LAB7

Name: Kushal Sourav

Regno: 2347125

In [1]:
```
import pandas as pd
import re
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout
import random
```

In [2]:
```
data = pd.read_csv('./PoetryFoundationData.csv')
```

In [3]:
```
print(data.describe())
```

```
        Unnamed: 0
count  13854.000000
mean      93.204417
std       57.493544
min        0.000000
25%       42.000000
50%       92.000000
75%      142.000000
max      199.000000
```

In [4]:
```
print(data.head())
```

```
   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
```

In [5]:
```
print(data.shape)
```

```
(13854, 5)
```

In [6]:
```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13854 entries, 0 to 13853
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  13854 non-null  int64
 1   Title       13854 non-null  object
 2   Poem        13854 non-null  object
 3   Poet        13854 non-null  object
 4   Tags        12899 non-null  object
dtypes: int64(1), object(4)
memory usage: 541.3+ KB
None
```

In [7]:
```
corpus = "\n".join(data['Poem'].values)
```

In [8]:
```
corpus = corpus.lower()
corpus = re.sub(r'[^\w\s]', '', corpus)
```

In [9]:
```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([corpus])
total_words = len(tokenizer.word_index) + 1

# Convert text into sequences of integers
input_sequences = []
corpus_words = corpus.split()
for i in range(5, len(corpus_words)):
    sequence = corpus_words[i-5:i+1]
    tokenized_seq = tokenizer.texts_to_sequences([" ".join(sequence)])[0]
    input_sequences.append(tokenized_seq)

# Pad sequences
max_sequence_len = 5  # Length of each sequence
input_sequences = pad_sequences(input_sequences, maxlen=max_sequence_len + 1)
```

In [10]:
```
X, y = input_sequences[:, :-1], input_sequences[:, -1]
X, y = X[:10000], y[:10000]
y = np.array(y)
```

In [11]:
```
model = Sequential()
model.add(Embedding(input_dim=total_words, output_dim=100, input_length=max_sequ
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(total_words, activation='softmax'))
```

```
c:\Users\USER\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src
\layers\core\embedding.py:90: UserWarning: Argument `input_length` is deprecated.
Just remove it.
  warnings.warn(
```

In [12]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=
```

In [13]:
```
model.fit(X, y, epochs=50, batch_size =128, verbose=1)
```

```
Epoch 1/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 100s 1s/step - accuracy: 0.0611 - loss: 11.4866
Epoch 2/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 102s 1s/step - accuracy: 0.0700 - loss: 7.1831
Epoch 3/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.0715 - loss: 6.8709
Epoch 4/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 87s 1s/step - accuracy: 0.0685 - loss: 6.7412
Epoch 5/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 88s 1s/step - accuracy: 0.0728 - loss: 6.6478
Epoch 6/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 87s 1s/step - accuracy: 0.0720 - loss: 6.5806
Epoch 7/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.0730 - loss: 6.5769
Epoch 8/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.0658 - loss: 6.5776
Epoch 9/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 88s 1s/step - accuracy: 0.0718 - loss: 6.5145
Epoch 10/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.0746 - loss: 6.5027
Epoch 11/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.0730 - loss: 6.4584
Epoch 12/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.0717 - loss: 6.4738
Epoch 13/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.0721 - loss: 6.3878
Epoch 14/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 97s 1s/step - accuracy: 0.0723 - loss: 6.2694
Epoch 15/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 92s 1s/step - accuracy: 0.0683 - loss: 6.2925
Epoch 16/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 93s 1s/step - accuracy: 0.0720 - loss: 6.1667
Epoch 17/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 92s 1s/step - accuracy: 0.0701 - loss: 6.0604
Epoch 18/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 91s 1s/step - accuracy: 0.0736 - loss: 6.0141
Epoch 19/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.0730 - loss: 5.9294
Epoch 20/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 89s 1s/step - accuracy: 0.0756 - loss: 5.8962
Epoch 21/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 93s 1s/step - accuracy: 0.0702 - loss: 5.8566
Epoch 22/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.0710 - loss: 5.7648
Epoch 23/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 90s 1s/step - accuracy: 0.0718 - loss: 5.7368
Epoch 24/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 93s 1s/step - accuracy: 0.0789 - loss: 5.6473
Epoch 25/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.0773 - loss: 5.6052
Epoch 26/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 83s 1s/step - accuracy: 0.0825 - loss: 5.5511
Epoch 27/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 83s 1s/step - accuracy: 0.0807 - loss: 5.5099
Epoch 28/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.0847 - loss: 5.4542
Epoch 29/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 93s 1s/step - accuracy: 0.0879 - loss: 5.4085
Epoch 30/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 89s 1s/step - accuracy: 0.0952 - loss: 5.3137
```

```
Epoch 31/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 87s 1s/step - accuracy: 0.0981 - loss: 5.3002
Epoch 32/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.0977 - loss: 5.2499
Epoch 33/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.0994 - loss: 5.2025
Epoch 34/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1078 - loss: 5.1263
Epoch 35/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 88s 1s/step - accuracy: 0.1070 - loss: 5.0803
Epoch 36/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 88s 1s/step - accuracy: 0.1186 - loss: 5.0134
Epoch 37/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 87s 1s/step - accuracy: 0.1125 - loss: 4.9919
Epoch 38/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.1240 - loss: 4.9412
Epoch 39/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.1254 - loss: 4.8659
Epoch 40/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1346 - loss: 4.8210
Epoch 41/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.1336 - loss: 4.7920
Epoch 42/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.1413 - loss: 4.7068
Epoch 43/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 86s 1s/step - accuracy: 0.1474 - loss: 4.6701
Epoch 44/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1403 - loss: 4.6403
Epoch 45/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.1442 - loss: 4.6171
Epoch 46/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1523 - loss: 4.5310
Epoch 47/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1561 - loss: 4.4950
Epoch 48/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.1559 - loss: 4.4816
Epoch 49/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 84s 1s/step - accuracy: 0.1648 - loss: 4.3937
Epoch 50/50
79/79 ━━━━━━━━━━━━━━━━━━━━ 85s 1s/step - accuracy: 0.1598 - loss: 4.4085
```

Out[13]: <keras.src.callbacks.history.History at 0x2a2df9e5390>

In [14]:

```python
def generate_poetry(seed_text, next_words=5000):
    generated_words = set()
    poem = seed_text

    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([poem])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len, paddin

        predicted_probs = model.predict(token_list, verbose=0)
        predicted = np.argmax(predicted_probs, axis=-1)

        next_word = tokenizer.index_word.get(predicted[0], None)
        if next_word is None or next_word in generated_words:
            continue

        generated_words.add(next_word)
        poem += " " + next_word
```

```
    return poem

print(generate_poetry("The Morning Sun Shine", next_words=500))
```

```
The Morning Sun Shine his way is the dead of
```

In [15]:

```
def generate_poetry(seed_text, next_words=5000, words_per_line=10):
    generated_words = set()
    poem = seed_text

    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([poem])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len, paddin

        predicted_probs = model.predict(token_list, verbose=0)
        predicted = np.argmax(predicted_probs, axis=-1)

        next_word = tokenizer.index_word.get(predicted[0], None)
        if next_word is None or next_word in generated_words:
            continue

        generated_words.add(next_word)
        poem += " " + next_word

    return poem

print(generate_poetry("The Morning Sun Shine", next_words=500))
```

```
The Morning Sun Shine his way is the dead of
```

In [16]:

```
# Generate multiple lines of poetry using different starting phrases
seed_texts = ["the moonlight whispers", "in the quiet of night", "stars shine br

for seed in seed_texts:
    print(f"Seed: {seed}")
    print(generate_poetry(seed, next_words=100, words_per_line=100))
    print("\n" + "="*50 + "\n")
```

```
Seed: the moonlight whispers
the moonlight whispers that faces in one look like

==================================================

Seed: in the quiet of night
in the quiet of night that ride of her will is still in the way and are days as

==================================================

Seed: stars shine brightly
stars shine brightly its long juice and these gods from

==================================================

Seed: a gentle breeze flows
a gentle breeze flows in is

==================================================

Seed: echoes in silence
echoes in silence and are the map one left as ride

==================================================
```

In [ ]: