Oishi_136_Lab_7

November 21, 2024

[ ]:
```
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
```

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

[ ]:
```
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
```

[ ]:
```
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
```

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

```
(13854, 5)
```

[ ]:
```
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
```

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

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

[ ]:
```
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)
```

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

[ ]:
```
model = Sequential()
model.add(Embedding(input_dim=total_words, output_dim=100,␣
 ↪input_length=max_sequence_len))
```

```
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'))
```

[ ]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
model.fit(X, y, epochs=100, batch_size =128, verbose=1)
```

```
Epoch 1/100
79/79            1s 7ms/step -
accuracy: 0.7714 - loss: 1.3504
Epoch 2/100
79/79            1s 6ms/step -
accuracy: 0.7811 - loss: 1.3091
Epoch 3/100
79/79            1s 6ms/step -
accuracy: 0.7811 - loss: 1.2659
Epoch 4/100
79/79            0s 6ms/step -
accuracy: 0.7857 - loss: 1.2438
Epoch 5/100
79/79            1s 6ms/step -
accuracy: 0.7917 - loss: 1.2037
Epoch 6/100
79/79            1s 6ms/step -
accuracy: 0.8036 - loss: 1.1524
Epoch 7/100
79/79            1s 6ms/step -
accuracy: 0.8012 - loss: 1.1356
Epoch 8/100
79/79            1s 6ms/step -
accuracy: 0.8065 - loss: 1.0973
Epoch 9/100
79/79            1s 6ms/step -
accuracy: 0.8164 - loss: 1.0513
Epoch 10/100
79/79            1s 6ms/step -
accuracy: 0.8228 - loss: 1.0161
Epoch 11/100
79/79            1s 6ms/step -
accuracy: 0.8239 - loss: 1.0104
Epoch 12/100
79/79            1s 6ms/step -
accuracy: 0.8358 - loss: 0.9581
Epoch 13/100
```

```
79/79              1s 6ms/step -
accuracy: 0.8294 - loss: 0.9520
Epoch 14/100
79/79              1s 6ms/step -
accuracy: 0.8375 - loss: 0.9020
Epoch 15/100
79/79              1s 6ms/step -
accuracy: 0.8448 - loss: 0.8877
Epoch 16/100
79/79              1s 6ms/step -
accuracy: 0.8510 - loss: 0.8562
Epoch 17/100
79/79              1s 8ms/step -
accuracy: 0.8501 - loss: 0.8447
Epoch 18/100
79/79              1s 9ms/step -
accuracy: 0.8588 - loss: 0.8105
Epoch 19/100
79/79              1s 9ms/step -
accuracy: 0.8509 - loss: 0.8166
Epoch 20/100
79/79              1s 6ms/step -
accuracy: 0.8604 - loss: 0.7820
Epoch 21/100
79/79              1s 6ms/step -
accuracy: 0.8635 - loss: 0.7668
Epoch 22/100
79/79              1s 6ms/step -
accuracy: 0.8658 - loss: 0.7424
Epoch 23/100
79/79              1s 6ms/step -
accuracy: 0.8703 - loss: 0.7263
Epoch 24/100
79/79              1s 6ms/step -
accuracy: 0.8711 - loss: 0.6959
Epoch 25/100
79/79              1s 7ms/step -
accuracy: 0.8735 - loss: 0.7042
Epoch 26/100
79/79              0s 6ms/step -
accuracy: 0.8705 - loss: 0.6965
Epoch 27/100
79/79              1s 6ms/step -
accuracy: 0.8809 - loss: 0.6543
Epoch 28/100
79/79              0s 6ms/step -
accuracy: 0.8862 - loss: 0.6335
Epoch 29/100
```

```
79/79              1s 6ms/step -
accuracy: 0.8883 - loss: 0.6213
Epoch 30/100
79/79              1s 6ms/step -
accuracy: 0.8855 - loss: 0.6080
Epoch 31/100
79/79              1s 6ms/step -
accuracy: 0.8836 - loss: 0.6073
Epoch 32/100
79/79              1s 6ms/step -
accuracy: 0.8972 - loss: 0.5741
Epoch 33/100
79/79              1s 6ms/step -
accuracy: 0.8853 - loss: 0.5732
Epoch 34/100
79/79              1s 6ms/step -
accuracy: 0.8966 - loss: 0.5528
Epoch 35/100
79/79              1s 6ms/step -
accuracy: 0.8989 - loss: 0.5460
Epoch 36/100
79/79              1s 7ms/step -
accuracy: 0.8945 - loss: 0.5438
Epoch 37/100
79/79              1s 9ms/step -
accuracy: 0.8914 - loss: 0.5437
Epoch 38/100
79/79              1s 11ms/step -
accuracy: 0.9043 - loss: 0.4992
Epoch 39/100
79/79              1s 7ms/step -
accuracy: 0.9032 - loss: 0.4899
Epoch 40/100
79/79              1s 6ms/step -
accuracy: 0.9053 - loss: 0.4872
Epoch 41/100
79/79              1s 6ms/step -
accuracy: 0.9006 - loss: 0.4876
Epoch 42/100
79/79              0s 6ms/step -
accuracy: 0.9026 - loss: 0.4905
Epoch 43/100
79/79              1s 6ms/step -
accuracy: 0.9094 - loss: 0.4580
Epoch 44/100
79/79              1s 6ms/step -
accuracy: 0.9066 - loss: 0.4573
Epoch 45/100
```

```
79/79             0s 6ms/step -
accuracy: 0.9068 - loss: 0.4536
Epoch 46/100
79/79             1s 6ms/step -
accuracy: 0.9096 - loss: 0.4406
Epoch 47/100
79/79             1s 6ms/step -
accuracy: 0.9086 - loss: 0.4388
Epoch 48/100
79/79             1s 7ms/step -
accuracy: 0.9099 - loss: 0.4318
Epoch 49/100
79/79             1s 6ms/step -
accuracy: 0.9152 - loss: 0.4124
Epoch 50/100
79/79             1s 6ms/step -
accuracy: 0.9163 - loss: 0.4131
Epoch 51/100
79/79             1s 6ms/step -
accuracy: 0.9109 - loss: 0.4138
Epoch 52/100
79/79             1s 6ms/step -
accuracy: 0.9130 - loss: 0.4053
Epoch 53/100
79/79             1s 6ms/step -
accuracy: 0.9138 - loss: 0.3917
Epoch 54/100
79/79             1s 6ms/step -
accuracy: 0.9183 - loss: 0.3878
Epoch 55/100
79/79             1s 6ms/step -
accuracy: 0.9257 - loss: 0.3825
Epoch 56/100
79/79             1s 9ms/step -
accuracy: 0.9165 - loss: 0.3925
Epoch 57/100
79/79             1s 9ms/step -
accuracy: 0.9232 - loss: 0.3657
Epoch 58/100
79/79             1s 8ms/step -
accuracy: 0.9251 - loss: 0.3552
Epoch 59/100
79/79             1s 6ms/step -
accuracy: 0.9237 - loss: 0.3591
Epoch 60/100
79/79             1s 6ms/step -
accuracy: 0.9171 - loss: 0.3600
Epoch 61/100
```

```
79/79               1s 6ms/step -
accuracy: 0.9200 - loss: 0.3630
Epoch 62/100
79/79               1s 6ms/step -
accuracy: 0.9215 - loss: 0.3436
Epoch 63/100
79/79               1s 6ms/step -
accuracy: 0.9217 - loss: 0.3477
Epoch 64/100
79/79               1s 6ms/step -
accuracy: 0.9243 - loss: 0.3365
Epoch 65/100
79/79               1s 6ms/step -
accuracy: 0.9293 - loss: 0.3334
Epoch 66/100
79/79               1s 6ms/step -
accuracy: 0.9236 - loss: 0.3262
Epoch 67/100
79/79               0s 6ms/step -
accuracy: 0.9284 - loss: 0.3200
Epoch 68/100
79/79               1s 6ms/step -
accuracy: 0.9298 - loss: 0.3117
Epoch 69/100
79/79               1s 7ms/step -
accuracy: 0.9211 - loss: 0.3125
Epoch 70/100
79/79               1s 6ms/step -
accuracy: 0.9274 - loss: 0.3091
Epoch 71/100
79/79               1s 7ms/step -
accuracy: 0.9262 - loss: 0.3156
Epoch 72/100
79/79               1s 6ms/step -
accuracy: 0.9318 - loss: 0.3005
Epoch 73/100
79/79               1s 6ms/step -
accuracy: 0.9283 - loss: 0.3039
Epoch 74/100
79/79               1s 8ms/step -
accuracy: 0.9283 - loss: 0.2991
Epoch 75/100
79/79               1s 9ms/step -
accuracy: 0.9264 - loss: 0.2967
Epoch 76/100
79/79               1s 9ms/step -
accuracy: 0.9343 - loss: 0.2823
Epoch 77/100
```

```
79/79             1s 10ms/step -
accuracy: 0.9335 - loss: 0.2797
Epoch 78/100
79/79             1s 7ms/step -
accuracy: 0.9331 - loss: 0.2874
Epoch 79/100
79/79             1s 6ms/step -
accuracy: 0.9340 - loss: 0.2825
Epoch 80/100
79/79             1s 7ms/step -
accuracy: 0.9307 - loss: 0.2919
Epoch 81/100
79/79             1s 6ms/step -
accuracy: 0.9267 - loss: 0.2947
Epoch 82/100
79/79             1s 6ms/step -
accuracy: 0.9320 - loss: 0.2729
Epoch 83/100
79/79             1s 6ms/step -
accuracy: 0.9394 - loss: 0.2617
Epoch 84/100
79/79             1s 6ms/step -
accuracy: 0.9323 - loss: 0.2693
Epoch 85/100
79/79             1s 6ms/step -
accuracy: 0.9290 - loss: 0.2724
Epoch 86/100
79/79             1s 6ms/step -
accuracy: 0.9349 - loss: 0.2651
Epoch 87/100
79/79             1s 6ms/step -
accuracy: 0.9363 - loss: 0.2584
Epoch 88/100
79/79             1s 6ms/step -
accuracy: 0.9404 - loss: 0.2590
Epoch 89/100
79/79             1s 6ms/step -
accuracy: 0.9295 - loss: 0.2685
Epoch 90/100
79/79             1s 6ms/step -
accuracy: 0.9359 - loss: 0.2548
Epoch 91/100
79/79             1s 6ms/step -
accuracy: 0.9376 - loss: 0.2414
Epoch 92/100
79/79             1s 6ms/step -
accuracy: 0.9336 - loss: 0.2576
Epoch 93/100
```

```
79/79                1s 6ms/step -
accuracy: 0.9359 - loss: 0.2446
Epoch 94/100
79/79                1s 6ms/step -
accuracy: 0.9313 - loss: 0.2482
Epoch 95/100
79/79                1s 7ms/step -
accuracy: 0.9369 - loss: 0.2518
Epoch 96/100
79/79                1s 9ms/step -
accuracy: 0.9298 - loss: 0.2510
Epoch 97/100
79/79                1s 10ms/step -
accuracy: 0.9351 - loss: 0.2461
Epoch 98/100
79/79                1s 7ms/step -
accuracy: 0.9339 - loss: 0.2478
Epoch 99/100
79/79                1s 6ms/step -
accuracy: 0.9397 - loss: 0.2417
Epoch 100/100
79/79                1s 7ms/step -
accuracy: 0.9341 - loss: 0.2526
```

[ ]: <keras.src.callbacks.history.History at 0x7e220371e6e0>

[ ]:
```
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,␣
 ↪padding='pre')

        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
```

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

```
The Morning Sun Shine sean stevens wallace
```

Evaluation and Experimentation

[ ]:
```python
# Generate multiple lines of poetry using different starting phrases
seed_texts = ["the moonlight whispers", "in the quiet of night", "stars shine␣
↪brightly", "a gentle breeze flows", "echoes in silence"]

for seed in seed_texts:
    print(f"Seed: {seed}")
    print(generate_poetry(seed, next_words=50, words_per_line=10))
    print("\n" + "="*50 + "\n")
```

```
Seed: the moonlight whispers
the moonlight whispers traxler sarah morgan

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

Seed: in the quiet of night
in the quiet of night ann percy

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

Seed: stars shine brightly
stars shine brightly traxler quincy cavalieri victor

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

Seed: a gentle breeze flows
a gentle breeze flows quincy cavalieri kate moses anya silver franco

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

Seed: echoes in silence
echoes in silence traxler quincy cavalieri victor

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