LAPORAN UAS DEEP LEARNING

CHATBOT INFORMASI SKRIPSI INFORMATIKA UNIVERSITAS BENGKULU

DISUSUN OLEH :

1. **Rimaya Dwi Atika (G1A021021)**
2. **Rahayu Ningrum Puspa Ridha (G1A021071)**

DOSEN PENGAMPU :

Arie Vatresia, S.T., M.T.I., Ph.D

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

1. Pendahuluan

Menyelesaikan skripsi adalah salah satu tantangan terbesar yang dihadapi selama pendidikan strata 1, terutama bagi mahasiswa yang mengikuti program studi informatika di Universitas Bengkulu. Skripsi bukan hanya merupakan syarat kelulusan, tetapi juga menunjukkan bahwa mahasiswa memiliki kemampuan untuk menerapkan apa yang mereka pelajari selama berkuliah dalam penelitian ilmiah. Walaupun terdapat pedoman skripsi yang sudah disediakan, seringkali ada mahasiswa yang kebingungan dan bahkan malas membaca dokumen. Oleh karena itu, memilih subjek, mengumpulkan data, memahami peraturan akademik, administrasi, menyelesaikan sidang dan publikasi ilmiah merupakan masalah yang dihadapi mahasiswa.

Jumlah persyaratan administrasi membuat mahasiswa bingung terkait hal yang perlu disiapkan, bagaimana menulis, atau kapan sidang harus diakhiri. Ada petunjuk yang jelas dalam skripsi tentang cara memilih judul atau topik penelitian yang sesuai dengan Kerangka Kualifikasi Nasional Indonesia (KKNI). Oleh karena itu, menjawab pertanyaan dengan cepat sangat membantu dalam menyelesaikan tantangan ini.

Terdapat peluang untuk mengatasi masalah ini dengan cara yang lebih inovatif berkat kemajuan teknologi, terutama dalam *deep learning*. Sistem cerdas berbasis *natural language processing* (NLP) adalah teknologi pembelajaran yang mencakup sistem informasi terintegrasi yang membantu mahasiswa dapat mengikuti jalan skripsi, chatbot yang menjawab pertanyaan mahasiswa, dan rekomendasi otomatis berdasarkan data historis tentang topik skripsi yang tepat untuk mereka. Diharapkan chatbot ini akan membuat mahasiswa informatika universitas bengkulu lebih efektif dan efisien dalam menemukan informasi yang berguna untuk menyelesaikan skripsi mereka.

2. Analisis Model

Sistem chatbot yang dikembangkan ini menggunakan algoritma *Long Short Term Memory* (LSTM) yang termasuk dalam algoritma *deep learning* karena terdiri dari arsitektur berlapis seperti *input layer*, *hidden layers*, dan *output layer*. LSTM ini memiliki kemampuan dalam mempelajari pola kompleks dalam data sekuensial sehingga dapat menangkap depedensi temporal dalam jangka panjang yang menjadi ciri khas *deep learning* agar dapat mengekstraksi fitur otomatis dari data besar. Sedangkan, algoritma yang dapat dikatakan *shallow learning* adalah algoritma yang hanya memiliki sedikit lapisan dan membutuhkan rekayasa fitur manual dalam pelatihan sehingga lebih cocok untuk mengenali pola sederhana dan dataset kecil. Contoh algoritma shallow learning adalah SVM atau *Decision Tree*.

Pada proyek ini menghasilkan model dengan akurasi yang tinggi dan loss yang rendah sehingga LSTM terbukti dapat memberikan performa yang baik dalam pelatihan data teks. Pada LSTM terdapat lapisan *embedding* yang akan mengubah data teks menjadi numerik agar data dapat diolah. Selain itu, data melewati pra-processing mulai dari menghapus punktuasi, lematisasi, tekenisasi, memberikan padding dan melalukan konversi keluaran dengan enkoding. Dari hasil pelatihan terlihat bahwa model dapat mempelajari pola dan melakukan prediksi dengan baik.

3. Penjelasan Kode

3.1 Pengumpulan Data

```
[ ] import json

    intents = {
        "intents": [

        # Contoh pembagian intent manual dari bagian tertentu
        {
            "tag": "greetings",
            "patterns": [
                "hai",
                "halo",
                "hallo",
                "hy",
                "hi",
                "morning",
                "pagi",
                "siang",
                "malam",
                "Good morning",
                "good afternoon",
                "good evening",
                "good night",
                "kawan",
                "bro",
                "sis",
                "Kon'nichiwa",
                "What's new?",
                "yo",
                "Hi there"

            ],
            "responses": [
                "Hai, Ada yang bisa Skripsi Informatika Assistant (SIA) UNIB bantu?",
                "Halo, Ada yang bisa Skripsi Informatika Assistant (SIA) UNIB bantu?",
                "Saya Skripsi Informatika Assistant (SIA) UNIB akan membantumu dalam menemukan jawaban",
                "Hi, Ada yang bisa Skripsi Informatika Assistant (SIA) UNIB bantu?",
                "Yo, Ada yang bisa Skripsi Informatika Assistant (SIA) UNIB bantu?"

            ],
            "context": [""]
        },
        {
                "tag": "alur_skripsi",
                "patterns": [
                    "Bagaimana alur pengambilan skripsi?",
                    "Alur pengambilan skripsi bagaimana?"
                ],
                "responses": [
                    """
                    Berikut adalah alur dalam pengambilan mata kuliah skripsi:\n
                    1. Mengambil Mata Kuliah Skripsi\n
```

Penjelasan :

Data yang digunakan dalam proyek SIAbot_UNIB ini adalah data yang dikumpulkan dari buku Panduan Skripsi Informatika Universitas Bengkulu dan informasi umum terkait rekomendasi judul dari laman internet. Pertama, informasi dari buku panduan akan disaring menjadi dataset. Kemudian dataset tersebut akan diolah menjadi susunan percakapan yang mencakup pertanyaan dan jawaban seputar informasi skripsi informatika Universitas Bengkulu dan diubah kedalam format .json . Dataset ini yang akan digunakan dalam melatih

SIAbot_UNIB agar dapat memberikan respon yang akurat dari pertanyaan yang diajukan pengguna.

3.2 Memuat Data .json

```
[ ]  !pip -q install gtts
```

Penjelasan :

Kode ini digunakan untuk menginstall pustaka dari *Google text to speech* yang akan menjadi fitur tambahan untuk SIAbot_UNIB agar dapat mengkonversi jawaban teks menjadi suara.

```
[ ]  # Kumpulan Pustaka

     import json  # Library untuk bekerja dengan data JSON
     import nltk  # Library untuk Natural Language Processing (NLP)
     import time  # Library untuk pengukuran waktu
     import random  # Library untuk menghasilkan angka acak
     import string  # Library untuk operasi pada string
     import pickle  # Library untuk serialisasi dan deserialisasi data Python
     import numpy as np  # Library untuk operasi array dan komputasi numerik
     import pandas as pd  # Library untuk manipulasi data berbasis tabel
     from io import BytesIO  # Modul untuk operasi input/output berbasis byte
     import tensorflow as tf  # Library untuk machine learning dan deep learning
     import IPython.display as ipd  # Modul untuk menampilkan konten multimedia di Jupyter Notebook
     import matplotlib.pyplot as plt  # Library untuk visualisasi data
     from nltk.stem import WordNetLemmatizer  # Modul untuk lemmatization pada NLP
     from tensorflow.keras.models import Model  # Modul untuk membangun model menggunakan Keras
     from keras.utils import plot_model  # Modul untuk memvisualisasikan arsitektur model
     from sklearn.preprocessing import LabelEncoder  # Modul untuk mengkodekan label target
     from tensorflow.keras.preprocessing.text import Tokenizer  # Modul untuk tokenisasi teks
     from tensorflow.keras.layers import Input, Embedding, LSTM  # Layer yang digunakan untuk membangun model
     from tensorflow.keras.preprocessing.sequence import pad_sequences  # Modul untuk padding sequence teks
     from tensorflow.keras.layers import Flatten, Dense, GlobalMaxPool1D  # Layer tambahan untuk model
```

Penjelasan :

Kode ini digunakan untuk mengimpor pustaka yang dibutuhkan untuk melakukan pelatihan model dengan LSTM. Pustaka yang digunakan berupa *NumPy* untuk komputasi matematika, *Matplotlib* untuk visualisasi model data, *Natural Language Toolkit* atau NLTK untuk pengolahan teks, *Pandas* untuk membaca data, serta *Tensorflow* untuk model pada data menggunakan algoritma LSTM (*Long Short Term Memory*).

```
[ ]  # Paket untuk tokenisasi kalimat
     nltk.download('punkt')
     # Download the 'punkt_tab' package untuk sentence tokenization.
     nltk.download('punkt_tab')
     # Paket untuk lemmatization
     nltk.download('wordnet')
     # Paket data WordNet multibahasa
     nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

Penjelasan :

Kode diatas digunakan untuk mengunduh paket yang dibutuhkan untuk tokenisasi kalimat dan kata, mengunduh basis data WordNet untuk lematisasi, dan mengunduh multibahasa dari multilingual WordNet untuk mendukung operasi berbagai bahasa.

```
# Importing the dataset
with open('/content/SIAbot_UNIB.json', encoding='utf-8') as content:
  data1 = json.load(content)

# Mendapatkan semua data ke dalam list
tags = [] # data tag
inputs = [] # data input atau pattern
responses = {} # data respon
words = [] # Data kata
classes = [] # Data Kelas atau Tag
documents = [] # Data Kalimat Dokumen
ignore_words = ['?', '!'] # Mengabaikan tanda spesial karakter
# Tambahkan data intents dalam json
for intent in data1['intents']:
  responses[intent['tag']]=intent['responses']
  for lines in intent['patterns']:
    inputs.append(lines)
    tags.append(intent['tag'])
    # digunakan untuk pattern atau teks pertanyaan dalam json
    for pattern in intent['patterns']:
      w = nltk.word_tokenize(pattern)
      words.extend(w)
      documents.append((w, intent['tag']))
      # tambahkan ke dalam list kelas dalam data
      if intent['tag'] not in classes:
        classes.append(intent['tag'])

# Konversi data json ke dalam dataframe
data = pd.DataFrame({"patterns":inputs, "tags":tags})
```

```
# Cetak data keseluruhan
data
```

	patterns	tags
0	hai	greetings
1	halo	greetings
2	hallo	greetings
3	hy	greetings
4	hi	greetings
...	...	...
97	Sampai jumpa	goodbye
98	Bye	goodbye
99	Selamat tinggal	goodbye

Penjelasan :

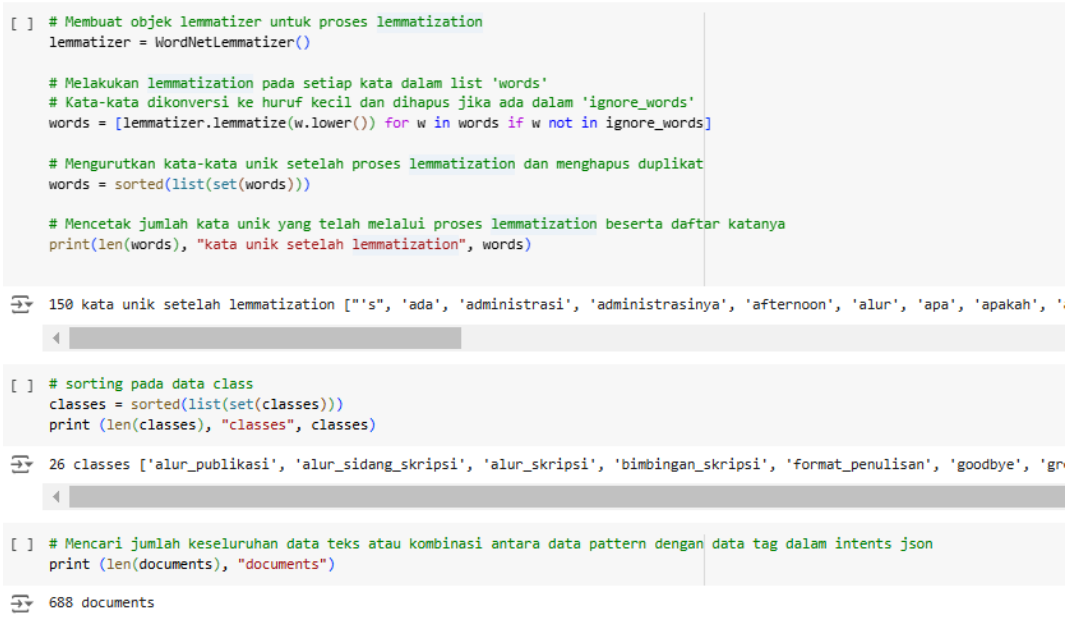
Kode diatas digunakan untuk membaca dataset SIAbot_UNIB dalam format JSON, memprosesnya, dan mengonversinya menjadi DataFrame terstruktur. Dataset ini berisi intents (tujuan), pola teks (patterns), dan respons (responses). Kode dimulai dengan membuka file JSON menggunakan modul `json` dan membaca isinya.

3.3 Melakukan Pengolahan Data

```
[ ] # Removing Punctuations (Menghilangkan Punktuasi) atau tanda baca seperti special character yaitu ! , ? dan tanda baca
    data['patterns'] = data['patterns'].apply(lambda wrd:[ltrs.lower() for ltrs in wrd if ltrs not in string.punctuation])
    data['patterns'] = data['patterns'].apply(lambda wrd: ''.join(wrd))
```

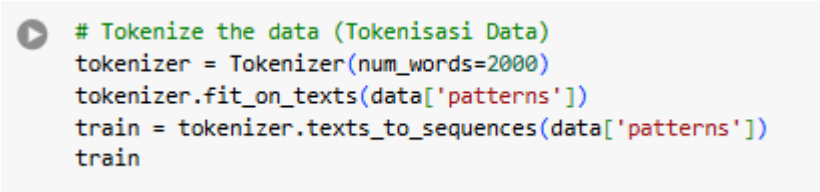
Penjelasan :

Kode diatas digunakan untuk menghapus punktuasi atau tanda baca seperti special character yaitu ! (tanda seru) , (tanda koma) . (tanda titik sebagai berhenti) ? (tanda tanya) dan tanda baca yang lain.



Penjelasan :

Kode diatas digunakan untuk melakukan proses *lemmatization* pada kata-kata dalam dataset SIAbot_UNIB, mengurutkan data kelas (*classes*), dan menghitung jumlah dokumen (*documents*) yang telah diproses. Pertama, sebuah objek *WordNetLemmatizer* dibuat untuk membantu normalisasi kata. Setelah itu, kata-kata unik hasil *lemmatization* diurutkan dan ditampilkan jumlahnya (150 kata). Selanjutnya, data kelas atau *classes* (kategori intent) diurutkan secara alfabetis, ditampilkan, dan dihitung jumlah uniknya (26 kelas). Terakhir, kode menghitung jumlah pasangan dokumen berupa kombinasi antara *patterns* dan *tags*, menghasilkan total 688 dokumen. Semua langkah ini bertujuan untuk memastikan data SIAbot_UNIB telah dinormalisasi dan siap untuk pelatihan model.



Penjelasan :

Kode tersebut digunakan untuk melakukan proses tokenisasi pada data teks dalam kolom patterns dari dataset chatbot. Proses dimulai dengan membuat objek Tokenizer dari Keras dengan parameter num_words=2000, yang membatasi jumlah kata unik yang akan

diproses hingga 2000 kata yang paling sering muncul. Hasil akhirnya disimpan dalam variabel train, yang berisi representasi numerik dari teks untuk digunakan sebagai input dalam model pembelajaran mesin atau deep learning. Proses ini bertujuan untuk mengubah teks menjadi bentuk yang dapat dipahami oleh algoritma komputasi.

```
# Melakukan proses padding pada data
x_train = pad_sequences(train)
# Menampilkan hasil padding
print(x_train)
```

Penjelasan :

Kode diatas digunakan untuk melakukan proses padding pada data menggunakan fungsi *pad_sequences* dan menampilkan hasilnya. *Padding* adalah teknik yang digunakan dalam pemrosesan data sekuensial untuk membuat semua data memiliki panjang yang sama.

```
[ ] # Melakukan konversi data label tags dengan encoding
    le = LabelEncoder()
    y_train = le.fit_transform(data['tags'])
    print(y_train)
```

Penjelasan :

Kode diatas digunakan untuk melakukan proses *encoding* pada label data menggunakan *LabelEncoder* dari pustaka *sklearn*. Fungsi ini digunakan untuk mengubah data label kategorikal menjadi representasi numerik agar dapat diproses oleh algoritma *deep learning.*

3.4 Pelatihan Model

```
[ ] # Membuat model (Modeling)
    i = Input(shape=(input_shape,))  # Layer input untuk menerima data dengan bentuk 'input_shape'
    x = Embedding(vocabulary+1, 10)(i)  # Layer embedding untuk merepresentasikan kata dalam bentuk vektor berdimensi 10
    x = LSTM(10, return_sequences=True, recurrent_dropout=0.2)(x)  # Layer LSTM dengan 10 unit, output berupa urutan, dropout recurrent 20%
    x = Flatten()(x)  # Layer flatten untuk meratakan output dari layer sebelumnya menjadi satu dimensi
    x = Dense(output_length, activation="softmax")(x)  # Layer dense sebagai output, menggunakan fungsi aktivasi softmax
    model = Model(i, x)  # Menyusun model dari layer input hingga output

    # Kompilasi model
    # Menggunakan 'sparse_categorical_crossentropy' sebagai fungsi loss, optimizer 'adam', dan metrik akurasi
    model.compile(loss="sparse_categorical_crossentropy", optimizer='adam', metrics=['accuracy'])
```

```
# Visualisasi Plot Arsitektur Model
# Membuat diagram arsitektur model dan menyimpannya sebagai file gambar 'model_plot.png'
# Parameter 'show_shapes=True' menampilkan bentuk (shape) setiap layer
# Parameter 'show_layer_names=True' menampilkan nama setiap layer dalam diagram

plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Penjelasan :

Kode diatas digunakan untuk proses pembangunan, kompilasi, dan visualisasi model berbasis *deep learning* menggunakan Keras. Pertama, model dibuat menggunakan beberapa lapisan. Lapisan Input menerima data dengan bentuk tertentu yang didefinisikan oleh input_shape. Data tersebut kemudian direpresentasikan sebagai vektor berdimensi 10

melalui lapisan Embedding. Setelah itu, data diteruskan ke lapisan LSTM dengan 10 unit, yang dilengkapi dengan dropout 20% untuk mencegah overfitting dan memastikan data diproses sebagai urutan. Lapisan Flatten kemudian digunakan untuk meratakan output dari LSTM menjadi satu dimensi sebelum diteruskan ke lapisan Dense yang menggunakan fungsi aktivasi softmax untuk menghasilkan output akhir. Model ini kemudian disusun menggunakan kelas Model dari Keras. Selanjutnya, Model ini dikonfigurasi menggunakan fungsi loss sparse_categorical_crossentropy, optimizer adam, dan metrik akurasi untuk mengukur performa. Setalah itu membuat diagram arsitektur model menggunakan fungsi plot_model. Diagram ini disimpan dalam file bernama model_plot.png, dengan menampilkan bentuk dan nama setiap lapisan di dalam diagram untuk memberikan pemahaman visual tentang struktur model yang telah dibangun.

```
# Menampilkan parameter pada model LSTM
model.summary()
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 9)	0
embedding (Embedding)	(None, 9, 10)	1,500
lstm (LSTM)	(None, 9, 10)	840
flatten (Flatten)	(None, 90)	0
dense (Dense)	(None, 26)	2,366

```
Total params: 4,706 (18.38 KB)
Trainable params: 4,706 (18.38 KB)
Non-trainable params: 0 (0.00 B)
```

```
[ ] # Training the model (Melatih model data sampai 450 kali)
    train = model.fit(x_train, y_train, epochs=450)
```

Penjelasan :

Kode dan gambar di atas menampilkan arsitektur model LSTM dan proses pelatihannya. Pada bagian pertama, model.summary() menampilkan rincian lapisan model, termasuk jenis lapisan, dimensi output, dan jumlah parameter yang dapat dilatih. Model memiliki empat lapisan utama: Input Layer, Embedding Layer untuk mengonversi kata menjadi vektor berdimensi 10, LSTM Layer dengan 10 unit memori yang memproses data sekuensial, dan Dense Layer dengan 66 unit untuk klasifikasi multi-kelas. Total parameter dalam model adalah 11.646, semuanya dapat dilatih. Pada bagian kedua, proses pelatihan model dilakukan dengan menggunakan data x_train dan y_train selama 450 epoch.

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 1.0000 - loss: 0.0305
Epoch 447/450
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 1.0000 - loss: 0.0353
Epoch 448/450
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 1.0000 - loss: 0.0314
Epoch 449/450
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 1.0000 - loss: 0.0293
Epoch 450/450
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 1.0000 - loss: 0.0321
```

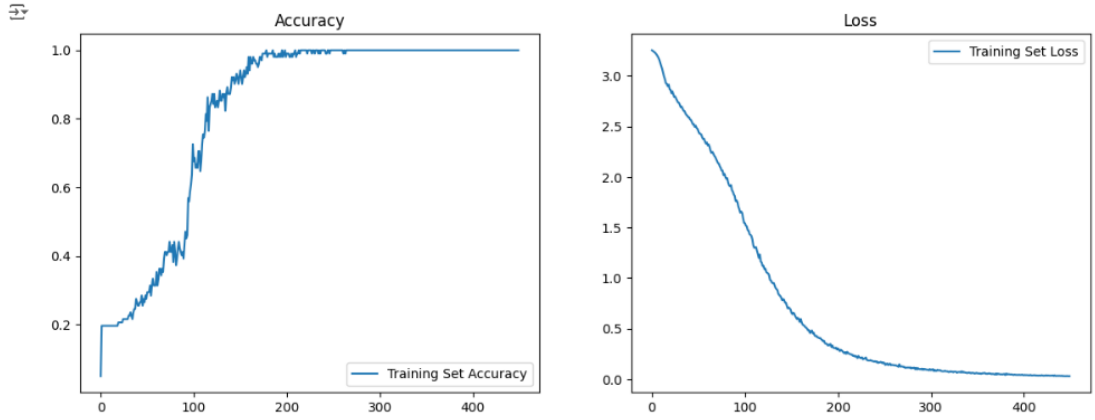
Penjelasan :

Gambar diatas menampilkan hasil pelatihan model pada beberapa epoch terakhir dari total 450 epoch. Model menunjukkan akurasi 1.0000 (100%) pada setiap epoch, yang menunjukkan bahwa model telah berhasil mengklasifikasikan seluruh data pelatihan dengan benar. Nilai loss juga terus menurun hingga mencapai angka yang sangat kecil, berkisar antara 0.0355 hingga 0.0293, yang menunjukkan bahwa model mampu memprediksi output dengan tingkat kesalahan yang sangat rendah.

```python
# Plotting model Accuracy and Loss (Visualisasi Plot Hasil Akurasi dan Loss)
# Plot Akurasi

plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(train.history['accuracy'], label='Training Set Accuracy')
plt.legend(loc='lower right')
plt.title('Accuracy')

# Plot Loss
plt.subplot(1, 2, 2)
plt.plot(train.history['loss'], label='Training Set Loss')
plt.legend(loc='upper right')
plt.title('Loss')
plt.show()
```



Penjelasan :

Gambar di atas menunjukkan dua grafik yang memvisualisasikan kinerja model selama proses pelatihan selama 450 epoch. Pola ini menunjukkan bahwa model mengalami konvergensi yang baik tanpa tanda-tanda overfitting, karena loss terus menurun dan akurasi meningkat stabil tanpa fluktuasi signifikan di akhir pelatihan.

```python
# Membuat Input Chat
from gtts import gTTS
import IPython.display as ipd

while True:
  texts_p = []
  prediction_input = input('🧑 Anda : ')

  # Menghapus punktuasi atau tanda baca dan konversi ke huruf kecil
  prediction_input = [letters.lower() for letters in prediction_input if letters not in string.punctuation]
  prediction_input = ''.join(prediction_input)
  texts_p.append(prediction_input)

  # Melakukan Tokenisasi dan Padding pada data teks
  prediction_input = tokenizer.texts_to_sequences(texts_p)
  # Konversi data teks menjadi array
  prediction_input = np.array(prediction_input).reshape(-1)
  prediction_input = pad_sequences([prediction_input],input_shape)

  # Mendapatkan hasil prediksi keluaran pada model
  output = model.predict(prediction_input)
  output = output.argmax()

  # Menemukan respon sesuai data tag dan memainkan suara bot
  response_tag = le.inverse_transform([output])[0]
  # Bot akan melakukan random jawaban percakapan dari hasil pertanyaan
  print("😈 SIAbot_UNIB : ", random.choice(responses[response_tag]))
  # Tambahkan suara bot dengan Google Text to Speech
  tts = gTTS(random.choice(responses[response_tag]), lang='id')
  # Simpan model voice bot ke dalam Google Drive dengan format .wav
  tts.save('SIAbot_UNIB.wav')
  # Atur waktu jeda sampai 5 detik
  time.sleep(0.05)
  # Ambil file model yang telah disimpan sebelumnya
  ipd.display(ipd.Audio('/content/SIAbot_UNIB.wav', autoplay=False))
  print("="*60 + "\n")
  # Tambahkan respon 'goodbye' agar bot bisa berhenti melakukan percakapan
  if response_tag == "goodbye":
    break
```

```
🧑 Anda : hallo
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 375ms/step
😈 SIAbot_UNIB :  Hi, Ada yang bisa Skripsi Informatika Assistant (SIA) UNIB bantu?
▶ 0:00 / 0:06 ——————

============================================================

🧑 Anda : Bagaimana alur pengambilan skripsi?
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
😈 SIAbot_UNIB :
                Berikut adalah alur dalam pengambilan mata kuliah skripsi:

                1. Mengambil Mata Kuliah Skripsi

                2. Membuat Proposal Skripsi

                3. Seminar Proposal Skripsi

                4. Penelitian dan Penulisan Skripsi

                5. Seminar Hasil Skripsi

                6. Sidang Skripsi

▶ 0:00 / 0:23 ——————

============================================================
```

Penjelasan :

Gambar diatas menampilkan kode pengujian dan output dari pengujian model. Proses dimulai dengan menerima input teks dari pengguna yang kemudian diproses dengan menghapus tanda baca dan mengubah teks menjadi huruf kecil. Input ini ditokenisasi dan diproses lebih lanjut menggunakan padding agar sesuai dengan format yang diperlukan oleh model. Model prediksi digunakan untuk menentukan keluaran berupa tag respons

berdasarkan input pengguna. SIAbot_UNIB ini juga mendukung penghasilan respons suara. Setelah respons teks dipilih secara acak dari daftar jawaban yang relevan, respons tersebut akan dikonversi menjadi suara menggunakan Google Text-to-Speech (gTTS), lalu disimpan dalam format audio (.wav). File audio ini diputar menggunakan IPython.display agar pengguna dapat mendengarkan respons bot.

4. Kesimpulan

Proyek SIAbot_UNIB ini berhasil menghasilkan sistem chatbot yang dapat memberikan respons otomatis terhadap pertanyaan mahasiswa informatika Universitas Bengkulu terkait panduan skripsi yang relevan. Hasil evaluasi model menunjukkan bahwa model memiliki performa yang baik dengan akurasi tinggi hampir mencapai 100% pada data pelatihan. Pada pengujian model juga terbukti menghasilkan respon yang tepat sesuai dengan pertanyaan pengguna dilengkapi dengan fitur suara yang akurat sesuai dengan jawaban teks yang dihasilkan.