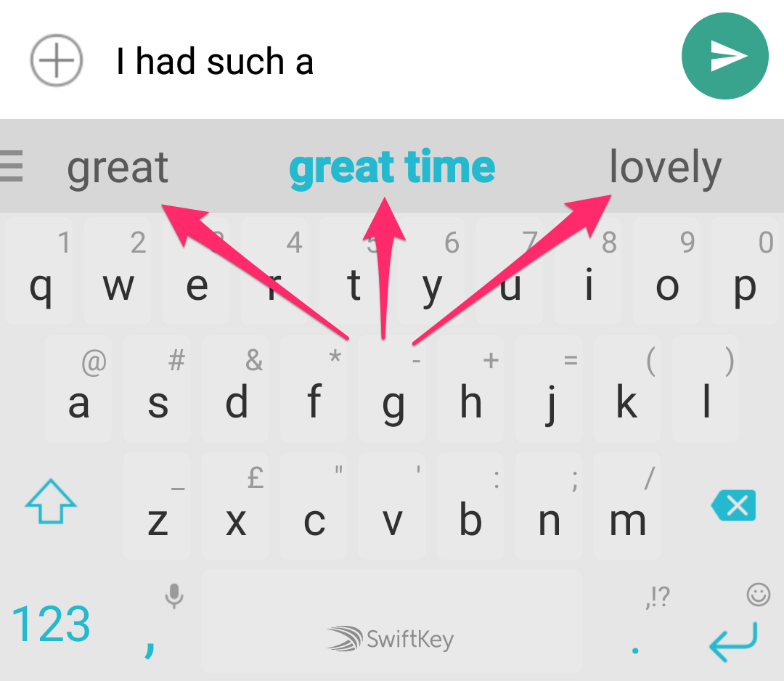
**Next Word Generator**

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

A close-up of a person's hand

Description automatically generated



LSTM (Long Short-Term Memory) -> salah satu jenis arsitektur jaringan saraf yang efektif dalam memahami dan memodelkan urutan data, seperti urutan kata dalam sebuah kalimat.

Flow pembuatan:

- Preprocessing (opsional)

- Tokenisasi

Tokenisasi adalah proses mengubah teks menjadi urutan kata atau token. Dalam konteks Next Word Generator, setiap kata diubah menjadi representasi numerik unik yang disebut token. Tokenisasi memungkinkan model untuk memahami dan memproses urutan kata.

- Pembuatan Sequences

Pada tahap ini, teks tokenized dipecah menjadi urutan kata yang disebut sequences. Setiap sequence terdiri dari beberapa kata yang membentuk konteks untuk memprediksi kata berikutnya.

- Pembuatan Model LSTM

Model LSTM digunakan untuk memahami dan memodelkan hubungan antar kata dalam sebuah sequence. LSTM memiliki kemampuan untuk mengatasi masalah vanishing gradient yang sering muncul dalam model rekuren tradisional. Model ini dilatih menggunakan sequences dari teks untuk memprediksi kata berikutnya

- Embedding

Layer embedding digunakan untuk mengkonversi setiap token ke dalam vektor ruang kata. Ini memungkinkan model untuk memahami representasi semantik dari setiap kata, meningkatkan kualitas prediksi.

- Training Model

Model dilatih menggunakan metode pembelajaran mesin supervised. Input model adalah sequences kata, dan outputnya adalah kata berikutnya. Proses pelatihan menggunakan optimizer dan fungsi loss untuk mengoptimalkan prediksi model.

- Generasi Kata Berikutnya

Setelah pelatihan, model dapat digunakan untuk memprediksi kata berikutnya berdasarkan input teks tertentu. Proses ini melibatkan tokenisasi input, penggunaan model untuk menghasilkan prediksi, dan konversi prediksi kembali ke dalam bentuk kata.

Langkah implementasi program:

**Import Library Needed**

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**

with open('C:/prak-nlp/next\_word/Artikel.txt', 'r', encoding='unicode\_escape') as myfile:

    mytext = myfile.read()

mytext

**Preprocessing**

my\_tokenizer = Tokenizer()

my\_tokenizer.fit\_on\_texts([mytext])

total\_words = len(my\_tokenizer.word\_index) + 1

my\_tokenizer.word\_index

Output:

A screen shot of a computer

Description automatically generated

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)*

Output:

A screen shot of a computer

Description automatically generated

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(n\_gram\_sequences)*

        my\_input\_sequences.append(my\_n\_gram\_sequence)

*# print(input\_sequences)*

Output:

A screenshot of a computer code

Description automatically generated

max\_sequence\_len = max([len(seq) for seq in my\_input\_sequences])

input\_sequences = np.array(pad\_sequences(my\_input\_sequences, maxlen=max\_sequence\_len, padding='pre'))

input\_sequences

Output:

A screenshot of a computer code

Description automatically generated

X = input\_sequences[:, :-1]

y = input\_sequences[:, -1]

**Define Models**

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())

Output:

A screenshot of a computer program

Description automatically generated

early\_stopping = EarlyStopping(monitor='accuracy', patience=5, min\_delta=0.01, mode='max', verbose=1)

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

**Training Models**

model.fit(X, y, epochs=100, verbose=1, callbacks=[early\_stopping])

A screenshot of a computer program

Description automatically generated

**Make Prediction**

input\_text = "peserta"

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, padding='pre')

    predicted = np.argmax(model.predict(token\_list), axis=-1)

    output\_word = ""

    for word, index in my\_tokenizer.word\_index.items():

        if index == predicted:

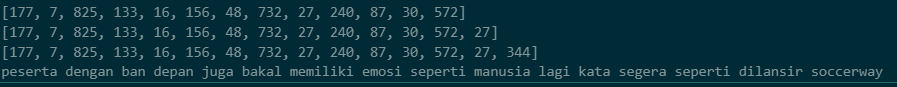
            output\_word = word

            break

    input\_text += " " + output\_word

print(input\_text)

Output:



**Save Model**

model.save("mymodel.h5")

**Load Model**

model\_loaded = load\_model("mymodel.h5")

­­­

Kesimpulan:

Model Next Word Generator menggunakan LSTM dapat digunakan untuk memahami dan memodelkan hubungan antar kata dalam teks. Penggunaannya dapat mencakup aplikasi otomatisasi penulisan, peningkatan antarmuka pengguna berbasis teks, dan banyak lagi. Penting untuk memiliki corpus data yang mencakup berbagai konteks agar model dapat memberikan hasil yang lebih baik.