Chiranjeev_113_LAB7

November 19, 2024

0.0.1 Import statements

```
[ ]: import numpy as np
     import pandas as pd

     import matplotlib.pyplot as plt
     import re

     from tensorflow.keras import layers
     from tensorflow.keras.preprocessing.text import Tokenizer
     from tensorflow.keras.preprocessing.sequence import pad_sequences
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Embedding, LSTM, Dropout, Dense
```

0.0.2 1. Dataset Preparation:

Import Dataset

```
[ ]: from google.colab import drive
     import pandas as pd

     drive.mount('/content/drive')

     dataset = pd.read_csv('/content/drive/MyDrive/PoetryFoundationData.csv')

     dataset = dataset.drop_duplicates(subset=['Poem'])

     dataset.head()
```

```
Mounted at /content/drive
```

```
[ ]:    Unnamed: 0                                            Title  \
     0           0  \r\r\n                   Objects Used to Prop…
     1           1  \r\r\n                   The New Church\r\r\n…
     2           2  \r\r\n                   Look for Me\r\r\n   …
     3           3  \r\r\n                   Wild Life\r\r\n     …
     4           4  \r\r\n                   Umbrella\r\r\n      …

                                                   Poem              Poet Tags
     0  \r\r\nDog bone, stapler,\r\r\ncribbage board, …  Michelle Menting  NaN
```

```
1  \r\r\nThe old cupola glinted above the clouds,…     Lucia Cherciu  NaN
2  \r\r\nLook for me under the hood\r\r\nof that …       Ted Kooser  NaN
3  \r\r\nBehind the silo, the Mother Rabbit\r\r\n…   Grace Cavalieri  NaN
4  \r\r\nWhen I push your button\r\r\nyou fly off…      Connie Wanek  NaN
```

Dataset Columns

[ ]:
```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13754 entries, 0 to 13833
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  13754 non-null  int64
 1   Title       13754 non-null  object
 2   Poem        13754 non-null  object
 3   Poet        13754 non-null  object
 4   Tags        12854 non-null  object
dtypes: int64(1), object(4)
memory usage: 644.7+ KB
```

Concatenate multiple poems into a single text corpus, separating them by newline characters for clarity

[ ]:
```python
df_cleaned = dataset.dropna(subset=['Tags'])
df_cleaned = df_cleaned.drop_duplicates(subset=['Poem'])

corpus = "\n".join(df_cleaned['Poem'].head(100).values)
# print(corpus)
```

0.0.3 2. Data Preprocessing:

Convert the text to lowercase and remove special characters or punctuation if necessary.

[ ]:
```python
cleaned_text = corpus.lower()

cleaned_text = re.sub(r'[^a-zA-Z\s,\'"-.]', '', cleaned_text)
cleaned_text = re.sub(r'\r\r\n\n\r\r\n', '\n', cleaned_text)
cleaned_text = re.sub(r'[\r\r]', '', cleaned_text)
cleaned_text = re.sub(r'\n \n', '\n\n', cleaned_text)

cleaned_text = re.sub(r'\n{2,}', '\n\n', cleaned_text)
cleaned_text = re.sub(r' +', ' ', cleaned_text)
cleaned_text = cleaned_text.strip()

print(cleaned_text[:5000])
print("============")
print(repr(cleaned_text[:5000]))
```

```
corpus = cleaned_text
```

```
invisible fish swim this ghost ocean now described by waves of sand, by water-
worn rock. soon the fish will learn to walk. then humans will come ashore and
paint dreams on the dying stone. then later, much later, the ocean floor will be
punctuated by chevy trucks, carrying the dreamers decendants, who are going to
the store.
dont bother the earth spirit who lives here. she is working on a story. it is
the oldest story in the world and it is delicate, changing. if she sees you
watching she will invite you in for coffee, give you warm bread, and you will be
obligated to stay and listen. but this is no ordinary story. you will have to
endure earthquakes, lightning, the deaths of all those you love, the most
blinding beauty. its a story so compelling you may never want to leave this is
how she traps you. see that stone finger over there that is the only one who
ever escaped.
hour in which i consider hydrangea, a salt or sand plant, varietal, the question
of varietals, the diet of every mother i know, pounds feels like , i have lost i
have lost, yes, a sense of my own possible beauty, grown external, i externalize
beauty. beauty occurs on the surface of plants the sun darkens the skin of my
child, he is so small, he is beautiful (i can see it is obvious) and everything
about him is beautiful. his hand swells from the bite spread of some insects
venom because he is small. he appears to feel nothing. he smashes his skull
against the floor. he screams. i hold him in my lap on the kitchen floor in
front of an open freezer, pressing a pack of frozen clay against his forehead.
he likes the cold. i see it is so obvious. hydrangea. when i move, when i walk
pushing my childs stroller (it is both walking and pushing or hauling,
sometimes, also, lifting it is having another body, an adjunct body composed of
errand and weight and tenderness and no small amount of power), i imagine i can
feel this small amount of weight, this pounds like , interfering with the twitch
of every muscle in my body. as an object, a mother is confusing, a middle-aged
mother with little spare flesh, i feel every inch of major muscle pulling
against gravity and against the weight of my child, now sleeping. this is the
hour for thinking hydrangea. let no man look at me. i stop to brush the drowsy
childs little eye. his face. he barely considers his mother. i am all around
him. why should he consider what is all around him perhaps what is missing is a
subtle power of differentiation. i am in, therefore, a time of mass
apprehensions.
my fathers body is a map
a record of his journey

he carries a bullet
lodged in his left thigh
there is a hollow where it entered
a protruding bump where it sleeps
the doctors say it will never awaken

it is the one souvenir he insists on keeping
```

mother has her own opinionsb ca con inyour father is crazy

as a child
i wanted a scar just like my fathers
bold and appalling a mushroom explosion
that said i too was at war
instead i settled for a grain of rice
a scar so small look closely there
here between the eyes
a bit to the right
there on the bridge of my nose

father says i was too young to remember
it happened while i was sleeping
leaking roof the pounding rain
drop after drop after drop

it has long been forgotten this practice of the mother
weaning a child she crushes the seeds of a green
chili rubs it to her nipple what the child feels
she too will share in this act of love
my own mother says it was not meant
to be cruel when cruelty she tells me
is a childs lips torn from breast as proof
back home the women wear teeth marks

why are you still seventeen
and drifting like a dog after dark,
dragging a shadow youve found

put it back where it belongs,
and that bend of river, too. thats not the road
you want, though you have it to yourself.

gone are the cars that crawl to town
from the reactors, a parade of insects, metallic,
fuming along the one four-lane street.

the poplars of the shelterbelt lean away
from the bypass that never had much to pass by
but coyote and rabbitbrush.

pinpricks stabbed in a map too dark to read
i stared at stars light-years away.
listen. that hissing just a sprinkler

damping down yesterday until its today.
the cottonwoods shiver, or i do,

every leaf rustling as if its the one

about to tear itself, not i.
memory takes the graveyard shift.

yes, your childhood now a legend of fountains
 jorge gulln

yes, your childhood, now a legend
gone to weeds, still remembers the gray road
that set out to cross the desert of the future.

and how, always just ahead,
gray water glittered, happy to be just a mirage.
who steps off the gray bus at the depot

sidewalks shudder all the way home.
blinds close their scratchy eyes.
who settles in your old room

sniffy air sprawls as if it owns the place,
and now your teenage secrets have no one to tell.
for the spider laying claim to the corner,

there is a stickiness to spin, that the living may beg
to be wrapped in silk and devoured,
leaving not even the flinch f
============
'invisible fish swim this ghost ocean now described by waves of sand, by water-worn rock. soon the fish will learn to walk. then humans will come ashore and paint dreams on the dying stone. then later, much later, the ocean floor will be punctuated by chevy trucks, carrying the dreamers decendants, who are going to the store.\ndont bother the earth spirit who lives here. she is working on a story. it is the oldest story in the world and it is delicate, changing. if she sees you watching she will invite you in for coffee, give you warm bread, and you will be obligated to stay and listen. but this is no ordinary story. you will have to endure earthquakes, lightning, the deaths of all those you love, the most blinding beauty. its a story so compelling you may never want to leave this is how she traps you. see that stone finger over there that is the only one who ever escaped.\nhour in which i consider hydrangea, a salt or sand plant, varietal, the question of varietals, the diet of every mother i know, pounds feels like , i have lost i have lost, yes, a sense of my own possible beauty, grown external, i externalize beauty. beauty occurs on the surface of plants the sun darkens the skin of my child, he is so small, he is beautiful (i can see it is obvious) and everything about him is beautiful. his hand swells from the bite spread of some insects venom because he is small. he appears to feel nothing. he smashes his skull against the floor. he screams. i hold him in my lap on the kitchen floor in front of an open freezer, pressing a pack of frozen clay

against his forehead. he likes the cold. i see it is so obvious. hydrangea. when i move, when i walk pushing my childs stroller (it is both walking and pushing or hauling, sometimes, also, lifting it is having another body, an adjunct body composed of errand and weight and tenderness and no small amount of power), i imagine i can feel this small amount of weight, this pounds like , interfering with the twitch of every muscle in my body. as an object, a mother is confusing, a middle-aged mother with little spare flesh, i feel every inch of major muscle pulling against gravity and against the weight of my child, now sleeping. this is the hour for thinking hydrangea. let no man look at me. i stop to brush the drowsy childs little eye. his face. he barely considers his mother. i am all around him. why should he consider what is all around him perhaps what is missing is a subtle power of differentiation. i am in, therefore, a time of mass apprehensions.\nmy fathers body is a map\na record of his journey\n\nhe carries a bullet\nlodged in his left thigh\nthere is a hollow where it entered\na protruding bump where it sleeps\nthe doctors say it will never awaken\n\nit is the one souvenir he insists on keeping\nmother has her own opinionsb ca con inyour father is crazy\n\nas a child\ni wanted a scar just like my fathers\nbold and appalling a mushroom explosion\nthat said i too was at war\ninstead i settled for a grain of rice\na scar so small look closely there\nhere between the eyes\na bit to the right\nthere on the bridge of my nose\n\nfather says i was too young to remember\nit happened while i was sleeping\nleaking roof the pounding rain\ndrop after drop after drop\n\nit has long been forgotten this practice of the mother\nweaning a child she crushes the seeds of a green\nchili rubs it to her nipple what the child feels\nshe too will share in this act of love\nmy own mother says it was not meant\nto be cruel when cruelty she tells me\nis a childs lips torn from breast as proof\nback home the women wear teeth marks\n\nwhy are you still seventeen\nand drifting like a dog after dark,\ndragging a shadow youve found\n\nput it back where it belongs,\nand that bend of river, too. thats not the road\nyou want, though you have it to yourself.\n\ngone are the cars that crawl to town\nfrom the reactors, a parade of insects, metallic,\nfuming along the one four-lane street.\n\nthe poplars of the shelterbelt lean away\nfrom the bypass that never had much to pass by\nbut coyote and rabbitbrush.\n\npinpricks stabbed in a map too dark to read\ni stared at stars light-years away.\nlisten. that hissing just a sprinkler\n\ndamping down yesterday until its today.\nthe cottonwoods shiver, or i do,\nevery leaf rustling as if its the one\n\nabout to tear itself, not i.\nmemory takes the graveyard shift.\n\nyes, your childhood now a legend of fountains\n jorge gulln\n\nyes, your childhood, now a legend\ngone to weeds, still remembers the gray road\nthat set out to cross the desert of the future.\n\nand how, always just ahead,\ngray water glittered, happy to be just a mirage.\nwho steps off the gray bus at the depot\n\nsidewalks shudder all the way home.\nblinds close their scratchy eyes.\nwho settles in your old room\n\nsniffy air sprawls as if it owns the place,\nand now your teenage secrets have no one to tell.\nfor the spider laying claim to the corner,\n\nthere is a stickiness to spin, that the living may beg\nto be wrapped in silk and devoured,\nleaving not even the flinch f'

[ ]:
```
paragraphs = corpus.split("\n\n")

# Now, split each paragraph into lines (split by single \n)
lines_in_paragraphs = [paragraph.split("\n") for paragraph in paragraphs]

temp_lines_in_paragraphs = []

lines = 0
for para in lines_in_paragraphs:
    temp = []
    for line in para:
      line = line.strip()
      if line == "":
        continue
      temp.append(line)
    if len(temp) > 0:
      temp_lines_in_paragraphs.append(temp)

    lines += len(para)

print(f"Paragraphs: {len(paragraphs)}")
print(f"Lines: {lines}")
```

```
Paragraphs: 612
Lines: 2790
```

Tokenize the text (e.g., convert each word to a unique integer).

[ ]:
```
flattened_lines = [line for paragraph in temp_lines_in_paragraphs for line in␣
  ↪paragraph]

# Tokenize the corpus
tokenizer = Tokenizer()
tokenizer.fit_on_texts(flattened_lines)

# Get the vocabulary size
vocab_size = len(tokenizer.word_index) + 1  # Add 1 for padding token
print("Vocabulary size:", vocab_size)

# Convert the corpus into a sequence of integers (tokens)
sequences = tokenizer.texts_to_sequences(flattened_lines)
print(sequences[:20])  # Print the first 20 tokenized words
```

```
Vocabulary size: 4685
[[528, 370, 529, 24, 701, 163, 62, 1013, 36, 530, 3, 371, 36, 121, 1014, 702,
703, 1, 370, 25, 704, 4, 222, 82, 705, 25, 122, 1702, 6, 531, 284, 12, 1, 317,
706, 82, 223, 176, 223, 1, 163, 199, 25, 32, 1015, 36, 1703, 1016, 1704, 1,
1705, 1706, 47, 50, 106, 4, 1, 532], [107, 1707, 1, 135, 1708, 47, 247, 98, 33,
10, 1017, 12, 2, 177, 11, 10, 1, 1709, 177, 5, 1, 99, 6, 11, 10, 1710, 1711, 45,
33, 1712, 9, 533, 33, 25, 1713, 9, 5, 17, 318, 319, 9, 372, 1018, 6, 9, 25, 32,
1714, 4, 430, 6, 373, 30, 24, 10, 46, 1019, 177, 9, 25, 40, 4, 1715, 1716, 534,
1, 707, 3, 37, 224, 9, 123, 1, 374, 1717, 285, 35, 2, 177, 48, 1718, 9, 248, 83,
124, 4, 286, 24, 10, 53, 33, 1719, 9, 85, 13, 706, 708, 90, 56, 13, 10, 1, 94,
31, 47, 200, 1020], [535, 5, 125, 7, 1021, 709, 2, 710, 28, 371, 711, 1720, 1,
536, 3, 1721, 1, 1722, 3, 136, 80, 7, 100, 1022, 537, 14, 7, 40, 154, 7, 40,
154, 201, 2, 431, 3, 8, 137, 712, 285, 538, 1723, 7, 1724, 285, 285, 1023, 12,
1, 713, 3, 1725, 1, 249, 1726, 1, 202, 3, 8, 155, 22, 10, 48, 126, 22, 10, 225,
7, 64, 85, 11, 10, 1024, 6, 138, 78, 75, 10, 225, 16, 139, 1727, 34, 1, 1728,
1025, 3, 116, 1026, 1729, 86, 22, 10, 126, 22, 1027, 4, 149, 150, 22, 1730, 16,
1028, 140, 1, 199, 22, 1731, 7, 203, 75, 5, 8, 539, 12, 1, 540, 199, 5, 375, 3,
38, 320, 1732, 1733, 2, 1734, 3, 541, 1735, 140, 16, 432, 22, 1736, 1, 321, 7,
85, 11, 10, 48, 1024, 709, 49, 7, 1029, 49, 7, 222, 1030, 8, 542, 1737, 11, 10,
204, 543, 6, 1030, 28, 1738, 205, 544, 545, 11, 10, 546, 164, 58, 38, 1739, 58,
1740, 3, 1741, 6, 322, 6, 1742, 6, 46, 126, 1031, 3, 323, 7, 324, 7, 64, 149,
24, 126, 1031, 3, 322, 24, 1022, 14, 1743, 15, 1, 1744, 3, 136, 714, 5, 8, 58,
18, 38, 1032, 2, 80, 10, 1745, 2, 1033, 1746, 80, 15, 190, 1747, 715, 7, 149,
136, 1748, 3, 547, 714, 716, 140, 1749, 6, 140, 1, 322, 3, 8, 155, 62, 376, 24,
10, 1, 535, 17, 717, 709, 91, 46, 72, 141, 27, 29, 7, 226, 4, 1750, 1, 1034,
542, 190, 250, 16, 95, 22, 1035, 1751, 16, 80, 7, 101, 37, 108, 75, 206, 377,
22, 1021, 43, 10, 37, 108, 75, 548, 43, 10, 1036, 10, 2, 1752, 323, 3, 1753, 7,
101, 5, 718, 2, 76, 3, 433, 1754], [8, 434, 58, 10, 2, 178], [2, 1755, 3, 16,
719], [22, 1037, 2, 378], [1756, 5, 16, 179, 1757], [56, 10, 2, 1758, 51, 11,
1038], [2, 1759, 1760, 51, 11, 1039], [1, 1040, 70, 11, 25, 83, 1041], [11, 10,
1, 31, 1761, 22, 1762, 12, 720], [80, 65, 23, 137, 1763, 1764, 1765, 1766, 117,
10, 1767], [18, 2, 155], [7, 207, 2, 721, 102, 14, 8, 434], [1042, 6, 1768, 2,
1769, 1770], [13, 57, 7, 66, 19, 27, 379], [722, 7, 1043, 17, 2, 1771, 3, 723],
[2, 721, 48, 126, 141, 1044, 56], [98, 227, 1, 129], [2, 724, 4, 1, 156]]
```

Use a sliding window to create sequences of words for the LSTM model. For example, if n=5, create sequences of 5 words with the 6th word as the target.

[ ]:
```
sequence_length = 5  # Length of the sequence

# Create input-output pairs (X, y) based on the sliding window
X, y = [], []

for line in sequences:
    for i in range(sequence_length, len(line)):
        X.append(line[i-sequence_length:i])
        y.append(line[i])

print("X shape:", len(X))
print("y shape:", len(y))
```

```
X shape: 7098
y shape: 7098
```

Pad the sequences so that they all have the same length.

[ ]:
```
X = pad_sequences(X, maxlen = sequence_length, padding='pre')
y = np.array(y)
print("Padded X shape:", X.shape)
```

```
Padded X shape: (7098, 5)
```

0.0.4 3. LSTM Model Development

o Define an LSTM model with the following structure:
- An embedding layer with an appropriate input dimension (based on vocabulary size) and output dimension (e.g., 100).
- One or two LSTM layers with 100 units each.
- A dropout layer with a rate of 0.2 to prevent overfitting.
- A dense output layer with softmax activation for word prediction.

[ ]:
```
# Define the LSTM model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=100, input_length=X.
  ↪shape[1]))
model.add(LSTM(100, return_sequences=True))
model.add(LSTM(100))
model.add(Dropout(0.2))  # Dropout layer to prevent overfitting
model.add(Dense(vocab_size, activation='softmax'))  # Output layer for word␣
  ↪prediction

# Compile the model
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
  ↪metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 5, 100)            468500    
                                                                 
 lstm (LSTM)                 (None, 5, 100)            80400     
                                                                 
 lstm_1 (LSTM)               (None, 100)               80400     
                                                                 
 dropout (Dropout)           (None, 100)               0         
                                                                 
 dense (Dense)               (None, 4685)              473185    
                                                                 
=================================================================
Total params: 1102485 (4.21 MB)
Trainable params: 1102485 (4.21 MB)
Non-trainable params: 0 (0.00 Byte)
```

```
----------------------------------------------------------------
```

0.0.5 4. Training:

Compile the model with categorical cross-entropy as the loss function and accuracy as the metric.

```
[ ]: from keras.utils import to_categorical

     # One-hot encode the target labels
     y = to_categorical(y, num_classes=vocab_size)
     print("y shape after one-hot encoding:", y.shape)
```

```
y shape after one-hot encoding: (7098, 4685)
```

Train the model on the sequences for 10-20 epochs (or until it achieves satisfactory performance).

```
[ ]: # Train the model
     model.fit(X, y, epochs=10, batch_size=64)
```

```
Epoch 1/10
111/111 [==============================] - 5s 19ms/step - loss: 7.5300 -
accuracy: 0.0533
Epoch 2/10
111/111 [==============================] - 2s 18ms/step - loss: 6.7217 -
accuracy: 0.0609
Epoch 3/10
111/111 [==============================] - 2s 19ms/step - loss: 6.6194 -
accuracy: 0.0609
Epoch 4/10
111/111 [==============================] - 2s 19ms/step - loss: 6.5517 -
accuracy: 0.0609
Epoch 5/10
111/111 [==============================] - 2s 19ms/step - loss: 6.4548 -
accuracy: 0.0609
Epoch 6/10
111/111 [==============================] - 2s 19ms/step - loss: 6.3262 -
accuracy: 0.0609
Epoch 7/10
111/111 [==============================] - 2s 19ms/step - loss: 6.2342 -
accuracy: 0.0614
Epoch 8/10
111/111 [==============================] - 2s 19ms/step - loss: 6.1572 -
accuracy: 0.0668
Epoch 9/10
111/111 [==============================] - 2s 19ms/step - loss: 6.0525 -
accuracy: 0.0727
Epoch 10/10
111/111 [==============================] - 2s 19ms/step - loss: 5.9259 -
accuracy: 0.0741
```

```
[ ]: <keras.src.callbacks.History at 0x78ba74085f30>
```

0.0.6 5. Text Generation:

```
[ ]: def generate_text(seed_text, next_words, model, tokenizer, max_sequence_len,␣
     ↪temperature=1.0, words_per_set=5):
         previous_word = ""  # Track the previous word to avoid consecutive␣
     ↪repetition
         word_count = 0  # Track the number of words generated in the current set
         for i in range(next_words):
             # Tokenize the seed text
             tokenized_text = tokenizer.texts_to_sequences([seed_text])[0]

             # Pad the tokenized sequence
             tokenized_text = pad_sequences([tokenized_text],␣
     ↪maxlen=max_sequence_len, padding='pre')

             # Predict the next word probabilities
             predicted = model.predict(tokenized_text, verbose=0)[0]

             # Apply temperature to predictions (to make them more diverse)
             predicted = np.log(predicted + 1e-7) / temperature
             predicted = np.exp(predicted) / np.sum(np.exp(predicted))  # Normalize␣
     ↪to get valid probabilities

             # Sample the next word based on the adjusted probabilities
             predicted_word_idx = np.random.choice(len(predicted), p=predicted)
             predicted_word = tokenizer.index_word.get(predicted_word_idx, '')

             # Ensure the predicted word is not the same as the previous word
             if predicted_word == previous_word:
                 continue  # Skip the word if it's the same as the previous one

             if word_count != 0:
               seed_text += " "

             # Append the predicted word to the seed text
             seed_text += predicted_word

             # Update the previous word
             previous_word = predicted_word

             # Track the number of words in the current set
             word_count += 1

             # If we've reached the words_per_set limit, add a comma and reset word␣
     ↪count
```

```
        if word_count >= words_per_set:
            seed_text += ",\n"
            word_count = 0  # Reset the word count for the next set of words

    return seed_text

seed = "once upon a time "
generated_poem = generate_text(seed, next_words=50, model=model,␣
 ↪tokenizer=tokenizer, max_sequence_len=X.shape[1], temperature=0.7,␣
 ↪words_per_set=5)
print(generated_poem)
```

```
once upon a time of say and a wars,
of you ambivalence and other,
head their better old still,
in the knew body what,
to me was be this,
to alone a just of,
the far we do our,
craving power a trove of,
the affliction and bon the,
mouth of is walked
```

0.0.7 6. Evaluation and Experimentation:

[ ]:
```
# Define the LSTM model
model2 = Sequential()
model2.add(Embedding(input_dim=vocab_size, output_dim=100, input_length=X.
 ↪shape[1]))
model2.add(LSTM(50, return_sequences=True))
model2.add(LSTM(50))
model2.add(Dropout(0.4))  # Dropout layer to prevent overfitting
model2.add(Dense(vocab_size, activation='softmax'))  # Output layer for word␣
 ↪prediction

# Compile the model
model2.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
model2.summary()
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 5, 100)            468500

 lstm_2 (LSTM)               (None, 5, 50)             30200
 lstm_3 (LSTM)               (None, 50)                20200

 dropout_1 (Dropout)         (None, 50)                0

 dense_1 (Dense)             (None, 4685)              238935

=================================================================
Total params: 757835 (2.89 MB)
Trainable params: 757835 (2.89 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Train the model
model2.fit(X, y, epochs=30, batch_size=64)
```

```
Epoch 1/30
111/111 [==============================] - 5s 14ms/step - loss: 7.6568 -
accuracy: 0.0538
Epoch 2/30
111/111 [==============================] - 2s 14ms/step - loss: 6.7824 -
accuracy: 0.0609
Epoch 3/30
111/111 [==============================] - 2s 14ms/step - loss: 6.6630 -
accuracy: 0.0609
Epoch 4/30
111/111 [==============================] - 2s 14ms/step - loss: 6.5983 -
accuracy: 0.0609
Epoch 5/30
111/111 [==============================] - 2s 14ms/step - loss: 6.5303 -
accuracy: 0.0609
Epoch 6/30
111/111 [==============================] - 2s 14ms/step - loss: 6.4455 -
accuracy: 0.0609
Epoch 7/30
111/111 [==============================] - 2s 14ms/step - loss: 6.3483 -
accuracy: 0.0609
Epoch 8/30
111/111 [==============================] - 2s 14ms/step - loss: 6.2845 -
accuracy: 0.0610
Epoch 9/30
111/111 [==============================] - 2s 14ms/step - loss: 6.2367 -
accuracy: 0.0611
Epoch 10/30
111/111 [==============================] - 1s 13ms/step - loss: 6.1914 -
accuracy: 0.0609
Epoch 11/30
111/111 [==============================] - 2s 14ms/step - loss: 6.1563 -
accuracy: 0.0610
```

```
Epoch 12/30
111/111 [==============================] - 2s 14ms/step - loss: 6.1152 -
accuracy: 0.0624
Epoch 13/30
111/111 [==============================] - 2s 14ms/step - loss: 6.0638 -
accuracy: 0.0633
Epoch 14/30
111/111 [==============================] - 2s 14ms/step - loss: 6.0131 -
accuracy: 0.0626
Epoch 15/30
111/111 [==============================] - 2s 14ms/step - loss: 5.9459 -
accuracy: 0.0668
Epoch 16/30
111/111 [==============================] - 2s 14ms/step - loss: 5.8729 -
accuracy: 0.0686
Epoch 17/30
111/111 [==============================] - 2s 14ms/step - loss: 5.8086 -
accuracy: 0.0711
Epoch 18/30
111/111 [==============================] - 2s 14ms/step - loss: 5.7446 -
accuracy: 0.0737
Epoch 19/30
111/111 [==============================] - 2s 14ms/step - loss: 5.6805 -
accuracy: 0.0752
Epoch 20/30
111/111 [==============================] - 1s 13ms/step - loss: 5.6209 -
accuracy: 0.0762
Epoch 21/30
111/111 [==============================] - 2s 14ms/step - loss: 5.5547 -
accuracy: 0.0776
Epoch 22/30
111/111 [==============================] - 1s 13ms/step - loss: 5.4939 -
accuracy: 0.0827
Epoch 23/30
111/111 [==============================] - 2s 14ms/step - loss: 5.4411 -
accuracy: 0.0868
Epoch 24/30
111/111 [==============================] - 2s 14ms/step - loss: 5.3797 -
accuracy: 0.0859
Epoch 25/30
111/111 [==============================] - 2s 14ms/step - loss: 5.3259 -
accuracy: 0.0869
Epoch 26/30
111/111 [==============================] - 2s 14ms/step - loss: 5.2682 -
accuracy: 0.0920
Epoch 27/30
111/111 [==============================] - 2s 14ms/step - loss: 5.2160 -
accuracy: 0.0940
```

```
Epoch 28/30
111/111 [==============================] - 1s 13ms/step - loss: 5.1631 -
accuracy: 0.0965
Epoch 29/30
111/111 [==============================] - 2s 14ms/step - loss: 5.1143 -
accuracy: 0.1026
Epoch 30/30
111/111 [==============================] - 2s 14ms/step - loss: 5.0515 -
accuracy: 0.1027
```

```
[ ]: <keras.src.callbacks.History at 0x78b94c7a0fd0>
```

```
[ ]: seed = "In a Faraway Land, "
     generated_poem = generate_text(seed, next_words=50, model=model2,␣
      ↪tokenizer=tokenizer, max_sequence_len=X.shape[1], temperature=0.7,␣
      ↪words_per_set=7)
     print(generated_poem)
```

```
In a Faraway Land, wondered nipple you the grave blessd de,
slap his hole scrap bowed flares as,
they billows until third of rideable shall,
around at smoke strong wheelchairs into to,
bed as in the blown of too,
light of a trucks holding it they,
tells of a wheel in the food,
out
```