LAPORAN UAS DEEP LEARNING
CHATBOT LAYANAN MAHASISWA BARU UNIB

DISUSUN OLEH :

1. **Yulia Pratiwi (G1A021029)**
2. **Zabrila Amrina Zadia Putri (G1A021053)**

DOSEN PEMBIMBING:

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

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

1. Pendahuluan

Universitas Bengkulu (UNIB) merupakan sebuah perguruan tinggi negeri yang terletak di Kota Bengkulu, Indonesia. UNIB didirikan pada 24 April 1982 dan memiliki visi untuk menjadi universitas unggul dalam pengembangan ilmu pengetahuan, teknologi, seni, dan budaya. Universitas ini menawarkan berbagai program studi dari jenjang diploma, sarjana, hingga pascasarjana di berbagai bidang ilmu. Selain itu, UNIB juga aktif dalam penelitian dan pengabdian kepada masyarakat untuk mendukung pembangunan regional dan nasional.

Penerimaan Mahasiswa Baru (PMB) di Universitas Bengkulu sangat penting, dan salah satu peran utama adalah memberikan informasi terkait PMB kepada calon mahasiswa. Bagian Tata Usaha (TU) sering kali harus menjawab pertanyaan yang sama berkali-kali, yang tidak efisien. Oleh karena itu, diperlukan peningkatan pelayanan informasi PMB dengan menggunakan Chatbot untuk memberikan jawaban yang konsisten dan terbaru kepada calon mahasiswa.

Chatbot adalah program komputer yang merespons pesan pengguna melalui teks atau suara menggunakan pemrosesan bahasa alami (NLP) sebagai bagian dari kecerdasan buatan. Chatbot pertama kali diperkenalkan oleh Joseph Weizenbaum pada tahun 1966 dan menjadi populer berkat tes Loebner untuk mengukur efektivitasnya. NLP merupakan salah satu cabang kecerdasan buatan yang memungkinkan sistem memahami teks atau ucapan seperti manusia. Aplikasi NLP mencakup perangkat pintar yang dapat mendengarkan dan merespons perintah manusia, mesin pencari yang memberikan informasi, dan chatbot yang merespons pengguna dengan cepat dan akurat (Adamopoulou & Moussiades, 2020).

Pada penelitian sebelumnya, diteliti menggunakan metode *Bag of Words* (BoW) dan *Feed-forward neural network* (FNN) kepada mengembangkan chatbot untuk digunakan di rumah sakit, hasil penelitian menunjukkan chatbot memiliki akurasi 100% pada iterasi ke-1000 *loss* 0% pada iterasi ke-697 (Mittal et al., 2021). Dalam penelitian sebelumnya, kami memeriksa klasifikasi teks menggunakan metode TF-IDF dan Word2Vec untuk ekstraksi fitur, dan jaringan saraf konvolusional (CNN). Hasil pencarian mencapai akurasi optimal 82% (Song et al., 2019).

Salah satu komponen utama dalam pengembangan chatbot ini adalah penggunaan *Long Short-Term Memory* (LSTM), sebuah jenis jaringan saraf tiruan yang dirancang

untuk menangani data sekuensial dan memiliki kemampuan memahami konteks temporal. Model klasifikasi yang dibangun dengan menggunakan metode LSTM akan mampu memahami dan mengklasifikasikan beberapa pertanyaan berdasarkan kategori yang telah ditentukan sebelumnya, seperti biaya pendaftaran, jadwal pendaftaran, biaya kuliah, atau informasi pencapaian program studi (Yu et al., 2019)

Proyek ini mencakup beberapa tahapan utama, seperti persiapan data, pembangunan model, pelatihan, dan evaluasi performa chatbot. Melalui laporan ini, pembaca akan mendapatkan gambaran menyeluruh tentang pengembangan chatbot mulai dari analisis kode hingga kesimpulan yang dapat ditarik dari hasil akhir.

2. Analisis Model

Proyek ini menggunakan metode *deep learning*, yang secara mendasar berbeda dari *shallow learning*. *Shallow learning* biasanya melibatkan model pembelajaran mesin sederhana seperti regresi logistik, Support Vector Machines (SVM), atau Decision Trees, yang hanya memiliki satu atau dua lapisan transformasi dan terbatas dalam memahami pola yang kompleks. Sebaliknya, proyek ini menggunakan *Long Short-Term Memory* (LSTM), sejenis jaringan saraf tiruan yang dirancang untuk mengelola data sekuensial dengan memahami hubungan temporal di dalamnya. LSTM memiliki arsitektur yang lebih dalam dengan mekanisme memori jangka panjang dan pendek, yang memungkinkannya menangkap konteks historis dalam data teks.

Selain itu, proyek ini menggunakan lapisan *embedding* untuk merepresentasikan kata-kata sebagai vektor numerik, LSTM untuk menangkap hubungan temporal antar kata, dan *dense layer* untuk menghasilkan klasifikasi. Kombinasi lapisan ini memungkinkan model memahami pola kompleks yang ada dalam data pemrosesan bahasa alami (NLP), yang menjadi inti dari pengembangan chatbot. Kemampuan ini jauh melampaui *shallow learning* yang tidak dirancang untuk menangani data sekuensial atau memahami konteks.

3. Penjelasan Kode

Data yang digunakan dalam proyek Chatbot ini adalah data yang dikumpulkan dari sumber terkait seperti pada laman https://regmaba.unib.ac.id/ dan https://www.unib.ac.id/ .Data-data yang didapatkan dari laman *website* terkait dikumpulkan hingga menjadi dataset yang dapat digunakan dalam Chatbot layanan mahasiswa baru UNIB. Kemudian data tersebut diolah sehingga menyusun dataset

percakapan yang mencakup pertanyaan dan jawaban mengenai hal-hal terkait penerimaan mahasiswa baru yang diperlukan oleh Mahasiswa UNIB terkait informasi yang ingin ditanyakan oleh Mahasiswa. Kemudian dataset ini dilakukan pelatihan agar dapat menghasilkan model yang dapat memberikan respons dan jawaban yang tepat terkait dengan pertanyaan yang diberikan.

```
# Import Libraries
import json
import nltk
import time
import random
import string
import pickle
import numpy as np
import pandas as pd
from io import BytesIO
import tensorflow as tf
import IPython.display as ipd
import matplotlib.pyplot as plt
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.models import Model
from keras.utils import plot_model
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Input, Embedding, LSTM
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Flatten, Dense, GlobalMaxPool1D
```

Penjelasan :

Kode pada gambar digunakan untuk menginstall *library* yang diperlukan dalam pembuataan model Chatbot ini, Pustaka dasar Python seperti json, random, dan pickle digunakan untuk pemrosesan data, sedangkan nltk mendukung pemrosesan bahasa alami (NLP) seperti tokenisasi dan lemmatization. Pustaka numerik seperti numpy dan pandas digunakan untuk manipulasi data, sedangkan matplotlib memfasilitasi visualisasi, tensorflow.keras menyediakan komponen untuk membangun model seperti input, embedding, dan LSTM yang membantu memahami data teks berurutan. pad_sequences digunakan untuk menyeimbangkan panjang dan kelas data seperti Dense dan GlobalMaxPool1D. Encoder label Sklearn mengubah label menjadi format digital dan memplot model untuk memvisualisasikan arsitektur model. Kombinasi perpustakaan ini mendukung keseluruhan proses pengembangan, mulai dari pra-pemrosesan hingga evaluasi model.

```
[ ] # Package sentence tokenizer
    nltk.download('punkt')
    # Package lemmatization
    nltk.download('wordnet')
    # Package multilingual wordnet data
    nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

Penjelasan:

Kode pada gambar diatas digunakan untuk mengunduh package yang dibutuhkan sepeti nltk.download(‘punkt’) yang digunakan untuk tokenisasi kalimat dan kata, nltk.download(‘wordnet’) yang digunakan untuk mengunduh basis data WordNet utnuk lemmatization (mengubah kata menjadi bentuk dasarnya), serta kode terakhir yakni nltk.download(‘omw-1.4’) yang digunakan untuk mengunduh multibahasa dari open multilingual wordnet untuk mendukung operasi berbagai bahasa.

```
[ ] import nltk
    nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
# Memuat data JSON
with open('data.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 pattern in intent['patterns']:
        inputs.append(pattern)
        tags.append(intent['tag'])

        # Tokenisasi dan pemrosesan kata
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        documents.append((w, intent['tag']))

        # Menambahkan kelas jika belum ada
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

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

# Tampilkan contoh data
print(data.head())
```

Penjelasan:

Kode di atas memproses data intents dari file JSON menjadi format siap pakai untuk melatih model chatbot. Tokenizer punkt_tab dari NLTK digunakan untuk memecah teks menjadi kata, sedangkan file JSON dimuat menggunakan pustaka json. Variabel seperti tags, inputs, responses, words, dan documents diinisialisasi untuk menyimpan pola input, tag, respons, token kata, dan pasangan dokumen-tag. Karakter khusus seperti tanda tanya diabaikan. Setiap pola dikodekan dengan NLTK dan ditambahkan ke daftar kata, sementara respons diurutkan berdasarkan tag. Data intents kemudian dikonversi menjadi DataFrame menggunakan Pandas untuk mempermudah manipulasi dan analisis, lalu ditampilkan untuk memastikan data siap digunakan dalam pelatihan model.

Remove Punctuations

Tahapan praproses pada data teks yang pertama adalah menghapus punktuasi atau tanda baca seperti *special character* yaitu '!' (**tanda seru**) ',' (**tanda koma**) '.' (**tanda titik sebagai berhenti**) '?' (**tanda tanya**) dan tanda baca yang lain. Tahapan ini gunanya untuk mempermudah pemrosesan data teks yang akan kita olah.

```
[ ] # Removing Punctuations (Menghilangkan Punktuasi)
    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))
```

Lemmatization (Lematisasi)

```
[ ] lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
    words = sorted(list(set(words)))

    print (len(words), "unique lemmatized words", words)
```

```
285 unique lemmatized words ['(', ')', ',', '.', '2024', 'absen', 'ada', 'adakah', 'adik', 'administrasi', 'akademik', 'akhir', 'akun', 'a
```

Menyortir Data Kelas Tags

```
[ ] # sorting pada data class
    classes = sorted(list(set(classes)))
    print (len(classes), "classes", classes)
```

```
66 classes ['akademik_tugas', 'alamat_pengiriman_berkas', 'bantuan', 'beasiswa', 'beasiswa_fakultas', 'beasiswa_jenis', 'beasiswa_kewajiba
```

Tokenization (Tokenisasi)

```
[ ] # Tokenize the data (Tokenisasi Data)
    tokenizer = Tokenizer(num_words=2000)
    tokenizer.fit_on_texts(data['patterns'])
    train = tokenizer.texts_to_sequences(data['patterns'])
    train
```

Penjelasan :

Kode di atas adalah bagian dari prapemrosesan data teks untuk keperluan pemrosesan bahasa alami (NLP). Kode pertama-tama menghapus tanda baca dari teks menggunakan fungsi “string.punctuation”, kemudian mengubah semua huruf menjadi huruf kecil untuk konsistensi. Lexisasi kemudian dilakukan dengan menggunakan “WordNetLemmatizer” untuk mengubah kata-kata tersebut ke bentuk dasarnya, dengan

mengabaikan beberapa kata dalam daftar “ignore_words”. Kemudian, daftar kata yang diproses akan diurutkan berdasarkan abjad dan kata duplikat akan dihapus. Untuk data berlapis (“lapisan”), kode ini juga menghapus duplikat dan mengurutkannya. Terakhir, tokenisasi dilakukan pada data teks menggunakan “Tokenizer” dari pustaka Keras, yang mengubah teks menjadi serangkaian angka berdasarkan frekuensi kata dengan batas maksimum 2.000 kata. Hasil akhirnya adalah data teks yang siap digunakan untuk melatih model NLP.

Padding

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

```
[[  0   0   0 ...   0   0 150]
 [  0   0   0 ...   0   0 151]
 [  0   0   0 ...   0  74 152]
 ...
 [  0   0   0 ...   2   5   6]
 [  0   0   0 ...  44  12  15]
 [  0   0   0 ...  21 281  64]]
```

Hasil setelah padding adalah setiap sequence memiliki panjang yang sama. Padding dapat melakukan ini dengan menambahkan 0 secara default pada awal sequence yang lebih pendek.

Encoding Text

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

```
[58 58 58 58 58 44 44 44 44 44 61 61 61 61 61  2  2  2  2  2 28 28 28 28
 24 24 24 24 14 14 14 14 63 63 63 63 26 26 26 26 19 19 19 19 20 20 20 20
 41 41 41 41 41 22 22 22 16 16 16 16  1  1  1  1 60 60 60 60 64 64 64 18
 18 18 18 16 16 16 16 40 40 40 40 40 42 42 42 42 42 45 45 45 45 45 65 65
 65 65 65 38 38 38 29 29 29 33 33 33 31 31 31 34 34 34 30 30 30 35 35 35
 36 36 36 32 32 32 39 39 39  9  9  9 11 11 11  5  5  5  4  4  4 10 10 10
  8  8  8  6  6  6  7  7  7 25 25 25 25 23 23 23 23 27 27 27 27 46 46 46
 46 12 12 12 12 62 62 62 62 51 51 51 51 48 48 47 47 49 49 53 53 52 52 54 54
 50 50 59 59 59 59 57 57 57 43 43 43 21 21 21 21 15 15 15 55 55 55 55 13 13 13  3
  3  3 37 37 37 37  0  0  0 56 56 56 17 17 17]
```

```
[ ] # Melihat hasil input pada data teks
    input_shape = x_train.shape[1]
    print(input_shape)
```

```
12
```

Penjelasan :

Kode di atas selanjutnya memproses awal data teks untuk menyiapkan input bagi model NLP. Pertama, data teks telah diubah menjadi string numerik tokenized yang akan di pad menggunakan fungsi 'pad_sequences' sehingga setiap string memiliki panjang yang sama dengan menambahkan nilai null ke string tersebut. terpendek Berikutnya, label kategori di kolom data['tags'] diubah menjadi angka menggunakan LabelEncode, di mana setiap kategori unik direpresentasikan sebagai nilai numerik. Kemudian panjang maksimum string teks setelah padding dihitung menggunakan 'x_train.shape[1]', yang akan digunakan untuk menentukan ukuran input model. Proses ini memastikan bahwa data teks dan label berada dalam format yang konsisten dan siap untuk pelatihan model.

```
[ ] # Simpan hasil pemrosesan teks dengan menggunakan pickle
    pickle.dump(words, open('words.pkl','wb'))
    pickle.dump(classes, open('classes.pkl','wb'))
```

Save Label Encoder & Tokenizer

```
[ ] pickle.dump(le, open('le.pkl','wb'))
    pickle.dump(tokenizer, open('tokenizers.pkl','wb'))
```

```
# Creating the model (Membuat Modelling)
i = Input(shape=(input_shape,)) # Layer Input
x = Embedding(vocabulary+1,10)(i) # Layer Embedding
x = LSTM(10, return_sequences=True, recurrent_dropout=0.2)(x) # Layer Long Short Term Memory
x = Flatten()(x) # Layer Flatten
x = Dense(output_length, activation="softmax")(x) # Layer Dense
model  = Model(i,x) # Model yang telah disusun dari layer Input sampai layer Output

# Compiling the model (Kompilasi Model)
model.compile(loss="sparse_categorical_crossentropy", optimizer='adam', metrics=['accuracy'])
```

```
[ ] # Visualization Plot Architecture Model (Visualisasi Plot Arsitektur Model)
    plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)
```

Show hidden output

Penjelasan:

Kode di atas terdiri dari beberapa langkah penting dalam membangun dan menyimpan model NLP. Pertama, data hasil preprocessing, seperti daftar kata unik (words), kelas label (classes), label encoder (le), dan tokenizer (tokenizer), disimpan ke dalam file menggunakan library pickle agar dapat digunakan kembali tanpa melakukan preprocessing ulang. Selanjutnya, model neural network dibuat menggunakan arsitektur bertingkat: lapisan input (Input), lapisan embedding (Embedding) untuk merepresentasikan kata dalam bentuk vektor, lapisan LSTM dengan dropout untuk menangani data sekuensial, lapisan Flatten untuk meratakan output LSTM, dan lapisan dense dengan aktivasi softmax untuk klasifikasi multi-kelas. Model kemudian dikompilasi menggunakan fungsi kerugian sparse_categorical_crossentropy, optimizer Adam, dan metrik akurasi. Terakhir, arsitektur model divisualisasikan dan disimpan sebagai gambar menggunakan fungsi plot_model untuk memudahkan pemahaman struktur model.

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

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 12)	0
embedding (Embedding)	(None, 12, 10)	2,820
lstm (LSTM)	(None, 12, 10)	840
flatten (Flatten)	(None, 120)	0
dense (Dense)	(None, 66)	7,986

Total params: 11,646 (45.49 KB)
Trainable params: 11,646 (45.49 KB)
Non-trainable params: 0 (0.00 B)

```
# Training the model (Melatih model data sampai 300 kali)
train = model.fit(x_train, y_train, epochs=200)
```

Penjelasan:

Gambar di atas menunjukkan ringkasan 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 200 epoch. Proses ini bertujuan untuk mengoptimalkan bobot model sehingga dapat memprediksi output dengan akurasi yang tinggi.

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9874 - loss: 0.2371
Epoch 195/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9705 - loss: 0.2778
Epoch 196/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9742 - loss: 0.2726
Epoch 197/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9776 - loss: 0.2651
Epoch 198/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9759 - loss: 0.2448
Epoch 199/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9748 - loss: 0.2520
Epoch 200/200
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9730 - loss: 0.2918
```

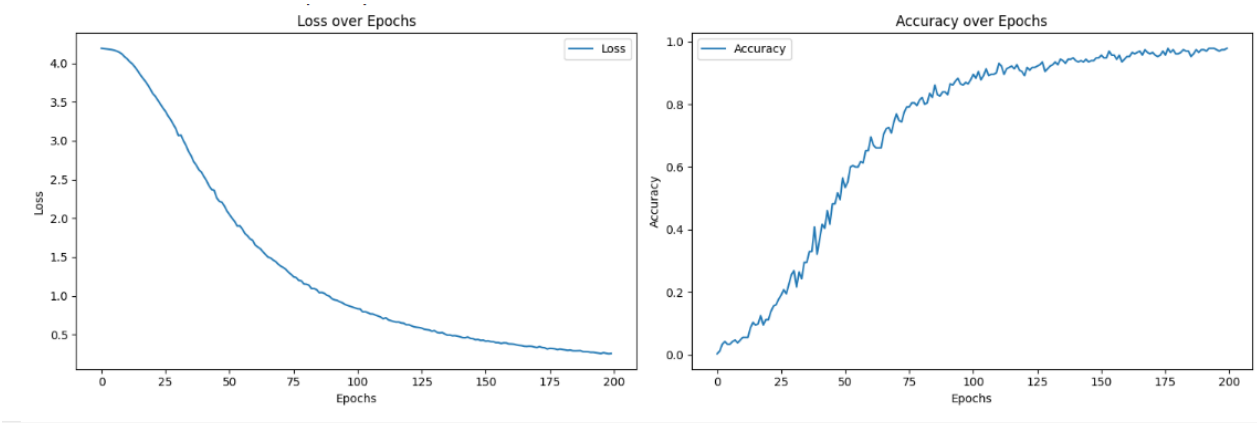
Dengan akurasi akhir pelatihan model pada gambar diatas adalah 0.9730 dengan loss sekitar 0.2918.

```
[ ] # Evaluasi model pada data latih
    loss, accuracy = model.evaluate(x_train, y_train)
    print(f"Model evaluation - Loss: {loss}, Accuracy: {accuracy}")

    # **Plotting Grafik Akurasi dan Kerugian Selama Pelatihan**

    # Mengambil history pelatihan dari model
    history = train.history

    # Membuat Grafik untuk Loss dan Akurasi
    fig, ax = plt.subplots(1, 2, figsize=(15, 5))

    # Grafik Loss
    ax[0].plot(history['loss'], label='Loss')
    ax[0].set_title('Loss over Epochs')
    ax[0].set_xlabel('Epochs')
    ax[0].set_ylabel('Loss')
    ax[0].legend()

    # Grafik Akurasi
    ax[1].plot(history['accuracy'], label='Accuracy')
    ax[1].set_title('Accuracy over Epochs')
    ax[1].set_xlabel('Epochs')
    ax[1].set_ylabel('Accuracy')
    ax[1].legend()

    plt.tight_layout()
    plt.show()
```

```
8/8 ━━━━━━━━━━━━━━━━━━━━ 2s 3ms/step - accuracy: 0.9795 - loss: 0.2887
Model evaluation - Loss: 0.23924916982650757, Accuracy: 0.9826086759567261
```

Penjelasan:

Kode di atas melakukan evaluasi performa model dan visualisasi hasil pelatihan. Pertama, model dievaluasi pada data latih (x_train dan y_train) menggunakan metode model.evaluate, yang mengembalikan nilai loss (kerugian) dan akurasi. Hasil evaluasi dicetak untuk memberikan gambaran seberapa baik model mempelajari data latih. Selanjutnya, riwayat pelatihan (train.history) digunakan untuk membuat grafik perubahan loss dan akurasi selama epoch pelatihan. Grafik pertama menunjukkan perubahan loss (kerugian) di setiap epoch, sedangkan grafik kedua menunjukkan perubahan akurasi. Visualisasi ini membantu memahami bagaimana model berkembang selama pelatihan, apakah mengalami overfitting atau underfitting, serta apakah konvergensi model berjalan dengan baik.



Penjelasan:

Gambar di atas menunjukkan dua grafik yang memvisualisasikan kinerja model selama proses pelatihan selama 200 epoch. Grafik pertama di sebelah kiri menggambarkan penurunan loss (kerugian) secara konsisten dari awal hingga akhir pelatihan, menunjukkan bahwa model berhasil meminimalkan error saat mempelajari data. Grafik kedua di sebelah kanan menunjukkan peningkatan akurasi secara bertahap hingga mencapai nilai mendekati 1, yang berarti model menjadi semakin baik dalam memprediksi dengan benar selama pelatihan. 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.

4. Kesimpulan

Proyek ini berhasil menghasilkan chatbot yang dapat memberikan respons otomatis terhadap pertanyaan mahasiswa baru di UNIB. Hasil evaluasi menunjukkan bahwa model memiliki kinerja yang baik dengan akurasi tinggi pada data pengujian. Namun, terdapat beberapa hal yang dapat ditingkatkan, seperti menambah jumlah data pelatihan untuk meningkatkan generalisasi model dan mengintegrasikan chatbot ke dalam platform yang lebih interaktif. Pengembangan lebih lanjut juga dapat mencakup penggunaan model bahasa yang lebih canggih, untuk meningkatkan pemahaman dan relevansi respons.

DAFTAR PUSTAKA

Adamopoulou, E., & Moussiades, L. (2020). An Overview of Chatbot Technology. In *IFIP Advances in Information and Communication Technology: Vol. 584 IFIP*. Springer International Publishing. https://doi.org/10.1007/978-3-030-49186-4_31

Mittal, M., Battineni, G., Singh, D., Nagarwal, T., & Yadav, P. (2021). Web-based chatbot for Frequently Asked Queries (FAQ) in Hospitals. *Journal of Taibah University Medical Sciences*, *16*(5), 740–746. https://doi.org/10.1016/j.jtumed.2021.06.002

Song, P., Geng, C., & Li, Z. (2019). Research on Text Classification Based on Convolutional Neural Network. *Proceedings - 2nd International Conference on Computer Network, Electronic and Automation, ICCNEA 2019*, 229–232. https://doi.org/10.1109/ICCNEA.2019.00052

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). *A Review of Recurrent Neural Networks : LSTM Cells and Network Architectures*. *1270*, 1235–1270. https://doi.org/10.1162/neco