Nama : Virgie Yunita Salsabil

NIM : 21110022

Kelas : S1-SD02A

`Next Word Generator adalah aplikasi NLP (Natural Language Processing) yang dapat memprediksi kata berikutnya dalam sebuah kalimat berdasarkan konteks yang diberikan.`

TUGAS

1. Pilih satu file txt dari file zip "data" sebagai dataset untuk pelatihan
2. Running program yang ada di folder dengan dataset yang kamu pilih
3. Lakukan uji coba model kamu
4. Kumpulkan dalam bentuk pdf (ipynb di pdfkan)

Import Library Needed

```
# Upgrade tensorflow ke versi terbaru
!pip install --upgrade tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0.post1)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.23.5)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.35.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.60.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.1)
```

```
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.42.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->t
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.
```

```
# Utk operasi numerik, seperti manipulasi array dan matriks.
import numpy as np
# Utk membangun dan melatih model machine learning.
import tensorflow as tf

# Import class Tokenizer dari sub-module preprocessing.text dlm library Keras.
# Utk mengonversi teks menjadi urutan token (kata-kata).
from tensorflow.keras.preprocessing.text import Tokenizer

# Import class Sequential dan load_model dari sub-module models dlm library Keras.
# Sequential : membuat model secara berurutan, lapisan demi lapisan.
# load_model : utk memuat model yg telah disimpan sebelumnya.
from tensorflow.keras.models import Sequential, load_model

# Import class pad_sequences dari sub-module preprocessing.sequence dlm library Keras
# pad_sequences : Utk mengonversi urutan token menjadi matriks dg panjang yg seragam (misal: kasus padding).
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Import class Embedding, LSTM & Dense dari sub-module layers dlm library Keras
# Embedding : Utk mengonversi token menjadi vektor angka real. Berguna utk representasi teks.
# LSTM : Utk memahami dan memproses urutan data.
# Dense : layer fully connected yg digunakan dlm output model.
from tensorflow.keras.layers import Embedding, LSTM, Dense

# Import class EarlyStopping dari sub-module callbacks dlm library Keras
# EarlyStopping : Utk menghentikan pelatihan model jika suatu kondisi terpenuhi, seperti tidak adanya peningkatan performa.
from tensorflow.keras.callbacks import EarlyStopping
```

Load Data

```
# Membuka file 'lady-gaga.txt' untuk dibaca ('r'), menggunakan encoding 'unicode_escape'
with open('lady-gaga.txt', 'r', encoding='unicode_escape') as myfile:
    # Membaca seluruh konten file dan menyimpannya dalam variabel mytext
    mytext = myfile.read()
```

```
# Tampilkan text
mytext
```

```
'I\'ll undress you, \'cause you\'re tired\nCover you as you desire\nWhen you fall asleep inside my arms\nMay not have the fancy things\nBut I\'ll give you everyt
hing\nYou could ever want, it\'s in my arms So baby tell me yes\nAnd I will give you everything\nSo baby tell me yes\nAnd I will be all yours tonight\nSo baby te
ll me yes\nAnd I will give you everything\nI will be right by your side If I can\'t find the cure, I\'ll\nI\'ll fix you with my love\nNo matter what you know, I
\'ll\nI\'ll fix you with my love\nAnd if you say you\'re okay\nI\'m gonna heal you anyway\nPromise I\'ll always be there\nPromise I\'ll be the cure (be the cure)
Rub your feet, your hands, your legs\nLet me take care of it, babe\nClose your eyes, I\'ll sing your favorite song\nI wrote you this lullaby\nHush now baby, don
\'t you cry\nAnything you want could not be wrong So baby tell me yes\nAnd I will give you everything\nSo baby tell me yes\nAnd I will be all yours tonight\nSo b
aby tell me yes\nAnd I will give you…'
```

Preprocessing

```
# Membuat objek Tokenizer utk mengonversi teks menjadi urutan token
my_tokenizer = Tokenizer()

# Tokenizer akan menghitung frekuensi kemunculan setiap token & membuat indeks utk setiap token
my_tokenizer.fit_on_texts([mytext])

# Menghitung total kata unik dalam teks dengan menambahkan satu, karena indeks dimulai dari 1 (indeks 0 biasanya tidak digunakan)
total_words = len(my_tokenizer.word_index) + 1
```

```
# Memetakan kata-kata unik ke indeks numerik berdasar frekuensi dlm teks
my_tokenizer.word_index
```

```
{'i': 1,
 'you': 2,
 'the': 3,
 'a': 4,
 'me': 5,
 'my': 6,
 'to': 7,
 'and': 8,
 'oh': 9,
 "i'm": 10,
 'your': 11,
 'that': 12,
 'love': 13,
 'be': 14,
 'in': 15,
```

```
 'on': 16,
 'it': 17,
 "don't": 18,
 'want': 19,
 'just': 20,
 'eh': 21,
 'baby': 22,
 'with': 23,
 'of': 24,
 'but': 25,
 'we': 26,
 'what': 27,
 'wanna': 28,
 'like': 29,
 'do': 30,
 'know': 31,
 'so': 32,
 'is': 33,
 'this': 34,
 "i'll": 35,
 'can': 36,
 "you're": 37,
 'not': 38,
 "can't": 39,
 'could': 40,
 'p': 41,
 'up': 42,
 'one': 43,
 'no': 44,
 'good': 45,
 'as': 46,
 'got': 47,
 'm': 48,
 'for': 49,
 'if': 50,
 'million': 51,
 "it's": 52,
 'way': 53,
 'reasons': 54,
 'all': 55,
 'when': 56,
 'heart': 57,
 'hair': 58,
```

INTERPRETASI

- `my_tokenizer.word_index`:
  Atribut dari objek Tokenizer berisi kamus/dictionary yang memetakan kata-kata unik dlm teks ke indeks numerik.
- Setiap kata diberi indeks berdasarkan frekuensinya dalam teks. Indeks ini digunakan saat membuat representasi numerik dari teks menggunakan token.
- Dalam contoh di atas -> indeks 1 diberikan pada kata 'I', indeks 2 pada 'You', dan seterusnya.

```
# Membuat list kosong 'my_input_sequences' utk menyimpan urutan token berdasarkan n-gram
my_input_sequences = []

# Melakukan iterasi pada setiap baris teks yg dipisahkan berdasarkan karakter newline ('\n')
for line in mytext.split('\n'):
    # Mengonversi setiap baris teks menjadi urutan token menggunakan my_tokenizer
    token_list = my_tokenizer.texts_to_sequences([line])[0]
    print(token_list)

    # Melakukan iterasi pada setiap indeks dalam token_list
    for i in range(1, len(token_list)):
        # Membuat n-gram sequence dari token_list hingga indeks ke-i
        my_n_gram_sequence = token_list[:i+1]
        # Menambahkan n-gram sequence ke dalam my_input_sequences
        my_input_sequences.append(my_n_gram_sequence)
        # print(input_sequences)
```

```
[35, 826, 2, 61, 37, 827]
[407, 2, 46, 2, 723]
[56, 2, 297, 828, 244, 6, 580]
[361, 38, 108, 3, 643, 232]
[25, 35, 78, 2, 179]
[2, 40, 408, 19, 52, 15, 6, 580, 32, 22, 81, 5, 264]
[8, 1, 139, 78, 2, 179]
[32, 22, 81, 5, 264]
[8, 1, 139, 14, 55, 644, 105]
[32, 22, 81, 5, 264]
[8, 1, 139, 78, 2, 179]
[1, 139, 14, 140, 343, 11, 502, 50, 1, 39, 214, 3, 198, 35]
[35, 193, 2, 23, 6, 13]
[44, 382, 27, 2, 31, 35]
[35, 193, 2, 23, 6, 13]
[8, 50, 2, 90, 37, 298]
[10, 60, 383, 2, 581]
[233, 35, 199, 14, 151]
[233, 35, 14, 3, 198, 14, 3, 198, 582, 11, 829, 11, 137, 11, 830]
[109, 5, 82, 323, 24, 17, 129]
[645, 11, 155, 35, 468, 11, 583, 384]
[1, 831, 2, 34, 982]
[469, 106, 22, 18, 2, 143]
[503, 2, 19, 40, 38, 14, 276, 32, 22, 81, 5, 264]
[8, 1, 139, 78, 2, 179]
[32, 22, 81, 5, 264]
[8, 1, 139, 14, 55, 644, 105]
[32, 22, 81, 5, 264]
[8, 1, 139, 78, 2, 179]
[1, 139, 14, 140, 343, 11, 502, 50, 1, 39, 214, 3, 198, 35]
[35, 193, 2, 23, 6, 13]
[44, 382, 27, 2, 31, 35]
[35, 193, 2, 23, 6, 13]
[8, 50, 2, 90, 37, 298]
[10, 60, 383, 2, 581]
[233, 35, 199, 14, 151]
```

```
[233, 35, 14, 3, 198, 14, 3, 198, 35, 193, 2, 23, 6, 13]
[35, 193, 2, 23, 6, 13, 724, 13]
[35, 193, 2, 23, 6, 13]
[35, 193, 2, 23, 6, 13, 724, 13, 50, 1, 39, 214, 3, 198, 35]
[35, 193, 2, 23, 6, 13]
[44, 382, 27, 2, 31, 35]
[35, 193, 2, 23, 6, 13]
[8, 50, 2, 90, 37, 298]
[10, 60, 383, 2, 581]
[233, 35, 199, 14, 151]
[233, 35, 14, 3, 198, 14, 3, 198]
[233, 35, 14, 3, 198, 14, 3, 198, 35, 193, 2, 23, 6, 13, 37, 200, 5, 4, 51, 54, 7, 109, 2, 113]
[37, 200, 5, 4, 51, 54, 7, 548, 3, 65]
[37, 201, 5, 4, 51, 54]
[78, 5, 4, 51, 54]
[201, 5, 4, 51, 54]
[121, 4, 51, 54, 50, 1, 170, 4, 549, 1, 277, 234, 49, 3, 504]
[50, 2, 40, 214, 4, 584, 53, 470, 245, 14, 171]
[25, 37, 200, 5, 4, 51, 54]
[78, 5, 4, 51, 54]
[201, 5, 4, 51, 54]
```

```
# Membuat list kosong 'my_input_sequences' untuk menyimpan urutan token berdasarkan n-gram
my_input_sequences = []

# Melakukan iterasi pada setiap baris teks yang dipisahkan berdasarkan karakter newline ('\n')
for line in mytext.split('\n'):
    # Mengonversi setiap baris teks menjadi urutan token menggunakan my_tokenizer
    token_list = my_tokenizer.texts_to_sequences([line])[0]

    # Melakukan iterasi pada setiap indeks dalam token_list
    for i in range(1, len(token_list)):
        # Membuat n-gram sequence dari token_list hingga indeks ke-i
        my_n_gram_sequence = token_list[:i+1]

        # Menampilkan n-gram sequence ke layar (dapat di-comment atau dihapus jika tidak diperlukan)
        print(my_n_gram_sequence)

        # Menambahkan n-gram sequence ke dalam my_input_sequences
        my_input_sequences.append(my_n_gram_sequence)
        # print(input_sequences)
```

```
Streaming output truncated to the last 5000 lines.
[213, 3, 178, 213, 3, 1253, 213, 3, 2084, 72, 94, 166, 12, 269, 846, 718]
[213, 3, 178, 213, 3, 1253, 213, 3, 2084, 72, 94, 166, 12, 269, 846, 718, 2085]
[65, 130]
[65, 130, 97]
[65, 130, 97, 26]
[65, 130, 97, 26, 30]
[65, 130, 97, 26, 30, 17]
[381, 20]
[381, 20, 353]
```

```
[381, 20, 353, 53]
[381, 20, 353, 53, 7]
[381, 20, 353, 53, 7, 1170]
[381, 20, 353, 53, 7, 1170, 1]
[381, 20, 353, 53, 7, 1170, 1, 13]
[381, 20, 353, 53, 7, 1170, 1, 13, 2]
[381, 20, 353, 53, 7, 1170, 1, 13, 2, 8]
[381, 20, 353, 53, 7, 1170, 1, 13, 2, 8, 52]
[381, 20, 353, 53, 7, 1170, 1, 13, 2, 8, 52, 329]
[381, 20, 353, 53, 7, 1170, 1, 13, 2, 8, 52, 329, 660]
[35, 335]
[35, 335, 159]
[35, 335, 159, 15]
[35, 335, 159, 15, 6]
[35, 335, 159, 15, 6, 2086]
[35, 335, 159, 15, 6, 2086, 8]
[35, 335, 159, 15, 6, 2086, 8, 214]
[35, 335, 159, 15, 6, 2086, 8, 214, 4]
[35, 335, 159, 15, 6, 2086, 8, 214, 4, 53]
[8, 56]
[8, 56, 2]
[8, 56, 2, 90]
[8, 56, 2, 90, 12]
[8, 56, 2, 90, 12, 165]
[8, 56, 2, 90, 12, 165, 12]
[8, 56, 2, 90, 12, 165, 12, 2]
[8, 56, 2, 90, 12, 165, 12, 2, 90]
[8, 56, 2, 90, 12, 165, 12, 2, 90, 12]
[8, 56, 2, 90, 12, 165, 12, 2, 90, 12, 236]
[8, 56, 2, 90, 12, 165, 12, 2, 90, 12, 236, 5]
[8, 56, 2, 90, 12, 165, 12, 2, 90, 12, 236, 5, 1189]
[35, 225]
[35, 225, 152]
[35, 225, 152, 35]
[35, 225, 152, 35, 225]
[35, 225, 152, 35, 225, 152]
[35, 225, 152, 35, 225, 152, 35]
[35, 225, 152, 35, 225, 152, 35, 225]
[35, 225, 152, 35, 225, 152, 35, 225, 152]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838, 953]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838, 953, 7]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838, 953, 7, 157]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838, 953, 7, 157, 6]
[35, 225, 152, 35, 225, 152, 35, 225, 152, 8, 125, 24, 838, 953, 7, 157, 6, 1271]
```

```
# Menghitung panjang maksimum dari semua n-gram sequence dlm my_input_sequences
max_sequence_len = max([len(seq) for seq in my_input_sequences])

# Menggunakan pad_sequences dari NumPy untuk mengonversi my_input_sequences menjadi matriks 2D dengan panjang yang seragam
# Matriks ini akan berisi n-gram sequence yang sudah dipad dengan nilai 0 di bagian awal (padding='pre')
# digunakan sebagai input untuk model LSTM
input_sequences = np.array(pad_sequences(my_input_sequences, maxlen=max_sequence_len, padding='pre'))
```

```
# Tampilkan
input_sequences
```

```
array([[  0,   0,   0, ...,   0,  35, 826],
       [  0,   0,   0, ...,  35, 826,   2],
       [  0,   0,   0, ..., 826,   2,  61],
       ...,
       [  0,   0,   0, ...,  33,   4, 981],
       [  0,   0,   0, ...,   4, 981,  94],
       [  0,   0,   0, ..., 981,  94, 360]], dtype=int32)
```

```
# Memisahkan matriks input_sequences menjadi dua bagian: X dan y
X = input_sequences[:, :-1] # Mengambil semua kolom kecuali kolom terakhir
y = input_sequences[:, -1] # Mengambil kolom terakhir
```

```
X[2]
```

```
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,   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,  35, 826,   2], dtype=int32)
```

```
y[2]
```

```
61
```

```
X
```

```
array([[  0,   0,   0, ...,   0,   0,  35],
       [  0,   0,   0, ...,   0,  35, 826],
       [  0,   0,   0, ...,  35, 826,   2],
       ...,
       [  0,   0,   0, ...,  94,  33,   4],
       [  0,   0,   0, ...,  33,   4, 981],
       [  0,   0,   0, ...,   4, 981,  94]], dtype=int32)
```

```
y
```

```
array([826,   2,  61, ..., 981,  94, 360], dtype=int32)
```

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

```
y
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 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)
```

```
y[0]
```

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

Define Models

```
# Membuat objek model sequential untuk membuat model secara berurutan
model = tf.keras.models.Sequential()

# Menambahkan layer Embedding sebagai layer pertama dari model
# Embedding akan mengonversi input token menjadi vektor dg dimensi 100
# Input_length diatur sebagai panjang maksimum dari n-gram sequence dikurangi satu
model.add(Embedding(total_words, 100, input_length=max_sequence_len-1))

# Menambahkan layer LSTM (Long Short-Term Memory)
# LSTM utk memproses urutan data secara temporal
# jumlah unit LSTM yg digunakan : 150
model.add(LSTM(150))

# Menambahkan layer Dense sebagai layer output
# Dense dgn total_words unit dan fungsi aktivasi softmax
# Menghasilkan distribusi probabilitas utk setiap token pada output layer
model.add(Dense(total_words, activation='softmax'))

# Menampilkan ringkasan (summary) dari arsitektur model
print(model.summary())
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
 embedding_5 (Embedding)     (None, 69, 100)           227100

 lstm_5 (LSTM)               (None, 150)               150600

 dense_5 (Dense)             (None, 2271)              342921

=================================================================
Total params: 720621 (2.75 MB)
Trainable params: 720621 (2.75 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

`Model akan berhenti dilatih jika tidak ada peningkatan yang signifikan pada akurasi selama jumlah epoch yang ditentukan oleh patience:`

```
early_stopping = EarlyStopping(monitor='accuracy', patience=5, min_delta=0.01,
                               mode='max', verbose=1)
```

PENJELASAN

- monitor='accuracy': Memonitor metrik akurasi model selama pelatihan.
- patience=5: Menentukan jumlah epoch yang akan dipertahankan tanpa adanya peningkatan pada metrik yang dimonitor sebelum pelatihan dihentikan. Dalam contoh ini, pelatihan akan dihentikan jika akurasi tidak meningkat selama 5 epoch berturut-turut.
- min_delta=0.01: Menentukan perubahan minimum yang dianggap sebagai peningkatan. Jika perubahan akurasi kurang dari min_delta, itu tidak dianggap sebagai peningkatan.
- mode='max': Menunjukkan bahwa kita ingin memaksimalkan metrik yang dimonitor (dalam hal ini, akurasi). Jika kita memonitor loss dan ingin meminimalkannya, kita bisa menggunakan mode='min'.
- verbose=1: Menampilkan pesan informasi saat early stopping diaktifkan.

```
# Kompilasi model dg konfigurasi tertentu
# Loss function yg digunakan -> categorical crossentropy
# Optimizer yg digunakan -> Adam
# Metrik evaluasi yang digunakan -> akurasi
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

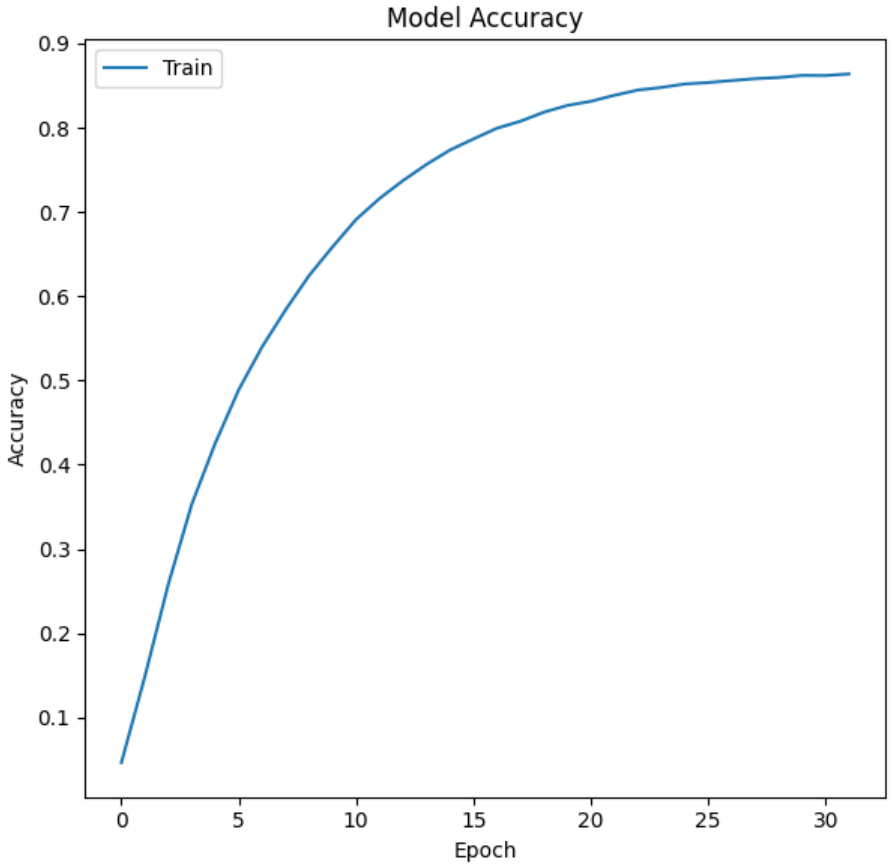
TRAINING MODEL

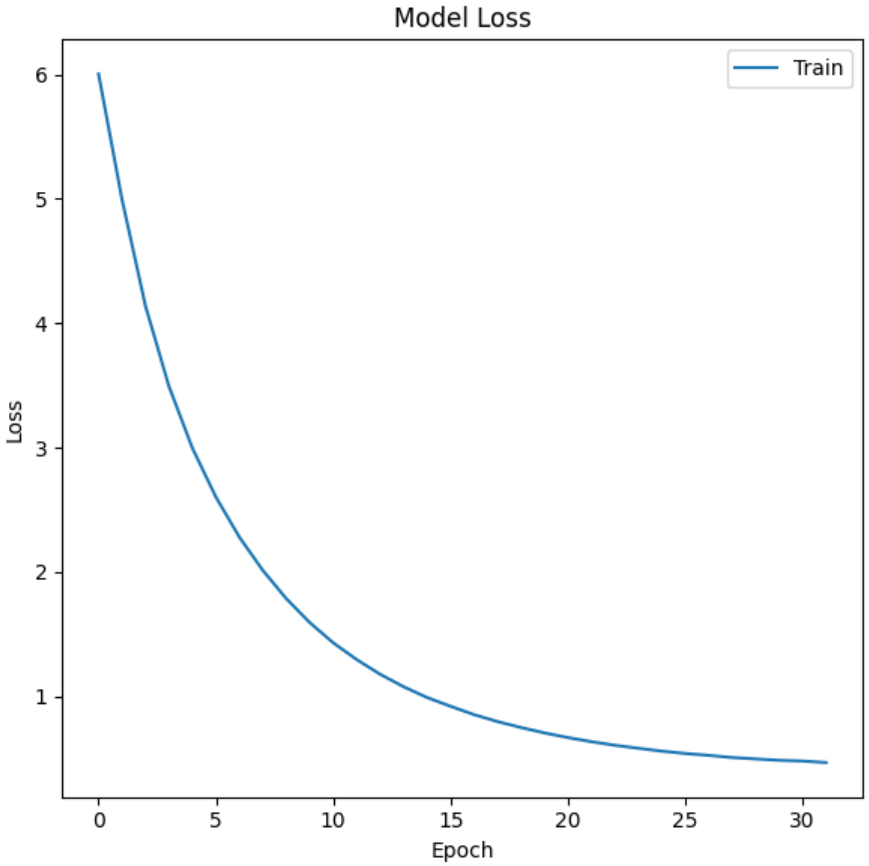
```
# Melatih model menggunakan data X dan y
# Model akan melatih dirinya sendiri utk memahami pola dlm urutan token
# Proses pelatihan akan dilakukan selama 100 epochs (iterasi)
# Verbose=1 mengaktifkan tampilan progres pelatihan
hist = model.fit(X, y, epochs=100, verbose=1, callbacks=[early_stopping])
```

```
Epoch 1/100
828/828 [==============================] - 69s 81ms/step - loss: 6.0039 - accuracy: 0.0465
Epoch 2/100
828/828 [==============================] - 65s 79ms/step - loss: 4.9953 - accuracy: 0.1486
Epoch 3/100
828/828 [==============================] - 66s 80ms/step - loss: 4.1391 - accuracy: 0.2583
Epoch 4/100
828/828 [==============================] - 66s 79ms/step - loss: 3.4932 - accuracy: 0.3533
Epoch 5/100
828/828 [==============================] - 65s 79ms/step - loss: 2.9960 - accuracy: 0.4257
Epoch 6/100
828/828 [==============================] - 64s 77ms/step - loss: 2.6035 - accuracy: 0.4890
Epoch 7/100
828/828 [==============================] - 64s 77ms/step - loss: 2.2815 - accuracy: 0.5401
Epoch 8/100
828/828 [==============================] - 63s 76ms/step - loss: 2.0138 - accuracy: 0.5840
Epoch 9/100
828/828 [==============================] - 63s 77ms/step - loss: 1.7866 - accuracy: 0.6246
Epoch 10/100
828/828 [==============================] - 64s 77ms/step - loss: 1.5942 - accuracy: 0.6586
Epoch 11/100
828/828 [==============================] - 63s 77ms/step - loss: 1.4316 - accuracy: 0.6911
Epoch 12/100
828/828 [==============================] - 63s 76ms/step - loss: 1.2970 - accuracy: 0.7160
Epoch 13/100
828/828 [==============================] - 63s 77ms/step - loss: 1.1787 - accuracy: 0.7372
Epoch 14/100
828/828 [==============================] - 64s 77ms/step - loss: 1.0785 - accuracy: 0.7564
Epoch 15/100
828/828 [==============================] - 65s 78ms/step - loss: 0.9918 - accuracy: 0.7734
Epoch 16/100
828/828 [==============================] - 63s 76ms/step - loss: 0.9213 - accuracy: 0.7866
Epoch 17/100
828/828 [==============================] - 64s 77ms/step - loss: 0.8545 - accuracy: 0.7993
Epoch 18/100
828/828 [==============================] - 63s 76ms/step - loss: 0.7986 - accuracy: 0.8076
Epoch 19/100
828/828 [==============================] - 64s 77ms/step - loss: 0.7508 - accuracy: 0.8183
Epoch 20/100
828/828 [==============================] - 63s 76ms/step - loss: 0.7075 - accuracy: 0.8264
Epoch 21/100
828/828 [==============================] - 63s 76ms/step - loss: 0.6700 - accuracy: 0.8313
Epoch 22/100
828/828 [==============================] - 63s 77ms/step - loss: 0.6369 - accuracy: 0.8383
Epoch 23/100
828/828 [==============================] - 63s 76ms/step - loss: 0.6081 - accuracy: 0.8446
Epoch 24/100
828/828 [==============================] - 63s 77ms/step - loss: 0.5841 - accuracy: 0.8477
```

```
Epoch 25/100
828/828 [==============================] - 65s 79ms/step - loss: 0.5608 - accuracy: 0.8518
Epoch 26/100
828/828 [==============================] - 65s 78ms/step - loss: 0.5421 - accuracy: 0.8535
Epoch 27/100
828/828 [==============================] - 64s 77ms/step - loss: 0.5270 - accuracy: 0.8559
Epoch 28/100
828/828 [==============================] - 64s 78ms/step - loss: 0.5097 - accuracy: 0.8581
Epoch 29/100
828/828 [==============================] - 64s 77ms/step - loss: 0.4984 - accuracy: 0.8595
```

```
# Mengimpor library matplotlib untuk membuat visualisasi plot
import matplotlib.pyplot as plt

# Membuat plot untuk melihat perubahan akurasi dan loss selama proses pelatihan model
plt.figure(figsize=(12, 6))  # Mengatur ukuran gambar plot

# Subplot pertama untuk plot akurasi
plt.subplot(1, 2, 1)
plt.plot(hist.history['accuracy'], label='Train')  # Plot akurasi data latih
plt.title('Model Accuracy')  # Menyertakan judul plot
plt.xlabel('Epoch')  # Label sumbu x
plt.ylabel('Accuracy')  # Label sumbu y
plt.legend()  # Menampilkan legenda

# Subplot kedua untuk plot loss
plt.subplot(1, 2, 2)
plt.plot(hist.history['loss'], label='Train')  # Plot loss data latih
plt.title('Model Loss')  # Menyertakan judul plot
plt.xlabel('Epoch')  # Label sumbu x
plt.ylabel('Loss')  # Label sumbu y
plt.legend()  # Menampilkan legenda

plt.tight_layout()  # Menyesuaikan tata letak plot untuk tampilan yang lebih rapi
plt.show()  # Menampilkan plot
```





Save Model

```
# Simpan model
model.save("modelku.h5")
```

Load Model

```
model_loaded = load_model("modelku.h5")
```

Make Prediction

```
# Membuat teks input awal yang akan digunakan untuk memprediksi kata-kata selanjutnya
input_text = "baby"

# Menentukan jumlah kata yang akan diprediksi selanjutnya
predict_next_words = 20

# Melakukan iterasi untuk memprediksi kata-kata selanjutnya berdasarkan teks input
for _ in range(predict_next_words):
    # Mengonversi teks input menjadi urutan token menggunakan tokenizer yang telah dipelajari sebelumnya
    token_list = my_tokenizer.texts_to_sequences([input_text])[0]

    # Melakukan padding pada urutan token untuk memastikan panjang yang sesuai dengan model
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')

    # Memprediksi probabilitas untuk kata-kata selanjutnya berdasarkan urutan token
    predictions = model_loaded.predict(token_list)[0]

    # Mengambil indeks dari kata-kata teratas yang diprediksi
    top_indices = np.argsort(predictions)[-5:][::-1]  # Mengambil 5 kata teratas, sesuaikan dengan kebutuhan

    # Mengambil kata-kata yang sesuai dengan indeks yang diprediksi
    next_words = [word for word, index in my_tokenizer.word_index.items() if index in top_indices]

    # Menampilkan teks input dan kata-kata selanjutnya beserta probabilitasnya
    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}")

    # Memilih kata dengan probabilitas tertinggi sebagai kata selanjutnya
    output_word = my_tokenizer.index_word[top_indices[0]]

    # Menambahkan kata yang dipilih ke dalam teks input untuk iterasi selanjutnya
    input_text += " " + output_word

# Menampilkan teks hasil prediksi
display(input_text)
```

```
1/1 [==============================] - 0s 21ms/step
Input Text: baby
Next Words and Probabilities:
i: 0.2244
don't: 0.1568
you're: 0.1271
cause: 0.1047
loves: 0.0815
1/1 [==============================] - 0s 24ms/step
Input Text: baby don't
Next Words and Probabilities:
you: 0.8108
give: 0.1574
have: 0.0072
stop: 0.0040
cry: 0.0037
1/1 [==============================] - 0s 21ms/step
Input Text: baby don't cry
Next Words and Probabilities:
you: 0.5243
it: 0.1404
for: 0.0899
handle: 0.0653
anymore: 0.0365
1/1 [==============================] - 0s 23ms/step
Input Text: baby don't cry anymore
Next Words and Probabilities:
to: 0.3839
love: 0.2994
can't: 0.1139
could: 0.0803
keep: 0.0183
1/1 [==============================] - 0s 23ms/step
Input Text: baby don't cry anymore to
Next Words and Probabilities:
feel: 0.6099
keep: 0.1356
leave: 0.0670
loose: 0.0392
hate: 0.0285
1/1 [==============================] - 0s 26ms/step
Input Text: baby don't cry anymore to keep
Next Words and Probabilities:
the: 0.2013
on: 0.1980
it: 0.1709
with: 0.1495
out: 0.0394
1/1 [==============================] - 0s 23ms/step
Input Text: baby don't cry anymore to keep it
Next Words and Probabilities:
in: 0.5187
dreams: 0.1232
last: 0.0300
inside: 0.0276
```

```
work: 0.0259
1/1 [==============================] - 0s 26ms/step
Input Text: baby don't cry anymore to keep it work
Next Words and Probabilities:
you: 0.9910
it: 0.0051
up: 0.0011
no: 0.0007
here: 0.0005
1/1 [==============================] - 0s 26ms/step
Input Text: baby don't cry anymore to keep it work it
Next Words and Probabilities:
in: 0.5213
can't: 0.3814
back: 0.0292
work: 0.0148
burn: 0.0093
1/1 [==============================] - 0s 28ms/step
Input Text: baby don't cry anymore to keep it work it back
Next Words and Probabilities:
and: 0.6127
in: 0.1804
with: 0.0544
of: 0.0314
last: 0.0257
1/1 [==============================] - 0s 25ms/step
Input Text: baby don't cry anymore to keep it work it back of
Next Words and Probabilities:
my: 0.6847
this: 0.1894
him: 0.0563
here: 0.0245
two: 0.0151
1/1 [==============================] - 0s 26ms/step
Input Text: baby don't cry anymore to keep it work it back of him
Next Words and Probabilities:
a: 0.4025
me: 0.1484
he: 0.0731
or: 0.0477
ain't: 0.0424
1/1 [==============================] - 0s 23ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't
Next Words and Probabilities:
is: 0.7383
no: 0.0581
him: 0.0316
enough: 0.0266
cheap: 0.0260
1/1 [==============================] - 0s 34ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no
Next Words and Probabilities:
your: 0.2393
we: 0.1185
enough: 0.0734
```

```
other: 0.0717
reason: 0.0587
1/1 [==============================] - 0s 22ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other
Next Words and Probabilities:
and: 0.3318
i'm: 0.2942
of: 0.1648
way: 0.0308
man: 0.0289
1/1 [==============================] - 0s 23ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other and
Next Words and Probabilities:
i: 0.4962
oh: 0.2665
i'm: 0.0699
if: 0.0395
that's: 0.0174
1/1 [==============================] - 0s 21ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other and if
Next Words and Probabilities:
i: 0.7738
you: 0.0659
my: 0.0629
i'm: 0.0450
you're: 0.0107
1/1 [==============================] - 0s 27ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other and if i
Next Words and Probabilities:
know: 0.1784
can't: 0.1483
am: 0.1232
left: 0.0830
promised: 0.0583
1/1 [==============================] - 0s 24ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other and if i left
Next Words and Probabilities:
i: 0.7184
you: 0.1197
my: 0.0741
it: 0.0431
everyone: 0.0139
1/1 [==============================] - 0s 26ms/step
Input Text: baby don't cry anymore to keep it work it back of him ain't no other and if i left my
Next Words and Probabilities:
heart: 0.7516
hair: 0.0744
head: 0.0636
phone: 0.0600
girlfriend: 0.0103
'baby don't cry anymore to keep it work it back of him ain't no other and if i left my head'
```

KESIMPULAN

- Dari output di atas, terlihat bahwa model telah menghasilkan prediksi untuk kata-kata selanjutnya berdasarkan teks input awal "baby". Iterasi dilakukan sebanyak 20 kali, dan pada setiap iterasi, model memprediksi beberapa kata yang memiliki probabilitas tertinggi.

```
# Menentukan teks input awal yang akan digunakan untuk memulai prediksi
input_text = "i will give you"
# Menentukan jumlah kata yang akan diprediksi selanjutnya
predict_next_words = 15

# Melakukan iterasi sebanyak predict_next_words untuk memprediksi kata-kata selanjutnya berdasarkan teks input
for _ in range(predict_next_words):
    # Mengonversi teks input menjadi urutan token menggunakan tokenizer yang telah dipelajari sebelumnya
    token_list = my_tokenizer.texts_to_sequences([input_text])[0]
    print(token_list)

    # Melakukan padding pada urutan token untuk memastikan panjang yang sesuai dengan model
    token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')

    # Memprediksi indeks token dengan probabilitas tertinggi menggunakan model
    predicted = np.argmax(model_loaded.predict(token_list), axis=-1)

    # Mendapatkan kata yang sesuai dengan indeks yang diprediksi
    output_word = ""
    for word, index in my_tokenizer.word_index.items():
        if index == predicted:
            output_word = word
            break
    # Menambahkan kata yang diprediksi ke dalam teks input untuk iterasi selanjutnya
    input_text += " " + output_word

# Tampilkan hasil
print(input_text)
```

```
[1, 139, 78, 2]
1/1 [==============================] - 0s 24ms/step
[1, 139, 78, 2, 179]
1/1 [==============================] - 0s 22ms/step
[1, 139, 78, 2, 179, 72]
1/1 [==============================] - 0s 22ms/step
[1, 139, 78, 2, 179, 72, 1]
1/1 [==============================] - 0s 22ms/step
```

```
[1, 139, 78, 2, 179, 72, 1, 31]
1/1 [==============================] - 0s 21ms/step
[1, 139, 78, 2, 179, 72, 1, 31, 12]
1/1 [==============================] - 0s 23ms/step
[1, 139, 78, 2, 179, 72, 1, 31, 12, 52]
1/1 [==============================] - 0s 22ms/step
[1, 139, 78, 2, 179, 72, 1, 31, 12, 52, 32]
1/1 [==============================] - 0s 21ms/step
[1, 139, 78, 2, 179, 72, 1, 31, 12, 52, 32, 563]
1/1 [==============================] - 0s 22ms/step
[1, 139, 78, 2, 179, 72, 1, 31, 12, 52, 32, 563, 25]
```