AN AUTOMATIC LYRICS MAKER BASED ON TEXT USING MARCHINE LEARNING

Nama : **Siti Aisya**

NIM : **09011182025001**

Jurusan : **Sistem Komputer**

Dosen Pengampu : **Prof. Dr. Ir. Siti Nurmaini, M.T.**

PROGRAM STUDI SISTEM KOMPUTER

UNIVERSITAS SRIWIJAYA

2022/2023

I. PENDAHULUAN

Project ini dibuat untuk memenuhi tugas mata perkuliah Kecerdasan Buatatan atau Artificial Intelligence (AI). Pada project tersebut saya mengunakan Google Colab untuk mencoba menjalankan perintah program yang sudah dibuat. Program tersebut merupakan program untuk membuat aplikasi AI menggunakan Deep learning untuk membuat lirik lagu otomatis mengguanakan Bahasa Indonesia. Yang mana jika program tersebut dijalankan dan kita memasukkan sebuah inputan kata atau kalimat pada perintah program yang telah ditentukan, maka program tersebut akan otomatis menyambungkan lirik dari kata atau kalimat yang sudah kita masukkan pada program tersebut.

Pembuat Lirik lagu otomatis adalah aplikasi yang menggunakan Kecerdasan Buatan (AI) untuk membantu kita agar dapat menulis lirik sendiri untuk lagu. Menggunakan pembuat lirik berbasis AI adalah cara luar biasa untuk memulai membuat dan mengarang lagu sendiri. Mereka umumnya memiliki database kata, frasa yang besar, dll. yang dapat kita gunakan untuk membuat lirik lagu.

Selain itu, AI dapat menghilangkan elemen emosional manusia dalam seni penulisan lagu karena mesin tidak dapat memahami emosi manusia. Namun para pendukung berpendapat bahwa penulisan lagu AI dapat memungkinkan peningkatan kualitas musik karena memungkinkan penulis memusatkan perhatian mereka pada aspek lain dari penciptaan lagu seperti melodi, nada, aransemen, dan lainnya, serta menghadirkan bentuk musik yang inovatif. Jadi, projek tersebut dibuat untuk dapat memberikan inovasi dan kreatifitas seseorang dalam menentukan sebuah kata untuk dijadikan dalam sebuah lagu.

II. DATASET

Dataset ini didapatkan dari hasil scraping dari beberapa situs penyedia lirik lagu. Yang mana Scraping adalah proses pengambilan data atau esktraksi dari sebuah website, lalu data tersebut umumnya disimpan dalam sebuah format tertentu.

Berikut merupakan link dari dasate dan ny pre-trained model yang saya gunakan pada project tersebut.

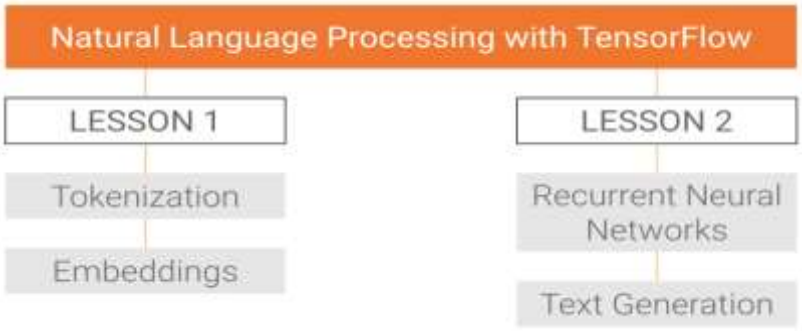
Link Download Dataset : https://s.id/AKgGy

Link Download Pre-Trained Model : https://s.id/AKgO9

Source code program scrapping lirik lagu : https://github.com/share424/lyric-scrapper

III. METODE

Disini saya menggunakan Deep Learning dengan metode Natural Language Processing (NLP), dapat dilihat pada gambar dibawah ini



Adapun penjelasannya yaitu sebagai berikut :

- **Deep Learning**

  Deep learning merupakan subbidang machine learning yang algoritmanya terinspirasi dari struktur otak manusia. Saat ini, teknik deep learning sangat populer di kalangan praktisi data dan menarik perhatian banyak pihak. Hal ini karena teknologi deep learning telah diterapkan dalam berbagai produk berteknologi tinggi seperti self-driving car. Selain itu, ia juga ada di balik produk dan layanan yang kita gunakan sehari-hari. Contohnya antara lain, asisten digital, Google Translate, dan voice-activated device (perangkat cerdas yang bisa diaktifkan dengan suara).

- **Natural language processing (NLP)**

  Natural language processing (NLP) adalah cabang dari kecerdasan buatan yang berhubungan dengan interaksi antara komputer dan manusia menggunakan bahasa alami. Menurut Textmetrics, NLP digunakan untuk mengukur sentimen dan menentukan bagian mana dari bahasa manusia yang penting.

- **Tokenization**

  Tokenisasi adalah proses untuk membagi teks yang dapat berupa kalimat, paragraf atau dokumen, menjadi token-token/bagian-bagian tertentu. Sebagai contoh, tokenisasi dari kalimat "Aku baru saja makan bakso pedas" menghasilkan enam token, yakni: "Aku", "baru", "saja", "makan", "bakso", "pedas". Biasanya, yang menjadi acuan pemisah antar token adalah spasi dan tanda baca. Tokenisasi sering kali dipakai dalam linguistik dan hasil tokenisasi berguna untuk analisis teks lebih lanjut. Contoh program tokenisasi yang dapat diakses dan digunakan secara daring

adalah morphadorner dan NLTK Tokenizer. Tokenization juga merupakan proses pengiriman data sensitif melalui panggilan Application Programming Interface (API) atau file batch ke penyedia tokenisasi yang kemudian menggantikan data tersebut dengan placeholder berbentuk tidak sensitif yang disebut token.

- **Embedding dan Word Embedding**

  Istilah embeddings, kata sebenarnya adalah kelas teknik di mana kata-kata individual direpresentasikan sebagai vektor bernilai nyata dalam ruang vektor yang telah ditentukan.Setiap kata dipetakan ke satu vektor dan nilai-nilai vektor dipelajari dengan cara yang menyerupai jaringan saraf dalam Artificial Intelligence atau yang sering dikenal dengan kecerdasan buatan, dan karenanya teknik ini sering disamakan dengan bidang pembelajaran yang mendalam.

  Word embeddings adalah representasi terdistribusi untuk teks yang mungkin merupakan salah satu terobosan kunci untuk kinerja yang mengesankan dari metode pembelajaran mendalam pada masalah pemrosesan bahasa alami yang menantang.

- **Recurrent Neural Networks (RNN)**

  Recurrent Neural Networks (RNN) merupakan salah satu bentuk arsitektur Artificial Neural Networks (ANN) yang dirancang khusus untuk memproses data yang bersambung/ berurutan (sequential data). RNN biasanya digunakan untuk menyelesaikan permasalahan data historis atau time series, contohnya data ramalan cuaca. Selain itu, RNN juga dapat diimplementasikan pada bidang natural language understanding (pemahaman bahasa alami), misalnya translasi bahasa.

- **Text Generation**

  Text generation adalah step yang dilakukan untuk memprediksi the next most likely word atau kata berikutnya yang paling besar.

IV. PENJELASAN OPEN SOURCE

Import Library

Terdapat beberapa library yang digunakan untuk menjalankan program dari project ini, yaitu sebagai berikut :

```
import tensorflow as tf
import numpy as np
import json
import re
import os
import logging
```

- **Tensorflow** : library open source yang digunakan untuk komputasi numerik dan project machine learning berskala besar.
- **Numpy** : memiliki fungsi yang bekerja dalam domain aljabar linier, transformasi fourier, dan matriks
- **Json** : sebuah format data yang digunakan untuk menyimpan sebuah informasi (data).
- **Re** : untuk mencari sebuah string untuk match (match)
- **Os** : berfungsi untuk path atau memanggil alamat folder.
- **Logging** : untuk proses debug dan pemantauan kode

Melakukan Preprocess

Proses tersebut bertujuan untuk menghapus karakter dan simbol aneh pada lirik yang akan dihasilkan nantinya karena sebelumnya dataset yang diambil adalah hasil scraping dan masih berantakan.

```
def preprocess(lyric, max_length=None):
    lyric = lyric.lower().strip()
    lyric = lyric.replace("<newline>", " <newline> ")

    lyric = re.sub(r"([?.!,])", r" \1 ", lyric)
    lyric = re.sub(r'([" "]+)', " ", lyric)
    lyric = re.sub(r"[^a-zA-Z?.!,<>]", " ", lyric)
    lyric = lyric.strip()

    if max_length != None:
        lyric = " ".join(lyric.split(" ")[:max_length])

    return "<start> " + lyric + " <end>"

preprocess("Aku... dan<newline> kamu", 5)
```

Memasukkan Dataset

Disini kita memanggil dataset yang sudah didapatkan sebelumnya dengan file name yaitu : “lyric_bahasa.json”, yang mana pada setiap lagu memiliki panjang 162.

```
def create_dataset(filename, max_length=None):
    dataset = []
    with open(filename, "r") as file:
        dataset = json.loads(file.read())
    preprocessed_lyric = [preprocess(song["lyric"], max_length) for song in dataset if len(song["lyric"])
    return preprocessed_lyric

dataset = create_dataset("lyric_bahasa.json", 162)
```

Membuat Tokenizer

Tokenizer dibuat untuk mengubah semua lirik dari huruf menjadi sebuah angka

```
def create_tokenizer(lyrics, num_words=None):
    tokenizer = tf.keras.preprocessing.text.Tokenizer(filters="", num_words=num_words, oov_token="<unk>")
    tokenizer.fit_on_texts(lyrics)
    return tokenizer

def tokenize(tokenizer, lyrics):
    tensor = tokenizer.texts_to_sequences(lyrics)
    tensor = tf.keras.preprocessing.sequence.pad_sequences(tensor, padding='post')
    return tensor
def load_dataset(filename, num_words, max_length):
    dataset = create_dataset(filename, max_length)
    tokenizer = create_tokenizer(dataset, num_words)
    input_tensor = tokenize(tokenizer, dataset)

    return tokenizer, input_tensor
```

Jumlah Tensor

Dapat dilihat pada gambar dibawah ini merupakan ukuran tensor yang berjumlah 23181 lagu dan pada setiap lagu memiliki panjang 162.

```
tokenizer, input_tensor = load_dataset("lyric_bahasa.json", 10000, 160)
input_tensor.shape
```

```
(23181, 162)
```

Proses Mapping

Jika ingin melakukan proses mapping dari angka index menjadi kata yang sebenanya dapat dilihat pada gambar berikut

```
for t in input_tensor[0]:
    if t == 0:
        continue
    print(t, "=>", tokenizer.index_word[t])
```

```
14 => <start>
16 => ada
79 => rindu
9 => di
1254 => malaamku
2 => <newline>
16 => ada
460 => resah
9 => di
1892 => tidurku
2 => <newline>
16 => ada
783 => tangis
9 => di
32 => hatiku
2 => <newline>
16 => ada
723 => hasrat
5 => yang
```

Membuat Label

```
def split_input_target(sequence):
    input_tensor = sequence[:-1]
    target_tensor = sequence[1:]
    return input_tensor, target_tensor
```

```
split_input_target(["saya", "dan", "dia"])
```

```
(['saya', 'dan'], ['dan', 'dia'])
```

Membuat Object Dataset

Adapun object pada dataset dapat dilihat pada gambar berikut :

- Buffer Size : untuk mengshuffle data agar ukurannya sesuai dengan input tensor
- Dataset diambil dari tensor dan di input tensornya
- Data yang diambil sebanyak 64 data yang mana di satu data itu terdapat 161 untuk input dan juga targetnya.

```
BUFFER_SIZE = len(input_tensor)
BATCH_SIZE = 64
embedding_dim = 256
units = 1024

dataset = tf.data.Dataset.from_tensor_slices(input_tensor).shuffle(BUFFER_SIZE).map(split_input_target)
dataset = dataset.batch(BATCH_SIZE, drop_remainder=True)
```

```
example_input_batch, example_target_batch = next(iter(dataset))
example_input_batch.shape, example_target_batch.shape
```

```
(TensorShape([64, 161]), TensorShape([64, 161]))
```

Membuat Model

Terdapat 3 cara untuk membuat model dari tensorflow, tetapi disini model yang digunakan yaitu menggunakan Sequential, dan disini kita akan membuat tunpukan layernya. Dapat dilihat pada gambar berikut :

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim),
    tf.keras.layers.GRU(units, return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.GRU(units, return_sequences=True),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(vocab_size, activation='softmax')
])
```

Model Summary

Hasil output di bawah merupakan model yang akan dimasukkan ke dalam rumus output shape sehingga dapat diketahui output shape dari setiap layer. Total dari parameter yang akan dikerjakan oleh deep learning adalah **23,047,185** Perhitungan.

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 256)         2560256

 gru (GRU)                   (None, None, 1024)        3938304

 dropout (Dropout)           (None, None, 1024)        0

 gru_1 (GRU)                 (None, None, 1024)        6297600

 dropout_1 (Dropout)         (None, None, 1024)        0

 dense (Dense)               (None, None, 10001)       10251025

=================================================================
Total params: 23,047,185
Trainable params: 23,047,185
Non-trainable params: 0
```

Membuat Checkpoint

Bertujuan untuk menyimpan model yang sudah ada untuk jaga-jaga jika nanti ketika di running atau dijalankan terjadinya pemadaman listrik.

```
checkpoint_dir = 'training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, 'ckpt_{epoch}.h5')

checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_prefix,
    save_weights_only=True,
    monitor='loss',
    mode='min',
    save_best_only=True
)
```

Load Model

```
model.load_weights("checkpoint.h5")
```

Membuat Function

Bertujuan untuk mengenerate text

```
def greedy_search(seed, max_length=150):
    start = seed.strip()
    sequences = [tokenizer.word_index[i] for i in start.lower().split(" ")]
    for i in range(max_length):
        x = np.array([sequences])
        pred = model.predict(x)
        pred_id = np.argmax(pred[0][-1])
        if pred_id == 0 or pred_id == tokenizer.word_index["<end>"]:
            break
        sequences.append(pred_id)
    print_sequence(sequences)
```

V. OUTPUT

- **Menggunakan Greedy Seacrh**

```
greedy_search("<start> hujan")
```

```
hujan beta arti , luka tak juga duda
hangati melayang , hatimu televisi
hai ini coba
tapi jangan kau lupa indah kasih turun
ayo yang sekarang patah kau rasa
pasti kan ada tinggal
tak akan jiwaku pintaku ini
di milikku  kan ada pelangi
niat cemburu kan terhebat hati
jangan ayo lagi
hujan beta arti , luka tak juga duda
hangati melayang , hatimu televisi
hai ini coba
tapi jangan kau lupa indah kasih turun
ayo yang sekarang patah kau rasa
pasti kan ada tinggal
tak akan jiwaku pintaku ini , di milikku  kan ada pelangi
niat cemburu kan terhebat hati , jangan ayo lagi
tak akan jiwaku pintaku ini , di milikku  kan ada pelangi
niat cemburu kan terhebat hati , jangan ayo lagi
hujan kan tenggelam ,
```

- **Menggunakan Beam Seacrh**

```
beam_search("<start> engkau ")
```

```
engkau datang padaku
kala  gelap hatiku
engkau datang padaku
membawa luka yang kini ada

jangan biarkan rindu yang terluka
jangan datang lalu datang

aku takkan ada
perhatianku kasih kita coba
```

Dapat dilihat disini untuk percobaan membuat lirik lagu sendiri menggunakan dua buah search yaitu greedy search dan beam search. Dari hasil yang di dapatkan perbandingan jika menggunakan beam search maka lirik lagu yang keluar lebih baik daripada menggunakan greedy search. Dapat ditarik kesimpulan bahwa program untuk membuat aplikasi membuat lirik lagu sendiri menggunakan AI tersebut dikatakan berhasil, karena ouput nya sudah tampil atau sudah di dapatkan, tetapi jika ingin lebih baik dan lirik lebih banyak lagi bisa diatur sendiri panjang kata dari setiap liriknya juga dapat diatur sendiri untuk mengatur dari setiap katanya agar tidak terjadinya perulangan kata.

DAFTAR PUSTAKA

https://www.dicoding.com/blog/kecerdasan-buatan-adalah/

https://algorit.ma/blog/natural-language-processing/

https://www.dicoding.com/blog/mengenal-deep-learning/

https://glints.com/id/lowongan/natural-language-processing-adalah/#.YzbvNtdBzIU\

https://id.wikipedia.org/wiki/Tokenisasi

https://rifqimulyawan.com/blog/pengertian-tokenization/

https://rifqimulyawan.com/blog/pengertian-embedding/