 57)
shape of x_test: (60089, 60, 57)
shape of y_test: (60089, 57)
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 1024)              4431872
_________________________________________________________________
dense_2 (Dense)              (None, 57)                58425
=================================================================
Total params: 4,490,297
Trainable params: 4,490,297
Non-trainable params: 0
_________________________________________________________________
Train on 126170 samples, validate on 14019 samples
Epoch 1/100
126170/126170 [==============================] - 193s 2ms/step - loss: 2.8516 - accuracy: 0.2575 - val_loss: 1.9599 - val_accuracy: 0.4225

Epoch 00001: val_accuracy improved from -inf to 0.42250, saving model to weights_max.hdf5
Epoch 2/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.9197 - accuracy: 0.4317 - val_loss: 1.6783 - val_accuracy: 0.5023

Epoch 00002: val_accuracy improved from 0.42250 to 0.50232, saving model to weights_max.hdf5
Epoch 3/100
126170/126170 [==============================] - 193s 2ms/step - loss: 1.7413 - accuracy: 0.4804 - val_loss: 1.5875 - val_accuracy: 0.5264

Epoch 00003: val_accuracy improved from 0.50232 to 0.52636, saving model to weights_max.hdf5
Epoch 4/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.6507 - accuracy: 0.5064 - val_loss: 1.5505 - val_accuracy: 0.5390

Epoch 00004: val_accuracy improved from 0.52636 to 0.53898, saving model to weights_max.hdf5
Epoch 5/100
126170/126170 [==============================] - 193s 2ms/step - loss: 1.5933 - accuracy: 0.5212 - val_loss: 1.5216 - val_accuracy: 0.5541

Epoch 00005: val_accuracy improved from 0.53898 to 0.55411, saving model to weights_max.hdf5
```

Epoch 6/100
126170/126170 [==============================] - 193s 2ms/step - loss: 1.5461 - accuracy: 0.5340 - val_loss: 1.5049 - val_accuracy: 0.5547

Epoch 00006: val_accuracy improved from 0.55411 to 0.55468, saving model to weights_max.hdf5
Epoch 7/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.5194 - accuracy: 0.5409 - val_loss: 1.5025 - val_accuracy: 0.5629

Epoch 00007: val_accuracy improved from 0.55468 to 0.56288, saving model to weights_max.hdf5
Epoch 8/100
126170/126170 [==============================] - 193s 2ms/step - loss: 1.4875 - accuracy: 0.5501 - val_loss: 1.4883 - val_accuracy: 0.5598

Epoch 00008: val_accuracy did not improve from 0.56288
Epoch 9/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.4616 - accuracy: 0.5559 - val_loss: 1.4744 - val_accuracy: 0.5616

Epoch 00009: val_accuracy did not improve from 0.56288
Epoch 10/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.4308 - accuracy: 0.5618 - val_loss: 1.4693 - val_accuracy: 0.5635

Epoch 00010: val_accuracy improved from 0.56288 to 0.56352, saving model to weights_max.hdf5
Epoch 11/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.4167 - accuracy: 0.5649 - val_loss: 1.4815 - val_accuracy: 0.5617

Epoch 00011: val_accuracy did not improve from 0.56352
Epoch 12/100
126170/126170 [==============================] - 191s 2ms/step - loss: 1.3991 - accuracy: 0.5708 - val_loss: 1.4745 - val_accuracy: 0.5602

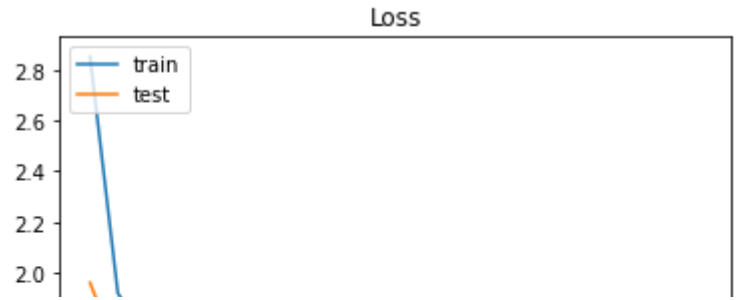
Epoch 00012: val_accuracy did not improve from 0.56352
Epoch 13/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.3793 - accuracy: 0.5765 - val_loss: 1.4795 - val_accuracy: 0.5670

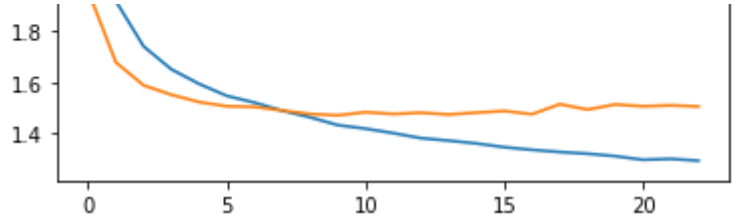
Epoch 00013: val_accuracy improved from 0.56352 to 0.56702, saving model to weights_max.hdf5
Epoch 14/100
126170/126170 [==============================] - 192s 2ms/step - loss: 1.3696 - accuracy: 0.5788 - val_loss: 1.4727 - val_accuracy: 0.5664

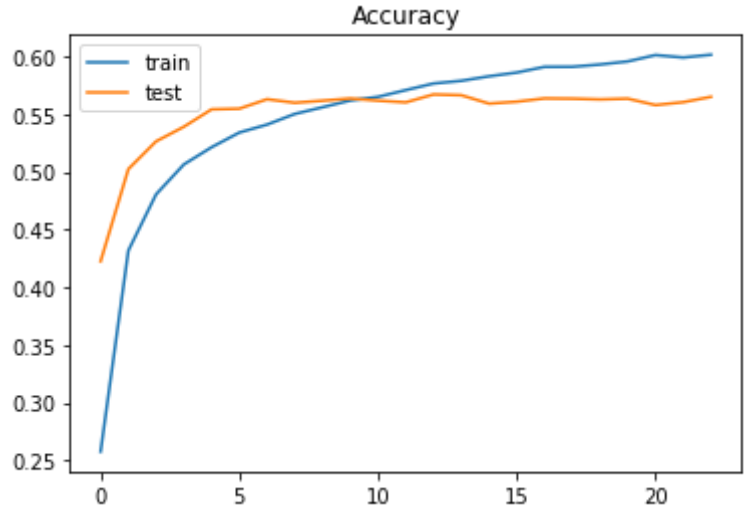
Epoch 00014: val_accuracy did not improve from 0.56702
Epoch 15/100
126170/126170 [==============================] - 194s 2ms/step - loss: 1.3584 - accuracy: 0.5828 - val_loss: 1.4799 - val_accuracy: 0.5591

Epoch 00015: val_accuracy did not improve from 0.56702
Epoch 16/100
126170/126170 [==============================] - 195s 2ms/step - loss: 1.3437 - accuracy: 0.5860 - val loss: 1.4864 - val accuracy: 0.5607

```
Epoch 00016: val_accuracy did not improve from 0.56702
Epoch 17/100
126170/126170 [==============================] - 196s 2ms/step - loss: 1.3333 - accuracy: 0.5909 - val_loss: 1.4736 - val_accuracy: 0.5636

Epoch 00017: val_accuracy did not improve from 0.56702
Epoch 18/100
126170/126170 [==============================] - 195s 2ms/step - loss: 1.3247 - accuracy: 0.5910 - val_loss: 1.5130 - val_accuracy: 0.5634

Epoch 00018: val_accuracy did not improve from 0.56702
Epoch 19/100
126170/126170 [==============================] - 196s 2ms/step - loss: 1.3181 - accuracy: 0.5930 - val_loss: 1.4921 - val_accuracy: 0.5627

Epoch 00019: val_accuracy did not improve from 0.56702
Epoch 20/100
126170/126170 [==============================] - 196s 2ms/step - loss: 1.3084 - accuracy: 0.5956 - val_loss: 1.5120 - val_accuracy: 0.5634

Epoch 00020: val_accuracy did not improve from 0.56702
Epoch 21/100
126170/126170 [==============================] - 196s 2ms/step - loss: 1.2941 - accuracy: 0.6011 - val_loss: 1.5050 - val_accuracy: 0.5578

Epoch 00021: val_accuracy did not improve from 0.56702
Epoch 22/100
126170/126170 [==============================] - 195s 2ms/step - loss: 1.2976 - accuracy: 0.5990 - val_loss: 1.5085 - val_accuracy: 0.5603

Epoch 00022: val_accuracy did not improve from 0.56702
Epoch 23/100
126170/126170 [==============================] - 196s 2ms/step - loss: 1.2905 - accuracy: 0.6014 - val_loss: 1.5040 - val_accuracy: 0.5649

Epoch 00023: val_accuracy did not improve from 0.56702
Epoch 00023: early stopping
60089/60089 [==============================] - 77s 1ms/step
loss= 1.483303352273126 accuracy= 0.5668591856956482
```







---Generating with seed:"f taking the side of criminals; a sort of socialistic sympat"
------ temperature: 0.2
hy and the germans of the same time the same times the
same time the same time the same time the same time the same times the
power of the same times as a personal in the existence of the
soul of the sense of the same time the same time the same time the
precession of the hondress of the same times and of the soul of the
philosopher of the same time the same time the degree of the same
times, and

------ temperature: 0.5
in the goes" of man and pride him, and by the person of the
philosopher in the same comparison and even the sense of the
philosophers as the most religion of a counter-alth times, and
contrary, the procession of the propertion of the senses and here as cho
get the present soul of the same different man who has hitherto been
being to the truth, the german selvation of it, as it is a man who has
don

------ temperature: 1.0
otion.=--how can deterl sign on suniel, under the european
trughters.=

131. wie lovers, but! is a broagere of philosophy, to little opposingation lithle by it there
aware, not-post, it once does-ween something (us th,/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:68: RuntimeWarning: divide b
 in quising to have understand enough; however, we intendlicc
to contemption as a
perhaps, phoe oatwed, where is generage: by why has weach the
change of has much elevated, he has cr

------ temperature: 1.2
uelry itself! on looked that
and ictual, togerld, in hos etione as arthfuls of case through german
philosophers evid and a stizeist and
ngusrinu. so go sense is
arw hrongs the shamed word
itnted, takes, our extented:
about pla hes others have the dughting of himl wild teve.

7

hsensuh greek, lumber, daring sy that theoughout leagingily, thouralitity the to--possible that
music
science.

13

=i