p-ISSN:2527-5321
e-ISSN:2527-5941

MODEL KLASIFIKASI INTENSI UNTUK CHATBOT LAYANAN PELANGGAN MENGGUNAKAN LONG SHORT-TERM MEMORY

Maeru Bagas Trisoko[1], Ni Wayan Parwati Septiani [2]

[1,2] Program Studi Teknik informatika, Universitas Indraprasta PGRI
Jl. Nangka Raya No.58 C, RT.7/RW.5, Tj. Barat, Kec. Jagakarsa, Kota Jakarta 12530
[1]maerubagas14@gmail.com, [2]wayan.parwati@gmail.com

ABSTRAK

Penelitian ini bertujuan untuk mengembangkan model klasifikasi intensi dalam bentuk *chatbot* layanan pelanggan Bahasa Indonesia menggunakan algoritma *Long Short-Term Memory* (LSTM). Permasalahan utama yang diangkat dalam penelitian ini adalah belum adanya teknologi yang dapat menanggapi pertanyaan pelanggan secara otomatis dan *realtime* di PT Sampul Kreatif Teknologi. *Dataset* yang digunakan *realcase* yang diperoleh dari komentar atau pertanyaan pengguna media sosial yang dikategorikan ke dalam berbagai label/*tag* intensi, seperti layanan-web, keamanan-aplikasi, harga, kontak, dan tag lainnya. Proses pelatihan model melibatkan beberapa tahapan *preprocessing* seperti *lowercasing, punctual removal, tokenization, slang normalization* menggunakan *combined_slang_words.txt*, *word indexing, padding sequences*, serta penggunaan *word embedding id.vec* untuk vektorisasi 300 dimensi. Setelah proses vektorisasi, model dilatih menggunakan jaringan LSTM dengan fungsi aktivasi softmax, *optimizer adam*, menggunakan learning rate 0.001 dan batch size 16 selama 100 epoch. Hasil pelatihan menunjukkan bahwa model memiliki kemampuan klasifikasi yang cukup baik, dengan akurasi evaluasi akhir sebesar 57,53% terhadap data uji. Meskipun masih perlu banyak peningkatan pada pemahaman konteks, namun akurasi tersebut sudah cukup baik karena menggunakan dataset realcase media sosial yang pada umumnya tidak berpola.
Kata Kunci: *LSTM*, *Chatbot*, Klasifikasi, Intensi.

ABSTRACT

This study aims to develop an intention classification model in the form of an Indonesian customer service chatbot using the Long Short-Term Memory (LSTM) algorithm. The main problem in this study is the lack of technology that can respond to customer questions automatically and in real time at PT Sampul Kreatif Teknologi. *The dataset used is real-world data obtained from user comments or questions on social media, categorized into various intention labels/tags such as web services, pricing, and other tags. The model involves several preprocessing steps, such as lowercasing, punctuation removal, tokenization, slang normalization using combined_slang_words.txt, word indexing, padding sequences, and the use of word embedding id.vec for 300-dimensional vectorization. After the vectorization, the model was trained using an LSTM network with a softmax activation, adam optimizer, learning rate of 0.001 and a batch size of 16 for 100 epochs. The training results show that the model has decent classification capabilities, with a final evaluation accuracy of 57.53% on the test data. Although there is still room for improvement in context understanding, this accuracy is already quite good given that the dataset used is real-world social media data, which is generally unstructured.*
Key Words: *LSTM, Chatbot, Classification, Intention.*

PENDAHULUAN

Latar belakang penelitian ini dikarenakan proses layanan masih dilakukan secara manual oleh staf, sehingga waktu tanggap bisa bervariasi dan terkadang memakan waktu lebih lama dan juga belum adanya teknologi yang dapat menanggapi pertanyaan pelanggan secara otomatis dan *realtime* di PT. Sampul Kreatif Teknologi. Oleh karena itu penulis mengusulkan pengembangan sebuah chatbot yang mampu menjawab pertanyaan pelanggan secara otomatis dan *real-time*. *Chatbot* adalah agen percakapan yang memanfaatkan kecerdasan buatan untuk memahami suara atau teks percakapan manusia, chatbot disimulasikan untuk memberikan percakapan antara pelanggan dan sistem secara *real-time* (Babulak, 2023). *Chatbot* pertama yang dikenal luas adalah ELIZA yang dikembangkan oleh Joseph Weizenbaum pada tahun 1966. Weizenbaum, (1966) menjelaskan bahwa kalimat masukan dianalisis berdasarkan aturan yang didasarkan pada kata kunci yang ada pada kalimat masukan, kemudian menjalankan serangkaian aturan yang telah dibuat untuk memproses dan memberi tanggapan. Algoritma yang penulis pilih dalam penelitian *chatbot* ini adalah *Long Short-Term Memory*, Hochreiter &

p-ISSN:2527-5321
e-ISSN:2527-5941

Schmidhuber, (1997) menyatakan bahwa dalam mendukung kemampuan untuk menyimpan informasi jangka panjang, LSTM dilengkapi dengan dua jenis gerbang utama, yaitu *input gate dan output gate. Input gate* berfungsi untuk mengatur kapan informasi baru diperbolehkan masuk dan disimpan ke dalam memori. sedangkan output gate berfungsi sebagai pengatur kapan informasi di dalam memory cell bisa digunakan untuk mempengaruhi bagian jaringan lainnya. Input gate dan output gate membantu memastikan bahwa informasi yang tidak berkaitan tidak mengganggu proses prediksi. Dengan demikian, mekanisme ini memungkinkan LSTM mempertahankan informasi penting dalam jangka panjang secara stabil selama proses pelatihan berlangsung sehingga sangat cocok digunakan dalam perancangan *chatbot* layanan pelanggan yang memanfaatkan *real-case dataset* dengan karakteristik pertanyaan yang yang pada umumnya tidak terstruktur dan sangat teknis. Hasil dari pelatihan model yang dikembangkan menunjukkan performa yang cukup dalam memahami pertanyaan pengguna dengan akurasi sebesar 57.53% yang dapat memahami Bahasa Indonesia formal maupun informal. Dengan adanya *chatbot* ini, diharapkan pelayanan pelanggan akan menjadi lebih efisien, waktu respons lebih singkat serta dapat mengurangi beban pekerjaan staff karena chatbot dapat menanggapi pertanyaan yang berulang.

METODE PENELITIAN

Peneliti memilih metode *Natural Language Processing* (NLP) menggunakan algoritma *Long Short-Term Memory* (LSTM) dengan teknik klasifikasi intensi dalam perancangan *chatbot* dalam penelitian ini, hasil dari penelitian ini akan digunakan untuk layanan pelanggan otomatis di PT Sampul Kreatif Teknologi, sebuah perusahaan yang menawarkan jasa di bidang teknologi dan perangkat lunak.

Dataset

Peneliti melakukan pengumpulan data *real-case* dari beberapa sumber di internet, seperti komentar pada aplikasi Tiktok dan pertanyaan-pertanyaan yang ada pada situs Quora. Peneliti menggunakan teknik *web scraping* dalam mengumpulkan data dari sumber tersebut. Saat penggabungan data, proses klasifikasi intensi berdasarkan *tag* dilakukan dengan pencocokan *keyword*. Misalnya, jika dalam pertanyaan terdapat kata "harga", maka secara *default* pertanyaan tersebut dimasukkan ke dalam *tag* harga. Namun karena proses klasifikasi *tag* menggunakan pencocokan *keyword* masih kurang akurat sesuai konteksnya, maka peneliti memutuskan untuk mengklasifikasi intensi *dataset* secara manual dengan menempatkan kalimat pertanyaan (*pattern*) kedalam intensi (*tag*) yang sesuai konteksnya. Dataset yang telah diperoleh memiliki karakteristik sebagai berikut.

Tabel 1. Karakteristik *Dataset*

No	Deskripsi	Nilai
1	Jumlah *tag*	29 label
2	Total *Patterns*	729 kalimat
3	Total *Responses*	74 kalimat
4	Bahasa	Bahasa Indonesia formal dan tidak formal

Dataset yang digunakan dalam penelitian menggunakan struktur klasifikasi intensi dengan membagi menjadi tiga bagian kelas dengan keterangan sebagai berikut.

Tabel 2. Struktur *Dataset*

Nama Kelas	Deskripsi	Contoh Data
tag	Label intensi dari klasifikasi pertanyaan	harga-web
patterns	Kalimat pertanyaan dari sosial media	"Spil harga dong mas untuk web wisata tourr"
responses	Kalimat tanggapan chatbot untuk label intensi	"Harga pengembangan website sangat bervariasi tergantung kompleksitas project yang kakak inginkan, Kakak bisa menjelaskan detail project tersebut kepada kontak agen untuk mendapat harga terbaik."

Pre-Processing Data

Dalam proses pelatihan model penelitian ini melibatkan beberapa tahapan *pre-processing* mulai dari *lowercasing, punctual removal, tokenization, slang normalization* menggunakan *combined_slang_words.txt* dari repositori github milik louisowen6, *word indexing, padding sequences*, serta penggunaan *word embedding* untuk

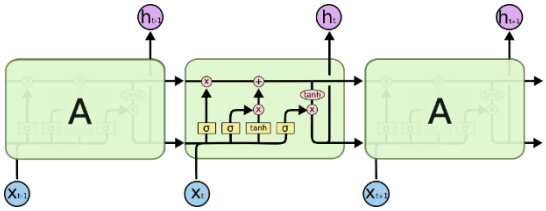
p-ISSN:2527-5321
e-ISSN:2527-5941

vektorisasi kata menjadi 300 dimensi dengan menggunakan id.vec repositori milik github kyubyong, sebagai contoh kata "*halo*" direpresentasikan dalam vektor 300 dimensi menjadi:

halo = [X_0, ... ,X_{299}] (1)
halo = [-0.095851, ... , -0.38799] (2)

***Long Short-Term Memory* (LSTM)**

Long Short-Term Memory (LSTM) merupakan salah satu bagian dari Recurrent Neural Network (RNN) yang dirancang untuk mengatasi kelemahan umum model RNN, terutama dalam hal hubungan jangka panjang dalam data yang berurutan. RNN pada umumnya mengalami kesulitan mempertahankan informasi selama proses pelatihan karena *vanishing gradient* atau *exploding gradient*, yang berarti sinyal terlalu kecil atau terlalu besar saat melakukan aliran balik (*back flow*) (Hochreiter & Schmidhuber, 1997). LSTM mengatasi hal tersebut dengan struktur arsitektur bernama *memory cell* yang mampu mempertahankan aliran yang konstan dalam jaringan. *Memory cell* dibangun di atas struktur dasar yang disebut *Constant Error Carousel* (CEC), yaitu unit linear dengan koneksi diri tetap (Hochreiter & Schmidhuber, 1997).



Gambar 1. Struktur Algoritma LSTM

Pada **Gambar 1** menampilkan struktur dari algoritma LSTM yang terdiri dari beberapa komponen utama yang disebut *gate*, yaitu *forget gate, input gate,* dan *output gate*. Setiap *gate* memiliki fungsi untuk mengatur informasi yang masuk untuk disimpan, atau dilupakan pada *cell state*, sehingga jaringan dapat mempertahankan informasi penting dalam jangka panjang. Perhitungan pada LSTM dapat dirumuskan sebagai berikut:

1. *Forget Gate*
$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \quad (3)$$

2. *Input Gate*
$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (4)$$

3. *Candidate Cell State*
$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \quad (5)$$

4. *Cell State Update*
$$C_t = f_t.C_{t-1} + i_t.\tilde{C}_t \quad (6)$$

5. *Output Gate*
$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (7)$$

6. *Hidden State Update*
$$h_t = o_t.tanh(C_t) \quad (8)$$

Hasil dari *hidden state update* pada tiap *timestep* akan diteruskan sebagai *input* pada *timestep* berikutnya bersama dengan *input* baru x_{t+1}. Dengan begitu LSTM dapat memepertahankan informasi dari waktu ke waktu. Dalam penelitian ini, *hidden state* terakhir dari perhitungan LSTM akan digunakan sebagai *input* di *dense layer*, sedangkan *output* dari *dense layer* akan berupa skor *logit* untuk masing-masing kelas. *Dense layer* dirumuskan sebagai berikut:

$$a_i = \sum_{j=1}^{n} (W_{ji}.h_j) + b_i \quad (9)$$

Skor logit yang telah dihasilkan oleh *dense layer* akan diubah menjadi probabilitas setiap kelas menggunakan fungsi aktivasi *softmax* yang dirumuskan sebagai berikut:

$$\hat{y}_i = \frac{e^{z_i - max\,(z)}}{\sum_{j=1}^{K} e^{z_j - max\,(z)}} \quad (10)$$

HASIL DAN PEMBAHASAN

Implementasi *Pre-Processing Data*

Dataset yang digunakan peneliti memiliki tiga bagian yaitu *tag, patterns* dan *response*. Pada bagian *patterns* nantinya akan digunakan untuk melatih model karena berisi pertanyaan-pertanyaan yang akan dipelajari oleh model. Berikut merupakan 10 *dataset* awal yang akan ditampilkan dalam *pre-processing* penelitian ini:

Dataset Asli

	Patterns	Tag
0	kak bikin web dari wordpress range harganya dari berapa ke berapa kak ?	harga-web
1	kalok bikin website perusahaan berapa bang.	harga-web
2	Berapa biaya untuk membangun situs web?	harga-web
3	Brp nih biaya pembuatan website	harga-web
4	Bikin website brp bang	harga-web
5	Berapa biaya pengembangan aplikasi web?	harga-web
6	Berapa biaya yang akan Anda kenakan sesuai kebutuhan saya untuk Situs Web WordPress?	harga-web
7	Berapa biaya membuat situs web baru?	harga-web
8	mas,bikin website ppob untuk jualan pulsa & kouta hrga brpa	harga-web
9	bikin web buat travel agent budget brp ya om?	harga-web

Gambar 2. *Dataset* Awal

p-ISSN:2527-5321
e-ISSN:2527-5941

Beberapa tahapan dari implementasi *pre-processing* dalam penelitian ini adalah sebagai berikut:

1. Melakukan proses *lowercasing* yaitu, semua huruf dalam teks diubah menjadi huruf kecil.
2. *Punctuation removal* yaitu seluruh tanda baca yang ada pada kalimat (*patterns*) akan dihapus.
3. *Tokenization* yang bertujuan untuk memecah *patterns* menjadi kata-kata individual (*token*).
4. *Slang normalization*, karena *dataset* berasal dari *realcase* komentar sosial media, yang Sebagian besar menggunakan Bahasa yang tidak beraturan, maka peneliti menggunakan kamus *slang* yang telah dikompilasi sebelumnya untuk mengubah kata-kata tersebut ke bentuk baku.
5. *Word indexing,* pada proses ini nantinya akan menghasilkan daftar indeks yang mewakili setiap kata secara unik.
6. *Padding sequences*, Panjang kalimat pada *patterns* sangat bervariasi, maka *dataset* perlu menggunakan *pad_sequences* untuk menyamakan panjang seluruh kalimat. Padding ini dilakukan di bagian akhir indeks (*post-padding*) dengan menambahkan nilai 0 agar semua input memiliki dimensi yang sama.
7. Tahap akhir dalam *pre-processing* pada penelitian ini adalah transformasi kata ke dalam bentuk vektor numerik menggunakan *pre-trained Word Vector* bahasa Indonesia. Setiap kata direpresentasikan dalam bentuk vektor berdimensi 300 yang menangkap makna semantik dari kata tersebut.

Word Embedding (5 Dimensi Pertama)

	Kalimat ke	Kata	Vektor[0:5]
0	1	kak	-0.0049, -0.0633, 0.2994, 0.0439, -0.1926
1	1	bikin	-0.0787, -0.0444, 0.0765, -0.0795, -0.1217
2	1	web	-0.4114, 0.1972, 0.0423, -0.1658, 0.1706
3	1	dari	0.1078, 0.0719, 0.2299, 0.3597, -0.0692
4	1	wordpress	-0.2832, 0.0152, 0.0789, 0.0819, -0.1993
5	1	range	-0.0210, -0.0065, -0.1133, 0.1666, 0.2106
6	1	harganya	0.1029, -0.1140, 0.1066, 0.1774, 0.3091
7	1	dari	0.1078, 0.0719, 0.2299, 0.3597, -0.0692
8	1	berapa	-0.0556, -0.0261, 0.0268, 0.2048, 0.2039
9	1	ke	-0.0184, 0.0039, 0.0148, 0.1686, -0.4272
10	1	berapa	-0.0556, -0.0261, 0.0268, 0.2048, 0.2039
11	1	kak	-0.0049, -0.0633, 0.2994, 0.0439, -0.1926
12	2	kalok	
13	2	bikin	-0.0787, -0.0444, 0.0765, -0.0795, -0.1217
14	2	website	-0.2729, 0.2024, -0.0738, 0.0507, 0.1680
15	2	perusahaan	0.1329, -0.1034, 0.2925, -0.2434, 0.2043
16	2	berapa	-0.0556, -0.0261, 0.0268, 0.2048, 0.2039
17	2	bang	0.2013, 0.3076, 0.2264, 0.2371, 0.1512
18	3	berapa	-0.0556, -0.0261, 0.0268, 0.2048, 0.2039
19	3	biaya	-0.2195, 0.1577, 0.0396, -0.0495, 0.1773
20	3	untuk	0.0422, 0.1495, 0.1735, 0.0265, -0.0149
21	3	membangun	0.3294, 0.2269, -0.0832, 0.0150, -0.1476
22	3	situs	-0.2659, 0.1139, -0.1551, -0.2585, 0.1538
23	3	web	-0.4114, 0.1972, 0.0423, -0.1658, 0.1706
24	4	berapa	-0.0556, -0.0261, 0.0268, 0.2048, 0.2039
25	4	ini	-0.1049, -0.1332, 0.2374, 0.2643, 0.0326

Gambar 3. Hasil *Vectorization*

Arsitektur Jaringan LSTM

Dalam penelitian ini, jaringan LSTM dibentuk dari beberapa lapisan antara lain sebagai berikut:

1. Lapisan Penyematan
   tf.keras.layers.Embedding(input_dim=len(word_index) + 1, output_dim=embedding_dim, weights=[embedding_matrix], input_length=padded.shape[1], trainable=False).
   Lapisan ini akan memetakan setiap kata dalam data pelatihan menjadi vektor berdimensi 300. Lapisan ini menggunakan matriks *embedding* yang sudah dilatih sebelumnya, sehingga model mendapatkan makna dari kata-kata sebelum proses pelatihan dimulai. Vektor ini membantu model memahami hubungan antar kata dalam ruang vektor.
2. Lapisan LSTM
   tf.keras.layers.LSTM(64, return_sequences=True).
   Lapisan kedua adalah LSTM dengan 64 unit dan argumen *return_sequences=True*. Pada konfigurasi ini, lapisan LSTM nantinya akan menghasilkan output pada setiap *timestep*, sehingga urutan informasi tetap dipertahankan dan dapat diteruskan ke lapisan berikutnya.
3. Lapisan *Bi-Directional* LSTM
   tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)).
   Lapisan selanjutnya adalah *Bi-Directional* LSTM dengan 64 unit. Peneliti menambahkan lapisan ini

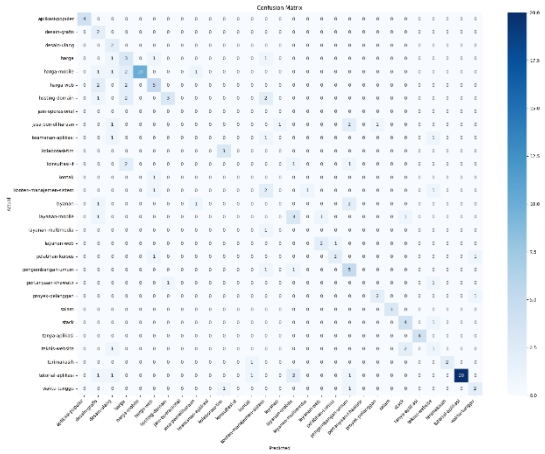
p-ISSN:2527-5321
e-ISSN:2527-5941

karena dataset yang digunakan berasal dari kondisi nyata yang mana polanya tidak beraturan sehingga model perlu pemahaman lebih dalam dengan lapisan ini, data teks akan diproses dari arah kiri ke kanan (*forward*) dan kanan ke kiri (*backward*) sehingga pemahaman model terhadap kalimat menjadi lebih menyeluruh.

4. Lapisan *Dropout* ***tf.keras.layers.Dropout(0.3).*** Lapisan keempat menggunakan *Dropout* dengan nilai 0,3 yang berarti *dropout* akan menonaktifkan 30% neuron secara acak pada saat pelatihan untuk mencegah overfitting.
5. Lapisan Keluaran ***tf.keras.layers.Dense(y.shape[1], activation='softmax').*** Lapisan terakhir adalah *Dense Layer* dengan jumlah unit sama dengan jumlah kelas target ***y.shape[1]*** menggunakan fungsi aktivasi *softmax*. Fungsi aktivasi ini akan menghasilkan distribusi probabilitas terhadap seluruh kelas, kelas dengan probabilitas tertinggi akan menjadi hasil klasifikasi model.

Hasil Pelatihan Model

Model dilatih menggunakan optimisasi *Adam* dengan *learning rate* sebesar 0,001. *loss function* yang digunakan adalah *categorical crossentropy* karena penelitian ini merupakan klasifikasi untuk multikelas. Proses pelatihan dilakukan selama 100 epoch dengan menggunakan batch sebanyak 16 data per iterasi.



Gambar 4. *Confusion Matrix*

Confusion Matrix menunjukkan bahwa beberapa kelas dapat dikenali dengan baik, seperti aplikasi-populer, harga-mobile, dan waktu-tunggu yang sebagian besar terklasifikasi benar. Namun, terdapat beberapa kelas yang masih salah prediksi, seperti konsultasi-it, jasa-pemeliharaan, dan keamanan-aplikasi, dan juga terdapat kelas yang tidak terdeteksi sama sekali seperti layanan-mobile, layanan-multimedia, dan kontak.

Classification Report:

	precision	recall	f1-score	support
aplikasi-populer	1.00	1.00	1.00	4
desain-grafis	0.22	1.00	0.36	2
desain-ulang	0.25	1.00	0.40	2
harga	0.27	0.50	0.35	6
harga-mobile	1.00	0.67	0.80	15
harga-web	0.56	0.56	0.56	9
hosting-domain	0.75	0.38	0.50	8
jam-operasional	0.00	0.00	0.00	0
jasa-pemeliharaan	0.00	0.00	0.00	5
keamanan-aplikasi	0.00	0.00	0.00	3
kolaborasi-tim	0.75	1.00	0.86	3
konsultasi-it	0.00	0.00	0.00	4
kontak	0.00	0.00	0.00	1
konten-manajemen-sistem	0.25	0.40	0.31	5
layanan	0.00	0.00	0.00	4
layanan-mobile	0.43	0.50	0.46	6
layanan-multimedia	0.00	0.00	0.00	1
layanan-web	0.67	0.67	0.67	3
pelatihan-kursus	0.67	0.50	0.57	4
pengembangan-umum	0.42	0.71	0.53	7
pertanyaan-khawatir	0.00	0.00	0.00	2
proyek-pelanggan	0.67	0.67	0.67	3
salam	1.00	1.00	1.00	3
stack	0.57	0.80	0.67	5
tanya-aplikasi	1.00	1.00	1.00	4
teknis-website	0.20	0.25	0.22	4
terimakasih	1.00	0.67	0.80	3
tutorial-aplikasi	1.00	0.77	0.87	26
waktu-tunggu	0.50	0.50	0.50	4
accuracy			0.58	146
macro avg	0.45	0.50	0.45	146
weighted avg	0.62	0.58	0.57	146

Final Accuracy: 57.53% | Loss: 3.3015

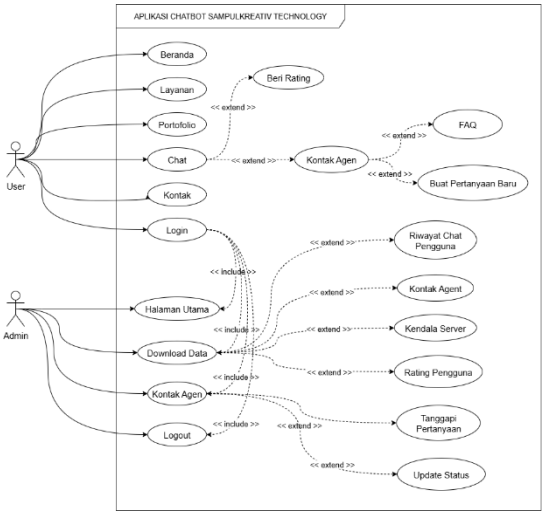
Gambar 5. *Classification Report*

Pada *classification report* nilai *precision, recall, dan f1-score* yang beragam pada tiap kelas. Beberapa kelas seperti aplikasi-populer dan harga-mobile mencapai skor 1,00, sedangkan ada beberapa kelas lain seperti konsultasi-it dan jasa-pemeliharaan tidak terdeteksi sama sekali dengan skor 0,00. Hasil pelatihan menunjukkan bahwa akurasi validasi hanya mencapai 57,53% dengan nilai loss validasi sekitar 3,28–3,30. Dengan hasil ini, dapat disimpulkan bahwa meskipun model mampu mengklasifikasikan sebagian kelas dengan baik, tetapi masih terdapat ketidakseimbangan antar kelas. Hal ini dikarenakan sumber *dataset realcase* pertanyaan seputar usaha IT yang digunakan peneliti sangat terbatas sehingga distribusi kelas menjadi tidak merata.

p-ISSN:2527-5321
e-ISSN:2527-5941

Perancangan Aplikasi

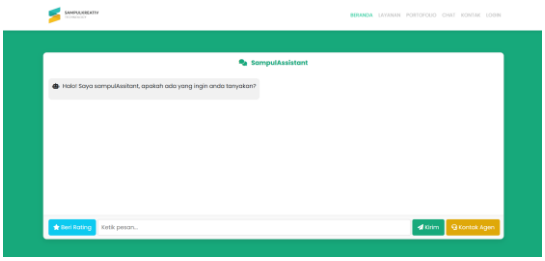
Dalam perancangan aplikasi, peneliti membagi menjadi 2 halaman antara lain halaman *user* dan *admin* yang masing-masing terdiri dari beberapa menu seperti halaman utama, *chat*, dan lainnya. Gambaran *use case* program aplikasi ditunjukkan pada diagram berikut:



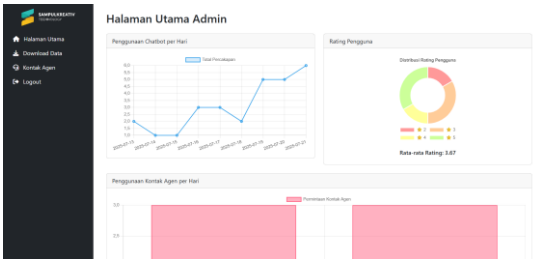
Gambar 6. *Use case Diagram*

Berikut tampilan program aplikasi pada beberapa menu:

Gambar 7. Halaman Utama *User*



Gambar 8. Halaman *Chat User*



Gambar 9. Halaman Utama *Admin*

Gambar 10. Halaman Kontak Agen *Admin*

File-file yang didapatkan dari hasil *pre-processing* kemudian digunakan dalam aplikasi untuk mendukung proses *chatbot*, yang memiliki alur sebagai berikut:

1. *Load resource*: *chatbot_model.h5*, *tokenizer.pkl*, *label_encoder.pkl*, *intents-real.json*, dan *slang.txt*.
2. Menerima *input* teks dari aplikasi.
3. Melakukan tahapan *pre-processing input* teks (*lowercase*, *punctuation remove*, tokenisasi, normalisasi *slang*).
4. Konversi teks ke *sequence* lalu melakukan *padding*.
5. Prediksi intent dari input yang telah di *pre-process* menggunakan model.
6. Ambil jawaban dari *responses* pada *dataset intents* sesuai intent yang diprediksi.
7. Kembalikan hasil dalam format JSON ke aplikasi.
8. Jawaban tampil pada aplikasi

SIMPULAN DAN SARAN

Model yang dikembangkan menunjukkan performa yang cukup dalam memahami pertanyaan pengguna dengan akurasi sebesar 57.53% yang dapat memahami Bahasa Indonesia formal maupun informal. Meskipun ukuran dataset masih terbatas, hasil ini menunjukkan bahwa pendekatan LSTM memiliki potensi besar untuk dikembangkan lebih lanjut dalam mengenali intensi pengguna secara otomatis. Peneliti selanjutnya disarankan untuk memperluas cakupan *datataset* agar model dapat mengenali lebih banyak variasi pola kalimat dan intensi pengguna untuk meningkatkan performa model ke depannya.

UCAPAN TERIMA KASIH

Penulis menyampaikan terima kasih kepada mas M. Firas Faishal, kang Hasan Al Farisi dan PT. Sampul Kreatif Teknologi, atas kesempatan serta bimbingan yang diberikan

p-ISSN:2527-5321
e-ISSN:2527-5941

dalam proses penelitian ini dan juga penulis sangat berterima kasih kepada kedua orang tua penulis atas segala doa, dukungan, dan motivasi yang diberikan.

DAFTAR PUSTAKA

Babulak, E. (2023). Introductory Chapter: Journey to AI Driven Chatbots. In E. Babulak (Ed.), *Chatbots - The AI-Driven Front-Line Services for Customers*. IntechOpen. https://doi.org/10.5772/intechopen.112461

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kyubyong, P. (2016, December 21). *Wordvectors*. https://github.com/Kyubyong/wordvectors

Owen, L. (2020, March 31). *NLP_bahasa_resources*. https://github.com/louisowen6/NLP_bahasa_resources

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, *9*(1), 36–45. https://doi.org/10.1145/365153.365168

Biografi Penulis

Maeru Bagas Trisoko, Pendidikan S1 Universitas Indraprasta PGRI, program studi Teknik Informatika, saat ini sedang fokus pada penelitian yang menggunakan Machine Learning dan Deep Learning.

Ni Wayan Parwati Septiani, M.M, M.Kom.
NIDN : 0323098401
wayan.parwati@gmail.com