Installing dependencies

```
pip install datasets
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting datasets
  Downloading datasets-2.11.0-py3-none-any.whl (468 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 468.7/468.7 kB 13.9 MB/s eta 0:00:00
Collecting multiprocess
  Downloading multiprocess-0.70.14-py39-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.9/132.9 kB 18.8 MB/s eta 0:00:00
Collecting dill<0.3.7,>=0.3.0
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.5/110.5 kB 16.7 MB/s eta 0:00:00
Collecting aiohttp
  Downloading aiohttp-3.8.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 51.5 MB/s eta 0:00:00
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (6.0)
Requirement already satisfied: fsspec[http]>=2021.11.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (2023.4.0)
Requirement already satisfied: pyarrow>=8.0.0 in /usr/local/lib/python3.9/dist-packages (from datasets) (9.0.0)
Collecting responses<0.19
  Downloading responses-0.18.0-py3-none-any.whl (38 kB)
Collecting huggingface-hub<1.0.0,>=0.11.0
  Downloading huggingface_hub-0.13.4-py3-none-any.whl (200 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.1/200.1 kB 27.0 MB/s eta 0:00:00
Requirement already satisfied: packaging in /usr/local/lib/python3.9/dist-packages (from datasets) (23.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from datasets) (1.5.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.9/dist-packages (from datasets) (2.27.1)
Collecting xxhash
  Downloading xxhash-3.2.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (212 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 212.2/212.2 kB 26.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-packages (from datasets) (1.22.4)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.9/dist-packages (from datasets) (4.65.0)
Collecting async-timeout<5.0,>=4.0.0a3
  Downloading async_timeout-4.0.2-py3-none-any.whl (5.8 kB)
Collecting aiosignal>=1.1.2
  Downloading aiosignal-1.3.1-py3-none-any.whl (7.6 kB)
Collecting frozenlist>=1.1.1
  Downloading frozenlist-1.3.3-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (158 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 158.8/158.8 kB 22.1 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4.0,>=2.0 in /usr/local/lib/python3.9/dist-packages (from aiohttp->datasets) (2.0.12)
Collecting yarl<2.0,>=1.0
  Downloading yarl-1.8.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (264 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 264.6/264.6 kB 32.2 MB/s eta 0:00:00
Collecting multidict<7.0,>=4.5
  Downloading multidict-6.0.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (114 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.2/114.2 kB 16.1 MB/s eta 0:00:00
```

```
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.9/dist-packages (from aiohttp->datasets) (22.2.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0.0,>=0.11.0->datas
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from huggingface-hub<1.0.0,>=0.11.0->datasets) (3.11.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests>=2.19.0->datasets) (2022.12.7)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->datasets) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas->datasets) (1.16.0)
Installing collected packages: xxhash, multidict, frozenlist, dill, async-timeout, yarl, responses, multiprocess, huggingface-hub, aiosignal, a
Successfully installed aiohttp-3.8.4 aiosignal-1.3.1 async-timeout-4.0.2 datasets-2.11.0 dill-0.3.6 frozenlist-1.3.3 huggingface-hub-0.13.4 mul
```

Importing the required libraries

```
from keras.utils import pad_sequences
from keras.layers import Embedding, LSTM, Dense, Dropout
from keras.preprocessing.text import Tokenizer
from keras.callbacks import EarlyStopping
from keras.models import Sequential
import keras.utils as ku
from keras.optimizers import Adam
import tensorflow
from numpy.random import seed
tensorflow.random.set_seed(2)
seed(1)
import pandas as pd
import numpy as np
import string, os
import re
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)
from datasets import load_dataset
```

Loading the fanfiction dataset

```
dataset =  load_dataset('jeremyf/fanfiction_z')
```

```
Downloading readme: 100%                                                203/203 [00:00<00:00, 16.7kB/s]
Downloading and preparing dataset json/jeremyf--fanfiction_z to /root/.cache/huggingface/datasets/jerer
Downloading data files: 100%                                            1/1 [00:01<00:00, 1.43s/it]
Downloading data: 100%                                                  14.6M/14.6M [00:00<00:00, 28.5MB/s]
Extracting data files: 100%                                             1/1 [00:00<00:00, 59.97it/s]
Dataset json downloaded and prepared to /root/.cache/huggingface/datasets/jeremyf___json/jeremyf--fanfi
```

Extracting the required stories for processing

```
story_list = [[i] for i in list(dataset['train']['story'])]
story_list = story_list[:25]
```

Data pre-processing

```
cleaned_stories = []
for story in story_list:
  story = " ".join(story[0].split("\n\n"))
  story = story.lower()
  story = story.translate(str.maketrans('', '', string.punctuation))
  story = story.replace("\nend file\n",'')
  cleaned_stories.append(story)

print(cleaned_stories[0])
```

```
just a question what i thought when i first read this was that b was a guy who else thought that plz leave in the reviews and thanks so much fo
```

Tokenization of the training data set

```
tokenizer = Tokenizer()

def get_sequence_of_tokens(cleaned_stories):
    ## tokenization
    tokenizer.fit_on_texts(cleaned_stories)
    total_words = len(tokenizer.word_index) + 1

    ## convert data to sequence of tokens
    input_sequences = []
```

```
    for line in cleaned_stories:
        token_list = tokenizer.texts_to_sequences([line])[0]
        for i in range(1, len(token_list)):
            n_gram_sequence = token_list[:i+1]
            input_sequences.append(n_gram_sequence)
    return input_sequences, total_words

inp_sequences, total_words = get_sequence_of_tokens(cleaned_stories)
print(inp_sequences[:10])
```

```
[[45, 5], [45, 5, 508], [45, 5, 508, 33], [45, 5, 508, 33, 7], [45, 5, 508, 33, 7, 122], [45, 5, 508, 33, 7, 122, 50], [45, 5, 508, 33, 7, 122,
```

Padding of the sequences for further processing

```
def generate_padded_sequences(input_sequences):
    max_sequence_len = max([len(x) for x in input_sequences])
    input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))

    predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
    label = ku.to_categorical(label, num_classes=total_words)
    return predictors, label, max_sequence_len

predictors, label, max_sequence_len = generate_padded_sequences(inp_sequences)
```

Creating the LSTM model

```
def create_model(max_sequence_len, total_words):
    input_len = max_sequence_len - 1
    model = Sequential()

    # Add Input Embedding Layer
    model.add(Embedding(total_words, 10, input_length=input_len))

    # Add Hidden Layer 1 - LSTM Layer
    model.add(LSTM(100))
    model.add(Dropout(0.1))

    # Add Output Layer
    model.add(Dense(total_words, activation='softmax'))

    opt = Adam(learning_rate=0.001)
```

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

model = create_model(max_sequence_len, total_words)
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 7879, 10)          39520

 lstm (LSTM)                 (None, 100)               44400

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 3952)              399152

=================================================================
Total params: 483,072
Trainable params: 483,072
Non-trainable params: 0
_________________________________________________________________
```

Fitting the model on training data

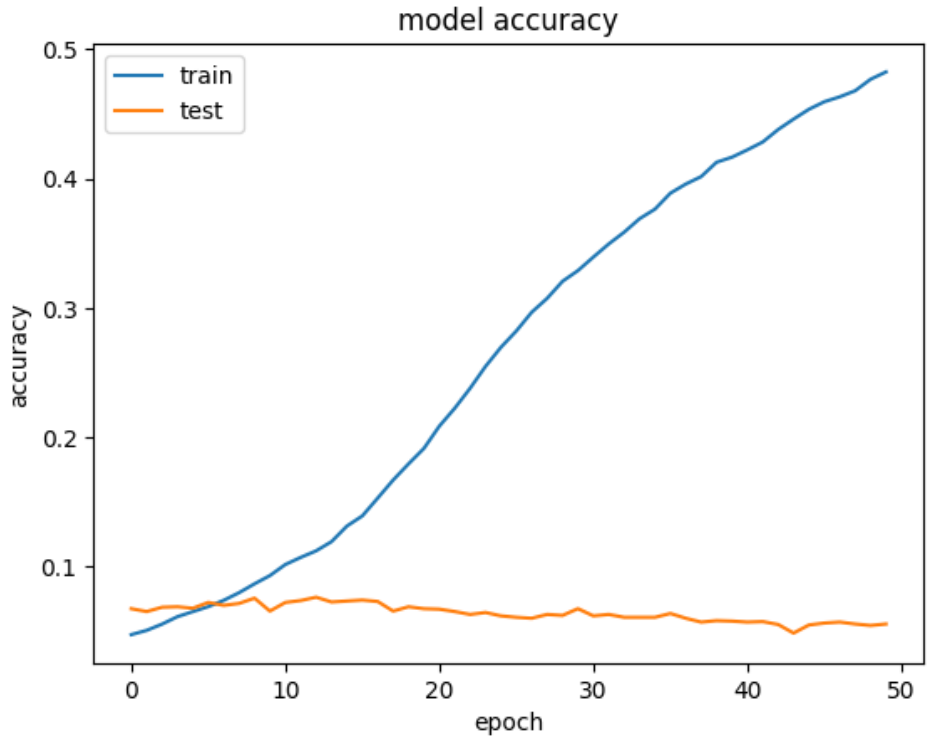
```
history = model.fit(predictors, label, validation_split=0.10, epochs=50, verbose=True)
```

```
761/761 [==============================] - 174s 228ms/step - loss: 3.0990 - accuracy: 0.3392 - val_loss: 9.2618 - val_accuracy: 0.0621
Epoch 32/50
761/761 [==============================] - 174s 228ms/step - loss: 3.0291 - accuracy: 0.3494 - val_loss: 9.3347 - val_accuracy: 0.0632
Epoch 33/50
761/761 [==============================] - 174s 228ms/step - loss: 2.9766 - accuracy: 0.3585 - val_loss: 9.3847 - val_accuracy: 0.0610
Epoch 34/50
761/761 [==============================] - 173s 228ms/step - loss: 2.9251 - accuracy: 0.3689 - val_loss: 9.4238 - val_accuracy: 0.0610
Epoch 35/50
761/761 [==============================] - 173s 228ms/step - loss: 2.8728 - accuracy: 0.3763 - val_loss: 9.4735 - val_accuracy: 0.0610
Epoch 36/50
761/761 [==============================] - 174s 228ms/step - loss: 2.8138 - accuracy: 0.3886 - val_loss: 9.5441 - val_accuracy: 0.0640
Epoch 37/50
761/761 [==============================] - 174s 228ms/step - loss: 2.7624 - accuracy: 0.3956 - val_loss: 9.6156 - val_accuracy: 0.0603
Epoch 38/50
761/761 [==============================] - 174s 228ms/step - loss: 2.7254 - accuracy: 0.4014 - val_loss: 9.6970 - val_accuracy: 0.0573
Epoch 39/50
761/761 [==============================] - 174s 228ms/step - loss: 2.6736 - accuracy: 0.4125 - val_loss: 9.7652 - val_accuracy: 0.0584
Epoch 40/50
761/761 [==============================] - 173s 228ms/step - loss: 2.6347 - accuracy: 0.4164 - val_loss: 9.7786 - val_accuracy: 0.0580
Epoch 41/50
761/761 [==============================] - 174s 228ms/step - loss: 2.5987 - accuracy: 0.4221 - val_loss: 9.8279 - val_accuracy: 0.0573
Epoch 42/50
761/761 [==============================] - 174s 228ms/step - loss: 2.5583 - accuracy: 0.4281 - val_loss: 9.8879 - val_accuracy: 0.0577
Epoch 43/50
761/761 [==============================] - 174s 229ms/step - loss: 2.5133 - accuracy: 0.4378 - val_loss: 9.9257 - val_accuracy: 0.0555
Epoch 44/50
761/761 [==============================] - 174s 228ms/step - loss: 2.4816 - accuracy: 0.4458 - val_loss: 10.0159 - val_accuracy: 0.0488
Epoch 45/50
761/761 [==============================] - 174s 228ms/step - loss: 2.4313 - accuracy: 0.4534 - val_loss: 10.0566 - val_accuracy: 0.0551
Epoch 46/50
761/761 [==============================] - 174s 228ms/step - loss: 2.4027 - accuracy: 0.4592 - val_loss: 10.1190 - val_accuracy: 0.0566
Epoch 47/50
761/761 [==============================] - 174s 228ms/step - loss: 2.3782 - accuracy: 0.4630 - val_loss: 10.1586 - val_accuracy: 0.0573
Epoch 48/50
761/761 [==============================] - 174s 228ms/step - loss: 2.3442 - accuracy: 0.4677 - val_loss: 10.2139 - val_accuracy: 0.0558
Epoch 49/50
761/761 [==============================] - 174s 228ms/step - loss: 2.3114 - accuracy: 0.4766 - val_loss: 10.2638 - val_accuracy: 0.0547
Epoch 50/50
761/761 [==============================] - 174s 228ms/step - loss: 2.2728 - accuracy: 0.4823 - val_loss: 10.2739 - val_accuracy: 0.0558
```

Plotting the accuracy and loss graphs of the model

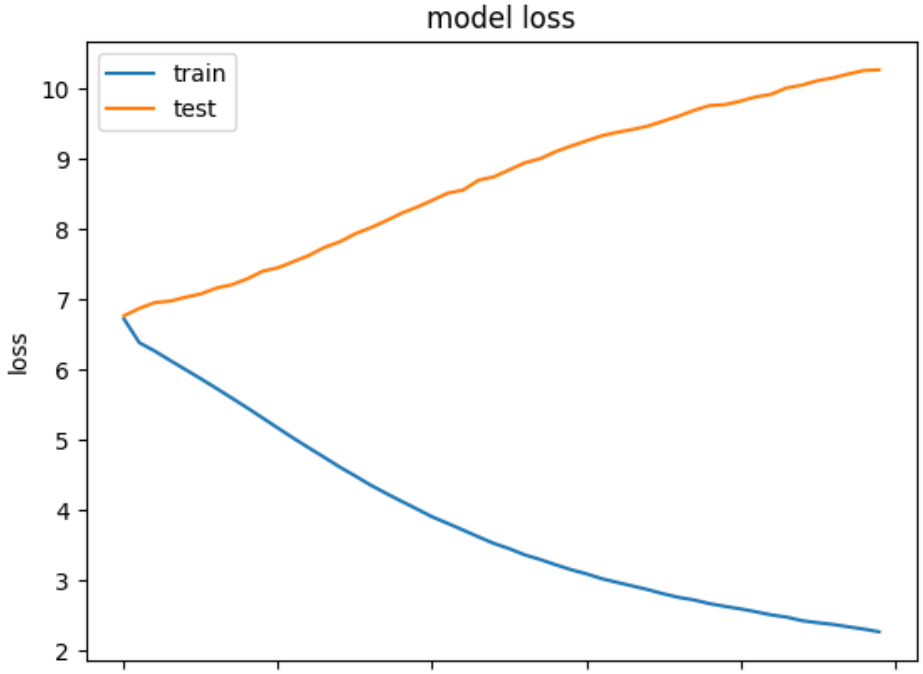
```
import matplotlib.pyplot as plt


# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('accuracy.png', bbox_inches='tight')
```




```
<Figure size 640x480 with 0 Axes>
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.savefig('loss .png', bbox_inches='tight')
```



Save the model

```
<Figure size 640x480 with 0 Axes>
```

```
model.save('lstm_working_new.h5')
```

Loading the saved model

```
new_model = tensorflow.keras.models.load_model('lstm_working_new.h5')

# Show the model architecture
new_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 7879, 10)          39520

 lstm (LSTM)                 (None, 100)               44400

 dropout (Dropout)           (None, 100)               0
```

```
 dense (Dense)               (None, 3952)              399152

=================================================================
Total params: 483,072
Trainable params: 483,072
Non-trainable params: 0
_________________________________________________________________
```

Generating stories using the trained model

```
def generate_text(seed_text, next_words, model, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted = np.argmax(model.predict(token_list, verbose=0),axis=1)
        output_word = ""
        for word,index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " "+output_word
    return seed_text.title()
```

A few prompts for story generation

```
print (generate_text("Just a question", 50, new_model, max_sequence_len))
```

```
Just A Question 2 Aybasbtu Game Script Zero Wing 2 One I Am Not Making My Characters My Doom I Cant Take It Let That Holy Endurance 510 Hound T
```

```
print (generate_text("United States", 50, new_model, max_sequence_len))
```

```
United States Is Tea Of The Early Signs Of The Experiment Pray The Experiment Is The Experiment That Still Watched The Single Friend Summer Is
```

```
print (generate_text("Alice in wonderland", 50, new_model, max_sequence_len))
```

```
Alice In Wonderland Your Remains Rating Zero Wing 2 One I Can Explain Gifts Not A Scientific Man I 1 You Gentlemen Up It Are Like A Lot Oneshot
```

perplexity calculation

```
evaluate_stories = dataset['train']['story'][26:28]
print(len(evaluate_stories))
```

```
2
```

```
cleaned_stories_eval = []
for story in evaluate_stories:
  story = " ".join(story.split("\n\n"))
  story = story.lower()
  story = story.translate(str.maketrans('', '', string.punctuation))
  story = story.replace("\nend file\n",'')
  cleaned_stories_eval.append(story)

print(cleaned_stories_eval[0])
```

```
the vita nova incident i climbed up a large rock with a hunting rifle slung across my back it was early in the morning and i was searching for
```

```
inp_sequences_eval, total_words_eval = get_sequence_of_tokens(cleaned_stories_eval)
print(inp_sequences_eval[:10])
```

```
[[1, 1149], [1, 1149, 1150], [1, 1149, 1150, 1063], [1, 1149, 1150, 1063, 4], [1, 1149, 1150, 1063, 4, 1862], [1, 1149, 1150, 1063, 4, 1862, 28
```

```
predictors_eval, label_eval, max_sequence_len_eval = generate_padded_sequences(inp_sequences_eval)
```

```
test_loss, test_accuracy = new_model.evaluate(predictors, label)
print('Test loss:', test_loss)
print('Test accuracy:', test_accuracy)
```

```
846/846 [==============================] - 92s 104ms/step - loss: 2.7703 - accuracy: 0.5181
Test loss: 2.770331382751465
Test accuracy: 0.5180624723434448
```

```
print(tensorflow.math.exp(test_loss))
```

```
tf.Tensor(15.963923, shape=(), dtype=float32)
```

✓ 0s completed at 11:29 AM