

**Laporan UAS *Deep Learning*: Chatbot Informasi Akademik Fakultas Teknik
Universitas Bengkulu "EngiBot"**



Disusun Oleh:

1. Seprina Dwi Cahyani (G1A021010)
2. Redo Hariyadi (G1A021034)
3. Afzal Alfaraz (G1A021098)

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
2024**

1. Latar Belakang

Portal akademik adalah sistem informasi yang berfungsi sebagai wadah informasi akademik dan sarana komunikasi antar civitas akademika. Sistem ini memungkinkan mahasiswa untuk mengakses berbagai informasi terkait studi, seperti jadwal kuliah, mata kuliah yang ditawarkan, dan kalender akademik. Selain itu, portal akademik juga memfasilitasi mahasiswa dalam mengelola rencana studi, seperti pengisian Kartu Rencana Studi (KRS), serta memberikan akses untuk melihat hasil studi secara real-time, termasuk nilai mata kuliah dan IPK (Indeks Prestasi Kumulatif).

Keberadaan portal akademik sangatlah penting dalam kehidupan mahasiswa, khususnya di era digital saat ini. Portal akademik ini memberikan kemudahan bagi mahasiswa dalam urusan administrasi khususnya dalam bidang akademik. Namun, dibalik manfaatnya, terdapat beberapa kendala dalam penggunaan portal akademik terutama bagi mahasiswa baru yang belum memahami penggunaan sistem informasi ini. Mahasiswa baru sering kali belum memiliki pengetahuan tentang cara kerja portal seperti cara mengisi KRS, mencari informasi jadwal, atau mengakses transkrip nilai. Kendala ini diperburuk oleh keterbatasan jumlah staf administrasi yang dapat membantu secara langsung, terutama di masa-masa sibuk seperti awal semester atau saat pengisian KRS.

Permasalahan serupa juga sering dihadapi oleh mahasiswa di Fakultas Teknik Universitas Bengkulu. Banyak mahasiswa yang belum memahami cara penggunaan portal akademik dan merasa kebingungan mengenai tempat atau orang yang dapat mereka ajak bertanya untuk mendapatkan informasi. Untuk mengatasi permasalahan tersebut, perkembangan teknologi yang semakin pesat menawarkan solusi inovatif, salah satunya melalui penggunaan *chatbot*. *Chatbot* adalah sebuah aplikasi berbasis kecerdasan buatan yang mampu berinteraksi dengan pengguna secara otomatis melalui antarmuka percakapan. Dengan kemampuan untuk merespons pertanyaan, memberikan informasi, serta menawarkan solusi, chatbot dapat menjadi asisten virtual yang efektif dalam menyediakan layanan akademik kepada mahasiswa.

Chatbot ini diharapkan dapat membantu mahasiswa Fakultas Teknik dalam mengakses dan memanfaatkan portal akademik secara lebih efektif. Dengan menyediakan panduan langkah demi langkah, menjawab pertanyaan umum, serta membantu menyelesaikan masalah teknis secara cepat, chatbot ini mampu menjadi solusi yang inovatif untuk mengatasi kendala yang dihadapi mahasiswa, terutama mereka yang masih baru dalam menggunakan sistem informasi akademik.

2. Pengembangan Model

2.1 Data Understanding

Data pada penelitian ini merupakan data primer yang diambil langsung dari laman website <https://pak.unib.ac.id/>. Data yang diambil merupakan data yang berhubungan dengan informasi akademik, seperti jadwal kuliah, mata kuliah yang ditawarkan, kalender akademik, serta data terkait pengisian Kartu Rencana Studi (KRS) dan transkrip nilai mahasiswa. Data yang dikumpulkan akan dianalisis untuk memahami pola-pola tertentu yang dapat memberikan wawasan terkait kebutuhan dan kendala yang dihadapi mahasiswa, terutama yang berhubungan dengan interaksi mereka dengan sistem portal akademik. Setelah melalui proses analisis, data akan disusun dalam format JSON untuk digunakan dalam pengembangan model. Setiap pola dalam JSON akan dikelompokkan berdasarkan kategori atau tag tertentu, yang mewakili jenis informasi yang diberikan. Format JSON yang digunakan akan mencakup elemen-elemen seperti tags untuk mengelompokkan informasi, *patterns* yang berisi data terkait portal akademik, dan responses yang berisi informasi atau solusi yang relevan untuk setiap pola yang ditemukan.

	patterns	tags
100	sampai kapan bisa isi krs	krs_deadline
101	krs terakhir kapan	krs_deadline
102	deadline input krs	krs_deadline
103	apakah krs bisa diubah	krs_edit
104	bagaimana cara mengubah krs	krs_edit
105	bisa nggak ganti mata kuliah di krs	krs_edit
106	cara revisi krs	krs_edit
107	bagaimana cara mendapat persetujuan krs	krs_approval
108	krs harus disetujui siapa	krs_approval
109	krs belum disetujui dosen pembimbing	krs_approval

Gambar 1 Dataset

2.2 Data Preparation

2.2.1 Data Cleaning

```
def clean_text(text):  
    # Menghapus tanda baca dan mengubah huruf menjadi lowercase  
    cleaned_text = [char.lower() for char in text if char not in string.punctuation]  
    # Menggabungkan kembali menjadi satu string  
    return ''.join(cleaned_text)  
  
# Terapkan fungsi clean_text ke kolom 'patterns'  
data['patterns'] = data['patterns'].apply(clean_text)  
  
# Menampilkan data dari baris 1 hingga 10  
data[1:10]
```

Gambar 2 Data Cleaning

Tahapan pertama pada tahap data preparation adalah pembersihan data. Pembersihan data ini dilakukan dengan menghapus tanda baca dan mengubah teks menjadi huruf kecil (*lowercase*). Proses ini dilakukan untuk memastikan bahwa teks yang digunakan dalam model tidak terpengaruh oleh tanda baca dan memiliki konsistensi dalam format huruf kecil, yang memudahkan pemrosesan lebih lanjut.

2.2.2 Lematisasi

```
lemmatizer = WordNetLemmatizer()  
  
# Fungsi untuk melakukan lemmatization  
def lemmatize_text(text):  
    return lemmatizer.lemmatize(text)  
  
# Terapkan fungsi lemmatize_text ke kolom 'patterns'  
data['patterns'] = data['patterns'].apply(lemmatize_text)  
  
# Gabungkan hasil lemmatization menjadi satu string  
data['patterns'] = data['patterns'].apply(lambda wrd: ''.join(wrd))  
  
# Menampilkan data dari baris 1 hingga 10  
data[1:10]
```

Gambar 3 Lematisasi

Selanjutnya data yang telah melalui proses data cleaning, dilanjutkan ketahap lemmatisasi. Lemmatisasi adalah proses yang dilakukan untuk menyederhanakan kata-kata menjadi bentuk dasar. Pada code diatas dapat dilihat

bahwa proses lemmatisasi dilakukan dengan menggunakan kelas *WordNetLemmatizer* dari Pustaka *nlk*. Fungsi lemmatisasi ini kemudian diterapkan pada semua elemen dalam kolom *patterns*, sehingga setiap kata dalam teks akan dilemmatisasi untuk mendapatkan bentuk dasar. Hal ini membantu model memahami maksud pengguna tanpa harus mencocokkan setiap bentuk kata secara manual.

2.2.3 Tokenisasi

```
tokenizer = Tokenizer(num_words=1000)

# Fungsi untuk melakukan tokenisasi
def fit_and_tokenize(text_data):
    tokenizer.fit_on_texts(text_data)
    return tokenizer.texts_to_sequences(text_data)

# Terapkan fungsi fit_and_tokenize ke kolom 'patterns' dan
train = fit_and_tokenize(data['patterns'])

# Menampilkan hasil tokenisasi
train
```

Gambar 4 Tokenisasi

Tokenisasi merupakan proses yang dilakukan untuk membagi teks menjadi unit-unit kecil seperti kata atau kalimat. Kode diatas digunakan untuk melakukan tokenisasi pada data teks yang ada pada kolom *patterns*. Proses dilakukan dengan memanfaatkan objek *Tokenizer* dari Pustaka Keras. Hasil dari tokenisasi ini menghasilkan setiap kata dalam teks akan digantikan dengan angka yang sesuai berdasarkan frekuensinya dalam data, dan hasil akhirnya adalah data yang sudah di-tokenisasi dalam bentuk urutan angka.

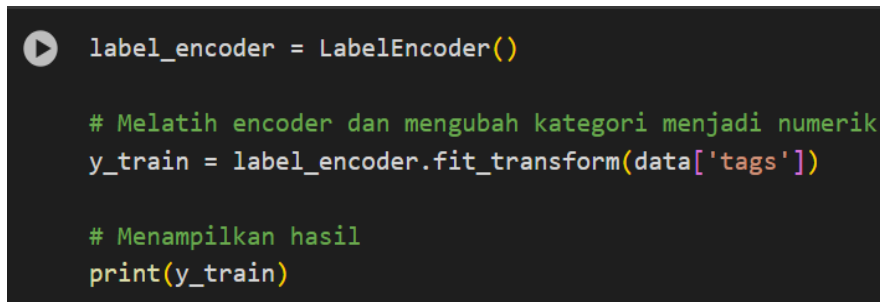
2.2.4 Padding

```
[38] x_train = pad_sequences(train)
      print(x_train)
```

Gambar 5 Padding

Selanjutnya dilakukan proses *padding* untuk menyamakan panjang dari semua teks input dengan menambahkan nilai-nilai nol (*padding*) untuk teks yang lebih pendek. Kode diatas melakukan proses *padding* pada data yang sudah di-tokenisasi, yaitu *train*, menggunakan fungsi *pad_sequences* dari Keras. Dengan *padding*, teks yang lebih pendek akan memiliki jumlah kata yang sama dengan teks yang lebih panjang, memastikan input untuk model dalam bentuk yang seragam.

2.2.5 Encoding

A screenshot of a code editor with a dark background. It shows a Python script for encoding categorical data. The code starts with creating a LabelEncoder object, then fits it to the 'tags' column of a dataset and transforms it into numerical values. Finally, it prints the transformed data. The code is as follows:

```
label_encoder = LabelEncoder()

# Melatih encoder dan mengubah kategori menjadi numerik
y_train = label_encoder.fit_transform(data['tags'])

# Menampilkan hasil
print(y_train)
```

Gambar 6 Encoding

Proses *Encoding* merupakan proses mengubah teks menjadi numerik atau format angka yang dapat diproses oleh model. Kode ini menggunakan *LabelEncoder* dari *scikit-learn* untuk mengonversi data kolom tags menjadi bentuk numerik. Proses ini diperlukan karena model pembelajaran mesin biasanya memerlukan data numerik sebagai input dan output.

2.3 Modeling

Model dibangun dengan menggunakan metode *Long Short-Term Memory* (LSTM), yang merupakan pengembangan dari metode *Recurrent Neural Network* (RNN). LSTM dirancang untuk mengatasi masalah yang sering muncul pada RNN, yaitu *vanishing gradient problem*. Metode LSTM digunakan dalam model ini karena mampu memberikan akurasi yang baik untuk data berbentuk teks dan memiliki keunggulan dalam memproses data dengan ketergantungan jangka panjang (*long-term dependency*). Dengan struktur jaringan yang lebih kompleks dibandingkan dengan RNN standar, LSTM dapat menangani dependensi jangka panjang antar kata dalam kalimat atau paragraf, yang sangat penting dalam memahami konteks percakapan atau teks secara keseluruhan.

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 13)	0
embedding_1 (Embedding)	(None, 13, 20)	9,200
lstm_1 (LSTM)	(None, 13, 32)	6,784
dropout_1 (Dropout)	(None, 13, 32)	0
flatten_1 (Flatten)	(None, 416)	0
dense_1 (Dense)	(None, 144)	60,048

Total params: 76,832 (297.00 KB)
Trainable params: 76,832 (297.00 KB)
Non-trainable params: 0 (0.00 B)

Gambar 7 Arsitektur Model

Model dibangun dengan menambahkan beberapa lapisan didalam nya, lapisan *embedding* digunakan untuk mengonversi input teks menjadi representasi *vector*. Selanjutnya ditambahkan lapisan LSTM untuk memproses urutan input dan mengingat informasi penting. Lapisan *dropout* digunakan untuk mengurangi risiko *overfitting* dengan menonaktifkan sejumlah neuron secara acak. Kemudian terdapat lapisan *flatten* yang digunakan mengubah *output* yang memiliki bentuk *multi-dimensional* (berurutan) dari layer LSTM menjadi bentuk satu dimensi, sehingga dapat diteruskan ke layer berikutnya (*Dense layer*). Lapisan terakhir merupakan lapisan *dense* yang memberikan *output* akhir dengan probabilitas setiap tag kelas.

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer='adam',
              metrics=['accuracy'])
```

Gambar 8 Penambahan Parameter

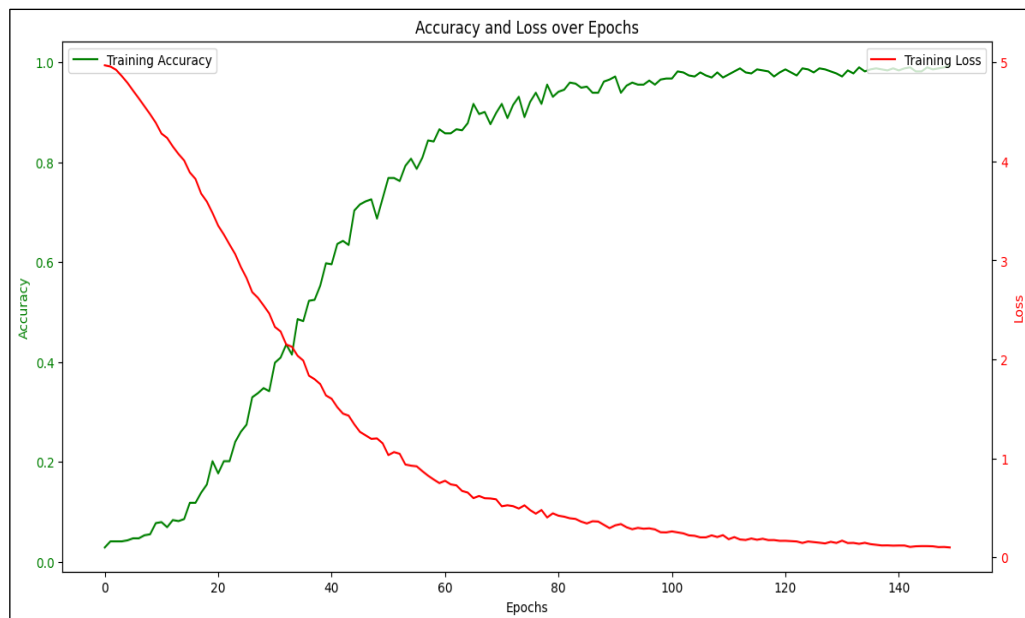
Pada model ini terdapat beberapa parameter yang digunakan untuk pengoptimalan dan evaluasi model. Fungsi loss yang digunakan adalah *sparse categorical crossentropy*. Fungsi *loss* ini umum digunakan untuk masalah klasifikasi multi-kelas di mana target atau label kelas berupa integer (bukan *one-hot encoded*). Selanjutnya terdapat *optimizer* yang digunakan yaitu Adam yang menggabungkan keunggulan dari dua algoritma optimisasi sebelumnya, yaitu Momentum dan RMSprop. *metrics=['accuracy']* digunakan sebagai metrik evaluasi untuk melihat seberapa baik model dalam memprediksi kelas yang benar.

```
[ ] history = model.fit(
    X_train,
    y_train,
    epochs=150)
```

Gambar 9 Pelatihan Model

Proses pelatihan model dilakukan menggunakan metode `model.fit()`, di mana data latih `X_train` (*input*) dan `y_train` (*label*) digunakan untuk melatih model selama 150 *epoch*. Pada setiap *epoch*, model akan memperbarui bobot-bobotnya berdasarkan perhitungan *loss function* dan algoritma *optimizer* yang telah didefinisikan sebelumnya. Dalam proses ini, model belajar untuk mengurangi kesalahan prediksi terhadap data latih dengan tujuan meningkatkan akurasi prediksi.

2.4 Evaluasi

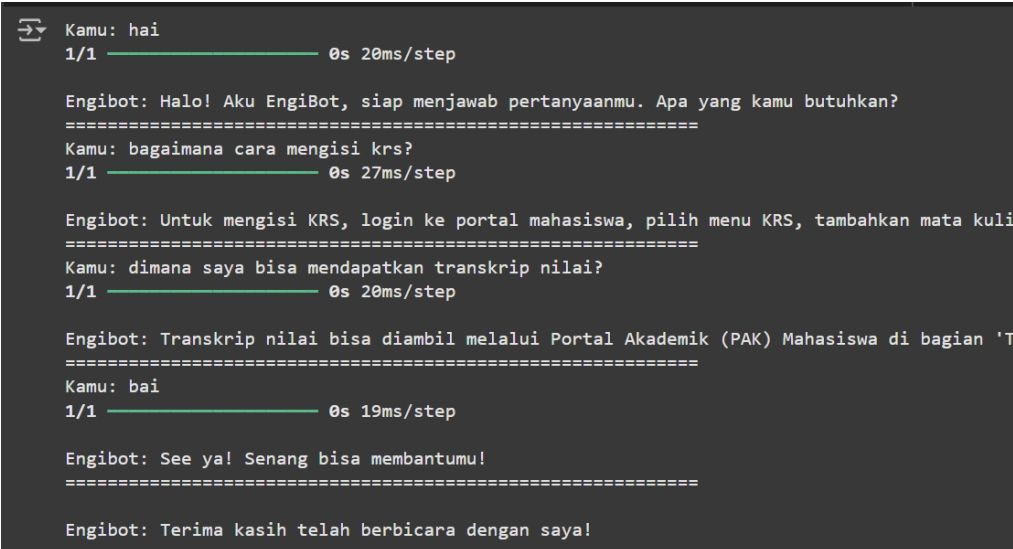


Gambar 10 Grafik Evaluasi

Evaluasi dilakukan dengan melihat perkembangan nilai *loss* dan akurasi model selama proses pelatihan. Pada grafik yang ditampilkan, sumbu horizontal merepresentasikan jumlah *epoch*, sedangkan sumbu vertikal kiri menunjukkan akurasi, dan sumbu vertikal kanan menunjukkan nilai *loss*. Dari grafik tersebut, terlihat bahwa seiring bertambahnya *epoch*, nilai *loss* cenderung menurun, yang

menandakan bahwa model semakin baik dalam memprediksi data latih. Sebaliknya, nilai akurasi meningkat, menunjukkan bahwa model semakin akurat dalam klasifikasi. Hasil ini menunjukkan bahwa model belajar dengan baik dan tidak mengalami *overfitting* atau *underfitting* yang signifikan, karena kedua metrik tersebut stabil mendekati nilai optimal di akhir proses pelatihan. Proses pelatihan menghasilkan akurasi sebesar 0.997, yang mencerminkan kemampuan model untuk melakukan prediksi dengan tingkat keakuratan yang hampir sempurna pada data latih.

2.5 Testing



```
➡ Kamu: hai
1/1 ————— 0s 20ms/step

Engibot: Halo! Aku EngiBot, siap menjawab pertanyaanmu. Apa yang kamu butuhkan?
=====
Kamu: bagaimana cara mengisi krs?
1/1 ————— 0s 27ms/step

Engibot: Untuk mengisi KRS, login ke portal mahasiswa, pilih menu KRS, tambahkan mata kuli
=====
Kamu: dimana saya bisa mendapatkan transkrip nilai?
1/1 ————— 0s 20ms/step

Engibot: Transkrip nilai bisa diambil melalui Portal Akademik (PAK) Mahasiswa di bagian 'T
=====
Kamu: bai
1/1 ————— 0s 19ms/step

Engibot: See ya! Senang bisa membantumu!
=====

Engibot: Terima kasih telah berbicara dengan saya!
```

Gambar 11 *Testing*

Hasil *testing* menunjukkan bahwa model *chatbot* dapat merespons pertanyaan pengguna dengan baik. Ketika pengguna menanyakan cara mengisi KRS, chatbot dengan cepat memberikan instruksi yang tepat, yakni mengarahkan pengguna untuk *login* ke portal mahasiswa dan memilih menu KRS. Selain itu, saat pengguna bertanya tentang transkrip nilai, chatbot juga berhasil menjawab bahwa transkrip dapat diperoleh melalui Portal Akademik Mahasiswa di bagian "Transkrip". Selain itu, respons pembuka dan penutup dari chatbot juga memberikan kesan yang ramah, sehingga mendukung interaksi yang positif dengan pengguna. Hal ini menunjukkan bahwa *chatbot* berhasil dikembangkan dengan baik untuk memahami dan menjawab pertanyaan terkait layanan akademik khususnya yang berkaitan dengan portal akademik.

3. Analisis Model

Penelitian ini menggunakan metode *Long Short-Term Memory* (LSTM) dalam membangun model. LSTM merupakan salah satu jenis jaringan saraf tiruan yang termasuk dalam kategori *deep learning*, bukan *shallow learning*. Perbedaannya terletak pada karakteristik LSTM yang memiliki arsitektur dengan banyak lapisan (*multi-layer*) serta kemampuan untuk menangkap pola yang kompleks dalam data. Dalam pembangunan model ini, digunakan beberapa lapisan tambahan seperti *embedding*, *dropout*, *flatten*, dan *dense*. Lapisan *embedding* berfungsi merepresentasikan data teks ke dalam bentuk vektor, lapisan *dropout* digunakan untuk mencegah *overfitting* dengan mengurangi ketergantungan pada fitur tertentu, lapisan *flatten* mengubah keluaran multidimensi menjadi bentuk satu dimensi, dan lapisan *dense* dengan fungsi aktivasi *softmax* memungkinkan prediksi berbasis probabilitas untuk setiap kelas. Lapisan-lapisan ini memperkuat ciri khas *deep learning* untuk membangun dan melatih jaringan saraf yang dalam (*deep neural networks*). Kombinasi dari lapisan-lapisan ini bekerja secara hierarkis, memungkinkan model menangkap pola-pola yang lebih kompleks dari data, yang merupakan ciri khas dari *deep learning*. Hal ini membedakannya dari *shallow learning*, yang umumnya menggunakan arsitektur sederhana dengan hanya satu atau sedikit lapisan, sehingga kurang mampu menangkap pola kompleks.

4. Kesimpulan

Chatbot ini berhasil menjawab pertanyaan-pertanyaan terkait informasi akademik, seperti pengisian KRS dan akses ke transkrip nilai, sesuai dengan kebutuhan pengguna. Pembuka dan penutup yang ramah semakin memperkuat kualitas interaksi, menciptakan pengalaman pengguna yang positif. Dengan demikian, *chatbot* ini diharapkan dapat menjadi solusi yang bermanfaat bagi mahasiswa dalam mengatasi berbagai kesulitan terkait urusan akademik, terutama dalam hal mendapatkan informasi secara cepat dan akurat melalui layanan *chatbot* otomatis.