Import Library Needed

In [ ]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping
```

Load Data

In [ ]:
```
with open('Kanye_West.txt', 'r', encoding='unicode_escape') as myfile:
    mytext = myfile.read()
```

In [ ]:
```
mytext
```

Preprocessing

In [ ]:
```
my_tokenizer = Tokenizer()
my_tokenizer.fit_on_texts([mytext])
total_words = len(my_tokenizer.word_index) + 1
```

In [ ]:
```
my_tokenizer.word_index
```

In [ ]:
```
my_input_sequences = []
for line in mytext.split('\n'):
    # print(line)
    token_list = my_tokenizer.texts_to_sequences([line])[0]
    print(token_list)
    for i in range(1, len(token_list)):
        my_n_gram_sequence = token_list[:i+1]
        my_input_sequences.append(my_n_gram_sequence)
        # print(input_sequences)
```

In [ ]:
```
my_input_sequences = []
for line in mytext.split('\n'):
    # print(line)
    token_list = my_tokenizer.texts_to_sequences([line])[0]
    # print(token_list)
    for i in range(1, len(token_list)):
        my_n_gram_sequence = token_list[:i+1]
        print(my_n_gram_sequence)
        my_input_sequences.append(my_n_gram_sequence)
        # print(input_sequences)
```

In [ ]:
```
max_sequence_len = max([len(seq) for seq in my_input_sequences])
input_sequences = np.array(pad_sequences(my_input_sequences, maxlen=max_sequ
```

In [ ]:
```
input_sequences
```

Out[ ]:
```
array([[   0,    0,    0, ...,    0, 2668,  240],
       [   0,    0,    0, ..., 2668,  240, 1097],
       [   0,    0,    0, ...,  240, 1097,  240],
       ...,
       [   0,    0,    0, ...,  814,    6,   10],
       [   0,    0,    0, ...,    0,   25,    9],
       [   0,    0,    0, ...,   25,    9,    3]], dtype=int32)
```

In [ ]:
```
X = input_sequences[:, :-1]
y = input_sequences[:, -1]
```

In [ ]:
```
X[2]
```

Out[ ]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0, 2668,  240, 1097], dtype=int32)
```

In [ ]:
```
y[2]
```

Out[ ]:
```
240
```

In [ ]:
```
X
```

Out[ ]:
```
array([[   0,    0,    0, ...,    0,    0, 2668],
       [   0,    0,    0, ...,    0, 2668,  240],
       [   0,    0,    0, ..., 2668,  240, 1097],
       ...,
       [   0,    0,    0, ...,   61,  814,    6],
       [   0,    0,    0, ...,    0,    0,   25],
       [   0,    0,    0, ...,    0,   25,    9]], dtype=int32)
```

In [ ]:
```
y
```

Out[ ]:
```
array([ 240, 1097,  240, ...,   10,    9,    3], dtype=int32)
```

In [ ]:
```
# lakukan one hot encoding
y = np.array(tf.keras.utils.to_categorical(y, num_classes=total_words))
```

In [ ]:
```
y
```

Out[ ]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]], dtype=float32)
```

In [ ]:
```
y[0]
```

Out[ ]:
```
array([0., 0., 0., ..., 0., 0., 0.], dtype=float32)
```

Define Models

In [ ]:
```
model = tf.keras.models.Sequential()
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))
model.add(LSTM(150))
model.add(Dense(total_words, activation='softmax'))
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 18, 100)           614200

 lstm (LSTM)                 (None, 150)               150600

 dense (Dense)               (None, 6142)              927442

=================================================================
Total params: 1692242 (6.46 MB)
Trainable params: 1692242 (6.46 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

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

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

```
Epoch 1/100
1413/1413 [==============================] - 23s 14ms/step - loss: 6.7223 -
accuracy: 0.0414
Epoch 2/100
1413/1413 [==============================] - 11s 8ms/step - loss: 6.1304 - a
ccuracy: 0.0636
Epoch 3/100
1413/1413 [==============================] - 10s 7ms/step - loss: 5.7624 - a
ccuracy: 0.0838
Epoch 4/100
1413/1413 [==============================] - 10s 7ms/step - loss: 5.4345 - a
ccuracy: 0.1026
Epoch 5/100
1413/1413 [==============================] - 9s 7ms/step - loss: 5.0986 - ac
curacy: 0.1235
Epoch 6/100
1413/1413 [==============================] - 10s 7ms/step - loss: 4.7692 - a
ccuracy: 0.1428
Epoch 7/100
1413/1413 [==============================] - 10s 7ms/step - loss: 4.4509 - a
ccuracy: 0.1683
Epoch 8/100
1413/1413 [==============================] - 10s 7ms/step - loss: 4.1465 - a
ccuracy: 0.1997
Epoch 9/100
1413/1413 [==============================] - 10s 7ms/step - loss: 3.8543 - a
ccuracy: 0.2399
Epoch 10/100
1413/1413 [==============================] - 9s 7ms/step - loss: 3.5813 - ac
curacy: 0.2824
Epoch 11/100
1413/1413 [==============================] - 10s 7ms/step - loss: 3.3306 - a
ccuracy: 0.3225
Epoch 12/100
1413/1413 [==============================] - 10s 7ms/step - loss: 3.1027 - a
ccuracy: 0.3617
Epoch 13/100
1413/1413 [==============================] - 10s 7ms/step - loss: 2.8973 - a
ccuracy: 0.3979
Epoch 14/100
1413/1413 [==============================] - 9s 7ms/step - loss: 2.7124 - ac
curacy: 0.4313
Epoch 15/100
1413/1413 [==============================] - 10s 7ms/step - loss: 2.5383 - a
ccuracy: 0.4663
Epoch 16/100
1413/1413 [==============================] - 10s 7ms/step - loss: 2.3842 - a
ccuracy: 0.4942
Epoch 17/100
1413/1413 [==============================] - 12s 9ms/step - loss: 2.2403 - a
ccuracy: 0.5225
Epoch 18/100
1413/1413 [==============================] - 9s 7ms/step - loss: 2.1080 - ac
curacy: 0.5492
Epoch 19/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.9877 - a
```

```
ccuracy: 0.5757
Epoch 20/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.8765 - a
ccuracy: 0.5968
Epoch 21/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.7739 - a
ccuracy: 0.6182
Epoch 22/100
1413/1413 [==============================] - 9s 6ms/step - loss: 1.6821 - ac
curacy: 0.6357
Epoch 23/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.5947 - a
ccuracy: 0.6548
Epoch 24/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.5162 - a
ccuracy: 0.6731
Epoch 25/100
1413/1413 [==============================] - 11s 8ms/step - loss: 1.4434 - a
ccuracy: 0.6871
Epoch 26/100
1413/1413 [==============================] - 9s 6ms/step - loss: 1.3763 - ac
curacy: 0.7001
Epoch 27/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.3130 - a
ccuracy: 0.7137
Epoch 28/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.2575 - a
ccuracy: 0.7260
Epoch 29/100
1413/1413 [==============================] - 9s 7ms/step - loss: 1.2051 - ac
curacy: 0.7375
Epoch 30/100
1413/1413 [==============================] - 9s 6ms/step - loss: 1.1593 - ac
curacy: 0.7451
Epoch 31/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.1144 - a
ccuracy: 0.7548
Epoch 32/100
1413/1413 [==============================] - 10s 7ms/step - loss: 1.0750 - a
ccuracy: 0.7630
Epoch 33/100
1413/1413 [==============================] - 9s 7ms/step - loss: 1.0389 - ac
curacy: 0.7709
Epoch 34/100
1413/1413 [==============================] - 9s 6ms/step - loss: 1.0028 - ac
curacy: 0.7769
Epoch 35/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.9730 - a
ccuracy: 0.7829
Epoch 36/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.9454 - a
ccuracy: 0.7870
Epoch 37/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.9201 - ac
curacy: 0.7912
Epoch 38/100
```

```
1413/1413 [==============================] - 9s 7ms/step - loss: 0.8965 - ac
curacy: 0.7942
Epoch 39/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.8762 - a
ccuracy: 0.7988
Epoch 40/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.8554 - a
ccuracy: 0.8015
Epoch 41/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.8375 - ac
curacy: 0.8050
Epoch 42/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.8198 - a
ccuracy: 0.8069
Epoch 43/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.8077 - a
ccuracy: 0.8076
Epoch 44/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7932 - a
ccuracy: 0.8120
Epoch 45/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.7788 - ac
curacy: 0.8137
Epoch 46/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7671 - a
ccuracy: 0.8155
Epoch 47/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7610 - a
ccuracy: 0.8160
Epoch 48/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7488 - a
ccuracy: 0.8175
Epoch 49/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.7378 - ac
curacy: 0.8185
Epoch 50/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7283 - a
ccuracy: 0.8194
Epoch 51/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7220 - a
ccuracy: 0.8209
Epoch 52/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7157 - a
ccuracy: 0.8209
Epoch 53/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.7096 - ac
curacy: 0.8213
Epoch 54/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.7013 - a
ccuracy: 0.8221
Epoch 55/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6958 - a
ccuracy: 0.8231
Epoch 56/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6917 - a
ccuracy: 0.8235
```

```
Epoch 57/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6866 - ac
curacy: 0.8231
Epoch 58/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6817 - a
ccuracy: 0.8244
Epoch 59/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6797 - a
ccuracy: 0.8242
Epoch 60/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6736 - ac
curacy: 0.8246
Epoch 61/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6698 - ac
curacy: 0.8244
Epoch 62/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6670 - a
ccuracy: 0.8251
Epoch 63/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6637 - a
ccuracy: 0.8254
Epoch 64/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6629 - ac
curacy: 0.8250
Epoch 65/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6599 - ac
curacy: 0.8247
Epoch 66/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6545 - a
ccuracy: 0.8252
Epoch 67/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6542 - a
ccuracy: 0.8255
Epoch 68/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6488 - ac
curacy: 0.8254
Epoch 69/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6487 - ac
curacy: 0.8252
Epoch 70/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6446 - a
ccuracy: 0.8264
Epoch 71/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6489 - a
ccuracy: 0.8253
Epoch 72/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6409 - ac
curacy: 0.8255
Epoch 73/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6389 - a
ccuracy: 0.8263
Epoch 74/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6363 - a
ccuracy: 0.8269
Epoch 75/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6377 - a
```

```
ccuracy: 0.8266
Epoch 76/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6373 - ac
curacy: 0.8249
Epoch 77/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6330 - a
ccuracy: 0.8258
Epoch 78/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6290 - a
ccuracy: 0.8277
Epoch 79/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6324 - a
ccuracy: 0.8251
Epoch 80/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6321 - ac
curacy: 0.8261
Epoch 81/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6278 - a
ccuracy: 0.8259
Epoch 82/100
1413/1413 [==============================] - 11s 7ms/step - loss: 0.6260 - a
ccuracy: 0.8258
Epoch 83/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6262 - a
ccuracy: 0.8257
Epoch 84/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6269 - ac
curacy: 0.8250
Epoch 85/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6215 - a
ccuracy: 0.8271
Epoch 86/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6211 - a
ccuracy: 0.8270
Epoch 87/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6214 - ac
curacy: 0.8256
Epoch 88/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6222 - ac
curacy: 0.8268
Epoch 89/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6193 - a
ccuracy: 0.8274
Epoch 90/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6168 - a
ccuracy: 0.8274
Epoch 91/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6207 - ac
curacy: 0.8257
Epoch 92/100
1413/1413 [==============================] - 9s 7ms/step - loss: 0.6137 - ac
curacy: 0.8274
Epoch 93/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6160 - a
ccuracy: 0.8266
Epoch 94/100
```

```
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6159 - a
ccuracy: 0.8271
Epoch 95/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6140 - ac
curacy: 0.8260
Epoch 96/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6124 - a
ccuracy: 0.8267
Epoch 97/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6122 - a
ccuracy: 0.8268
Epoch 98/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6119 - a
ccuracy: 0.8268
Epoch 99/100
1413/1413 [==============================] - 9s 6ms/step - loss: 0.6159 - ac
curacy: 0.8260
Epoch 100/100
1413/1413 [==============================] - 10s 7ms/step - loss: 0.6102 - a
ccuracy: 0.8270
```

In [ ]:

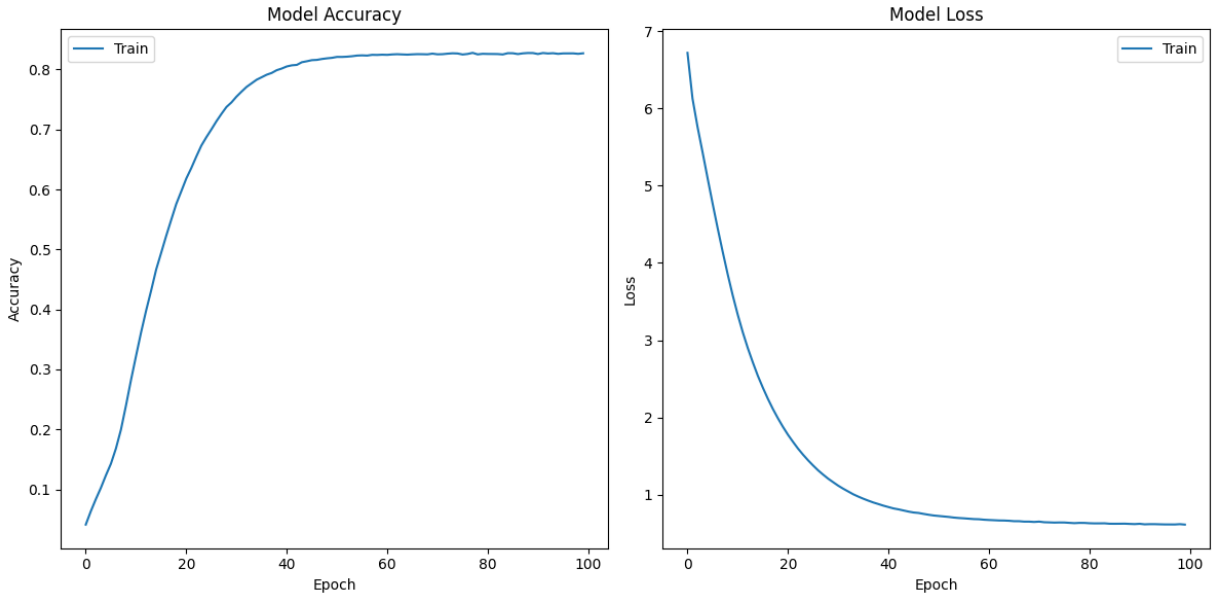
```python
import matplotlib.pyplot as plt

# Plot training & validation accuracy values
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(hist.history['accuracy'], label='Train')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(hist.history['loss'], label='Train')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```



Make Prediction

Save Model

In [ ]:
```
model.save("/content/mymodel.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: U
serWarning: You are saving your model as an HDF5 file via `model.save()`. Th
is file format is considered legacy. We recommend using instead the native K
eras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

Load Model

In [ ]:
```
model_loaded = load_model("/content/mymodel.h5")
```

In [ ]:
```
import numpy as np

input_text = "most rappers"
predict_next_words = 10

for _ in range(predict_next_words):
    token_list = my_tokenizer.texts_to_sequences([input_text])[0]
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padc
    predictions = model_loaded.predict(token_list)[0]

    # Get indices of top predicted words
    top_indices = np.argsort(predictions)[-5:][::-1]  # Adjust 5 to the numb

    # Get words corresponding to the indices
    next_words = [word for word, index in my_tokenizer.word_index.items() if

    # Print the list of next words along with their probabilities
```

```
    print("Input Text:", input_text)
    print("Next Words and Probabilities:")
    for word, index in zip(next_words, top_indices):
        probability = predictions[index]
        print(f"{word}: {probability:.4f}")

    # Choose the word with the highest probability as the next word
    output_word = my_tokenizer.index_word[top_indices[0]]

    input_text += " " + output_word

print(input_text)
```

```
1/1 [==============================] - 0s 35ms/step
Input Text: most rappers
Next Words and Probabilities:
and: 0.9986
to: 0.0003
so: 0.0003
got: 0.0002
taste: 0.0002
1/1 [==============================] - 0s 64ms/step
Input Text: most rappers taste
Next Words and Probabilities:
got: 0.9996
with: 0.0002
aint: 0.0001
girl: 0.0000
level: 0.0000
1/1 [==============================] - 0s 27ms/step
Input Text: most rappers taste level
Next Words and Probabilities:
is: 0.9991
for: 0.0003
aint: 0.0002
way: 0.0001
feel: 0.0001
1/1 [==============================] - 0s 20ms/step
Input Text: most rappers taste level aint
Next Words and Probabilities:
is: 0.9989
at: 0.0005
doin: 0.0001
pimp: 0.0001
hi: 0.0000
1/1 [==============================] - 0s 17ms/step
Input Text: most rappers taste level aint at
Next Words and Probabilities:
a: 0.9883
my: 0.0051
that: 0.0039
me: 0.0006
please: 0.0006
1/1 [==============================] - 0s 17ms/step
Input Text: most rappers taste level aint at my
Next Words and Probabilities:
watch: 0.9987
shoes: 0.0003
door: 0.0001
waist: 0.0001
hustle: 0.0001
1/1 [==============================] - 0s 16ms/step
Input Text: most rappers taste level aint at my waist
Next Words and Probabilities:
huh: 0.9999
doing: 0.0000
level: 0.0000
today: 0.0000
biggie: 0.0000
```

```
1/1 [==============================] - 0s 17ms/step
Input Text: most rappers taste level aint at my waist level
Next Words and Probabilities:
aint: 0.5305
yeah: 0.3709
why: 0.0445
gonna: 0.0100
theres: 0.0088
1/1 [==============================] - 0s 17ms/step
Input Text: most rappers taste level aint at my waist level gonna
Next Words and Probabilities:
never: 0.4741
had: 0.2630
try: 0.2142
watch: 0.0384
named: 0.0039
1/1 [==============================] - 0s 17ms/step
Input Text: most rappers taste level aint at my waist level gonna try
Next Words and Probabilities:
of: 0.4566
out: 0.1758
us: 0.1684
him: 0.1050
any: 0.0444
most rappers taste level aint at my waist level gonna try us
```

In [ ]:
```
input_text = "private jet"
predict_next_words = 15

for _ in range(predict_next_words):
    token_list = my_tokenizer.texts_to_sequences([input_text])[0]
    print(token_list)
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padc
    predicted = np.argmax(model_loaded.predict(token_list), axis=-1)
    output_word = ""
    for word, index in my_tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    input_text += " " + output_word

print(input_text)
```

```
[1103, 1104]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43]
1/1 [==============================] - 0s 18ms/step
[1103, 1104, 43, 7]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75, 18]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75, 18, 161]
1/1 [==============================] - 0s 19ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7]
1/1 [==============================] - 0s 18ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157]
1/1 [==============================] - 0s 19ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30]
1/1 [==============================] - 0s 20ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64]
1/1 [==============================] - 0s 18ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64, 190]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64, 190, 31]
1/1 [==============================] - 0s 19ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64, 190, 31, 356]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64, 190, 31, 356, 145]
1/1 [==============================] - 0s 17ms/step
[1103, 1104, 43, 7, 75, 18, 161, 7, 157, 30, 64, 190, 31, 356, 145, 86]
1/1 [==============================] - 0s 16ms/step
private jet out my love of head my home be off huh now far best bitch big
```

Interpretasi:

Penggunaan model LSTM untuk projek *next word prediction* ini sudah cukup baik untuk mengetahui kata-kata yang harus di generate selanjutnya. namun masih kurang dalam *generating* sebuah keseluruhan sentence yang memiliki makna, model cenderung memunculkan kata-kata baru yang hanya memiliki korelasi makna berjarak pendek terhadap kata-kata sebelumnya. Lalu penggunaan LSTM sebagai model juga belum cukup baik dalam hal efisiensi karena algoritma LSTM yang tidak memerhatikan penting atau tidaknya suatu kata untuk menjadi inti sebuah *sentence* dan mengambil keseluruhan sentence yang panjang untuk diingat pada algoritmanya.