RNN & LSTM

RNN (Recurrent Neural Network) & LSTM (Long Short-Term Memory)

Author: Irfan Ullah Khan

GITHUB PROFILE | KAGGLE PROFILE | LINKEDIN PROFILE

YOUTUBE PROFILE | EMAIL CONTACT ME | WEBSITE CONTACT ME

Import Libraries

```
import numpy as np
import tensorflow as tf
import tensorflow.keras.models as models
import tensorflow.keras.layers as layers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Sample data

```
# Sample data
sentences = [
"The cat slept on the windowsill, basking in the afternoon sun."

"She found a hidden note inside the old book."

"The coffee spilled, staining the tablecloth."

"The dog chased its tail in circles."

"An old man whistled a tune as he walked down the street."

"The cake was decorated with fresh strawberries and cream."

"He forgot his keys on the kitchen counter."

"The rain pitter-pattered against the rooftop all night."

"They laughed together, remembering their childhood adventures."

"The movie ended with a surprising plot twist."

"She wore a bright yellow scarf that caught everyone’s attention."

"The lighthouse stood tall against the stormy sea."

"He discovered a new hobby: collecting vintage postcards."

"The garden was full of blooming roses and lavender."

"The old clock on the wall ticked loudly in the quiet room."

"She hummed a melody while washing the dishes."

"The children built a fort out of blankets and pillows."

"A gentle breeze rustled the autumn leaves."
```

```
"He could hear the distant sound of a train whistle."

"The bookshop had a cozy reading nook by the fireplace."

"She danced gracefully across the stage in her new ballet shoes."

"The mountain view was breathtaking at sunrise."

"He made a wish while blowing out the candles on his birthday cake."

"The artist painted vibrant colors onto the canvas."

"The coffee shop had a welcoming aroma of freshly brewed beans."

"She found an old photograph in a dusty attic box."

"The wind chimes tinkled softly in the garden."

"He fixed the leaky faucet with a few twists of the wrench."

"The rainbows appeared after the storm had passed."

"The chef prepared a gourmet meal with exquisite precision."

"The little boat bobbed gently on the lake."

"She received a letter from an old friend she hadn’t seen in years."

"The concert was a spectacular display of lights and sound."

"The kitten played with a ball of yarn on the living room floor."

"The bakery window was filled with colorful pastries and cakes."

"He wore his favorite sweater even though it was warm outside."

"The stars shone brightly in the clear night sky."

"She sang a lullaby to her baby before bedtime."

"The wind carried the scent of fresh pine through the forest."

"The old man told stories of his youth to the fascinated children."

"She found a beautiful seashell on the beach."

"The library was silent except for the sound of turning pages."

"He sketched a portrait of his best friend in his notebook."

"The squirrel darted across the yard, clutching an acorn."

"The fog rolled in, enveloping the landscape in mystery."

"She practiced her guitar every evening on the porch."

"The fire crackled and popped in the fireplace."

"The road trip took them through stunning mountain passes."

"The old theater had ornate architecture and a grand chandelier."

"She carefully wrapped the gift with shiny paper and a bow."

"The snow covered the ground like a soft white blanket."
```

```
"The snow covered the ground like a soft white blanket."

"He enjoyed a quiet afternoon of fishing by the river."

"The artist’s studio was cluttered with paintbrushes and canvases."

"The baby giggled as she played with her toys."

"The street was lined with charming cafes and boutique shops."

"He cooked a hearty stew on a cold winter’s day."

"The wind rustled the pages of the open book."

"The park was filled with the sounds of children playing."

"She enjoyed a relaxing bath with lavender-scented candles."

"The old barn had a rustic charm and a creaky door."

"The train journey offered scenic views of rolling hills."

"She wore a necklace that had been passed down through generations."

"The museum displayed ancient artifacts and priceless paintings."

"He looked out the window and saw a colorful hot air balloon."

"The scent of freshly cut grass filled the air."

"The dog barked excitedly at the sight of its owner."

"The recipe called for a pinch of salt and a dash of pepper."

"She admired the intricate design of the stained glass window."

"The ice cream melted quickly under the hot sun."

"The children gathered around the storyteller for a magical tale."

"The old map showed hidden treasures and forgotten lands."

"He planted seeds in the garden and hoped for a bountiful harvest."

"The bell rang, signaling the end of the school day."

"She decorated her room with colorful posters and fairy lights."

"The aroma of freshly baked bread wafted through the house."

"He enjoyed reading detective novels on rainy weekends."

"The picnic was complete with sandwiches, fruit, and lemonade."

"The city skyline looked majestic against the setting sun."

"She wrote a heartfelt letter to her future self."

"The fireflies twinkled like tiny stars in the evening dusk."

"The playground was filled with the laughter of children."

"He carefully folded the map and tucked it into his pocket."

"The clouds formed whimsical shapes in the blue sky."
```

```
"She found a cozy spot by the lake to read her book."

"The bicycle ride through the park was invigorating."

"The old piano in the corner was covered in dust but still beautiful."

"The garden gnome had a mischievous smile."

"The sand felt warm beneath her feet as she walked along the beach."

"The snowflakes fell gently, creating a winter wonderland."

"He practiced his magic tricks in front of the mirror."

"The castle on the hill looked majestic in the morning mist."

"She carefully painted her nails a vibrant shade of red."

"The sunset painted the sky with hues of orange and pink."

"The dog curled up next to the fire and fell asleep."

"The old library had towering shelves filled with books."

"She wore a hat decorated with colorful feathers."

"The melody of the song lingered in her mind throughout the day."

"The garden was a haven for birds and butterflies."

"He discovered an interesting rock while hiking in the mountains."

"The vintage car had been lovingly restored to its former glory."
]
```

Tokenization & Preprocessing of Data

```
# Initialize the tokenizer
tokenizer = Tokenizer()

# Fit the tokenizer on the sentences
tokenizer.fit_on_texts(sentences)

# Get the total number of unique words (plus one for padding)
total_words = len(tokenizer.word_index) + 1
print(total_words)
print(tokenizer.word_index)

# Initialize a list to hold input sequences
input_sequences = []

# iterate over the sentences
for line in sentences:
    # Convert the sentence into a  sequence of integers
    token_list = tokenizer.texts_to_sequences([line])[0]
   # print(token_list)
   # Create n-gram sequences from the sentence
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

# Determine the maximum length sequence
max_sequence_len = max([len(x) for x in input_sequences])
```

```
# Pad sequences to ensure they are all the same length
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
print(input_sequences)
```

```
487
{'the': 1, 'a': 2, 'of': 3, 'and': 4, 'in': 5, 'she': 6, 'with': 7, 'he': 8, 'was': 9, 'on': 10, 'old': 11, 'her': 12, 'had': 13, 'his': 14, 'to': 15,
[[  0   0   0 ...   0   1 103]
 [  0   0   0 ...   1 103 104]
 [  0   0   0 ... 103 104  10]
 ...
 [  0   0   1 ... 484  15  28]
 [  0   1 103 ...  15  28 485]
 [  1 103 104 ...  28 485 486]]
```

Data Preparation:

```
# Split the data into inputs and labels

# Inputs: All elements of the sequences except the one
X=input_sequences[:,:-1]
print("Input Data:",X)

# Labels: The last element each  sequences
y=input_sequences[:,-1]
# print("Labels:",y)

# Convert the labels to one-hot encoding format
y = tf.keras.utils.to_categorical(y, num_classes=total_words)
# print("one hot encoded Vector: ",y)
```

```
Input Data: [[  0   0   0 ...   0   0   1]
 [  0   0   0 ...   0   1 103]
 [  0   0   0 ...   1 103 104]
 ...
 [  0   0   1 ... 483 484  15]
 [  0   1 103 ... 484  15  28]
 [  1 103 104 ...  15  28 485]]
```

Defining the Model Architecture:

Sequential Model: A linear stack of layers./

Embedding Layer:/ **SimpleRNN Layer:**

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense

# ... (rest of the code remains the same)

# Define the RNN

# Sequential model allows stacking layers in a linear fashion
model = Sequential([Embedding(total_words, 10, input_length=max_sequence_len-1),

# SimpleRNN layer with 20 hidden units,which processes the sequence
SimpleRNN(30),

# Dense output layer with softmax activation function
# Output dimension: total number of words (for multi-class classification)
Dense(total_words, activation='softmax')
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

Compile the Model

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

Train the Model

```
# Train the model
model.fit(X, y, epochs=10, verbose=1)
```

```
Epoch 1/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 10s 260ms/step - accuracy: 0.0070 - loss: 6.1734
Epoch 2/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 214ms/step - accuracy: 0.1204 - loss: 6.0234
Epoch 3/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 11s 224ms/step - accuracy: 0.1375 - loss: 5.7223
Epoch 4/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 284ms/step - accuracy: 0.1435 - loss: 5.3785
Epoch 5/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 8s 223ms/step - accuracy: 0.1460 - loss: 5.3086
Epoch 6/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 8s 259ms/step - accuracy: 0.1313 - loss: 5.3102
Epoch 7/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 9s 284ms/step - accuracy: 0.1280 - loss: 5.3302
Epoch 8/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 7s 215ms/step - accuracy: 0.1293 - loss: 5.3252
Epoch 9/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 12s 261ms/step - accuracy: 0.1223 - loss: 5.2335
Epoch 10/10
31/31 ━━━━━━━━━━━━━━━━━━━━ 11s 282ms/step - accuracy: 0.1324 - loss: 5.3527
<keras.src.callbacks.history.History at 0x7c7961c7c040>
```

Next Word Predication

Seed Text Converstion:

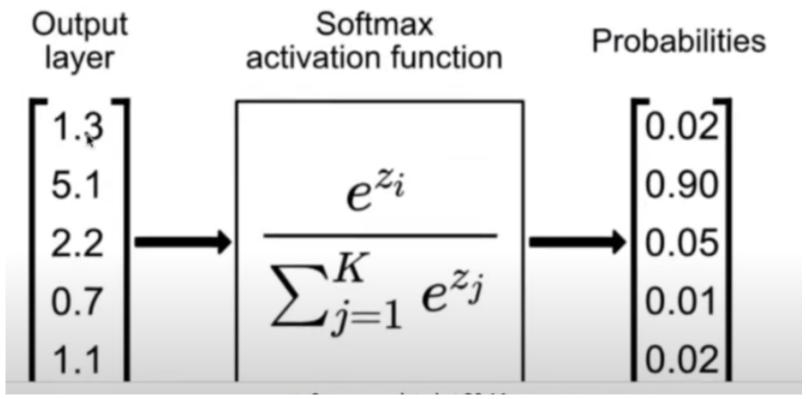
Padding:

Predication:

Selection and Update:

Iterative Predictioin:

```
# Function to predict the next word(s) given a seed text
def predict_next_word(seed_text, next_words=1):
    for _ in range(next_words):
        # Convert the seed text into a sequence of integers
        token_list = tokenizer.texts_to_sequences([seed_text])[0]

        # Pad the sequence to match the input length required by the model
        token_list= pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')

        # Predict the probability distribution of the next word
        predicted = model.predict(token_list, verbose=0)

        # Get the index of the predicted word ,with the highest probability
        predicted_word_index = np.argmax(predicted, axis=1)[0]

        #Retrieve the word corresponding to the predicted index
        predicted_word=tokenizer.index_word[predicted_word_index]

        # Append the predicted word to the seed text
        seed_text += " " + predicted_word
        # Return the updated seed text with the predicted word
        return seed_text
```


Output layer
Softmax activation function
Probabilities
1.3
5.1
2.2
0.7
1.1
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
0.02
0.90
0.05
0.01
0.02

```
# Test the prediction function with  a sample input
print(predict_next_word('Macine Learning'))
```

```
Macine Learning the
```

Building a Long Short-Term Memory(LSTM)

```
# Import necessary libraries
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Embedding
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Sample Data

```
# Sample data
sentences = [
"The cat slept on the windowsill, basking in the afternoon sun."

"She found a hidden note inside the old book."

"The coffee spilled, staining the tablecloth."

"The dog chased its tail in circles."

"An old man whistled a tune as he walked down the street."

"The cake was decorated with fresh strawberries and cream."

"He forgot his keys on the kitchen counter."

"The rain pitter-pattered against the rooftop all night."

"They laughed together, remembering their childhood adventures."

"The movie ended with a surprising plot twist."

"She wore a bright yellow scarf that caught everyone’s attention."

"The lighthouse stood tall against the stormy sea."

"He discovered a new hobby: collecting vintage postcards."
```

```
"The garden was full of blooming roses and lavender."

"The old clock on the wall ticked loudly in the quiet room."

"She hummed a melody while washing the dishes."

"The children built a fort out of blankets and pillows."

"A gentle breeze rustled the autumn leaves."

"He could hear the distant sound of a train whistle."

"The bookshop had a cozy reading nook by the fireplace."

"She danced gracefully across the stage in her new ballet shoes."

"The mountain view was breathtaking at sunrise."

"He made a wish while blowing out the candles on his birthday cake."

"The artist painted vibrant colors onto the canvas."

"The coffee shop had a welcoming aroma of freshly brewed beans."

"She found an old photograph in a dusty attic box."

"The wind chimes tinkled softly in the garden."

"He fixed the leaky faucet with a few twists of the wrench."

"The rainbows appeared after the storm had passed."

"The chef prepared a gourmet meal with exquisite precision."

"The little boat bobbed gently on the lake."

"She received a letter from an old friend she hadn’t seen in years."

"The concert was a spectacular display of lights and sound."

"The kitten played with a ball of yarn on the living room floor."

"The bakery window was filled with colorful pastries and cakes."

"He wore his favorite sweater even though it was warm outside."

"The stars shone brightly in the clear night sky."

"She sang a lullaby to her baby before bedtime."

"The wind carried the scent of fresh pine through the forest."

"The old man told stories of his youth to the fascinated children."

"She found a beautiful seashell on the beach."

"The library was silent except for the sound of turning pages."

"He sketched a portrait of his best friend in his notebook."

"The squirrel darted across the yard, clutching an acorn."

"The fog rolled in, enveloping the landscape in mystery."

"She practiced her guitar every evening on the porch."
```

```
"The fire crackled and popped in the fireplace."

"The road trip took them through stunning mountain passes."

"The old theater had ornate architecture and a grand chandelier."

"She carefully wrapped the gift with shiny paper and a bow."

"The snow covered the ground like a soft white blanket."

"He enjoyed a quiet afternoon of fishing by the river."

"The artist’s studio was cluttered with paintbrushes and canvases."

"The baby giggled as she played with her toys."

"The street was lined with charming cafes and boutique shops."

"He cooked a hearty stew on a cold winter’s day."

"The wind rustled the pages of the open book."

"The park was filled with the sounds of children playing."

"She enjoyed a relaxing bath with lavender-scented candles."

"The old barn had a rustic charm and a creaky door."

"The train journey offered scenic views of rolling hills."

"She wore a necklace that had been passed down through generations."

"The museum displayed ancient artifacts and priceless paintings."

"He looked out the window and saw a colorful hot air balloon."

"The scent of freshly cut grass filled the air."

"The dog barked excitedly at the sight of its owner."

"The recipe called for a pinch of salt and a dash of pepper."

"She admired the intricate design of the stained glass window."

"The ice cream melted quickly under the hot sun."

"The children gathered around the storyteller for a magical tale."

"The old map showed hidden treasures and forgotten lands."

"He planted seeds in the garden and hoped for a bountiful harvest."

"The bell rang, signaling the end of the school day."

"She decorated her room with colorful posters and fairy lights."

"The aroma of freshly baked bread wafted through the house."

"He enjoyed reading detective novels on rainy weekends."

"The picnic was complete with sandwiches, fruit, and lemonade."

"The city skyline looked majestic against the setting sun."
```

```
"She wrote a heartfelt letter to her future self."

"The fireflies twinkled like tiny stars in the evening dusk."

"The playground was filled with the laughter of children."

"He carefully folded the map and tucked it into his pocket."

"The clouds formed whimsical shapes in the blue sky."

"She found a cozy spot by the lake to read her book."

"The bicycle ride through the park was invigorating."

"The old piano in the corner was covered in dust but still beautiful.

"The garden gnome had a mischievous smile."

"The sand felt warm beneath her feet as she walked along the beach."

"The snowflakes fell gently, creating a winter wonderland."

"He practiced his magic tricks in front of the mirror."

"The castle on the hill looked majestic in the morning mist."

"She carefully painted her nails a vibrant shade of red."

"The sunset painted the sky with hues of orange and pink."

"The dog curled up next to the fire and fell asleep."

"The old library had towering shelves filled with books."

"She wore a hat decorated with colorful feathers."

"The melody of the song lingered in her mind throughout the day."

"The garden was a haven for birds and butterflies."

"He discovered an interesting rock while hiking in the mountains."
```

Tokenization & Preprocessing of Data

```
# Initialize the tokenizer
tokenizer = Tokenizer()

# Fit the tokenizer on the sentences
tokenizer.fit_on_texts(sentences)

# Get the total number of unique words (plus one for padding)
total_words = len(tokenizer.word_index) + 1
print(total_words)
print(tokenizer.word_index)

# Initialize a list to hold input sequences
input_sequences = []

# iterate over the sentences
for line in sentences:
    # Convert the sentence into a  sequence of integers
    token_list = tokenizer.texts_to_sequences([line])[0]
   # print(token_list)
   # Create n-gram sequences from the sentence
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

# Determine the maximum length sequence
max_sequence_len = max([len(x) for x in input_sequences])

# Pad sequences to ensure they are all the same length
input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_sequence_len, padding='pre'))
print(input_sequences)
```

```
487
{'the': 1, 'a': 2, 'of': 3, 'and': 4, 'in': 5, 'she': 6, 'with': 7, 'he': 8, 'was': 9, 'on': 10, 'old': 11, 'her': 12, 'had': 13, 'his': 14, 'to': 15,
[[  0   0   0 ...   0   1 103]
 [  0   0   0 ...   1 103 104]
 [  0   0   0 ... 103 104  10]
 ...
 [  0   0   1 ... 484  15  28]
 [  0   1 103 ...  15  28 485]
 [  1 103 104 ...  28 485 486]]
```

Data Preparation:

```
# Split the data into inputs and labels

# Inputs: All elements of the sequences except the one
X=input_sequences[:,:-1]
print("Input Data:",X)

# Labels: The last element each  sequences
y=input_sequences[:,-1]
# print("Labels:",y)

# Convert the labels to one-hot encoding format
y = tf.keras.utils.to_categorical(y, num_classes=total_words)
# print("one hot encoded Vector: ",y)
```

```
Input Data: [[  0   0   0 ...   0   0   1]
 [  0   0   0 ...   0   1 103]
 [  0   0   0 ...   1 103 104]
 ...
 [  0   0   1 ... 483 484  15]
 [  0   1 103 ... 484  15  28]
 [  1 103 104 ...  15  28 485]]
```

Defining the LSTM Model:

```
# Define the LSTM model
# Sequential Model allows stacking layers in a linear fashion
model = Sequential([
    # Embedding layer to represent words as dense vectors
    # Input dimension is the total number of unique words,Ouptut dimension:
    # Input length is the length of the sequence (excluding the last word)
    Embedding(total_words, 100, input_length=max_sequence_len-1),

    # LSTM layer with 30 units
    # LSTM (Long Shor-Term Memory) is a type of RNN that can capture long-term dependencie
    LSTM(100),

    # Dense output layer with a softmax activation function
    # Output dimension is the total number of unique words
    # Softmax activation converts the output  to probabilities for each word

    Dense(total_words, activation='softmax')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
```

Compiling the Model

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

Training the model

```
# Train the model
model.fit(X, y, epochs=50, verbose=1)
```

```
Epoch 1/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 881ms/step - accuracy: 0.1298 - loss: 5.3766
Epoch 2/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 878ms/step - accuracy: 0.1261 - loss: 5.3253
Epoch 3/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 866ms/step - accuracy: 0.1263 - loss: 5.2487
Epoch 4/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 873ms/step - accuracy: 0.1325 - loss: 5.1298
Epoch 5/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 849ms/step - accuracy: 0.1349 - loss: 4.9518
Epoch 6/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 880ms/step - accuracy: 0.1346 - loss: 4.8869
Epoch 7/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 869ms/step - accuracy: 0.1323 - loss: 4.8574
Epoch 8/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 869ms/step - accuracy: 0.1519 - loss: 4.7794
Epoch 9/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 869ms/step - accuracy: 0.1566 - loss: 4.5952
Epoch 10/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 865ms/step - accuracy: 0.1846 - loss: 4.5314
Epoch 11/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 28s 894ms/step - accuracy: 0.2071 - loss: 4.3196
Epoch 12/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.2110 - loss: 4.2573
Epoch 13/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 874ms/step - accuracy: 0.2158 - loss: 4.1311
Epoch 14/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 876ms/step - accuracy: 0.2159 - loss: 4.0609
Epoch 15/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.2132 - loss: 4.0988
Epoch 16/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.2459 - loss: 3.8299
Epoch 17/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 848ms/step - accuracy: 0.2649 - loss: 3.7636
Epoch 18/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 857ms/step - accuracy: 0.2776 - loss: 3.6202
Epoch 19/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 861ms/step - accuracy: 0.2774 - loss: 3.5590
Epoch 20/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 870ms/step - accuracy: 0.2879 - loss: 3.4803
Epoch 21/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.2885 - loss: 3.4578
Epoch 22/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 886ms/step - accuracy: 0.3307 - loss: 3.2203
Epoch 23/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 881ms/step - accuracy: 0.3191 - loss: 3.2652
Epoch 24/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 879ms/step - accuracy: 0.3010 - loss: 3.2763
Epoch 25/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 27s 869ms/step - accuracy: 0.3352 - loss: 3.0385
Epoch 26/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 882ms/step - accuracy: 0.3469 - loss: 3.0225
Epoch 27/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 41s 851ms/step - accuracy: 0.3472 - loss: 2.9730
Epoch 28/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 40s 843ms/step - accuracy: 0.4160 - loss: 2.7408
Epoch 29/50
31/31 ━━━━━━━━━━━━━━━━━━━━ 42s 880ms/step - accuracy: 0.4214 - loss: 2.6658
```

Next Word Predication

Seed Text Converstion:
Padding:
Predication:

Selection and Update:

Iterative Predictioin:

```
# Function to predict the next word(s) given a seed text
def predict_next_word(seed_text, next_words=1):  # Here if next_words=1 we will get only one word,if next_wor
    for _ in range(next_words):
        # Convert the seed text into a sequence of integers
        token_list = tokenizer.texts_to_sequences([seed_text])[0]

        # Pad the sequence to match the input length required by the model
        token_list= pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')

        # Predict the probability distribution of the next word
        predicted = model.predict(token_list, verbose=0)

        # Get the index of the predicted word ,with the highest probability
        predicted_word_index = np.argmax(predicted, axis=1)[0]

        #Retrieve the word corresponding to the predicted index
        predicted_word=tokenizer.index_word[predicted_word_index]

        # Append the predicted word to the seed text
        seed_text += " " + predicted_word
        # Return the updated seed text with the predicted word
        return seed_text
```

Test prediction

```
# Test the prediction function with  a sample input
print(predict_next_word('She found a'))
```

```
She found a cat
```