# LAPORAN UAS DEEP LEARNING

# CHATBOT KONSELING DAN DUKUNGAN MAHASISWA SEMESTER AKHIR "SAHABATAKHIRUNIB"


**Disusun Oleh:**

1. **Fachrurazi** **(G1A021016)**
2. **Fadilah Syakirah** **(G1A021022)**
3. **Adelia Ayu Lestari** **(G1A021066)**

**Dosen Mata Kuliah:**

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

**PROGRAM STUDI INFORMATIKA**
**FAKULTAS TEKNIK**
**UNIVERSITAS BENGKULU**

**2024**

## 1. Business Understanding

Chatbot telah menjadi alternatif untuk membantu mahasiswa akhir mengatasi tekanan mental, seperti yang dialami selama menyelesaikan skripsi. Teknologi ini dianggap efektif dalam membantu mengurangi masalah kesehatan mental, termasuk depresi (Huberta & Wijaya, 2023). Saat ini, berbagai teknologi, seperti aplikasi seluler, virtual reality, dan chatbot, menawarkan cara baru untuk menangani kesehatan mental (Ali et al., n.d.). Penelitian awal menunjukkan bahwa chatbot dapat membantu mengurangi gejala gangguan mental. Studi yang meneliti efektivitas chatbot dalam mendukung mahasiswa selama proses skripsi menemukan bahwa chatbot dapat membantu mahasiswa yang mengalami tekanan mental dengan menyediakan intervensi digital untuk mengelola stres. Teknologi ini dianggap sebagai solusi inovatif untuk memberikan dukungan psikologis jarak jauh. Pengguna chatbot untuk dukungan mental meningkat hingga 60% dalam dua minggu karena kecemasan yang dipicu oleh situasi pandemi (Maskur, 2016). Salah satu pengguna chatbot, Mr. Johnson, menyatakan bahwa ia menggunakan chatbot sebagai sarana untuk mendapatkan dukungan emosional. Mahasiswa semester akhir sering menghadapi berbagai tantangan, seperti tekanan menyelesaikan tugas akhir, kecemasan terhadap masa depan, serta kesulitan dalam mengatur waktu dan menjaga motivasi (Buatan et al., n.d.). Universitas Bengkulu (Unib) dapat menyediakan solusi berupa chatbot berbasis AI bernama "SahabatAkhirUnib". Chatbot ini dirancang untuk memberikan dukungan emosional, akademik, dan motivasional kepada mahasiswa semester akhir, sehingga mereka dapat menghadapi masa studi akhir mereka dengan lebih percaya diri dan terorganisir. Proyek ini bertujuan untuk menciptakan chatbot cerdas yang dapat menjadi teman belajar, konselor, dan sumber informasi bagi mahasiswa. Dengan fitur-fitur seperti bimbingan , dukungan emosional, dan akses ke informasi terkini, chatbot ini diharapkan dapat membantu mahasiswa mengatasi berbagai tantangan yang mereka hadapi selama masa studi, sehingga mereka dapat meraih prestasi akademik yang optimal (Maskur, 2016).

Proyek ini dibuat untuk membantu mahasiswa semester akhir dalam memenuhi kebutuhan mahasiswa. Proyek ini menyediakan fitur konseling

virtual untuk mendukung kesehatan mental, selain itu proyek ini juga menyediakan panduan untuk menyelesaikan tugas akhir, memberi informasi terkait administrasi kampus, dan juga membantu menjadi pengingat untuk tugas ataupun sebagai motivasi harian. Dalam era digital saat ini, kecerdasan buatan (AI) telah menjadi bagian integral dari berbagai aspek kehidupan, termasuk pendidikan. AI tidak hanya berfungsi sebagai alat untuk meningkatkan efisiensi, tetapi juga memiliki potensi untuk mempengaruhi motivasi dan kinerja akademik mahasiswa zi(Nur Hakim et al., 2023). Teknologi *Deep Learning*, yang menjadi inti dari chatbot ini, memungkinkan sistem untuk memahami konteks percakapan dengan lebih baik, memberikan jawaban yang lebih relevan, dan mengenali kebutuhan emosional pengguna secara lebih mendalam. Dengan memanfaatkan kecerdasan buatan yang canggih, chatbot ini dirancang untuk memberikan interaksi yang lebih manusiawi dan membantu mahasiswa semester akhir mengatasi berbagai kesulitan yang mereka hadapi, baik dari segi akademik maupun emosional.

**2. Data Understanding**

Dataset pada pembuatan *chatbot* “SahabatAkhirUnib” disusun dalam format JSON yang dibuat secara manual berdasarkan hasil riset dan analisis yang dilakukan terhadap pola konsultasi mahasiswa semester akhir Universitas Bengkulu. Isi dataset mencakup berbagai topik yang sering menjadi pembahasan mahasiswa semester akhir, seperti tantangan dan keluhan yang dialami selama mengerjakan skripsi, dukungan kesehatan mental, tips manajemen waktu agar sesuai dengan target yang telah ditentukan, dan perencanaan karir setelah lulus dari kampus. Perancangan dataset ini agar *chatbot* dapat memberikan respons yang relevan dan dapat membantu mahasiswa semester akhir.

Struktur dataset terdiri dari *intents*, yang merupakan kumpulan kategori yang akan mewakili setiap pertanyaan yang akan diajukan oleh mahasiswa seperti, kebingungan memulai skripsi dan tips mempersiapkan diri untuk dunia kerja. Setiap *intents* dilengkapi dengan *tags* yang berfungsi untuk mengelompokkan pola pertanyaan (*patterns*) yang akan menggambarkan

bagaimana mahasiswa memberikan pertanyaan lalu akan diberikan jawaban (*responses*) yang relevan sesuai dengan pertanyaan yang diajukan mahasiswa.

| | patterns | tags |
|---|---|---|
| 30 | Aku stres | stress |
| 31 | Saya merasa stres | stress |
| 32 | Aku lagi banyak tekanan | stress |
| 33 | Saya sedang stres | stress |
| 34 | Tekanan di kampus membuatku stres | stress |
| 35 | Aku merasa gagal | motivasi |
| 36 | Aku butuh motivasi | motivasi |
| 37 | Aku merasa tidak cukup baik | motivasi |
| 38 | Aku ingin semangat | motivasi |
| 39 | Motivasi dong | motivasi |
| 40 | Bagaimana cara mengakses konseling di UNIB? | konseling |

Gambar 2. Dataset

## 3. Praprocessing Data

### 3.1 Mengapus Tanda Baca

```
# Menghilangkan tanda baca (punctuation) dari kolom 'patterns' pada DataFrame 'data'
# Untuk setiap pola (pattern), mengiterasi setiap karakter dan menghapus karakter yang termasuk tanda baca
data['patterns'] = data['patterns'].apply(lambda wrd: [ltrs.lower() for ltrs in wrd if ltrs not in string.punctuation])

# Setelah tanda baca dihapus, menggabungkan kembali daftar karakter menjadi string
# Misalnya: ['h', 'e', 'l', 'l', 'o'] -> "hello"
data['patterns'] = data['patterns'].apply(lambda wrd: ''.join(wrd))

# Menampilkan baris ke-30 hingga ke-50 dari DataFrame untuk memeriksa hasil transformasi
data[30:50]
```

Gambar 3.1 Kode Menghapus Tanda Baca

Kode diatas adalah tahapan untuk melakukan proses pembersihan data dengan menghapus tanda baca dari kolom *patterns* dalam *DataFrame* data. Setiap karakter yang berada di dalam teks akan diperiksa, jika bukan tanda baca maka karakter akan disimpan. Selanjutnya terdapat kode untuk mengubah semua huruf menjadi huruf kecil agar data menjadi sama. Jika telah dilakukan pembersihan data dan huruf diubah menjadi kecil, selanjutnya akan menggabungkan kembali karakter yang telah dilakukan penghapusan tanda baca menjadi teks yang utuh kembali, sehingga akan menjadi teks bersih yang telah bebas dari tanda baca dan huruf besar.

### 3.2 Lemmatization

```
# Melakukan lemmatization pada setiap pola (pattern) di kolom 'patterns' pada DataFrame 'data'
# Setiap kata dalam pattern dilemmatize menggunakan fungsi 'lemmatizer.lemmatize()'
# Hasilnya adalah daftar kata setelah proses lemmatization
data['patterns'] = data['patterns'].apply(lambda wrd: [lemmatizer.lemmatize(wrd)])

# Setelah lemmatization, daftar kata digabungkan kembali menjadi string utuh
# Misalnya: ['bermain'] -> "bermain"
data['patterns'] = data['patterns'].apply(lambda wrd: ''.join(wrd))

# Menampilkan baris ke-30 hingga ke-50 dari DataFrame untuk memeriksa hasil transformasi
data[30:50]
```

Gambar 3.2 Kode *Lemmatization*

Kode ini adalah tahapan untuk melakukan *lemmatization* yang merupakan proses mengubah kata ke dalam bentuk dasarnya berdasarkan maka kata tersebut. Setiap kata yang telah dilakukan proses *lemmatization* akan disusun kembali menjadi sebuah teks utuh menggunakan fungsi *join*.

### 3.3 Menyortir dan Menyusun Kelas Tag

```
[ ]  # Menyortir data kelas tag
     classes = sorted(list(set(classes)))
     print (len(classes), "classes", classes)
```

Gambar 3.3 Kode Menyortir dan Menyusun Kelas Tag

Kode ini adalah tahapan untuk menyortir data yang terdapat pada variabel *classes* yang berisi daftar tag yang digunakan dalam dataset. Pada tahap ini dilakukan proses penghilangan duplikasi pada daftar *classes* dengan mengubahnya menjadi sebuah *seat*, lalu hasil yang telah bersih dari duplikasi akan diubah kembali menjadi daftar (*list*) dan disortir secara alfabetis.

### 3.4 Menghitung Jumlah Keseluruhan Dokumen dalam Dataset

```
     # Mencetak jumlah keseluruhan data teks
     # documents = kombinasi antara patterns and intents
     print (len(documents), "documents")
```

Gambar 3.4 Kode Menghitung Jumlah Keseluruhan Dokumen dalam Dataset

Kode ini adalah tahapan untuk menghitung dan mencetak jumlah total data teks yang terdapat pada variabel *documents*, yang merupakan gabungan antara *patterns* dan *intents*.

### 3.5 Tokenizer

```
[ ]
     # Membuat instance tokenizer dari TensorFlow Keras
     # Parameter 'num_words=2000' menunjukkan bahwa hanya 2000 kata yang paling sering muncul yang akan dipertimbangkan
     tokenizer = Tokenizer(num_words=2000)

     # Melatih tokenizer pada kolom 'patterns' dalam DataFrame 'data'
     # Proses ini menghasilkan indeks untuk setiap kata berdasarkan frekuensinya dalam dataset
     tokenizer.fit_on_texts(data['patterns'])

     # Mengubah teks dalam kolom 'patterns' menjadi urutan numerik berdasarkan indeks kata
     # Misalnya: Jika "main" memiliki indeks 1 dan "belajar" memiliki indeks 2, maka teks "main belajar" -> [1, 2]
     train = tokenizer.texts_to_sequences(data['patterns'])

     # Menampilkan hasil tokenisasi berupa daftar urutan numerik untuk setiap pola dalam dataset
     train
```

Gambar 3.5 Kode *Tokenizer*

Kode ini adalah tahapan untuk melakukan tokenisasi. Tokenisasi adalah proses mengubah kata-kata dalam teks menjadi angka, sehingga model akan dapat memprosesnya. Pada tahapan ini proses tokenisasi menggunakan *TensorFlow* Keras. *Tokenizer* yang dilatih pada data teks akan menghasilkan indeks numerik untuk setiap kata dan teks tersebut akan diubah menjadi urutan angka berdasarkan indeks kata dalam teks.

### 3.6 Padding

```
[ ] # Menerapkan padding pada data tokenisasi untuk memastikan bahwa semua urutan memiliki panjang yang sama
    # 'pad_sequences' menambahkan padding (biasanya nol) ke urutan yang lebih pendek agar sesuai dengan panjang urutan terpanjang
    X_train = pad_sequences(train)

    # Menampilkan hasil setelah padding diterapkan
    # Setiap urutan sekarang memiliki panjang yang sama, sesuai dengan panjang maksimum dalam dataset
    print(X_train)  # Padding sequences
```

Gambar 3.6 Kode *Padding*

Kode ini adalah tahapan untuk menerapkan *padding* pada data tokenisasi. Proses *padding* adalah proses untuk membuat setiap kalimat pada teks agar memiliki panjang yang sama.

### 3.7 Mengubah Label Kategori Menjadi Numerik

```
# Membuat instance dari LabelEncoder untuk mengubah data kategori menjadi angka
le = LabelEncoder()

# Mengubah kolom 'tags' menjadi nilai numerik
# LabelEncoder akan memberikan nilai unik (numerik) untuk setiap kategori di 'tags'
# Misalnya: "greeting" -> 0, "goodbye" -> 1, dll.
y_train = le.fit_transform(data['tags'])

# Menampilkan hasil encoding output
# Output ini adalah representasi numerik dari label yang sebelumnya berupa kategori teks
print(y_train)
```

Gambar 3.7 Kode Mengubah Label Kategori Menjadi Numerik

Kode ini adalah tahapan untuk mengubah data kategori menjadi angka dengan setiap kategori pada kolom tags akan diberi angka unik, sehingga hasilnya adalah representasi numerik dari label yang sebelumnya adalah teks.

### 3.8 Menentukan Panjang Input untuk Model

```
# Mendapatkan panjang input dari data pelatihan setelah proses padding
# 'X_train.shape' mengembalikan bentuk array NumPy dalam format (jumlah_sampel, panjang_urutan)
# '[1]' merujuk pada dimensi kedua, yaitu panjang urutan input
input_shape = X_train.shape[1]

# Menampilkan panjang input
# Panjang ini akan digunakan sebagai dimensi input dalam model deep learning
print(input_shape)
```

Gambar 3.8 Kode Menentukan Panjang Input untuk Model

Kode ini adalah tahapan untuk mendapatkan panjang *input* data pelatihan setelah dilakukan proses *padding*. Setelah memperoleh panjang urutan setiap *input* dalam dataset, maka akan memberikan nilai yang menunjukkan berapa banyak elemen yang ada di setiap urutan setelah *padding*.

### 3.9 Menghitung Jumlah Kata Unik

```
[ ] # Mendapatkan jumlah kata unik dalam tokenizer
    # 'tokenizer.word_index' adalah dictionary yang berisi kata-kata unik sebagai kunci dan indeksnya sebagai nilai
    # Fungsi 'len()' digunakan untuk menghitung jumlah kata unik dalam dictionary tersebut
    vocabulary = len(tokenizer.word_index)

    # Menampilkan jumlah kata unik dalam dataset
    print("number of unique words : ", vocabulary)
```

Gambar 3.9 Kode Menghitung Jumlah Kata Unik

Kode ini adalah tahapan untuk menghitung jumlah kata unik yang terdapat dalam dataset setelah dilakukan tokenisasi. Dengan melakukan tahapan ini akan mengetahui jumlah kata unik untuk menentukan ukuran vocabulary model.

### 3.10 Menghitung Jumlah Kelas Unik

```
[ ] # Mendapatkan panjang output dari jumlah kelas unik dalam label encoder
    # 'le.classes_' adalah array yang berisi semua label unik yang dipelajari oleh LabelEncoder
    # 'le.classes_.shape[0]' memberikan jumlah total kelas unik
    output_length = le.classes_.shape[0]

    # Menampilkan panjang output (jumlah kelas unik)
    print("output length: ", output_length)
```

Gambar 3.10 Kode Menghitung Jumlah Kelas Unik

Kode ini adalah tahapan untuk menghitung jumlah kelas unik yang berada dalam data label setelah dilakukan proses encoding menggunakan LabelEncoder. Dilakukan proses ini penting untuk menentukan jumlah unit pada output layer model.

## 4. Modeling

```
# Membuat arsitektur model menggunakan TensorFlow Keras Functional API

# Layer Input: Mendefinisikan input dengan bentuk sesuai panjang urutan (input_shape)
# 'input_shape' adalah panjang dari setiap urutan setelah padding
i = Input(shape=(input_shape,))

# Layer Embedding: Mengubah indeks kata menjadi representasi vektor berdimensi 20
# Dimensi kosakata adalah 'vocabulary + 1' untuk memasukkan indeks nol (padding)
x = Embedding(vocabulary + 1, 20)(i)

# Layer LSTM: Menambahkan Long Short-Term Memory layer untuk mempelajari urutan kata
# LSTM memiliki 20 unit, dan 'return_sequences=True' untuk mengembalikan urutan keluaran
x = LSTM(20, return_sequences=True)(x)

# Layer Flatten: Mengubah keluaran LSTM (3D tensor) menjadi vektor 1D untuk layer Dense
x = Flatten()(x)

# Layer Dense: Menambahkan layer fully connected dengan jumlah neuron sesuai jumlah kelas
# 'output_length' adalah jumlah kelas unik, menggunakan aktivasi softmax untuk probabilitas
x = Dense(output_length, activation="softmax")(x)

# Membuat model dengan TensorFlow Keras Functional API
# Input adalah 'i' dan output adalah 'x'
model = Model(i, x)

# Kompilasi model: Menentukan fungsi loss, optimizer, dan metrik evaluasi
# - Loss: 'sparse_categorical_crossentropy' cocok untuk target yang berupa integer label
# - Optimizer: 'adam' adalah optimizer adaptif yang bekerja dengan baik untuk berbagai kasus
# - Metrics: 'accuracy' digunakan untuk mengevaluasi performa model
model.compile(loss="sparse_categorical_crossentropy", optimizer='adam', metrics=['accuracy'])
```

Gambar 4. Modelling

Model klasifikasi menggunakan *TensorFlow* Keras *Functional* API. Model dimulai dengan layer input yang menerima data berurutan, kemudian *layer embedding* mengubah indeks kata menjadi vektor berdimensi 20. Layer LSTM digunakan untuk mempelajari pola urutan data, dan hasilnya diratakan menjadi vektor 1D dengan *layer Flatten*. Selanjutnya, *layer dense* dengan fungsi aktivasi *softmax* menghasilkan prediksi untuk sejumlah kelas tertentu. Model ini dikompilasi menggunakan *loss sparse_categorical_crossentropy*, *optimizer adam*, dan *metrik accuracy*, sehingga siap dilatih untuk tugas klasifikasi.

## 4.1 Menampilkan Parameter Model

```
# Menampilkan Parameter Model
model.summary()
```

**Model: "functional"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer (InputLayer) | (None, 17) | 0 |
| embedding (Embedding) | (None, 17, 20) | 7,220 |
| lstm (LSTM) | (None, 17, 20) | 3,280 |
| flatten (Flatten) | (None, 340) | 0 |
| dense (Dense) | (None, 93) | 31,713 |

**Total params:** 42,213 (164.89 KB)
**Trainable params:** 42,213 (164.89 KB)
**Non-trainable params:** 0 (0.00 B)

Gambar 4.1 Parameter Model

Model yang ditampilkan memiliki lima lapisan utama dengan total 42.213 parameter, semuanya dapat dilatih. Lapisan pertama adalah *InputLayer*, yang hanya mendefinisikan bentuk input tanpa parameter. Lapisan kedua adalah *Embedding*, yang mengubah data menjadi vektor berdimensi 20, dengan 7.220 parameter. Selanjutnya, lapisan LSTM memproses data berurutan dan memiliki 3.280 parameter. Lapisan *Flatten* mengubah data menjadi vektor satu dimensi tanpa menambah parameter. Terakhir, lapisan *Dense* menghasilkan 93 output dengan 31.713 parameter. Model ini cocok untuk tugas yang melibatkan data berurutan, seperti teks.

## 4.2 Melatih Model

```
# Melatih model dengan data pelatihan (X_train dan y_train)
# Parameter:
# - 'epochs=305': Model akan dilatih selama 300 epoch (siklus penuh melalui dataset)
# - 'X_train': Data input berupa urutan kata (tokenized dan padded)
# - 'y_train': Label output yang telah di-encode menjadi angka
train = model.fit(X_train, y_train, epochs=300)
```

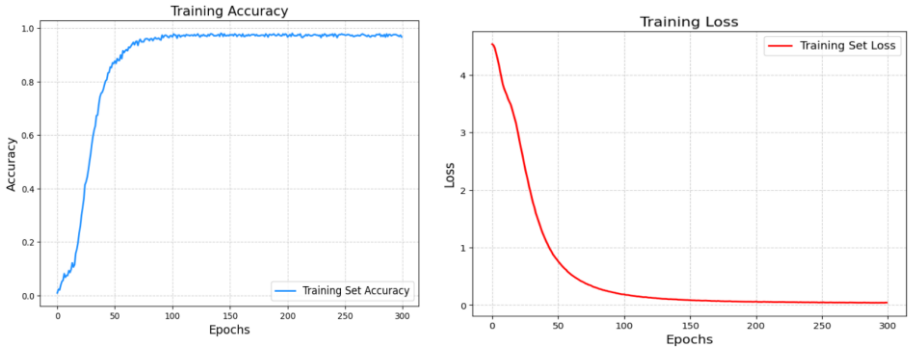
```
Epoch 272/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9779 - loss: 0.0348
Epoch 273/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9720 - loss: 0.0415
Epoch 274/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.9670 - loss: 0.0469
Epoch 275/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step - accuracy: 0.9758 - loss: 0.0409
Epoch 276/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9662 - loss: 0.0486
Epoch 277/300
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9715 - loss: 0.0420
Epoch 278/300
```

Gambar 4.2 Melatih Model dengan Data Latih

Gambar diatas merupakan proses pelatihan model selama 300 epoch menggunakan data pelatihan *X_train* dan *y_train.* Setiap epoch

merepresentasikan satu siklus penuh melalui dataset. Model mencatat metrik *accuracy* (tingkat keakuratan) dan *loss* (nilai kesalahan) di setiap epoch. Pada contoh yang ditampilkan, *accuracy* model berada di kisaran 0.9662 hingga 0.9783, menunjukkan performa yang cukup baik. Nilai *loss* berkisar antara 0.0348 hingga 0.0486, yang berarti model terus belajar untuk meminimalkan kesalahan. Proses pelatihan ini bertujuan agar model dapat mengenali pola dalam data dengan lebih baik untuk digunakan pada data baru.

**5. Evaluasi**




Gambar 5 Grafik Akurasi dan *Loss*

Pada kedua gambar diatas merupakan hasil visualisasi grafik *accuracy* dan *loss*.

1. Training accuracy

   Grafik *accurasy* ini menunjukkan bahwa akurasi model meningkat dengan bertambahnya jumlah *epoch*. Pada awal pelatihan (sekitar 0-50 *epoch*), akurasi meningkat, yang menunjukkan model dengan cepat belajar dari data pelatihan. Setelah sekitar 100 *epoch*, peningkatan akurasi mulai melambat dan mendekati 1, yang berarti model hampir sempurna dalam memprediksi data pelatihan.

2. Training Loss

   Grafik *Loss* ini menunjukkan penurunan *loss* selama pelatihan. *Loss* menurun dengan sangat cepat, menunjukkan bahwa model dengan cepat belajar meminimalkan kesalahan. Setelah sekitar 100 *epoch*, penurunan *loss* melambat dan mendekati 0, menunjukkan bahwa model hampir tidak melakukan kesalahan pada data pelatihan.

## 6. Testing

```
# Menentukan batas probabilitas
batas_probabilitas = 0.8

# Membuat Input Chat
while True:
  texts_p = []
  prediction_input = input(' Kamu : ')

  # Preprocessing teks
  prediction_input = [letters.lower() for letters in prediction_input if letters not in string.punctuation]
  prediction_input = ''.join(prediction_input)
  prediction_input = lemmatizer.lemmatize(prediction_input)
  texts_p.append(prediction_input)

  # Tokenisasi dan Padding
  prediction_input = tokenizer.texts_to_sequences(texts_p)
  prediction_input = np.array(prediction_input).reshape(-1)
  prediction_input = pad_sequences([prediction_input], input_shape)

  # Mendapatkan hasil keluaran pada model
  output = model.predict(prediction_input)
  output_probabilitas = round(output.max(), 2)
  output = output.argmax()
```

```
 # Memeriksa probabilitas
 if output_probabilitas < batas_probabilitas:
   print(" SahabatAkhirUnib : Maaf, saya tidak mengerti pertanyaan anda.")
 else:
   # Menemukan respon sesuai data tag dan memainkan voice bot
   response_tag = le.inverse_transform([output])[0]
   print(" Unibbot : ", random.choice(responses[response_tag]))

 time.sleep(0.08)
 print("="*60 + "\n")

 # Tambahkan respon 'goodbye' agar bot bisa berhenti
 if response_tag == "goodbye":
   break
```

```
 Kamu : halo
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
 Unibbot :  Hei sobat! Ini SahabatAkhirUnib, siap mendukungmu. Cerita apa nih yang ingin kamu sampaikan?
============================================================

 Kamu : dadah
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
 Unibbot :  Sampai jumpa lagi! Semangat terus ya!
```

Gambar 6. Testing

Hasil pengujian *chatbot* SahabatAkhirUnib, menunjukkan bahwa sistem dapat merespons *input* pengguna dengan cukup baik sesuai dengan konteks. Ketika pengguna memberikan *input* seperti "halo," *chatbot* merespons dengan sapaan ramah, yaitu "Hei sobat! Ini SahabatAkhirUnib, siap mendukungmu. Cerita apa nih yang ingin kamu sampaikan?" chatbot dapat memahami *input* tersebut sebagai sapaan. Selanjutnya, ketika pengguna mengetik "dadah," chatbot memberikan respons yang sesuai untuk perpisahan, yaitu "Sampai jumpa lagi! Semangat terus ya!" ini menunjukkan *chatbot* dapat membaca tag "goodbye" dan mengakhiri sesi obrolan dengan ucapan penutup yang sopan.

Secara keseluruhan, *chatbot* menunjukkan kemampuan yang baik dalam memahami konteks dan memberikan respons yang sesuai dengan kebutuhan pengguna.

**7. Analisa bagaimana model dapat dikatakan sebagai deep learning dan bukan shallow learn?**

Berdasarkan model yang dibuat untuk *chatbot* SahabatAkhirUnib merupakan model *deep learning* dan bukan *shallow learn*. Alasan model *chatbot* SahabatAkhirUnib adalah model *deep learning* karena menggunakan arsitektur multi-lapisan yang dirancang untuk memproses data kompleks. Terdapat layer LSTM (*Long Short-Term Memory*) dan *Embedding* yang merupakan ciri khas dari *deep learning*. Penggunaan LSTM digunakan untuk menangani data sekuensial seperti teks, sehingga akan membuat model dapat memahami hubungan antar kata, sedangkan penggunana *embedding* digunakan untuk mengubah kata-kata menjadi representasi vektor berdimensi tinggi. Model pada *chatbot* SahabatAkhirUnib dibangun menggunakan *framework TensorFlow* Keras yang dapat mendukung arsitektur jaringan saraf dengan lapisan-lapisan seperti *Dense, GlobalMaxPool1D,* dan lainnya. Sedangkan *shallow learn* adalah metode yaang hanya memiliki satu atau dua lapisan yang hanya dapat menangkap pola linear atau hubungan yang lebih sederhana dalam data.

**8. Kesimpulan**

Dari pengujian model LTSM dalam pembangunan ChatBot Konseling dan Dukungan mahasiswa semester akhir Universitas Bengkulu dapat disimpulkan bahwa model mendapat akurasi yang cukup bagus, yakni dengan rata-rata akurasi sebesar 0.9797. Kemudian, nilai loss pada model menunjukkan angka 0.0428 yang berarti nilai kesalahan atau error pada model terbilang sangat kecil. Model LSTM dengan total data sebesar 93 kelas dengan total pelatihan atau epoch sebanyak 300 ini sudah layak untuk diimplementasikan pada berbagai tugas yang melibatkan klasifikasi teks.

# DAFTAR PUSTAKA


Ali, R., Sari, Y. P., & Alawiyah, A. D. (n.d.). *Darmajaya Academic Chatbot Dengan Semantic Search*.

Buatan, K., Sumber, S., Mahasiswa, P., Menyelesaikan, D., Akhir, T., Hilal, S., Yusanto, Y., & Saylendra, A. (n.d.). *KECERDASAN BUATAN SEBAGAI SUMBER MOTIVASI: PERSPEKTIF MAHASISWA DALAM MENYELESAIKAN TUGAS AKHIR*. https://doi.org/10.56842/jp-ipa

Huberta, B., & Wijaya, A. B. (2023). PERANCANGAN CHATBOT WEBSITE PROGRAM STUDI INFORMATIKA MENGGUNAKAN FRAMEWORK CODEIGNITER. *Jurnal Informatika Dan Teknik Elektro Terapan*, *11*(3). https://doi.org/10.23960/jitet.v11i3.3225

Maskur. (2016). Perancangan CHATBOT Pusat Informasi Mahasiswa Menggunakan AIML Sebagai Virtual Assistant Berbasis Web. *KINETIK*, *1*(3).

Nur Hakim, I., Riset dan Inovasi Nasional Jl Gatot Subroto No, B., Bar, K., Mampang Prapatan, K., & Selatan, J. (2023). *Strategi Adaptif Kesehatan Mental Pekerja Pariwisata di Era Kecerdasan Buatan (AI)*. https://winco.cilacapkab.go.id