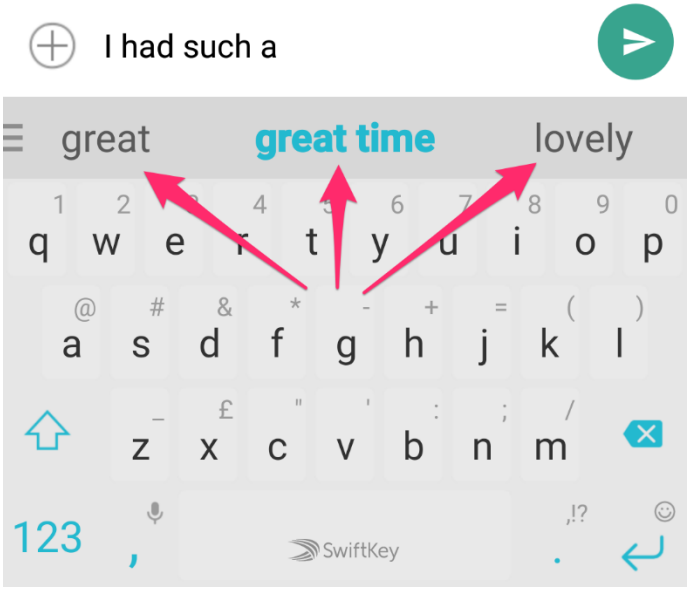
Next Word Generator

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



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:

```
{'di': 1,
 'yang': 2,
 'dan': 3,
 'ini': 4,
 'untuk': 5,
 'dari': 6,
```

```
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:

```
Jorginho Hindarkan Italia dari Kekalahan
jorginho adalah pemain italia dan jorginho adalah orang. Italia bermain imb
Polandia membuka ancaman ketika laga baru berjalan enam menit. Penjagaan ya
Lini pertahanan Italia terus diuji. Pada menit ke-26, Grzegorz Krychowiak y
Italia mencoba balik mengancam pada menit ke-37 melalui Federico Bernardesc
Pada menit ke-40, Zielinski mengantarkan Polandia memimpin. Tendangan first
Memasuki babak kedua, Italia mencoba bangkit. Bernardeschi kembali membuang
```

```
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:

```
[875, 2543, 350, 6, 630]
[875, 34, 71, 350, 3, 875, 34, 37, 350, 252, 737, 7, 543, 11, 72,
[543, 386, 1708, 144, 72, 100, 632, 740, 90, 2547, 2, 2, 15, 1709,
[484, 633, 350, 306, 2553, 11, 90, 20, 878, 2554, 2555, 2, 1710, 3
[350, 634, 1044, 1710, 11, 90, 20, 1265, 125, 2559, 1712, 388, 206
[11, 90, 20, 635, 1264, 1714, 543, 1045, 877, 2564, 2565, 1264, 12
[636, 545, 150, 350, 634, 1715, 1712, 126, 1716, 227, 5, 880, 281,
[11, 90, 20, 1046, 350, 389, 877, 739, 67, 2567, 2568, 190, 1267,
[637, 62, 62, 17, 174, 327, 72, 145, 350, 3, 543, 11, 72, 638, 1,
```

```
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:

```
array([[   0,    0,    0, ...,    0,  875, 2543],
       [   0,    0,    0, ...,  875, 2543,  350],
       [   0,    0,    0, ..., 2543,  350,    6],
       ...,
       [   0,    0,    0, ...,  841,  113,  545],
       [   0,    0,    0, ...,  113,  545,  150],
       [   0,    0,    0, ...,  545,  150,  391]])
```

```
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:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)      (None, 386, 100)          494300

 lstm_1 (LSTM)                (None, 150)               150600

 dense_1 (Dense)              (None, 4943)              746393

=================================================================
Total params: 1,391,293
Trainable params: 1,391,293
Non-trainable params: 0
_________________________________________________________________
None
```

```
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])
```

```
Epoch 1/100
740/740 [==============================] - 37s 48ms/step - loss: 7.6537 - accuracy: 0.0216
Epoch 2/100
740/740 [==============================] - 34s 47ms/step - loss: 7.1776 - accuracy: 0.0297
Epoch 3/100
740/740 [==============================] - 34s 47ms/step - loss: 6.8336 - accuracy: 0.0416
Epoch 4/100
740/740 [==============================] - 37s 51ms/step - loss: 6.3630 - accuracy: 0.0578
Epoch 5/100
740/740 [==============================] - 36s 48ms/step - loss: 5.8120 - accuracy: 0.0842
Epoch 6/100
740/740 [==============================] - 37s 50ms/step - loss: 5.2472 - accuracy: 0.1215
```

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:

```
[177, 7, 825, 133, 16, 156, 48, 732, 27, 240, 87, 30, 572]
[177, 7, 825, 133, 16, 156, 48, 732, 27, 240, 87, 30, 572, 27]
[177, 7, 825, 133, 16, 156, 48, 732, 27, 240, 87, 30, 572, 27, 344]
peserta dengan ban depan juga bakal memiliki emosi seperti manusia lagi kata segera seperti dilansir soccerway
```

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.