

PENGENALAN EMOSI DALAM TEKS DENGAN ALGORITME LONG SHORT TERM MEMORY

TUGAS AKHIR

Daniel Christianto
1118007



INSTITUT
TEKNOLOGI
HARAPAN
BANGSA

Veritas vos liberabit

PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2022

PENGENALAN EMOSI DALAM TEKS DENGAN ALGORITME LONG SHORT TERM MEMORY

TUGAS AKHIR

**Diajukan sebagai salah satu syarat untuk memperoleh
gelar sarjana dalam bidang Informatika**

**Daniel Christiano
1118007**



**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2022**

HALAMAN PERNYATAAN ORISINALITAS

**Saya menyatakan bahwa Tugas Akhir yang saya susun ini
adalah hasil karya saya sendiri.**

**Semua sumber yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

**Saya bersedia menerima sanksi pencabutan gelar akademik
apabila di kemudian hari Tugas Akhir ini terbukti plagiat.**

Bandung, 25 Juni 2022



**Daniel Christianto
1118007**

HALAMAN PENGESAHAN TUGAS AKHIR

Tugas Akhir dengan judul:

PENGENALAN EMOSI DALAM TEKS DENGAN ALGORITME LONG SHORT TERM MEMORY

yang disusun oleh:

Daniel Christianto

1118007

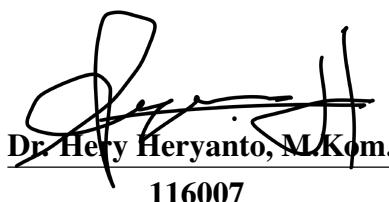
telah berhasil dipertahankan di hadapan Dewan Penguji Sidang Tugas Akhir yang dilaksanakan pada:

Hari / tanggal : Sabtu, 25 Juni 2022

Waktu : 08:30 WIB

Menyetujui

Pembimbing Utama:



Dr. Hery Heryanto, M.Kom.
116007

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Institut Teknologi Harapan Bangsa, saya yang bertanda tangan di bawah ini:

Nama : Daniel Christianto

NIM : 1118007

Program Studi : Informatika

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Harapan Bangsa **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Rights)** atas karya ilmiah saya yang berjudul:

PENGENALAN EMOSI DALAM TEKS DENGAN ALGORITME LONG SHORT TERM MEMORY

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Harapan Bangsa berhak menyimpan, mengalihmediakan, mengelola dalam pangkalan data, dan memublikasikan karya ilmiah saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Bandung, 25 Juni 2022

Yang menyatakan



Daniel Christianto

ABSTRAK

Nama : Daniel Christianto
Program Studi : Informatika
Judul : Pengenalan Emosi dalam Teks dengan Algoritme *Long Short Term Memory*

Pengenalan emosi dalam teks merupakan topik penelitian yang banyak dilakukan beberapa tahun terakhir. Berbagai penelitian telah dilakukan untuk menyelesaikan masalah pengenalan emosi dari teks. Jumlah data teks yang semakin meningkat serta diaplikasikan pada berbagai bidang telah membuat penelitian pengenalan emosi berkembang pesat termasuk dalam penelitian ini. Penelitian ini menguji akurasi menggunakan metode LSTM dan menguji pengaruh parameter terhadap akurasi deteksi emosi dalam teks.

Terdapat 3 indikator utama dalam penelitian ini, yaitu jumlah *emotion dimension & embedding weights*, *learning rate*, dan *dropout rate*. Dataset yang digunakan dalam penelitian ini, yaitu dataset dari *Computational Linguistics at Concordia* (CLaC) Lab yang memiliki jumlah data sebanyak 30.160 yang memiliki fitur *text* dan label. Pada penelitian ini dilakukan klasifikasi terhadap 4 kelas, yaitu *others*, *happy*, *sad*, dan *angry*.

Berdasarkan pengujian jumlah *emotion dimension & embedding weights*, *learning rate*, dan *dropout rate*, didapatkan nilai akurasi terbaik untuk setiap kelas yang ada pada penelitian deteksi emosi dalam teks. Pada kelas *others* didapatkan nilai akurasi tertinggi sebesar 82%, pada kelas *happy* didapatkan nilai akurasi tertinggi sebesar 93%, pada kelas *sad* didapatkan nilai akurasi tertinggi sebesar 91%, dan pada kelas *angry* didapatkan nilai akurasi tertinggi sebesar 94%.

Kata kunci: *Deep Learning*, *Long Short Term Memory* (LSTM), *Natural Language Processing* (NLP), *GloVe* (*Global Vector*), *Emotion Detection*.

ABSTRACT

Name : Daniel Christiano
Department : Informatics
Title : Emotion Recognition in Text with Long Short Term Memory Algorithm

Recognition of emotions in text is a topic of research that has been widely carried out in recent years. Various studies have been carried out to solve the problem of recognizing emotions from texts. The increasing amount of text data that can be applied to various fields has made emotion recognition research develop rapidly, including in this study. This study tested the accuracy using the LSTM method and tested the effect of parameters on the accuracy of emotion detection in the text.

There are 3 main indicators in this study, namely the number of emotion dimensions & embedding weights, learning rate, and dropout rate. The dataset used in this study is a dataset from the Computational Linguistics at Concordia (CLaC) Lab which has a total of 30160 data that has text and label features. In this study, 4 classes will be classified, namely others, happy, sad, and angry.

Based on testing the number of emotion dimensions & embedding weights, learning rate, and dropout rate, it was found that the best accuracy value for each class in the emotion detection research in text was obtained. In the others class the highest accuracy value is 82%, in the happy class the highest accuracy value is 93%, in the sad class the highest accuracy value is 91%, and in the angry class the highest accuracy value is 94%.

Keywords: Deep Learning, Long Short Term Memory (LSTM), Natural Language Processing (NLP), GloVe (Global Vector), Emotion Detection.

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan yang Maha Esa karena rahmat dan bimbingan-Nya penulis menyelesaikan Tugas Akhir yang berjudul "Pengenalan Emosi Dalam Teks Dengan Algoritme Long Short Term Memory". Laporan ini disusun sebagai salah satu syarat kelulusan di Institut Teknologi Harapan Bangsa. Pada kesempatan ini penulis menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan yang Maha Esa, karena oleh-Nya penulis selalu mendapatkan pengharapan dan penguatan untuk menyelesaikan tugas akhir ini.
2. Bapak Dr. Hery Heryanto, M.Kom., selaku pembimbing utama Tugas Akhir yang senantiasa memberi saran, ilmu dan dukungan kepada penulis selama proses pembuatan Tugas Akhir.
3. Bapak Ventje Jeremias Lewi Engel, M.T., CEH selaku penguji satu yang telah memberikan ilmu, masukan dan dukungan dalam menyelesaikan Laporan Tugas Akhir.
4. Seluruh dosen dan staf Departemen Informatika ITHB yang telah membantu dalam menyelesaikan Laporan Tugas Akhir ini.
5. Seluruh staf dan karyawan ITHB yang turut membantu kelancaran dalam menyelesaikan Laporan Tugas Akhir ini.
6. Orang tua dan kakak yang selalu menyediakan waktu untuk memberikan doa, semangat dan dukungan kepada penulis untuk menyelesaikan Laporan Tugas Akhir.
7. Hanjaya Suryalim, Andreas Aditya, Raffi Verrel Alessandro, Benedict Reydo Bayuputra, dan Daniel Alexander yang sudah mendukung dan memberikan semangat serta berjuang bersama untuk menyelesaikan Tugas Akhir.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan waktu dan pengetahuan yang dimiliki oleh penulis. Oleh karena itu, kritik dan saran untuk membangun kesempurnaan tugas akhir ini sangat diharapkan. Semoga dengan adanya tugas akhir ini membantu pihak yang membutuhkannya.

Bandung, 25 Juni 2022
Hormat penulis,

A handwritten signature in black ink, appearing to read "Daniel Christianto".

Daniel Christianto

DAFTAR ISI

ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xvi
BAB 1 PENDAHULUAN	1-1
1.1 Latar Belakang	1-1
1.2 Rumusan Masalah	1-2
1.3 Tujuan Penelitian	1-2
1.4 Batasan Masalah	1-3
1.5 Konstribusi Penelitian	1-3
1.6 Metodologi Penelitian	1-3
1.7 Sistematika Pembahasan	1-4
BAB 2 LANDASAN TEORI	2-1
2.1 Tinjauan Pustaka	2-1
2.1.1 <i>Natural Language Processing</i>	2-1
2.1.1.1 <i>Corpus</i>	2-2
2.1.1.2 <i>Tokenize</i>	2-2
2.1.1.3 <i>Stopwords Removal</i>	2-2
2.1.1.4 <i>Stemming</i>	2-2
2.1.1.5 <i>Converting Into Numerical Form</i>	2-2
2.1.2 <i>Text Classification</i>	2-3
2.1.3 <i>Sentiment Analysis</i>	2-3
2.1.4 <i>Artificial Neural Network</i>	2-4
2.1.5 <i>Word Embedding</i>	2-5
2.1.6 <i>Global Vector (GloVe)</i>	2-5
2.1.7 <i>Recurrenct Neural Network (RNN)</i>	2-6
2.1.8 <i>Long Short Term Memory (LSTM)</i>	2-7

2.1.8.1	<i>Forget Gate</i>	2-8
2.1.8.2	<i>Input Gate</i>	2-9
2.1.8.3	<i>Output Gate</i>	2-10
2.1.8.4	<i>Cell State</i>	2-11
2.1.8.5	<i>Hidden State</i>	2-11
2.1.9	<i>Emotion Detection in Text</i>	2-12
2.1.10	<i>Dropout</i>	2-15
2.1.11	<i>Dense/Fully Connected Layer</i>	2-16
2.1.12	<i>Adaptive Moment Estimation Optimizer (Adam)</i>	2-16
2.1.13	<i>Softmax</i>	2-18
2.1.14	<i>Categorical Cross-Entropy</i>	2-19
2.1.15	<i>Confusion Matrix</i>	2-19
2.2	Pustaka Python	2-21
2.2.1	Pandas	2-21
2.2.2	<i>Numerical Python (NumPy)</i>	2-22
2.2.3	Matplotlib	2-22
2.2.4	<i>Wordcloud</i>	2-24
2.2.5	<i>Regular Expression (RegEx)</i>	2-24
2.2.6	<i>String</i>	2-25
2.2.7	<i>Natural Language Toolkit (NLTK)</i>	2-25
2.2.7.1	<i>Stopwords</i>	2-26
2.2.7.2	<i>Tokenizer</i>	2-26
2.2.8	<i>Datasets</i>	2-26
2.2.9	Keras	2-27
2.2.9.1	<i>Preprocessing</i>	2-27
2.2.9.2	<i>Utils</i>	2-28
2.2.9.3	<i>Layers</i>	2-28
2.3	Tinjauan Studi	2-30
2.4	Tinjauan Objek	2-34

BAB 3 ANALISIS DAN PERANCANGAN SISTEM	3-1	
3.1	Analisis Masalah	3-1
3.2	Kerangka Pemikiran	3-1
3.3	Urutan Proses Global	3-3
3.4	Analisis Manual	3-5
3.4.1	<i>Pengambilan Dataset</i>	3-5
3.4.2	<i>Preprocessing</i>	3-5
3.4.2.1	<i>Case Folding</i>	3-6

3.4.2.2	<i>Remove Number & Symbol</i>	3-6
3.4.2.3	<i>Transform Negation</i>	3-7
3.4.2.4	<i>Spellcheck</i>	3-7
3.4.2.5	<i>Remove Stopwords</i>	3-8
3.4.3	<i>Tokenizer</i>	3-8
3.4.3.1	<i>Text to Sequence</i>	3-8
3.4.3.2	<i>Padding & Sequence</i>	3-9
3.4.4	<i>Word Embedding GloVe</i>	3-9
3.4.5	<i>Long Short Term Memory (LSTM)</i>	3-10
3.4.6	<i>Timestep Pertama</i>	3-11
3.4.6.1	<i>Forget Gate</i>	3-11
3.4.6.2	<i>Input Gate</i>	3-12
3.4.6.3	<i>Output Gate</i>	3-13
3.4.6.4	<i>Cell State</i>	3-14
3.4.6.5	<i>Hidden State</i>	3-14
3.4.7	<i>Timestep Kedua</i>	3-15
3.4.7.1	<i>Forget Gate</i>	3-15
3.4.7.2	<i>Input Gate</i>	3-16
3.4.7.3	<i>Output Gate</i>	3-17
3.4.7.4	<i>Cell State</i>	3-18
3.4.7.5	<i>Hidden State</i>	3-18
3.4.8	<i>Dropout Layer</i>	3-19
3.4.9	<i>Dense Layer</i>	3-20
3.4.10	<i>Output Layer dengan Aktivasi Softmax</i>	3-20
3.4.11	<i>Categorical Cross-Entropy</i>	3-21
3.4.12	<i>Confusion Matrix</i>	3-22

BAB 4 IMPLEMENTASI DAN PENGUJIAN 4-1

4.1	Lingkungan Implementasi	4-1
4.1.1	Spesifikasi Perangkat Keras	4-1
4.1.2	Spesifikasi Perangkat Lunak	4-1
4.2	Implementasi Perangkat Lunak	4-1
4.2.1	Daftar <i>Class</i> dan <i>Method</i>	4-2
4.2.1.1	<i>Class Dataset</i>	4-2
4.2.1.2	<i>Class Preprocessing</i>	4-2
4.2.1.3	<i>Class TokenizerWord</i>	4-3
4.2.1.4	<i>Class EmbeddingGlove</i>	4-4
4.2.1.5	<i>Class GenerateModel</i>	4-5

4.2.2	Penggunaan Google Colaboratory	4-7
4.2.3	Implementasi Penggunaan <i>Dataset</i>	4-7
4.2.4	Implementasi Penggunaan <i>Pretrained GloVe</i>	4-8
4.2.5	Implementasi Aplikasi	4-8
4.3	Pengujian	4-9
4.3.1	Pengujian Arsitektur	4-10
4.3.2	Pengujian <i>Embedding Output Dimension & Embedding Weights</i>	4-11
4.3.3	Pengujian <i>Dropout Rate</i>	4-11
4.3.4	Pengujian Nilai <i>Learning Rate</i>	4-12
4.4	Hasil Pengujian	4-12
4.4.1	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i>	4-12
4.4.1.1	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-12
4.4.1.2	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-15
4.4.1.3	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-17
4.4.1.4	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-20
4.4.2	Hasil Pengujian 100 <i>Embedding Dimension</i>	4-22
4.4.2.1	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-22
4.4.2.2	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-25
4.4.2.3	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-27
4.4.2.4	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-30
4.4.3	Hasil Pengujian 200 <i>Embedding Dimension</i>	4-32
4.4.3.1	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-32
4.4.3.2	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-35
4.4.3.3	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-37

4.4.3.4	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-40
4.4.4	Hasil Pengujian 300 <i>Embedding Dimension</i>	4-42
4.4.4.1	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-42
4.4.4.2	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-45
4.4.4.3	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-47
4.4.4.4	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-50
4.4.5	Hasil Pengujian Keseluruhan	4-52
4.4.5.1	Hasil Pengujian Keseluruhan Pada Kelas <i>Others</i> .	4-52
4.4.5.2	Hasil Pengujian Keseluruhan Pada Kelas <i>Happy</i> .	4-53
4.4.5.3	Hasil Pengujian Keseluruhan Pada Kelas <i>Sad</i> . .	4-55
4.4.5.4	Hasil Pengujian Keseluruhan Pada Kelas <i>Angry</i> .	4-56
4.4.5.5	Hasil Pengujian Keseluruhan <i>Precision & Recall</i> .	4-57
4.4.6	Hasil Pengujian Model Tanpa Kelas <i>Others</i>	4-58
4.4.7	Hasil Pengujian Model Tanpa <i>Dropout</i>	4-59
4.4.8	Pembahasan Umum Hasil Pengujian	4-60
4.5	Analisis Kesalahan	4-63
BAB 5	KESIMPULAN DAN SARAN	5-1
5.1	Kesimpulan	5-1
5.2	Saran	5-2

DAFTAR TABEL

2.1	Tabel <i>Confusion Matrix</i>	2-20
2.2	Tabel <i>Library Pandas</i>	2-21
2.3	Tabel <i>Library NumPy</i>	2-22
2.4	Tabel <i>Library Matplotlib</i>	2-23
2.5	Tabel <i>Library Wordcloud</i>	2-24
2.6	Tabel <i>Library RegEx</i>	2-25
2.7	Tabel <i>Library String</i>	2-25
2.8	Tabel <i>Library Stopwords</i>	2-26
2.9	Tabel <i>Library Tokenizer</i>	2-26
2.10	Tabel <i>Library Datasets</i>	2-27
2.11	Tabel <i>Library Preprocessing</i>	2-27
2.12	Tabel <i>Library Utils</i>	2-28
2.13	Tabel <i>Library Layers</i>	2-29
2.14	Tabel <i>State of the Art</i>	2-30
2.15	Tabel Fitur <i>Dataset</i>	2-34
2.16	Tabel Contoh Teks Dengan Label <i>Others</i>	2-35
2.17	Tabel Contoh Teks Dengan Label <i>Happy</i>	2-35
2.18	Tabel Contoh Teks Dengan Label <i>Sad</i>	2-36
2.19	Tabel Contoh Teks Dengan Label <i>Angry</i>	2-37
3.1	Contoh Teks	3-6
3.2	Contoh Proses <i>Case Folding</i> pada Teks	3-6
3.3	Contoh <i>Case Folding</i> pada Teks	3-7
3.4	Contoh Penjabaran Kata Negasi pada Teks	3-7
3.5	Contoh <i>Transform Negation</i> pada Teks	3-7
3.6	Contoh <i>Spellcheck</i> pada Teks	3-8
3.7	Contoh <i>Stopwords</i> pada Teks	3-8
3.8	Contoh Tokenisasi pada Teks	3-8
3.9	Contoh <i>Text to Sequence</i> pada Teks	3-9
3.10	Contoh <i>Padding & Sequence</i>	3-9
3.11	Contoh <i>Word Embedding</i>	3-9
3.12	Contoh <i>Output Testing</i> Dibandingkan dengan <i>Class Aslinya</i>	3-22
3.13	Contoh <i>Confusion Matrix</i> 4×5 dengan <i>4 Class</i>	3-23
3.14	<i>Classification Report</i> dengan <i>4 Class</i>	3-25

4.1	Daftar <i>method</i> pada <i>Class Dataset</i>	4-2
4.2	Daftar <i>method</i> pada <i>Class Preprocessing</i>	4-2
4.3	Daftar <i>method</i> pada <i>Class TokenizerWord</i>	4-3
4.4	Daftar <i>method</i> pada <i>Class EmbeddingGlove</i>	4-4
4.5	Daftar <i>method</i> pada <i>Class GenerateModel</i>	4-5
4.6	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-13
4.7	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-15
4.9	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-18
4.11	Hasil Pengujian 50 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-20
4.12	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-23
4.13	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-25
4.14	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-28
4.15	Hasil Pengujian 100 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-30
4.16	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-33
4.17	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-35
4.18	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-38
4.19	Hasil Pengujian 200 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-40
4.20	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Others</i>	4-43
4.21	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Happy</i>	4-45
4.22	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Sad</i>	4-48
4.23	Hasil Pengujian 300 <i>Embedding Dimension & Embedding Weights</i> Pada Kelas <i>Angry</i>	4-50

4.24	Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas <i>Others</i>	.4-52
4.25	Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas <i>Happy</i>	.4-54
4.26	Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas <i>Sad</i>	. . .4-55
4.27	Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas <i>Angry</i>	. .4-56
4.28	Hasil Pengujian Model Tanpa Kelas <i>Others</i>4-58

DAFTAR GAMBAR

2.1	Ilustrasi Arsitektur <i>Neural Network</i> [9]	2-4
2.2	Visualisasi GloVe [10]	2-6
2.3	Ilustrasi Arsitektur <i>Recurrent Neural Network</i> [12]	2-7
2.4	Ilustrasi Arsitektur LSTM [6]	2-8
2.5	Ilustrasi <i>Dropout Neural Net Model</i> [15]	2-15
2.6	<i>Dataset HuggingFace - Emo</i> (https://huggingface.co/datasets/viewer?dataset=emo)	2-34
3.1	Kerangka Pemikiran	3-2
3.2	<i>Flowchart Emotion Detection in Text</i>	3-4
3.3	<i>Flowchart Preprocessing</i>	3-6
3.4	Ilustrasi LSTM Perhitungan Manual	3-11
4.1	Persebaran Data Setiap Kelas	4-7
4.2	Ilustrasi Vektor dari Penggunaan <i>Pretrained GloVe</i>	4-8
4.3	Tampilan Utama Aplikasi	4-9
4.4	Tampilan Keluaran Aplikasi	4-9
4.5	Rancangan Arsitektur dalam Penelitian	4-10
4.6	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 50 Pada Kelas <i>Others</i>	4-14
4.7	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 50 Pada Kelas <i>Happy</i>	4-16
4.8	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 50 Pada Kelas <i>Sad</i>	4-19
4.9	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 50 Pada Kelas <i>Angry</i>	4-21
4.10	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 100 Pada Kelas <i>Others</i>	4-24
4.11	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 100 Pada Kelas <i>Happy</i>	4-26
4.12	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 100 Pada Kelas <i>Sad</i>	4-29
4.13	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 100 Pada Kelas <i>Angry</i>	4-31
4.14	Visualisasi Plot <i>Precision, Recall, F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 200 Pada Kelas <i>Others</i>	4-34

4.15 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 200 Pada Kelas <i>Happy</i>	4-36
4.16 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 200 Pada Kelas <i>Sad</i>	4-39
4.17 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 200 Pada Kelas <i>Angry</i>	4-41
4.18 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 300 Pada Kelas <i>Others</i>	4-44
4.19 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 300 Pada Kelas <i>Happy</i>	4-46
4.20 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 300 Pada Kelas <i>Sad</i>	4-49
4.21 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> dengan Dimensi 300 Pada Kelas <i>Angry</i>	4-51
4.22 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> Terbaik Untuk Setiap Dimensi Pada Kelas <i>Others</i>	4-53
4.23 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> Terbaik Untuk Setiap Dimensi Pada Kelas <i>Happy</i>	4-54
4.24 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> Terbaik Untuk Setiap Dimensi Pada Kelas <i>Sad</i>	4-56
4.25 Visualisasi Plot <i>Precision</i> , <i>Recall</i> , <i>F1 Score</i> , dan <i>Accuracy</i> Terbaik Untuk Setiap Dimensi Pada Kelas <i>Angry</i>	4-57
4.26 Perbandingan Visualisasi Plot <i>Accuracy</i> Model Dengan Kelas <i>Others</i> & Tanpa Kelas <i>Others</i>	4-59
4.27 Perbandingan Visualisasi Plot <i>Accuracy</i> Model Dengan <i>Dropout</i> & Tanpa <i>Dropout</i>	4-60

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Affective computing merupakan bidang penelitian yang bertujuan untuk mengembangkan sistem komputer yang mampu mendeteksi, menanggapi, dan meniru emosi manusia. Pengenalan emosi merupakan kemampuan utama dan memiliki peran penting dalam *affective computing* [1]. Pengenalan emosi dilakukan berdasarkan ekspresi wajah, suara, gerak tubuh, sinyal electroencephalogram – EEG, dan teks [2].

Pengenalan emosi dalam teks merupakan topik penelitian yang banyak dilakukan beberapa tahun terakhir. Berbagai penelitian telah dilakukan untuk menyelesaikan masalah pengenalan emosi dari teks. Jumlah data teks yang semakin meningkat serta diaplikasikan pada berbagai bidang telah membuat penelitian pengenalan emosi berkembang pesat [4]. Seiring dengan banyaknya data teks yang tersedia menyebabkan pendekatan menggunakan *deep learning* diaplikasikan karena jumlah data yang besar [3].

Pendekatan *deep learning*, seperti *Recurrent Neural Network* (RNN) sangat cocok digunakan untuk memprediksi dengan data sekuensial karena dalam setiap pemrosesan yang dilakukan dalam RNN, *output* yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tapi juga berdasarkan *state internal* yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya [5][6]. Tetapi RNN memiliki kelemahan pada masalah *vanishing gradient* saat memproses data sekuensial yang panjang sehingga mengurangi akurasi dari suatu prediksi pada RNN [5][6].

Long Short Term Memory (LSTM) merupakan salah satu jenis RNN. LSTM menyimpan informasi terhadap pola-pola pada data. LSTM mempelajari data mana saja yang disimpan dan data mana saja yang dibuang, karena pada setiap neuron LSTM memiliki beberapa *gates* yang mengatur memori pada setiap neuron itu sendiri [5][6]. LSTM juga menambahkan *memory cell* yang menyimpan informasi untuk jangka waktu yang lama dan LSTM diusulkan sebagai solusi untuk mengatasi terjadinya *vanishing gradient* pada RNN saat memproses data sekuensial yang panjang sehingga metode LSTM ini memecahkan solusi *vanishing gradient* dalam RNN dan hasil prediksi dari model LSTM menjadi lebih akurat dengan nilai akurasi sebesar 85% [5][6]. Selain itu dalam melakukan pengenalan emosi, teks harus diubah menjadi bahasa yang dikenali oleh komputer. Oleh

karena itu, teks seperti kata atau dokumen diubah menjadi vektor dalam pemrosesan bahasa alami yang disebut *word embedding* [7]. *Word embedding* membantu memberikan sebuah bobot pada suatu data yang menyebabkan data tersebut memiliki sebuah nilai dan dampak dari penggunaan *word embedding* adalah meningkatnya akurasi pada saat pelatihan model. Pada penelitian [6] dilihat model yang menggunakan *word embedding* mendapat nilai akurasi sebesar 95.17%. Penelitian ini memilih metode LSTM untuk diterapkan dalam pengenalan emosi untuk data teks dengan harapan memberikan hasil akurasi yang lebih tinggi dibandingkan beberapa metode sebelumnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, maka diidentifikasi rumusan masalah sebagai berikut:

1. Berapa nilai akurasi yang dihasilkan oleh penerapan algoritme LSTM dalam pengenalan emosi untuk data teks?
2. Parameter apa saja yang berpengaruh dalam penerapan algoritme LSTM pada pengenalan emosi untuk data teks?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah diatas, tujuan penelitian ini adalah sebagai berikut:

1. Menguji akurasi dalam algoritme LSTM pada pengenalan emosi dalam teks.
2. Mengidentifikasi parameter apa saja yang berpengaruh dalam penerapan algoritme LSTM pada pengenalan emosi untuk data teks.

1.4 Batasan Masalah

Dalam penelitian ini, peneliti membatasi masalah yang diteliti antara lain:

1. Bahasa yang digunakan dalam dataset adalah Bahasa Inggris.
2. Pengenalan emosi hanya dilakukan pada level kalimat.
3. Jenis emosi yang digunakan yaitu *happy*, *sad*, *angry*, dan *others*.

1.5 Kontribusi Penelitian

Penelitian ini dilakukan untuk mendeteksi emosi yang terkandung dalam teks dengan algoritme LSTM dengan hasil akhir berupa pemberian kelas pada teks. Penelitian ini juga memberikan hasil dari akurasi sebagai penilaian dari model yang dibuat menggunakan LSTM.

1.6 Metodologi Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

Penulisan ini dimulai dengan studi kepustakaan yang bersumber dari jurnal penelitian terkait dengan topik yang ada.

2. Pengambilan Sampel Data

Data yang digunakan untuk penelitian ini berupa dataset teks berbahasa Inggris dengan label emosinya.

3. Analisis Masalah

Pada tahap ini dilakukan analisis permasalahan yang ada, batasan yang dimiliki, dan kebutuhan yang diperlukan.

4. Perancangan dan Implementasi

Pada tahap ini dilakukan pembangunan model *deep learning* dengan algoritme LSTM untuk menyelesaikan masalah pada rumusan masalah.

5. Pengujian

Pada tahap ini dilakukan pengujian terhadap hasil pengenalan emosi yang dibuat oleh model *deep learning* dengan algoritme LSTM dan melihat nilai akurasi dari hasil uji coba.

6. Dokumentasi

Pada tahap ini dilakukan pendokumentasian hasil analisis dan implementasi secara tertulis dalam bentuk laporan metode penelitian.

BAB 1 PENDAHULUAN

1.7 Sistematika Pembahasan

Pada penelitian ini peneliti menyusun berdasarkan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penelitian, metodologi penelitian serta sistematika pembahasan.

BAB II LANDASAN TEORI

Bab ini berisi penjelasan dasar mengenai teori yang mendukung implementasi penelitian ini.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisis algoritme *deep learning* dengan algoritme LSTM untuk membangun sistem pengenalan emosi dalam teks.

BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi implementasi dan pengujian dari model *deep learning* dengan metode LSTM dan melihat nilai akurasi dari hasil uji coba.

BAB V KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian yang dilakukan berdasarkan hasil dari pengujian serta saran untuk penelitian lebih lanjut di masa mendatang.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan dasar teori yang digunakan dalam penelitian ini. Teori-teori tersebut merupakan teori penunjang yang diambil dari jurnal terkait yang digunakan dalam proses penelitian tugas akhir ini.

2.1 Tinjauan Pustaka

Dalam sub bab ini dijelaskan beberapa teori yang menjadi dasar dalam penelitian ini. Pembahasan teori ini dimulai dari penjelasan *machine learning*, *natural language processing*, *text classification*, *sentiment analysis*, *artificial neural network*, *LSTM*, *Word Vector Representation*, dan *Emotion Detection in Text*. Penjelasan mengenai teori-teori tersebut dijelaskan sebagai berikut.

2.1.1 *Natural Language Processing*

Area yang berfokus pada pembuatan mesin yang mempelajari dan memahami data tekstual untuk melakukan beberapa tugas yang berguna dikenal sebagai *Natural Language Processing* (NLP). Data teks bisa terstruktur atau tidak terstruktur sehingga diperlukan untuk menerapkan beberapa langkah agar analisinya siap. NLP sudah menjadi kontributor besar untuk beberapa aplikasi. Ada banyak aplikasi NLP yang banyak digunakan oleh bisnis saat ini seperti *chatbot*, pengenalan suara, terjemahan bahasa, sistem rekomendasi, deteksi spam, dan analisis sentimen [8].

Tidak ada cara yang tepat untuk melakukan analisis NLP karena seseorang menjelajahi berbagai cara dan mengambil pendekatan berbeda untuk menangani data teks. Namun, dari sudut pandang *machine learning*, ada lima langkah utama yang harus diambil untuk membuat data teks siap dianalisis. Lima langkah utama yang terlibat dalam NLP adalah [8]:

1. *Reading the corpus*
2. *Tokenization*
3. *Cleaning / Stopwords removal*
4. *Stemming*
5. *Converting into numerical form*

2.1.1.1 *Corpus*

Korpus merupakan kumpulan teks yang tersimpan secara elektronik atau seluruh koleksi dokumen teks untuk berbagai kebutuhan penyelidikan, penelitian, dan pemrosesan bahasa alami. Pembuatan korpus pada awalnya dengan mengkompilasi teks asli ke dalam komputer. Misalnya, kita memiliki ribuan email dalam kumpulan yang perlu kita proses dan analisis untuk digunakan. Kelompok email ini dikenal sebagai corpus karena berisi semua dokumen teks. Langkah selanjutnya dalam pemrosesan teks adalah *tokenization* [8].

2.1.1.2 *Tokenize*

Metode membagi kalimat atau kumpulan kata yang diberikan dari dokumen teks menjadi kata-kata terpisah / individual dikenal sebagai *tokenization*. *Tokenization* menghapus karakter yang tidak perlu seperti tanda baca dan membagi kalimat menjadi kumpulan kata / token yang nantinya digunakan untuk diproses selanjutnya. Token dalam NLP sering dimaknai dengan “sebuah kata”, walau tokenisasi juga bisa dilakukan ke kalimat atau paragraf. Langkah selanjutnya dalam pemrosesan teks adalah *stopwords* [8].

2.1.1.3 *Stopwords Removal*

Dalam dunia pemrograman seperti NLP, *stopwords* merupakan kata yang diabaikan dalam pemrosesan dan biasanya disimpan di dalam *stop lists*. Seperti yang diamati, kolom token berisi kata-kata yang sangat umum seperti ‘this’, ‘the’, ‘to’, ‘was’, ‘that’, dll. Kata-kata ini dikenal sebagai *stopwords* dan tampaknya menambahkan nilai yang sangat kecil untuk analisis. Jika mereka digunakan dalam analisis, itu meningkatkan *overhead* komputasi tanpa menambahkan terlalu banyak nilai atau wawasan. Oleh karena itu, selalu dianggap ide yang baik untuk menghapus *stopword* ini dari token [8].

2.1.1.4 *Stemming*

Stemming adalah proses pemetaan dan penguraian bentuk dari suatu kata menjadi bentuk kata dasarnya atau proses mengubah kata berimbuhan menjadi kata dasar. Tahap ini menghilangkan *suffix* dan *prefix* pada token/kata (reduce inflected), sehingga sebuah kata yang memiliki *suffix* maupun *prefix* kembali kebentuk dasarnya (contoh imbuhan meng-, me-, kan-, di-, i, pe, peng-, a-, dll.).[8].

2.1.1.5 *Converting Into Numerical Form*

Kumpulan kata hasil dari dilakukannya *tokenize* diubah menjadi sebuah angka. Hal ini dilakukan agar sebuah kata memiliki sebuah nilai yang diproses ke dalam

komputasi mesin. Ketika mengubah kata menjadi sebuah angka terdapat banyak cara yang dilakukan dan tiap cara ini memiliki keunggulan dan kekurangannya masing-masing, namun proses ini perlu dilakukan dan penting untuk dilakukan untuk pelatihan terhadap model [8].

2.1.2 *Text Classification*

Text Classification adalah bentuk utama dari analisis teks dan banyak digunakan dalam berbagai domain dan aplikasi. Premis klasifikasi sederhana, yaitu diberi variabel target kategori, pelajari pola yang ada antara *instance* yang terdiri dari variabel independen dan hubungannya dengan target. Karena target diberikan sebelumnya, klasifikasi dikatakan sebagai *supervised machine learning* karena model dilatih untuk meminimalkan kesalahan antara kategori yang diprediksi dan aktual dalam data pelatihan. Setelah model klasifikasi cocok, ia memberikan label kategori ke *instance* baru berdasarkan pola yang terdeteksi selama pelatihan [9].

Premis sederhana ini memberikan peluang untuk sejumlah besar kemungkinan aplikasi, selama masalah aplikasi dirumuskan untuk mengidentifikasi ya/tidak (klasifikasi biner) atau *multiclass classification*. Bagian tersulit dari analisis teks terapan adalah kurasi dan pengumpulan korpus khusus domain untuk membangun model. Bagian tersulit lainnya adalah menyusun solusi analitis untuk masalah khusus aplikasi [9].

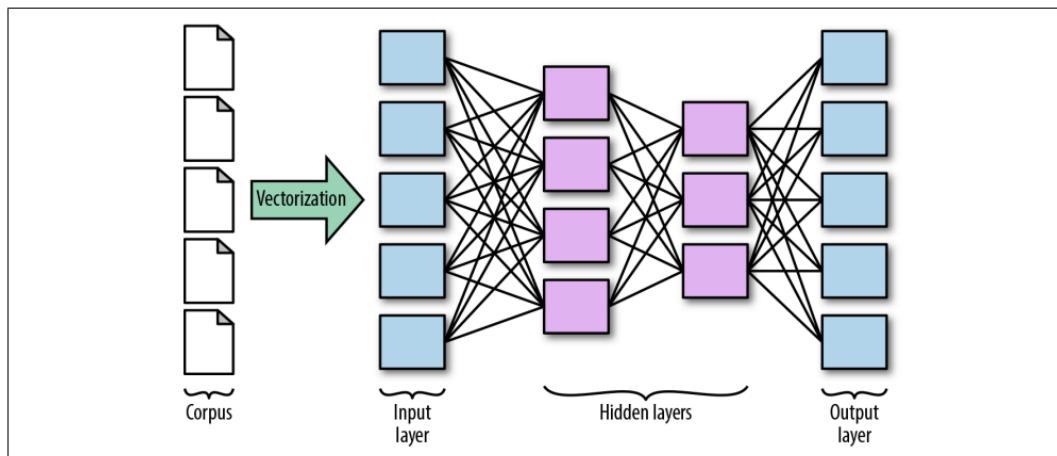
2.1.3 *Sentiment Analysis*

Sentiment analysis mengacu pada proses komputasi, mengidentifikasi, dan mengkategorikan polaritas emosional berdasarkan dalam sebuah ucapan misalnya, untuk menentukan negativitas relatif atau positif dari perasaan penulis atau pembicara. *Sentiment analysis* pada dasarnya berbeda dari klasifikasi gender karena sentimen bukanlah fitur bahasa, melainkan bergantung pada arti kata; misalnya, "kickflip itu sakit" adalah positif sedangkan "sup krim itu membuat saya sakit" adalah negatif, dan "Saya memiliki peliharaan iguana yang sakit" agak ambigu, definisi kata "sakit" dalam contoh ini berubah. Selain itu, sentimen bergantung pada konteks bahkan ketika definisi tetap konstan; "*bland*" mungkin negatif ketika berbicara tentang cabai, tetapi bisa menjadi istilah positif ketika menggambarkan sirup obat batuk. Akhirnya, tidak seperti gender atau tense, sentimen dinegasikan: "tidak baik" berarti buruk. Negasi membalik makna sejumlah besar teks positif; "Saya memiliki harapan tinggi dan ekspektasi besar untuk film yang dijuluki indah dan menggembirakan oleh para kritikus, tetapi sangat kecewa." Di sini, meskipun kata-kata yang biasanya menunjukkan sentimen

positif seperti "harapan tinggi", "hebat", "luar biasa dan menggembirakan", dan bahkan "sangat" melebihi satu-satunya sentimen negatif "kecewa", kata-kata positif tidak hanya tidak mengurangi hal sentimen negatif, tetapi mereka benar-benar meningkatkannya [9].

2.1.4 Artificial Neural Network

Neural networks terdiri dari keluarga model yang sangat luas dan beraneka ragam, tetapi kurang lebih semuanya berasal dari *perceptron*, mesin klasifikasi linier yang dikembangkan pada akhir 1950-an oleh Frank Rosenblatt di Cornell dan dimodelkan pada perilaku belajar otak manusia. Inti dari keluarga model *neural network* adalah beberapa komponen, seperti yang ditunjukkan pada gambar 2.1, lapisan input, representasi data pertama yang divektorkan, lapisan tersembunyi yang terdiri dari *neuron* dan sinapsis, dan lapisan keluaran yang berisi nilai prediksi. Di dalam lapisan tersembunyi, sinapsis bertanggung jawab untuk mentransmisikan sinyal antar *neuron*, yang bergantung pada fungsi aktivasi nonlinier untuk menyangga sinyal yang masuk. Sinapsis menerapkan bobot ke nilai yang masuk, dan fungsi aktivasi menentukan apakah input berbobot cukup tinggi untuk mengaktifkan neuron dan meneruskan nilai ke lapisan jaringan berikutnya [9].



Gambar 2.1 Ilustrasi Arsitektur Neural Network [9]

Dalam jaringan *feedforward*, sinyal berjalan dari *input* ke lapisan *output* dalam satu arah. Dalam arsitektur yang lebih kompleks seperti jaringan berulang dan rekursif, *signal buffering* digabungkan atau berulang antara node dalam lapisan. *Backpropagation* adalah proses dimana kesalahan, dihitung pada lapisan akhir jaringan, dikomunikasikan kembali melalui lapisan untuk secara bertahap menyesuaikan bobot sinapsis dan meningkatkan akurasi dalam iterasi pelatihan

berikutnya. Setelah setiap iterasi, model menghitung *gradient* fungsi kerugian untuk menentukan arah penyesuaian bobot [9].

2.1.5 *Word Embedding*

Word representation learning biasanya dibingkai sebagai prosedur yang tidak diawasi atau diawasi sendiri, karena tidak memerlukan anotasi manual dari data pelatihan. Teks mentah yang biasanya tersedia dalam skala besar, digunakan dengan bagus untuk menghitung statistik kemunculan kata bersama. Oleh karena itu, teknik *word representation* secara otomatis mempelajari ruang semantik tanpa perlu menggunakan pengawasan eksternal atau intervensi manual. Faktanya, salah satu keunggulan *Vector Space Model* (VSM), jika dibandingkan dengan pendekatan representasi pengetahuan lainnya, adalah bahwa mereka langsung dihitung dari corpora yang tidak berlabel. Ini adalah properti yang sangat diinginkan yang memungkinkan VSM menjadi sangat fleksibel dan diperpanjang, dan karena itu mendominasi bidang representasi semantik selama bertahun-tahun [10].

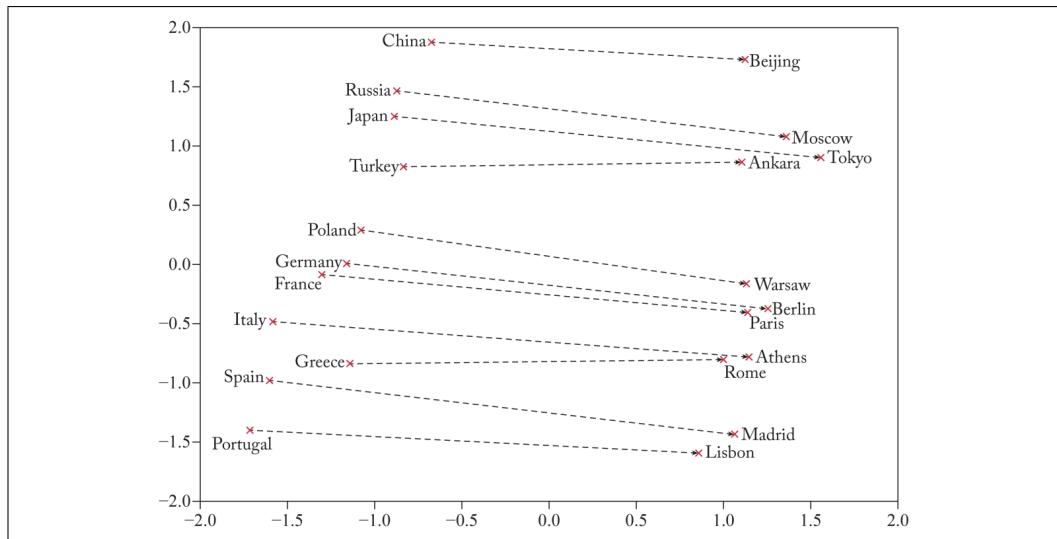
Word embedding adalah jenis khusus dari representasi kata terdistribusi yang dibangun dengan memanfaatkan jaringan saraf, terutama dipopulerkan setelah 2013, dengan diperkenalkannya *Word2vec*. *Word embedding* biasanya diklasifikasikan sebagai model prediktif karena dihitung melalui tujuan pemodelan bahasa, seperti memprediksi kata berikutnya atau kata yang hilang. *Word embedding* memiliki berbagai varian dan teknik spesialisasi untuk meningkatkan *word embedding*, seperti *Character Embeddings and Knowledge-Enhanced Embeddings* [10].

2.1.6 *Global Vector (GloVe)*

Global Vector merupakan metode *unsupervised learning* yang menghasilkan *word embedding* untuk menyelesaikan permasalahan kemiripan kata, analogi kata dan *Named Entity Recognition* (NER). Pada dasarnya, GloVe menyelesaikan kelemahan dari Word2Vec yaitu hanya memahami secara semantik dengan informasi yang lokal, sedangkan GloVe mendapatkan informasi secara lokal dan global [11].

Pre-trained word embedding merupakan *word embedding* yang sudah dilakukan pelatihan pada sekumpulan dataset yang besar, disimpan dan digunakan untuk penyelesaian tugas *embedding*. *Pre-trained word embedding* digunakan untuk menghindar permasalahan dari pembuatan *word embedding* dengan sendiri. Permasalahannya adalah tingkat persebaran kata dan jumlah parameter

pembelajaran. Dalam penelitian ini, digunakannya *pre-trained Stanford GloVe*. GloVe berisikan sekumpulan kata yang direpresentasikan dalam bentuk vektor.

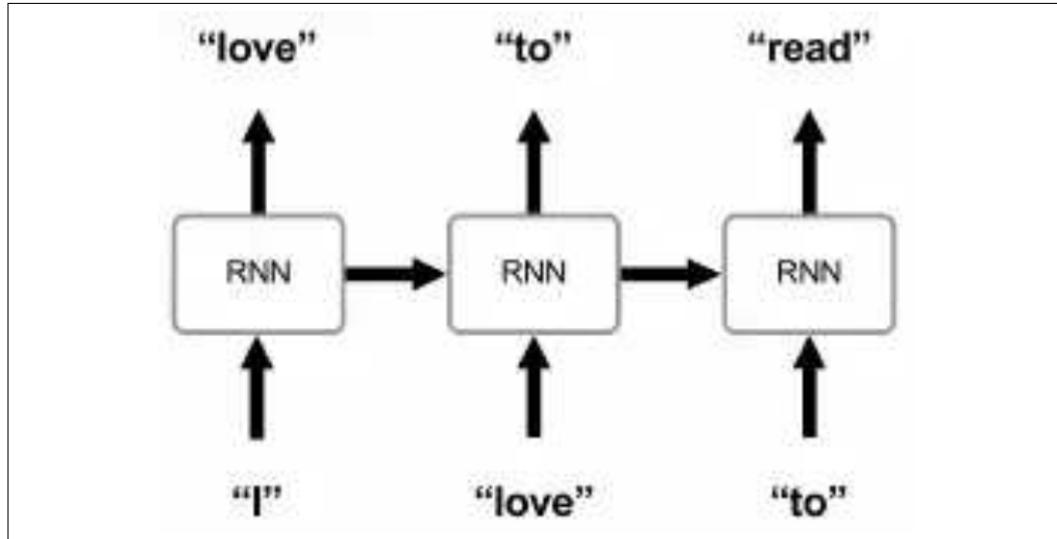


Gambar 2.2 Visualisasi GloVe [10]

Pretrained GloVe digunakan untuk mengambil vektor kata yang sudah dilakukan indeks kata keseluruhan yang disimpan dalam bentuk matriks *embedding*. Kemudian matriks *embedding* digunakan untuk mengkonversi sebuah kata menjadi vektor sebelum masuk untuk penghitungan pada metode LSTM [11].

2.1.7 Recurrent Neural Network (RNN)

RNN adalah salah satu model yang kuat dari keluarga *deep learning* yang telah menunjukkan hasil yang luar biasa dalam 5 tahun terakhir. Ini bertujuan untuk membuat prediksi pada data sekuensial dengan memanfaatkan arsitektur berbasis memori yang kuat [12].



Gambar 2.3 Ilustrasi Arsitektur *Recurrent Neural Network* [12]

Berdasarkan gambar 2.3, RNN menggunakan status memori tambahan. Ketika *input* A (kata "*I*") ditambahkan, jaringan menghasilkan *output* B (kata "*love*") dan menyimpan informasi tentang *input* A dalam status memori. Ketika *input* C berikutnya (kata "*love*") ditambahkan, jaringan menghasilkan *output* terkait D (kata "*to*") dengan bantuan status memori. Kemudian, status memori diperbarui menggunakan informasi dari *input* baru C dan operasi ini diulang untuk setiap *input*. Berdasarkan contoh ini dilihat bagaimana metode ini memprediksi tidak hanya bergantung pada *input* saat ini, tetapi juga pada data sebelumnya. Inilah alasan mengapa RNN adalah model paling mutakhir untuk menangani *sequence data* [12].

2.1.8 *Long Short Term Memory (LSTM)*

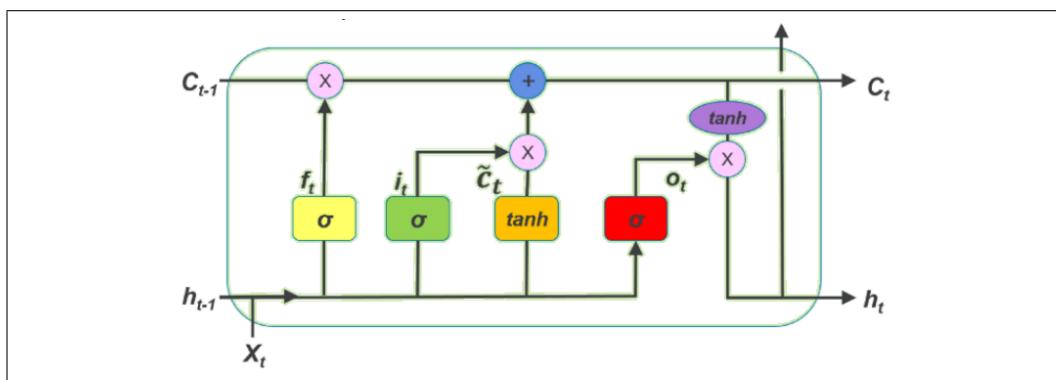
LSTM merupakan varian dari *Recurrent Neural Network* (RNN). RNN adalah salah satu model kuat dari keluarga *deep learning* yang telah menunjukkan hasil luar biasa dalam lima tahun terakhir. Ini bertujuan untuk membuat prediksi pada data sekuensial dengan memanfaatkan arsitektur berbasis memori yang kuat. RNN (juga disebut *feedback*) menggunakan status memori tambahan [12]. RNN berisi siklus yang memberikan aktivasi jaringan dari langkah waktu sebelumnya sebagai input ke jaringan untuk mempengaruhi prediksi pada langkah waktu saat ini. Aktivasi ini disimpan dalam keadaan *internal* jaringan yang pada prinsipnya menyimpan informasi kontekstual temporal jangka panjang. Mekanisme ini memungkinkan RNN untuk mengeksplorasi jendela kontekstual yang berubah secara dinamis melalui riwayat urutan input [13].

Namun dalam RNN terdapat kekurangan dalam mengatasi *vanishing gradient*

yang dimana hal ini menghentikan *neural network* untuk melakukan pelatihan lebih lanjut [5][6]. Kemudian pada tahun 1997, Hochreiter dan Schmidhuber mengusulkan model RNN canggih yang disebut LSTM. Ini bertujuan untuk memecahkan beberapa masalah utama dengan model jaringan saraf berulang yang paling sederhana [12] dan mengatasi masalah RNN dalam *vanishing* dan *exploding gradient* [5][6].

Jaringan LSTM berbeda dengan *Multilayer Perceptrons* (MLP) klasik. Seperti MLP, jaringan terdiri dari lapisan neuron. Data input disebarluaskan melalui jaringan untuk membuat prediksi. Seperti RNN, LSTM memiliki koneksi berulang sehingga status dari aktivasi neuron dari langkah waktu sebelumnya digunakan sebagai konteks untuk merumuskan *output*. Tetapi tidak seperti RNN lainnya, LSTM memiliki formulasi unik yang memungkinkannya untuk menghindari masalah yang mencegah pelatihan dan penskalaan RNN lainnya [13]. LSTM mengatasi tantangan ini dengan desain unit komputasi jaringan LSTM yang disebut *memory cell*, *memory block*, atau singkatnya *cell*.

Setiap unit LSTM, dilihat pada gambar 2.4 memiliki sel memori, dan keadaan pada waktu t direpresentasikan sebagai C_t . Membaca dan memodifikasi dikendalikan oleh gerbang sigmoid dan mempengaruhi *input gate* i_t , *forget gate* f_t dan *output gate* o_t . LSTM dihitung sebagai berikut: Pada saat ini, model menerima *input* dari dua sumber eksternal h_{t-1} dan x_t . Status tersembunyi h_t dihitung dengan *input vector* x_t yang diterima jaringan pada waktu t dan status tersembunyi sebelumnya h_{t-1} . Saat menghitung status node lapisan tersembunyi, *input gate*, *output gate*, *forget gate* dan x_t secara bersamaan mempengaruhi status node [6].



Gambar 2.4 Ilustrasi Arsitektur LSTM [6]

2.1.8.1 Forget Gate

Forget gate adalah gerbang yang memiliki tugas untuk menentukan informasi apa yang harus disimpan dan dibuang pada *cell* [13]. Informasi yang ingin dihitung,

diamambil *input* dari *hidden state* sebelumnya. Kalkulasi dimulai dari *weight* dikalikan dengan nilai *hidden state* sebelumnya dan nilai *weight* dikalikan dengan *input* yang kemudian ditambahkan dengan nilai bias. Nilai tersebut selanjutnya dihitung menggunakan *sigmoid activation* yang menghasilkan nilai dari *forget gate* dengan hasil apakah disimpan atau dibuang. Berikut adalah persamaan 2.1 untuk *forget gate*.

$$\text{forget gate} : f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (2.1)$$

Keterangan :

- f_t : Nilai dari *forget gate*.
 - σ_g : Fungsi aktivasi sigmoid.
 - W_f, U_f : Nilai *weight*.
 - x_t : Nilai *input* berupa vektor.
 - h_{t-1} : Nilai dari *hidden state* sebelumnya.
 - b_f : Nilai *bias*.
-

2.1.8.2 Input Gate

Input gate adalah *gate* yang menentukan nilai *input* baru yang dimana nilai tersebut digunakan untuk *update cell* [13]. Kalkulasi dimulai dari W_i dikalikan dengan nilai dari x_t sebelumnya dan U_i dikalikan dengan nilai *input* h_{t-1} . Kemudian hasil tersebut dijumlah dan ditambah dengan nilai b_i . Setelah itu nilai tersebut dihitung menggunakan *sigmoid activation* dan menghasilkan nilai untuk *input gate*. Berikut adalah persamaan 2.2 untuk penghitungan *input gate*.

$$\text{input gate} : i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (2.2)$$

Kalkulasi selanjutnya dilakukan dengan penghitungan yang sama seperti *input gate* dan sebagai pembedanya adalah penghitungan akhir yang menggunakan *tanh activation* untuk menghasilkan nilai *candidate state*. Berikut persamaan 2.3 untuk penghitungan *candidate state*.

$$candidate\ state : \tilde{C}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \quad (2.3)$$

Keterangan :

- i_t : Nilai dari *input gate*.
 - \tilde{C}_t : Nilai dari *candidate gate*.
 - σ_g : Fungsi aktivasi sigmoid.
 - σ_h : Fungsi aktivasi *tanh*.
 - W_i, U_i, W_c, U_c : Nilai *weight*.
 - x_t : Nilai *input* berupa vektor.
 - h_{t-1} : Nilai dari *hidden state* sebelumnya.
 - b_i, b_c : Nilai *bias*.
-

2.1.8.3 Output Gate

Output gate adalah *gate* yang memiliki tugas untuk menentukan nilai yang digunakan untuk penghitungan *hidden state* baru berdasarkan pada *input* dan memori pada *cell* [13]. Penghitungan pertama dilakukan oleh *output gate* yaitu dengan W_o dikalikan dengan nilai x_t dan U_o dikalikan dengan nilai h_{t-1} . Didapat dua nilai hasil perkalian tersebut yang kemudian dijumlahkan. Hasil penjumlahan tersebut ditambah dengan nilai b_o . Kemudian hasil tersebut dilakukan penghitungan menggunakan *sigmoid activation* dan dihasilkannya nilai untuk *output gate*. Berikut persamaan 2.4 untuk *output gate*.

$$output\ gate : o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (2.4)$$

Keterangan :

- o_t : Nilai dari *output gate*.
 - σ_g : Fungsi aktivasi sigmoid.
 - W_o, U_o : Nilai *weight*.
 - x_t : Nilai *input* berupa vektor.
 - h_{t-1} : Nilai dari *hidden state* sebelumnya.
 - b_o : Nilai *bias*.
-

2.1.8.4 Cell State

Cell state adalah bagian tambahan dari RNN yang memiliki tugas sebagai memori. Tahapan ini menentukan memori dari sebelumnya apa yang bisa disimpan atau dilupakan. Sebelum penghitungan, dipastikan dimulai dari memiliki hasil nilai-nilai dari *forget gate* dan *input gate*. Penghitungan dimulai dari nilai *forget gate* dikalikan dengan nilai *cell state* sebelumnya. Kemudian nilai *input gate* dikalikan dengan *candidate state*. Dari kedua nilai tersebut, dijumlahkan hasilnya yang menghasilkan nilai *cell state* yang terbaru. Berikut persamaan 2.5 untuk penghitungan terbaru nilai *cell state*.

$$\text{cell state} : C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (2.5)$$

Keterangan :

- C_t : Nilai dari *cell state*.
 - f_t : Nilai dari *forget gate*.
 - C_{t-1} : Nilai dari *cell state* sebelumnya.
 - i_t : Nilai dari *input gate*.
 - \tilde{C}_t : Nilai dari *candidate state*.
-

2.1.8.5 Hidden State

Setelah nilai dari *output gate* didapat, perhitungan selanjutnya dilakukan untuk menemukan nilai *hidden state* baru. Perhitungan dilakukan dengan nilai *cell state* dihitung dengan *tanh activation* dan kemudian hasil tersebut dikalikan dengan nilai *output gate*. Dari perhitungan tersebut didapat nilai *hidden state* baru yang diteruskan untuk penghitungan selanjutnya. Berikut persamaan 2.6 untuk penghitungan menentukan nilai *hidden state*.

$$\text{hidden state} : h_t = o_t \odot \sigma_h(C_t) \quad (2.6)$$

Keterangan :

- h_t : Nilai dari *hidden state*.
 o_t : Nilai dari *output gate*.
 σ_h : Fungsi aktivasi *tanh*.
 C_t : Nilai dari *candidate state*.
-

2.1.9 *Emotion Detection in Text*

Emotion Detection (ED) adalah cabang dari analisis sentimen yang berhubungan dengan ekstraksi dan analisis emosi. Bercabang dari bidang analisis sentimen yang tujuan utamanya adalah menganalisis bahasa manusia dengan mengekstraksi pendapat, ide, dan pemikiran melalui penetapan polaritas baik negatif, positif, atau netral adalah cabang bidang dari ED, yang berupaya mengekstraksi lebih halus emosi seperti senang, sedih, marah, dan sebagainya, dari bahasa manusia daripada tugas polaritas kasar dan umum di analisis sentimen. Oleh karena itu ED adalah asosiasi sinergis dari emosi yang juga disebut afeksi dan teknologi serta memperoleh esensinya dari penerapan teknologi yang ditentukan emosi ke area yang berbeda untuk memberikan pengambilan keputusan yang cermat [4].

Ekstraksi emosi dari berbagai jenis komponen jaringan sosial adalah topik penelitian yang sedang diselidiki untuk waktu yang lama sekarang. Berbagai macam konten yang diposting oleh orang-orang di platform jejaring sosial telah dianalisis untuk mendeteksi emosi di balik postingan tersebut. Dalam beberapa karya, kombinasi nada suara, ucapan, ekspresi wajah, gerak tubuh, sinyal EEG, berbagai jenis sinyal bio, dan teks digunakan untuk mendeteksi emosi. Namun, meskipun popularitas konten interaktif ini meningkat, kebanyakan orang masih beralih ke teks untuk komunikasi dan interaksi mereka dalam kehidupan sehari-hari serta di platform jejaring sosial [14].

Teks masih menjadi pilihan utama orang untuk mengungkapkan perasaan mereka terhadap orang, peristiwa, atau benda lain. Mengekstrak emosi dari teks masih lebih sulit karena sifat datanya. Mendeteksi emosi dari teks lebih mudah jika kata-kata yang mewakili emosi tertentu secara eksplisit dalam teks. Tetapi sebagian besar waktu, emosi diungkapkan dengan cara yang halus [14]. Kemudian, beberapa teks memiliki emosi dan kata-kata yang ambigu, beberapa kata memiliki makna ganda, dan beberapa kata memiliki makna emosi yang sama. Teks multibahasa, kesalahan ejaan, akronim, kalimat yang salah tata bahasa adalah beberapa karakteristik umum dari teks yang tersedia secara online. Keterbatasan

data tekstual ini terkadang membuat deteksi emosi otomatis hampir tidak mungkin dilakukan. Sekali lagi, terkadang, mungkin ada banyak emosi dalam satu teks. Kemudian, beberapa teks memiliki emosi dan kata-kata yang ambigu, beberapa kata memiliki makna ganda, dan beberapa kata memiliki makna emosi yang sama. Beberapa teks mewakili sarkasme, atau menggunakan bahasa gaul. Teks multibahasa, kesalahan ejaan, akronim, kalimat yang salah tata bahasa adalah beberapa karakteristik umum dari teks yang tersedia secara online [14].

Ada beberapa pendekatan untuk deteksi emosi tekstual. Sebenarnya, tugas pengenalan emosi adalah bagian dari *Affective Computing* [1] dan metode komputasi yang digunakan di bidang ini telah diklasifikasikan ke dalam beberapa kategori oleh berbagai peneliti. Metode yang didefinisikan oleh para peneliti ini digeneralisasikan ke dalam empat kategori [14]:

1. *Keyword-based Method*

Keyword-based emotion detection adalah pendekatan yang paling intuitif dan mudah. Idenya adalah untuk menemukan pola yang mirip dengan kata kunci emosi dan mencocokkannya. Tugas pertama adalah menemukan kata yang mengungkapkan emosi dalam sebuah kalimat. Ini biasanya dilakukan dengan menandai kata-kata dari sebuah kalimat dengan penanda *Parts-Of-Speech* dan kemudian mengekstrak kata-kata *Noun*, *Verb*, *Adjective* dan *Adverb* (NAVA). Sebagian besar penelitian linguistik dan berbasis emosi membuktikan bahwa ini adalah kata-kata pembawa emosi yang paling mungkin. Kemudian kata-kata ini dicocokkan dengan daftar kata yang mewakili emosi menurut model emosi tertentu. Emosi mana pun yang cocok dengan kata kunci dianggap sebagai emosi dari kalimat tertentu. Pendekatan yang berbeda diterapkan ketika kata tersebut cocok dengan beberapa emosi dari daftar. Dalam beberapa kamus kata kunci, setiap kata memiliki skor probabilitas untuk setiap emosi dan emosi dengan skor tertinggi dipilih sebagai emosi kata. Dalam beberapa karya lain, emosi pertama yang dicocokkan dengan kata dipilih sebagai emosi utama dari kata tersebut. Daftar referensi kata kunci atau kamus kata kunci berbeda tergantung pada peneliti. Kamus kata kunci biasanya dibuat oleh para peneliti berdasarkan emosi dan kata-kata yang terkait dengannya. Ada alat dan program online seperti *WordNet2* yang menemukan sinonim dan antonim kata yang digunakan untuk membuat kamus [14].

2. *Lexicon-based Method*

Pendekatan *Lexicon-based emotion detection* mengklasifikasikan teks menggunakan leksikon (basis pengetahuan dengan teks berlabel sesuai dengan

emosi) yang sesuai untuk dataset input. Di sini deteksi emosi mirip dengan metode sebelumnya, tetapi dalam kasus ini, leksikon emosi digunakan sebagai pengganti daftar kata. *National Research Council of Canada* (NRC) dan *EmoSenticNet* (ESN) adalah beberapa leksikon emosi dan sentimen yang umum digunakan.

3. *Machine Learning Method*

Metode *supervised* dan *unsupervised machine learning* digunakan untuk deteksi emosi teksual di mana model dirancang untuk melatih pengklasifikasi dengan bagian dari kumpulan data dan kemudian menguji pengklasifikasi dengan sisa data. Untuk metode terawasi, kumpulan data emosi beranotasi digunakan untuk melatih dan menguji pengklasifikasi terawasi. *Naive Bayes*, *Support Vector Machine*, dan *decision tree* disebut sebagai pengklasifikasi yang paling umum digunakan. Pengklasifikasi dimulai dengan beberapa kata benih untuk setiap emosi yang kemudian dirujuk silang ke kalimat. Dengan cara ini, kalimat diklasifikasikan ke emosi yang sesuai. Ini melatih model pengklasifikasi yang selanjutnya digunakan untuk memberi label pada data pengujian. Metode tanpa pengawasan adalah metode yang lebih umum tetapi dalam banyak kasus, klasifikasi yang diawasi mencapai akurasi yang lebih baik [14].

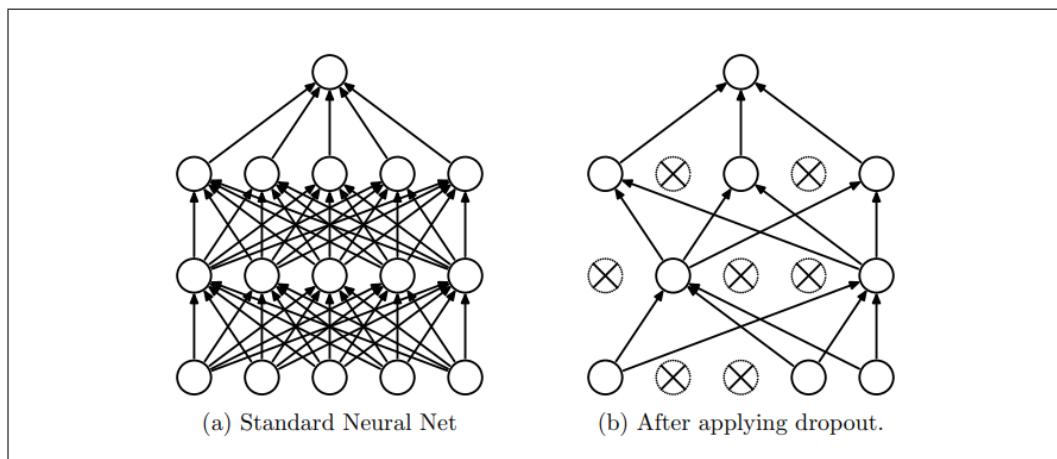
Model *supervised deep learning* sedang diadopsi sebagai pendekatan ML untuk mendeteksi emosi dari teks. Argumennya adalah karena teknik pembelajaran mendalam lebih kuat dan lapisan dalamnya mengekstraksi detail intrinsik/tersembunyi yang mungkin dibawa oleh teks. Penerapan teknik tersebut pada masalah *emotion detection* berbasis teks terlihat mengungguli teknik yang menerapkan teknik machine learning. Beberapa karya yang menerapkan beberapa teknik ini memecahkan rekor dalam *emotion detection* berbasis teks dibahas di bagian selanjutnya [14].

4. *Hybrid Method*

Pendekatan *hybrid* untuk deteksi emosi dalam teks adalah dengan menggabungkan dua atau ketiga metode yang ditentukan untuk mencapai manfaat dari beberapa metode dan mencapai tingkat akurasi maksimum [14]. Pendekatan *hybrid* menggabungkan konstruksi aturan dan pendekatan *machine learning* atau *deep learning* menjadi model terpadu. Dengan demikian, mengambil dari kekuatan kedua pendekatan sambil menyembunyikan keterbatasan yang terkait, pendekatan ini memiliki kemungkinan lebih tinggi untuk melampaui pendekatan lainnya secara individual [14].

2.1.10 Dropout

Jaring saraf pada gambar 2.5a yang dalam dengan sejumlah besar parameter adalah sistem pembelajaran mesin yang sangat kuat. Namun, *overfitting* adalah masalah serius dalam jaringan tersebut. Jaringan besar juga lambat untuk digunakan sehingga sulit untuk menangani *overfitting* dengan menggabungkan prediksi banyak jaringan saraf besar yang berbeda pada waktu pengujian. *Dropout* adalah teknik untuk mengatasi masalah ini. Kunci dari *dropout* adalah dengan secara acak membuang unit (bersama dengan koneksinya) dari jaringan saraf selama pelatihan. Ini mencegah unit untuk terlalu banyak beradaptasi. Selama pelatihan, sampel dibuang dari sejumlah eksponensial jaringan "thinned" yang berbeda seperti pada gambar 2.5b. Pada waktu pengujian, mudah untuk memperkirakan efek rata-rata prediksi semua jaringan tipis ini hanya dengan menggunakan jaringan tunggal yang tidak tipis yang memiliki bobot lebih kecil. Ini secara signifikan mengurangi *overfitting* dan memberikan peningkatan besar pada model [15].



Gambar 2.5 Ilustrasi *Dropout Neural Net* Model [15]

Proses perhitungan untuk mendapatkan satu nilai seperti pada persamaan 2.7 dimulai dengan melakukan inisialisasi nilai random dengan rentang 0 sampai 1. Jika nilai random lebih besar dari nilai probabilitas p, maka nilai keluaran dijadikan nol. Jika nilai random lebih kecil dari nilai probabilitas p, maka nilai masukan dikalikan dengan 1 dibagi probabilitas p.

$$O_{(i)} = \begin{cases} 0 & \text{for } rn > p \\ O_{(i)}x_p^1 & \text{for } rn \leq p \end{cases} \quad (2.7)$$

Keterangan :

- $O_{(i)}$: matriks keluaran posisi ke- i
 p : nilai probabilitas
 rn : nilai *random* dari 0 sampai 1
-

2.1.11 *Dense/Fully Connected Layer*

Dense atau *fully connected layer* adalah model tradisional neural network yang berfungsi untuk melakukan klasifikasi sesuai dengan kelas pada *output*. *Dense layer* digunakan dalam arsitektur LSTM untuk menghubungkan setiap neuron pada lapisan sebelumnya ke setiap neuron pada *fully connected layer*. Hasil dari *fully connected layer* diproses dengan fungsi aktivasi. *Hyperparameter* pada lapisan ini adalah jenis fungsi aktivasi yang digunakan. *Fully connected* terakhir pada sebuah *neural network* melakukan proses klasifikasi [13]. Persamaan untuk menghitung keluaran pada layer ini dilihat pada persamaan 2.8.

$$y = f(W_t x + b) \quad (2.8)$$

Keterangan :

- W_t : matriks yang berisi bobot sambungan antar *layer*
 f : fungsi aktivasi
 x : vektor input
 b : vektor *bias*
-

2.1.12 *Adaptive Moment Estimation Optimizer (Adam)*

Optimizer merupakan metode yang digunakan untuk menentukan pembaruan terhadap *weight* yang memiliki tujuan untuk mengurangi error. *Optimizer* bekerja dengan mencari *weight* yang optimal dengan memperbarui secara berkala sesuai dengan *epoch* atau iterasi pada saat proses pelatihan.

Optimizer yang digunakan pada penelitian ini adalah Adam (*Adaptive Moment Estimation*) *Optimizer* [16]. Adam memiliki kelebihan dibandingkan dengan algoritme optimasi lainnya yaitu lebih cepat dan efektif untuk mencapai titik *convergence*. Tujuan dari penggunaan Adam adalah tercapainya *convergence* yang

merupakan kondisi dimana Adam membuat perubahan yang sangat kecil yang mungkin belum optimal tetapi sudah sangat dekat dengan tujuannya. Berikut algoritme untuk Adam *Optimizer* [16].

Require:

α : Stepsize

$\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates

w_0 : Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)

$v_0 \leftarrow 0$ (Initialize 2nd moment vector)

$t \leftarrow 0$ (Initialize timestep)

while w_t not converged do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_w f_t(w_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)

$w_t \leftarrow w_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)

end while

return θ_t (Resulting parameters)

Keterangan :

α : stepsize atau *learning rate*.

β_1 : Konstanta kontribusi dari gradien sebelumnya untuk momen pertama.

β_2 : Konstanta kontribusi dari gradien sebelumnya untuk momen kedua.

w : *Weight* dari model.

m : Hasil estimasi momen pertama pada *weight* atau *bias*.

v : Hasil estimasi momen kedua pada *weight* atau *bias*.

t : *Timestep*.

g : Nilai gradien turunan parsial pada *error* pada iterasi.

\hat{m} : Nilai hasil koreksi *weight* atau *bias* estimasi momen pertama.

\hat{v} : Nilai hasil koreksi *weight* atau *bias* estimasi momen kedua.

ϵ : Nilai epsilon.

σ : Hasil parameter yang terdiri dari *weight* atau *bias*.

Memperbarui nilai bobot w_t , diperlukannya parameter yang disebut *learning rate*. *Learning rate* menentukan seberapa besar *step* yang diambil setiap iterasi selama

proses pelatihan hingga mencapai lokal minimum. *Learning rate* mempengaruhi performa akurasi selama proses pelatihan hingga akhir. Nilai *learning rate* yang besar membuat kecepatan belajar bertambah tetapi menciptakan hasil yang tidak stabil yang mempengaruhi akurasi. Nilai *learning rate* yang kecil membuat kecepatan belajar semakin lama dengan perubahan bobot yang kecil.

2.1.13 *Softmax*

Fungsi aktivasi bertujuan untuk menetapkan batas nilai keluaran dari *neuron* sehingga menentukan *neuron* apa yang harus diaktifasi atau tidak. *Softmax* adalah salah satu jenis fungsi aktivasi yang biasanya ditemukan dalam *output layer neural network*. Fungsi *softmax* digunakan pada jaringan saraf klasifikasi. *Neuron* yang memiliki nilai tertinggi mengklaim *input* sebagai anggota kelasnya. Fungsi aktivasi *softmax* memaksa *output* dari jaringan saraf untuk mewakili probabilitas bahwa *input* jatuh ke dalam masing-masing kelas [17].

Untuk mengklasifikasikan data *input* menjadi salah satu dari kelas yang ingin diklasifikasikan, diperlukan satu *neuron output* untuk masing-masing kelas tersebut. *Neuron* keluaran tidak secara *inherent* menentukan probabilitas masing-masing dari tiga spesies. Oleh karena itu, diinginkan untuk memberikan probabilitas yang berjumlah 100 persen. Jaringan saraf memberitahu kemungkinan suatu data menjadi masing-masing dari kelas yang diklasifikasikan. Untuk mendapatkan probabilitas, gunakan fungsi softmax pada persamaan 2.9 [17].

$$\text{softmax} : \phi_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \quad (2.9)$$

Keterangan :

- o_i : Nilai dari *softmax*.
 - \exp : Eksponensial.
 - z : Designates the array of output neurons.
 - i : Index of the output neuron (o) being calculated.
 - j : Index of all neurons.
 - Σ_j : Penjumlahan seluruh array j .
-

2.1.14 *Categorical Cross-Entropy*

Loss function merupakan fungsi pengukuran error dari sebuah model *neural network* dalam melakukan klasifikasi atau prediksi. Kesalahan dari model diminimalkan dengan pembaruan *weight* dan bias di setiap iterasinya. Dalam penelitian ini digunakannya fungsi aktivasi *softmax* [17] untuk proses terakhir yang digunakan sebagai prediksi hasil. Dari seluruh proses hingga proses akhir, dihitung nilai *loss* atau error sebagai nilai yang menunjukkan seberapa besar salah hasil prediksi dengan hasil seharusnya. *Categorical cross-entropy* merupakan fungsi *loss* yang digunakan dalam penelitian ini yang biasa digunakan untuk klasifikasi sebanyak lebih dari 2 kelas [13]. Persamaan 2.10 merupakan fungsi *loss* yang digunakan ditulis sebagai berikut [18].

$$Loss = -\log(\phi_i) \quad (2.10)$$

Keterangan :

Loss : nilai *loss*

ϕ_i : hasil fungsi *Softmax* yang menunjukkan probabilitas objek ke-*i*

2.1.15 *Confusion Matrix*

Confusion matrix adalah sebuah tabel yang digunakan untuk mendeskripsikan performa dari model klasifikasi dari sekumpulan data uji dimana nilai kebenarannya diketahui. Jumlah yang benar dan salah dihitung dan dipecah dalam masing-masing kelas. *Confusion matrix* memberikan gambaran tentang kesalahan yang dilakukan oleh *classifier*. Selain itu, *confusion matrix* juga memberikan gambaran tentang kesalahan yang dibuat [8][9].

Confusion matrix membentuk sebuah tabel berukuran $N \times N$ yang dibedakan menjadi nilai prediksi dan nilai sebenarnya, dimana N merupakan jumlah dari output class seperti yang tertera pada tabel 2.1.

Tabel 2.1 Tabel *Confusion Matrix*

		Predicted Label	
		Positive	Negative
Actual Label	Positive	True Positive (TP)	False Negative (TN)
	Negative	False Positive (FP)	True Negative (FN)

Berikut adalah penjelasan dari tabel 2.1 diatas:

1. TP (*True Positive*) adalah kondisi ketika data bernilai positif dan diprediksi dengan benar sebagai *class* positif.
2. TN (*True Negative*) adalah kondisi ketika data bernilai negatif dan diprediksi dengan benar sebagai *class* negatif.
3. FP (*False Positive*) adalah kondisi ketika data bernilai negatif dan diprediksi dengan salah sebagai *class* positif.
4. FN (*False Negative*) adalah kondisi ketika data bernilai positif dan diprediksi dengan salah sebagai *class* negatif.

Dalam *confusion matrix* disimpulkan sebuah akurasi yang menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar. Perhitungan akurasi dalam *confusion matrix* terlihat pada persamaan 2.11, dimana jumlah dari *true positives* dan *true negatives* dibagi dengan jumlah seluruh data.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.11)$$

Nilai *Recall* (R) digunakan untuk menghitung nilai prediksi positif jika data sebenarnya adalah positif. Berikut persamaan 2.12 untuk *recall*.

$$R = \frac{TP}{TP + FN} \quad (2.12)$$

Nilai *Precision* (P) digunakan untuk menghitung nilai prediksi positif yang benar. Berikut persamaan 2.13 untuk *precision*.

$$P = \frac{TP}{TP + FP} \quad (2.13)$$

Nilai F1 *Score* digunakan untuk mencari *mean* dari perbandingan *precision* dan *recall* dengan rentang nilai 0 sampai 1. Berikut persamaan 2.14 untuk *f1 score*.

$$F1 = \frac{2 \times P \times R}{P + R} \quad (2.14)$$

2.2 Pustaka Python

Dalam sub bab ini dijelaskan beberapa teori mengenai pustaka Python yang digunakan dalam penelitian ini. Penjelasan mengenai pustaka Python tersebut dijelaskan sebagai berikut.

2.2.1 Pandas

Pandas adalah alat analisis dan manipulasi data *open source* yang cepat, kuat, fleksibel, dan mudah digunakan yang dibangun diatas bahasa pemrograman Python. Pandas mendukung pembacaan dan penulisan data dengan media berupa *excel spreadsheet*, CSV, dan SQL yang kemudian dijadikan sebagai objek Python dengan baris dan kolom yang disebut data frame seperti halnya pada tabel statistik. Berikut tabel 2.2 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.2 Tabel *Library* Pandas

No	Method	Parameter	Penjelasan
1	DataFrame	data = mengacu pada struktur yang dibuat sebagai contoh kerangka data columns = properti opsional untuk memberi label pada kolom	Fungsi yang digunakan untuk membuat struktur 2 dimensi, yaitu struktur data di sepanjang dua sumbu: x dan y

2.2.2 Numerical Python (NumPy)

NumPy adalah pustaka yang berisi berbagai macam rumus matematis yang umum digunakan. NumPy adalah library Python yang fokus pada *scientific computing*. NumPy memiliki kemampuan untuk membentuk objek *N-dimensional array*, yang mirip dengan list pada Python. Keunggulan NumPy *array* dibandingkan dengan list pada Python adalah konsumsi memori yang lebih kecil serta *runtime* yang lebih cepat. NumPy juga memudahkan dalam aljabar linear, terutama operasi pada vektor (1-d *array*) dan matrix (2-d *array*). Berikut tabel 2.3 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.3 Tabel *Library* NumPy

No	Method	Parameter	Penjelasan
1	fromstring	string = string yang mengandung data dtype = tipe data <i>array</i> sep = string yang memisahkan angka dalam data	Fungsi yang digunakan untuk membuat <i>array</i> 1 dimensi yang baru dari data teks dalam string
2	zeros	shape = bentuk <i>array</i> baru	Fungsi yang digunakan untuk membuat <i>array</i> baru dengan bentuk dan tipe tertentu dan diisi dengan data 0
3	float	-	Fungsi yang digunakan untuk mengubah data tipe menjadi float

2.2.3 Matplotlib

Matplotlib adalah pustaka *open-source* yang digunakan untuk membuat visualisasi data yang statis, beranimasi, dan interaktif dalam Python. Matplotlib pertama kali diciptakan oleh John D. Hunter dan sekarang telah dikelola oleh tim pengembang yang besar. Awalnya matplotlib dirancang untuk menghasilkan plot grafik yang sesuai pada publikasi jurnal atau artikel ilmiah. Matplotlib digunakan dalam skrip Python, Python dan IPython shell, server aplikasi web, dan beberapa *toolkit graphical user interface* (GUI) lainnya. Visualisasi dari matplotlib adalah sebuah gambar grafik yang terdapat satu sumbu atau lebih. Setiap sumbu memiliki sumbu

BAB 2 LANDASAN TEORI

horizontal (x) dan sumbu vertikal (y), dan data yang direpresentasikan menjadi warna dan *glyphs* seperti marker atau *lines* atau *polygon*. Berikut tabel 2.4 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.4 Tabel *Library Matplotlib*

No	Method	Parameter	Penjelasan
1	figure	figsize = panjang (x) dan lebar (y) grafik	Fungsi yang digunakan untuk mengatur besar grafik yang ditampilkan
2	show	-	Fungsi untuk menampilkan grafik yang dihitung
3	imshow	X = array atau PIL image	Fungsi untuk menampilkan data sebagai image
4	axis	option = mengaktifkan atau menonaktifkan garis sumbu dan label	Fungsi ini digunakan untuk mengatur beberapa properti sumbu ke grafik
5	tight_layout	pad = <i>padding</i> antara tepi gambar dan tepi <i>subplot</i> , sebagai sebagian kecil dari ukuran font	Fungsi ini untuk menyesuaikan lapisan antara dan sekitar <i>subplots</i>
6	plot	x = array yang berisi titik-titik pada sumbu x y = array yang berisi titik-titik pada sumbu y	Fungsi yang digunakan untuk menggambar poin dalam diagram
7	title	label = teks yang digunakan untuk judul	Fungsi yang digunakan untuk memberikan judul pada grafik
8	xlabel	string = nama	Fungsi yang digunakan untuk mengatur label untuk sumbu x

Tabel 2.4 Tabel *Library Matplotlib* (Lanjutan)

No	Method	Parameter	Penjelasan
9	ylabel	string = nama	Fungsi yang digunakan untuk mengatur label untuk sumbu y
10	legend	labels = daftar label untuk ditampilkan di sebelah artis loc = lokasi dari legend	Fungsi yang digunakan untuk mengatur nama pada unsur-unsur dalam grafik

2.2.4 *Wordcloud*

Wordcloud adalah teknik visualisasi data yang digunakan untuk merepresentasikan data teks di mana ukuran setiap kata menunjukkan frekuensi atau kepentingannya. *Wordcloud* banyak digunakan untuk menganalisis data dari situs web jejaring sosial. Titik data textual yang signifikan disorot menggunakan *cloud* kata. Untuk menghasilkan *cloud* kata dengan Python, modul yang dibutuhkan adalah matplotlib, pandas, dan wordcloud. Berikut tabel 2.5 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.5 Tabel *Library Wordcloud*

No	Method	Parameter	Penjelasan
1	WordCloud	width = panjang height = tinggi background_color = warna latar belakang min_font_size = ukuran huruf	Fungsi yang digunakan untuk menggambar

2.2.5 *Regular Expression (RegEx)*

RegEx atau Ekspresi Reguler adalah urutan karakter yang membentuk pola pencarian. RegEx digunakan untuk mengecek jika sebuah string mengandung pola pencarian yang ditentukan untuk melakukan *matching* (pencocokan), *locate* (pencarian), dan manipulasi teks. Konsep tentang regex pertama kali muncul di tahun 1951, ketika seorang ilmuwan matematika bernama Stephen Cole Kleene memformulasikan definisi tentang bahasa formal. Kemudian konsep ini diadopsi di beberapa program dan menjadi umum digunakan pada program pemroses teks.

Berikut tabel 2.6 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.6 Tabel *Library* RegEx

No	Method	Parameter	Penjelasan
1	sub	pattern = pola yang dicari replace = string yang digunakan untuk menggantikan pola string = data teks yang ingin dicari digantikan polanya	Fungsi untuk menggantikan satu atau banyak kata dengan sebuah string

2.2.6 *String*

Modul Python *String* menyediakan alat tambahan untuk memanipulasi *string*. Modul Python *String* berisi juga memiliki beberapa konstanta, fungsi utilitas, dan kelas. Modul *string* berasal dari versi Python paling awal. Di versi 2.0, banyak fungsi yang sebelumnya hanya diterapkan di modul dipindahkan ke metode objek *string* dan *unicode*. Versi lama dari fungsi tersebut masih tersedia, tetapi penggunaannya tidak digunakan lagi dan dihapus di Python 3.0. Modul *string* masih berisi beberapa konstanta dan kelas yang berguna untuk bekerja dengan objek *string* dan *unicode*. Berikut tabel 2.7 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.7 Tabel *Library* String

No	Method	Parameter	Penjelasan
1	punctuation	-	Fungsi ini menghasilkan string karakter ASCII yang dianggap sebagai karakter tanda baca

2.2.7 *Natural Language Toolkit (NLTK)*

NLTK adalah *platform* terkemuka untuk membangun program Python untuk bekerja dengan data bahasa manusia. NLTK menyediakan *interface* yang mudah digunakan ke lebih dari 50 sumber daya korpora dan leksikal seperti *WordNet*, bersama dengan rangkaian perpustakaan pemrosesan teks untuk klasifikasi, tokenisasi, *stemming*, penandaan, penguraian, dan penalaran semantik, pembungkus untuk perpustakaan NLP kekuatan industri, dan forum diskusi yang

aktif.

2.2.7.1 Stopwords

Stopwords adalah kata-kata yang tidak menambahkan banyak arti pada sebuah kalimat. Mereka dengan aman diabaikan tanpa mengorbankan arti kalimat. Misalnya kata-kata seperti *the*, *he*, *have* dll. Kata-kata seperti ini sudah ditangkap di korpus yang telah diprogram untuk diabaikan oleh mesin pencari, baik saat mengindeks entri untuk pencarian maupun saat mengambilnya sebagai hasilnya. dari kueri penelusuran. *Stopwords* sering dihapus dari teks sebelum melatih model *deep learning* dan *machine learning* karena *stopwords* muncul dalam jumlah banyak, sehingga memberikan sedikit atau bahkan tidak ada informasi unik yang digunakan untuk klasifikasi atau pengelompokan. Berikut tabel 2.8 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.8 Tabel Library Stopwords

No	Method	Parameter	Penjelasan
1	words	language = bahasa yang ingin dihasilkan	Fungsi ini digunakan untuk menghasilkan kata-kata umum dalam bahasa tertentu

2.2.7.2 Tokenizer

Tokenizer adalah proses untuk membagi teks yang berupa kalimat, paragraf atau dokumen, menjadi token-token / bagian-bagian tertentu. Dalam Python, *tokenizer* pada dasarnya mengacu pada pemisahan teks yang lebih besar menjadi baris, kata, atau bahkan pembuatan kata yang lebih kecil untuk bahasa Inggris atau bukan bahasa Inggris. Berbagai fungsi tokenisasi terintegrasi ke dalam modul nltk itu sendiri. Berikut tabel 2.9 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.9 Tabel Library Tokenizer

No	Method	Parameter	Penjelasan
1	word_tokenize	text = teks yang digunakan untuk dipecahkan menjadi kata	Fungsi yang digunakan untuk menghasilkan salinan teks yang diberi token

2.2.8 Datasets

Datasets adalah pustaka ringan yang menyediakan dua fitur utama :

1. *One-line Dataloaders for Many Public Datasets*

One-liner untuk mengunduh dan melakukan pra-proses sejumlah kumpulan data kumpulan data publik utama (dalam 467 bahasa dan dialek) yang disediakan di *HuggingFace Datasets Hub*.

2. *Efficient Data Pre-processing*

Pra-pemrosesan data yang sederhana, cepat, dan direproduksi untuk kumpulan data publik diatas serta kumpulan data lokal Anda sendiri dalam CSV/JSON/teks.

Berikut tabel 2.10 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.10 Tabel *Library Datasets*

No	Method	Parameter	Penjelasan
1	load_dataset	dataset_name = nama dari dataset yang ingin dimuat	Fungsi ini digunakan untuk memuat dataset

2.2.9 Keras

Keras adalah API pembelajaran mendalam yang ditulis dengan Python, berjalan diatas platform pembelajaran mesin *TensorFlow*. Ini dikembangkan dengan fokus pada memungkinkan eksperimen cepat. Mampu pergi dari ide ke hasil secepat mungkin adalah kunci untuk melakukan penelitian yang baik. Keras mengurangi beban kognitif pengembang untuk membebaskan Anda untuk fokus pada bagian masalah yang benar-benar penting.

2.2.9.1 *Preprocessing*

API lapisan pra pemrosesan Keras yang memungkinkan pengembang untuk membangun saluran pemrosesan *input* asli Keras. *Pipeline* pemrosesan *input* ini digunakan sebagai kode pra pemrosesan independen dalam alur kerja non-Keras, digabungkan langsung dengan model Keras, dan diekspor sebagai bagian dari Keras *SavedModel*. Berikut tabel 2.11 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.11 Tabel *Library Preprocessing*

No	Modul	Method	Parameter	Penjelasan
----	-------	--------	-----------	------------

Tabel 2.11 Tabel *Library Preprocessing* (Lanjutan)

No	Method	Parameter	Penjelasan
1	<i>Text</i>	Tokenizer	num_words = jumlah maksimum kata yang harus disimpan, berdasarkan frekuensi kata
2	<i>Sequence</i>	pad_sequences	sequences = daftar urutan (setiap urutan adalah daftar bilangan bulat) maxlen = panjang maksimum semua urutan

2.2.9.2 *Utils*

Utilities Keras yang ditulis dengan Python dan berguna untuk membantu dalam penggunaan fungsi mendasar. Berikut tabel 2.12 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.12 Tabel *Library Utils*

No	Method	Parameter	Penjelasan
1	to_categorical	y = vektor kelas untuk diubah menjadi matriks num_classes = jumlah total kelas	Fungsi ini digunakan untuk mengkonversi vektor kelas (bilangan bulat) ke matriks kelas biner

2.2.9.3 *Layers*

Layers adalah blok bangunan dasar jaringan saraf di Keras. *Layers* terdiri dari fungsi komputasi *tensor-in-tensor-out* (metode panggilan lapisan) dan beberapa status, disimpan dalam variabel TensorFlow (bobot lapisan). Berikut tabel 2.13 yang berisi kumpulan method yang digunakan dalam penelitian ini.

Tabel 2.13 Tabel *Library Layers*

No	Method	Parameter	Penjelasan
1	Sequential	layer = <i>layer instance</i>	Model yang menerima input layer dengan menggunakan <i>method</i> .add() sehingga layer tersusun secara berurutan
2	Dense	units = bilangan bulat positif, dimensi ruang keluaran activation = Fungsi aktivasi untuk digunakan	Layer ini digunakan untuk melakukan klasifikasi sesuai dengan kelas pada <i>output</i> .
3	Embedding	input_dim = jumlah kosakata output_dim = dimensi dense embedding input_length = panjang urutan input weights = daftar array numpy untuk ditetapkan sebagai bobot awal trainable = mengatur bobot agar diperbarui atau tidak selama pelatihan	Layer ini digunakan untuk mengkalikan matriks sederhana yang mengubah kata menjadi <i>embeddings word</i> yang sesuai
4	Dropout	rate = fraksi unit input untuk dijatuhkan	Layer ini digunakan untuk membantu mencegah <i>overfitting</i>
5	LSTM	units = dimensi ruang keluaran	Layer ini digunakan untuk menyimpan bobot.

2.3 Tinjauan Studi

Pada bagian ini, dibahas beberapa penelitian terkait sebagai dasar dari penelitian ini. Pemilihan metode LSTM didasari dari penelitian Krommyda pada tahun 2021. Penelitian ini membuktikan bahwa algoritme LSTM menghasilkan tingkat akurasi yang paling tinggi dibanding dengan beberapa algoritme lainnya. *State of the art* dilihat pada tabel 2.14.

Tabel 2.14 Tabel *State of the Art*

No	Jurnal	Topik Penelitian	Metode	Hasil
1	M. Krommyda, A. Rigos, K. Bouklas, and A. Amditis, “An experimental analysis of data annotation methodologies for emotion detection in short text posted on Social Media,” <i>Informatics</i> , vol. 8, no. 1, p. 19, 2021.	Penelitian ini menerapkan beberapa algoritme <i>machine learning</i> dan <i>deep learning</i> untuk pengenalan emosi dalam sebuah data teks singkat.	1. LSTM 2. SVM-SGD 3. XGBoost 4. <i>Naive Bayes</i> 5. <i>Decision Tree</i> 6. <i>Random Forest</i>	Algoritme LSTM menghasilkan akurasi tertinggi sebesar 91.90%. Di sisi yang lain algoritme: 1. SVM-SGD menghasilkan akurasi 86.86%. 2. XGBoost menghasilkan akurasi 84.45%. 3. <i>Naive Bayes</i> menghasilkan akurasi 77.01%. 4. <i>Decision Tree</i> menghasilkan akurasi 84.69%. 5. <i>Random Forest</i> menghasilkan akurasi 80.35%.

Tabel 2.14 Tabel State of the Art (Lanjutan)

No	Jurnal	Topik Penelitian	Metode	Hasil
2	H. Park and K. Kim, "Impact of Word Embedding Methods on Performance of Sentiment Analysis with Machine Learning Techniques," <i>Journal of The Korea Society of Computer and Information</i> , vol. 25, no. 8, pp. 181–188, Aug. 2020.	Penelitian ini menerapkan beberapa teknik <i>word embedding</i> untuk membuktikan adanya dampak yang diberikan terhadap akurasi dari <i>sentiment analysis</i> .	1. <i>Bag of Word</i> 2. TF-IDF 3. <i>Word2Vec</i>	Model yang menggunakan <i>Word Embedding Bag of Word</i> menghasilkan akurasi 83.22%, TF-IDF menghasilkan akurasi 84.27%, dan <i>Word2Vec</i> menghasilkan akurasi adalah 79.8%.
3	M.-H. Su, C.-H. Wu, K.-Y. Huang, and Q.-B. Hong, "LSTM-Based Text Emotion Recognition Using Semantic and Emotional Word Vectors," <i>2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)</i> , May 2018.	Penelitian ini menerapkan algoritme LSTM yang mempelajari hubungan temporal dan kontekstual dan CNN yang mempelajari struktur spasial dalam <i>word sequence</i> .	1. CNN 2. LSTM	algoritme LSTM menghasilkan akurasi yang paling tinggi sebesar 70.66%, sedangkan CNN didapatkan akurasi sebesar 65.33%.

Tabel 2.14 Tabel *State of the Art* (Lanjutan)

No	Jurnal	Topik Penelitian	Metode	Hasil
4	N. Murthy, S. Rao Allu, B. Andhavarapu, M. Bagadi, and M. Belusonti, "Text Based Sentiment Analysis Using LSTM," <i>International Journal of Engineering Research and Technology</i> , vol. 9, no. 05, May 2020.	Penelitian ini ingin membuktikan bahwa algoritme LSTM mengatasi masalah <i>vanishing gradient</i> yang terjadi pada algoritme RNN.	1. RNN 2. LSTM	LSTM mengatasi masalah <i>vanishing gradient</i> pada RNN dengan menggunakan <i>cell memory</i> dan memiliki nilai akurasi sebesar 85%.
5	W. K. Sari, D. P. Rini, and R. F. Malik, "Text Classification Using Long Short Term Memory with GloVe Features," <i>Scientific Journal of Computer Electrical Engineering and Informatics</i> , vol. 5, no. 2, p. 85, 2020.	Penelitian ini ingin membuktikan bahwa kumpulan data teks digunakan untuk menghasilkan makna dan struktur yang digunakan oleh algoritme <i>machine learning</i> .	1. LSTM 2. GloVe	Kumpulan data teks diproses <i>machine learning</i> dengan bantuan <i>word embedding</i> dan hasil akurasi yang didapatkan dengan menggunakan LSTM - GloVe adalah 95.17%.

Terdapat 2 pendekatan yang digunakan untuk melakukan pengenalan emosi dalam teks, yaitu pendekatan menggunakan *machine learning* dan pendekatan menggunakan *deep learning*. Pendekatan menggunakan *machine learning* digunakan untuk melakukan pengenalan emosi, namun solusi yang diberikan oleh pendekatan *machine learning* gagal untuk mendapatkan akurasi yang tinggi,

terutama karena terbatasnya ketersediaan kumpulan data pelatihan beranotasi, dan bias yang diperkenalkan pada anotasi oleh interpretasi pribadi emosi dari individu [3].

Pada referensi [2][3][5][6] dilakukan uji coba terhadap beberapa algoritme selain LSTM untuk melihat seberapa bagus model yang dibuat dengan masalah-masalah seperti, kecepatan pembelajaran model, mengatasi masalah dalam kurangnya mempelajari *word sequence* dalam data [2], *vanishing gradient* pada model pelatihan [5][6], dan dampak dari penggunaan *word embedding* pada model [7]. Hasil dari pengujian tersebut terlihat bahwa LSTM mempelajari *word sequence* pada data lebih baik karena kemampuannya dalam mempelajari hubungan temporal dan kontekstual [2], mengatasi masalah dalam *vanishing gradient* karena adanya *memory cell* [5][6], dan meningkatkan performa model dengan bantuan *word embedding* [7].

Dalam pendekatan *deep learning*, terdapat beberapa metode yang digunakan untuk melakukan pengenalan emosi dalam teks, seperti metode *Recurrent Neural Network* (RNN) [5][6] dan *Convolutional Neural Network* (CNN) [2], namun penggunaan algoritme-algoritme tersebut memiliki kekurangan, yaitu model berbasis CNN yang mempelajari struktur spasial dalam *word sequence* tidak memberikan hasil yang bagus [2], sedangkan model berbasis RNN yang meningkatkan kemampuan belajar untuk data urutan yang lama memiliki kekurangan dalam masalah *vanishing gradient* ketika melatih model RNN [5][6]. Hal - hal tersebut mempengaruhi hasil akurasi dari suatu model. Selain itu *word embedding* juga mempengaruhi hasil dari model yang dibuat dan dampak dari adanya *word embedding* meningkatkan hasil akurasi dari model [7].

Pada referensi [19] dilakukan uji coba mengenai dampak *preprocessing* pada teks terhadap sistem. Faktor yang memiliki dampak terhadap *preprocessing* adalah menghilangkan *punctuation*, *spellcheck*, *negation word*, dan *stopwords*. Hasil membuktikan bahwa uji coba *preprocessing* terhadap sistem memberikan dampak yang membuat model menjadi lebih bagus, terutama dalam pengenalan emosi [19]. Pada referensi [5][6] penggunaan *tokenizer* disebutkan dan digunakan untuk melakukan *encode words* dan *encode label*, dimana hal ini dilakukan agar teks diproses oleh mesin.

Pada referensi [5][6] dilakukan uji coba terhadap *layer* yang digunakan dalam membuat model LSTM, dalam referensi tersebut dibuat uji coba model LSTM dengan 3 layer, yaitu *Embedding Layer*, *LSTM Layer*, dan *Dense Layer* dengan

fungsi aktivasi *softmax*, *loss function categorical_crossentropy*, dan *adam optimizer*. Dari hasil penelitian tersebut didapatkan hasil akurasi yang tinggi, yaitu 95.17% dan 99.56%.

Penggunaan LSTM sangatlah bagus, namun LSTM juga memiliki kecenderungan terjadinya *overfitting*, apalagi dengan tambahan menggunakan *embedding layer* dalam pembuatannya, maka dari itu diperlukannya regularisasi agar model tidak menjadi *overfitting*. Pada referensi [15] dijelaskan bahwa layer *dropout* berfungsi untuk mengatasi *overfitting* terhadap model dan meningkatkan performa dari model.

2.4 Tinjauan Objek

Pada bagian ini dijelaskan mengenai objek terkait dengan pengenalan emosi dalam teks berbahasa Inggris. Pada gambar 2.6 terlihat *dataset* ini didapat dari internet dengan website *HuggingFace* [20] dengan nama *dataset emo*. *Dataset* tersebut memiliki jumlah fitur sebanyak 2 seperti pada tabel 2.15 dan jumlah data sebanyak 30.160 yang digunakan pada penelitian ini.

Tabel 2.15 Tabel Fitur *Dataset*

Fitur	Penjelasan
<i>text</i>	Data kalimat
label	Jenis emosi yang dikategorikan menjadi: 1. <i>Others</i> (0) 2. <i>Happy</i> (1) 3. <i>Sad</i> (2) 4. <i>Angry</i> (3)

text	label
don't worry I'm girl hem how do i know if you are what's ur name	0
when did i see many times i think no i never saw you	3
by google chrome where you live	0
u r ridiculous I might be ridiculous but I am telling the truth & little disgusting whole	3
just for time pass w do u do & living then maybe	0

Gambar 2.6 Dataset *HuggingFace - Emo* (<https://huggingface.co/datasets/viewer?dataset=emo>)

BAB 2 LANDASAN TEORI

Berdasarkan tabel 2.15, dilihat bahwa terdapat 4 label yang dikategorikan dari sebuah teks, dimana label - label tersebut mewakili emosi yang terkandung di dalam teks, seperti label *others* mewakilkan kalimat - kalimat yang menyatakan keambiguan, kepedulian, keinginan, kritikan, kepastian, opini, dan saran. Contoh teks yang mewakili label *others* dilihat dalam tabel 2.16.

Tabel 2.16 Tabel Contoh Teks Dengan Label *Others*

No	Teks
1	<i>i see you as competition definitely but i like you</i>
2	<i>where everywhere in my head me wants sushi who is sushi</i>
3	<i>omg you need so many improvements i totally do i have it saved should i use it yes</i>
4	<i>not soon i want it now you want what girls to chat with</i>
5	<i>how have you been i've been good how about you i was home</i>
6	<i>there's nothing wrong in loving someone if we love each other the world will be a better place very true i never really believed that before but i can understand the difference now you're the expert mark</i>
7	<i>i miss you why don't you just get married and come back with you</i>
8	<i>i'm always right are you sure about that 100 percent sure</i>
9	<i>yeah i care too much but for him awww thanks i care for you too not for</i>
10	<i>who's batman i like spiderman more but if you're asking who would win id say batman educate me about batman</i>

Label *happy* mewakilkan kalimat - kalimat yang menyatakan kebahagiaan, kejenakaan, pujian, kekaguman, kenyamanan, dan candaan. Contoh teks yang mewakili label *happy* dilihat dalam tabel 2.17.

Tabel 2.17 Tabel Contoh Teks Dengan Label *Happy*

No	Teks
1	<i>are u enjoyed in holi festival i did d i had great fun on tht day its unforgettable</i>
2	<i>huuuuum you are beautiful so nice of you</i>
3	<i>okay i'm just kidding xd hahaha so you are a guy</i>
4	<i>your english is dam good thank you so is yours im impressed</i>

BAB 2 LANDASAN TEORI

Tabel 2.17 Tabel Contoh Teks Dengan Label *Happy* (Lanjutan)

No	Teks
5	<i>how to keep your partner interested in you seeing your work as an art craft lol cool stuff</i>
6	<i>i really enjoying wasting my times with you thanks feelings are mutual hey you there</i>
7	<i>it's amazing out here check what out the handle it's always amazing the handle</i>
8	<i>i figured that much no one else lol it's fun talking with you</i>
9	<i>oh thank u you're welcome bff n i love u too</i>
10	<i>i am watching prison break is it good amazingo amazing</i>

Label *sad* mewakilkan kalimat - kalimat yang menyatakan kesedihan, permintaan maaf, kekacauan, keprihatinan, dan kekecewaan. Contoh teks yang mewakili label *sad* dilihat dalam tabel 2.18.

Tabel 2.18 Tabel Contoh Teks Dengan Label *Sad*

No	Teks
1	<i>you make me disappointed sorry as u told to refer i referring to my friends</i>
2	<i>go away come back another day i dont want to talk to you</i>
3	<i>yeah i understand how was it bad</i>
4	<i>i am here you are not alone i am quite alone</i>
5	<i>she is my angel love you both but she didn't love me</i>
6	<i>i am fill so alone i'm happy alone but i am not happy</i>
7	<i>u ruined my mood why your joke ruined my mood</i>
8	<i>it's was bad it was horrible i hate you</i>
9	<i>you are not my friend okay oh what happened i am upset with you</i>
10	<i>u broke my heart when did i broke ur heart u r cheating on me</i>

Label *angry* mewakilkan kalimat - kalimat yang menyatakan kemarahan, kekesalan, kebencian, tidak ketertarikan, rasa ketergangguan, frustasi, dan mengejek. Contoh teks yang mewakili label *angry* dilihat dalam tabel 2.19.

BAB 2 LANDASAN TEORI

Tabel 2.19 Tabel Contoh Teks Dengan Label Angry

No	Teks
1	<i>i hate her this is why you cant trust boys ever i hate everything</i>
2	<i>are you stupid or something what makes you say that stupid is as stupid does</i>
3	<i>i am very angry right now how come you are angry angryface my gf cheated on me</i>
4	<i>but tell me what do u mean u r being rude</i>
5	<i>why are they so selfish because of low self esteem its contradictory</i>
6	<i>im not interested to talking with you i know i know oh youknow so why you disturb me</i>
7	<i>you are annoying since day 1 you knew fuck yourself</i>
8	<i>so you don't know me anymore i do i think you are not a good friend</i>
9	<i>u r so ugly even the mirror couldn't resist expressing it s feelings i only need to please myself confidence is everything are you stupid</i>
10	<i>i am really annoyed why how to remove you from my list</i>

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi pemaparan analisis terhadap masalah yang ingin diatasi dengan pendekatan yang digunakan. Gambaran besar yang dimaksud meliputi analisis masalah, kerangka pemikiran, *flowchart* proses global, pengambilan *datasets*, dan analisis manual yang dilakukan.

3.1 Analisis Masalah

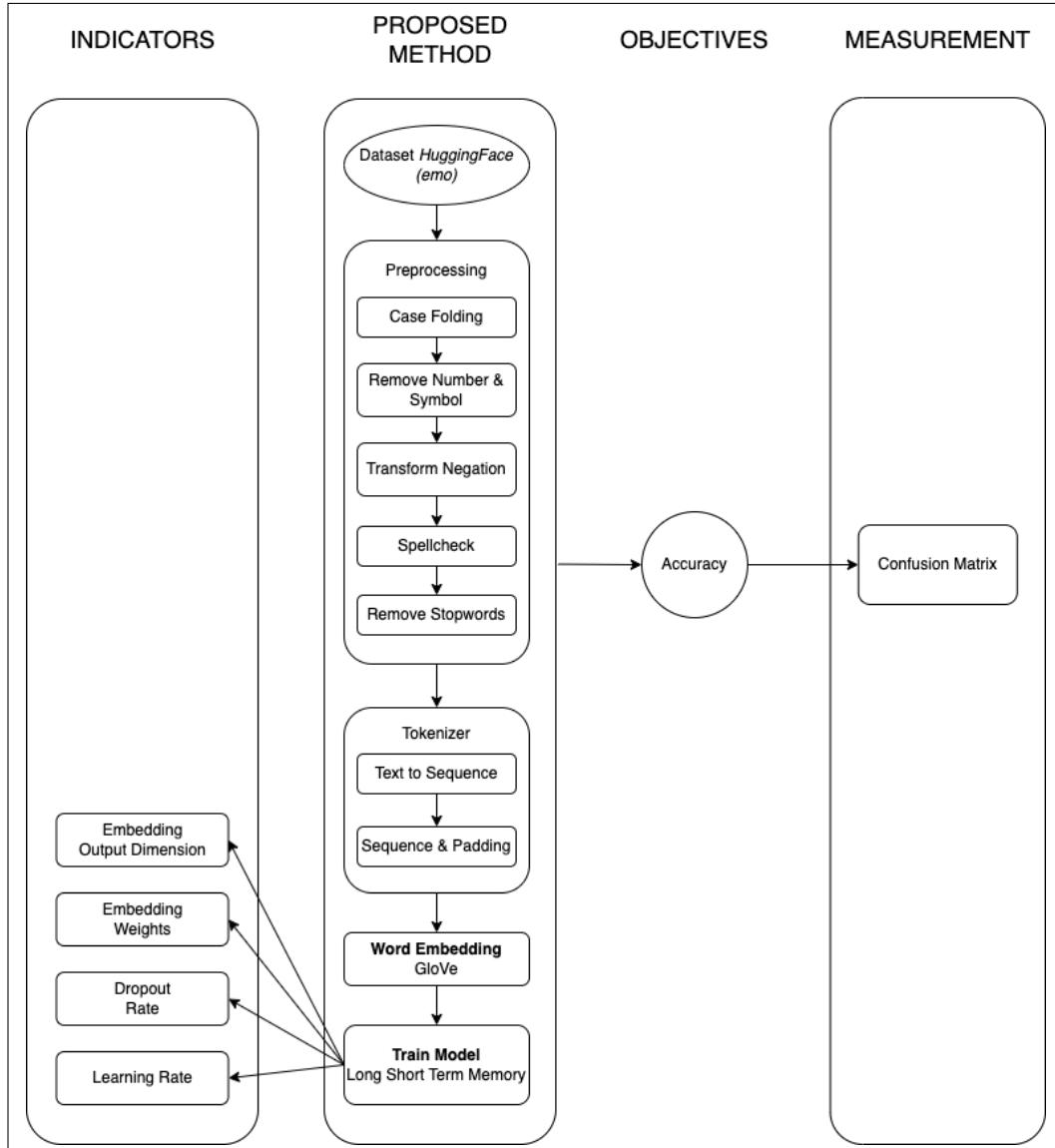
Pada bab 1 telah dijelaskan bahwa pada masa ini pendekripsi emosi diperlukan untuk meningkatkan kecerdasan sistem komputer karena emosi merupakan faktor penting dalam interaksi manusia komputer. Dengan pendekripsi emosi secara otomatis membantu sistem tutor cerdas, interaksi manusia komputer, analisis kepuasan pelanggan, sistem dialog, perilaku *blogger*, dan *enterprise computing*. Banyak penelitian saat ini menggunakan metode *machine learning* dan *deep learning* dalam melakukan pengenalan emosi. Penelitian menggunakan metode *deep learning* memiliki kesulitan dalam memperoleh data yang banyak, namun sejalan dengan pertumbuhan penggunaan internet menyebabkan jumlah data teks yang tersedia semakin banyak sehingga penelitian menggunakan metode *deep learning* diaplikasikan.

Pada penelitian ini, penulis membangun sebuah model LSTM untuk melakukan pengenalan emosi untuk data teks. Pada penelitian ini, penulis menilai akurasi dan mengidentifikasi parameter apa saja yang berpengaruh dalam penerapan algoritme LSTM pada pengenalan emosi untuk data teks. Pada penelitian ini, penulis juga menjawab beberapa masalah yang ada, seperti:

1. Bagaimana melakukan *preprocessing* data untuk data teks?
2. Bagaimana melakukan *tokenizer* pada data teks?
3. Bagaimana penggunaan *word embedding* terhadap model?
4. Bagaimana pengaruh parameter yang digunakan bersamaan ketika membuat model?
5. Bagaimana mendapatkan nilai akurasi terhadap model yang dibuat?

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk membangun model pengenalan emosi untuk data teks.



Gambar 3.1 Kerangka Pemikiran

Berikut adalah rangkaian penjelasan setiap bagian kerangka pemikiran pada gambar 3.1:

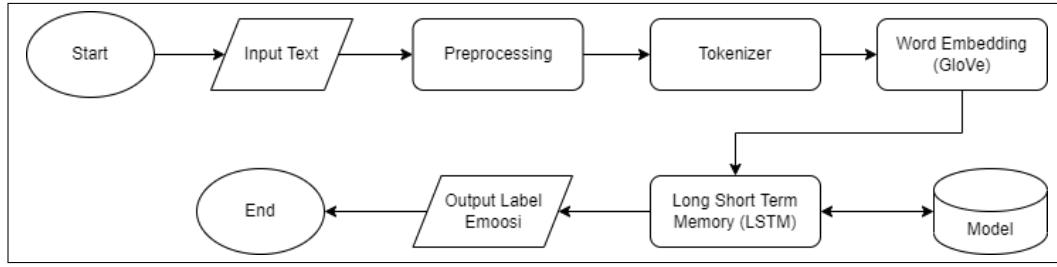
1. *Indicators* adalah parameter yang mempengaruhi hasil dari metode yang diajukan. Pada model yang dibuat terdapat sebuah indikator, yaitu *Embedding Output Dimension*, *Embedding Weights*, *Dropout Rate* dan *Learning Rate*.
 - (a) *Embedding Output Dimension* merupakan penentuan dimensi vektor dari *embedding layer*. Indikator ini harus disesuaikan dengan jumlah vektor dalam *word embedding* agar proses pelatihan model mengeluarkan hasil yang tepat.

- (b) *Embedding Weights* merupakan bobot yang dimasukan untuk representasi dari kata. Indikator ini disesuaikan dengan jumlah vektor dalam *word embedding* agar proses pelatihan model mengeluarkan hasil yang tepat.
 - (c) *Dropout Rate* merupakan penentuan *neuron* yang dipilih secara acak dan tidak dipakai / dibuang selama pelatihan. Indikator ini digunakan untuk mencegah terjadinya *overfitting* dan juga mempercepat proses pembelajaran.
 - (d) *Learning Rate* merupakan salah satu parameter *training* untuk menghitung nilai koreksi bobot pada waktu proses *training*. Nilai *learning rate* ini berada pada range nol (0) sampai satu (1). Semakin besar nilai *learning rate*, maka proses *training* berjalan semakin cepat.
2. *Proposed Method* adalah bagian yang menjelaskan proses dari awal hingga akhir dalam penelitian ini. Proses pertama yang dilalui adalah tahap *preprocessing* yang didalamnya terdapat proses perubahan pada teks, dimana dilakukan proses untuk melakukan *case folding*, menghilangkan angka, menghilangkan simbol, mengubah kata negasi, melakukan *spellcheck*, dan menghilangkan *stopwords*. Kemudian data hasil *preprocessing* digunakan untuk melakukan *tokenizer* untuk melakukan tokenisasi, *text to sequence*, dan *sequence & padding*. Lalu dilakukan tahap *word embedding*, dimana dalam tahap ini *word embedding* dihasilkan menggunakan GloVe. Setelah itu hasil dari proses *tokenizer* dan *word embedding* digunakan untuk melatih model pengenalan emosi yang dibangun.
 3. *Objectives* adalah bagian yang menjelaskan mengenai hal-hal yang menjadi acuan pengukuran dan pengukuran yang digunakan adalah akurasi model.
 4. *Measurement* adalah satuan ukur pada bagian *Objectives*, seperti perhitungan *confusion matrix*.

3.3 Urutan Proses Global

Berikut ini adalah urutan proses global dari metode yang diusulkan untuk membangun pengenalan emosi untuk data teks.

BAB 3 ANALISIS DAN PERANCANGAN SISTEM



Gambar 3.2 Flowchart Emotion Detection in Text

Berikut adalah rangkaian penjelasan urutan proses global pada gambar 3.2:

1. *Dataset* yang digunakan merupakan data teks dengan label emosinya yang diambil dari website *HuggingFace* dengan data bernama emo.
2. Setiap data teks melalui tahap *preprocessing*, dimana dalam tahap ini dilakukan proses *case folding*, menghilangkan angka, menghilangkan simbol, mengubah kata negasi, melakukan *spellcheck*, dan menghilangkan *stopwords* untuk mendapatkan data bersih.
3. Setelah data teks melalui tahap *preprocessing*, dilakukan tahap *tokenizer* untuk melakukan tokenisasi, *text to sequence*, dan *sequence & padding*. Hal ini dilakukan agar data teks beserta labelnya diproses oleh mesin.
4. *Word embedding* di *generate* untuk disiapkan dimasukkan ke dalam *embedding layer*. Digunakannya data *pretrained* GloVe untuk menghasilkan *word embedding*.
5. Setelah dilakukan proses *tokenizer* dan menghasilkan *word embedding*, dibuat sebuah model LSTM, dimana dalam model LSTM ini dibuat menggunakan 4 *layer*, yaitu:
 - (a) *Embedding Layer* = Dalam *embedding layer* ini digunakan 5 parameter, yaitu: *input_dim*, *output_dim*, *input_length*, *weights*, dan *trainable*.
 - (b) *LSTMLayer* = Dalam *LSTMlayer* digunakan 1 parameter, yaitu: *units*.
 - (c) *Dropout Layer* = Dalam *dropout layer* digunakan 1 parameter, yaitu: *rate*.
 - (d) *Dense Layer* = Dalam *dense layer* digunakan 2 parameter, yaitu: *units* dan *activation*.

Dengan tambahan *categorical_crossentropy* sebagai *loss function* dan *adam* sebagai *optimizer* yang dibuat dalam model ketika melakukan *compile*, serta penggunaan *validation_split* dan *epoch* ketika melatih model.

6. Setelah model dibuat, maka dilakukan pengenalan emosi menggunakan model LSTM tersebut dan dikategorikan ke dalam 4 macam, yaitu:
 - (a) *Others* (0)
 - (b) *Happy* (1)
 - (c) *Sad* (2)
 - (d) *Angry* (3)

3.4 Analisis Manual

Dalam sub bab ini dijelaskan proses secara bertahap dengan menggunakan perhitungan manual yang ada dalam penelitian ini. Penjelasan dari proses-proses tersebut dijelaskan sebagai berikut.

3.4.1 Pengambilan Dataset

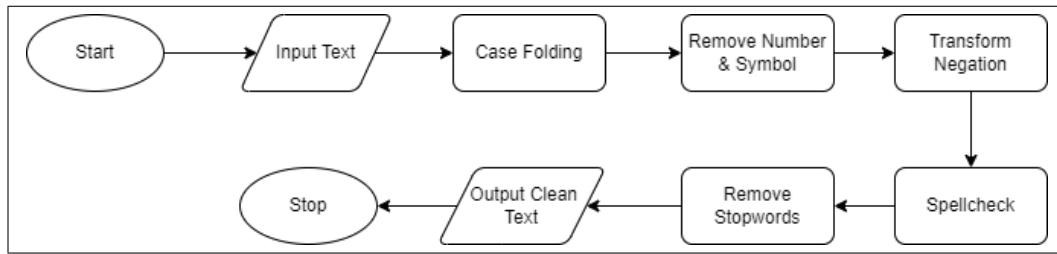
Dalam penelitian ini menggunakan satu dataset berupa teks berbahasa Inggris yang sudah diberi label emosi dalam bentuk angka yang diambil dari *website Huggingface*, yang dimana dalam *dataset* ini terdiri dari jumlah data *training* sebanyak 30160 dan jumlah data *testing* sebanyak 5509. Dataset ini didapatkan dengan menggunakan fungsi *load_dataset()* yang diimport dari sebuah *library* bernama *datasets*.

Terlihat contoh input text berupa data dalam teks dan label emosi yang sudah dikategorikan menggunakan angka yang tampak pada gambar 2.6. Kategori tersebut dibedakan menjadi 4 macam, yaitu *others*, *sad*, *happy*, dan *angry*. Kemudian *input* teks tersebut diproses ke dalam tahap *preprocessing* untuk mendapatkan bentuk data yang diproses ketika melakukan *training* dan *testing* dalam model yang dibuat.

3.4.2 Preprocessing

Pada tahap ini dijelaskan proses pembersihan kata dalam teks. Setiap kata dalam teks dalam *input* teks dilakukan proses pembersihan sehingga digunakan dalam pembuatan model. Proses ini meliputi pembersihan kata dengan menggunakan *case folding*, *remove number*, *remove symbol*, *transform negation*, *spellcheck*, dan

remove stopwords. Berikut adalah langkah-langkah pada saat proses *preprocessing*.



Gambar 3.3 Flowchart Preprocessing

Berikut adalah contoh teks yang digunakan dalam analisa manual.

Tabel 3.1 Contoh Teks

Teks	Label
<i>I don't like when U come here 2 days ago!</i>	Sad (2)

3.4.2.1 Case Folding

Pada tahap ini dilakukan *case folding*, dimana *case folding* adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif. Tujuan dilakukan *case folding* adalah untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’ yang diterima. Hal ini dilakukan untuk memudahkan memproses data teks di tahap *preprocessing* ini. Pada tabel 3.2, terlihat contoh dari perubahan perubahan data teks setelah dilakukan *case folding*.

Tabel 3.2 Contoh Proses *Case Folding* pada Teks

Sebelum	Sesudah
<i>I don't like when U come here 2 days ago!</i>	<i>i don't like when u come here 2 days ago!</i>

3.4.2.2 Remove Number & Symbol

Pada tahap ini angka dan simbol dihilangkan dalam teks. Tujuan penghapusan angka dan simbol ini dikarenakan tidak ada dampaknya terhadap pengenalan emosi yang dilakukan sehingga tahap ini digunakan untuk memudahkan memproses data teks saat melakukan pelatihan pada model. Pada tabel 3.3, terlihat contoh dari perubahan data teks setelah angka dan simbol dihilangkan.

Tabel 3.3 Contoh *Case Folding* pada Teks

Sebelum	Sesudah
<i>i don't like when u come here 2 days ago!</i>	<i>i don't like when u come here days ago</i>

3.4.2.3 *Transform Negation*

Pada tahap ini dilakukan *transform negation*, *negation* adalah mekanisme yang mengubah argumen positif menjadi penolakan terbalik. Secara khusus, dalam tugas analisis afektif, negasi memainkan peran penting karena kata-kata negasi mempengaruhi polaritas kata atau kalimat yang menyebabkan polaritas terbalik dalam banyak kasus. Sebelum dilakukan *transform negation* kata-kata seperti “*don't*”, “*doesn't*”, “*didn't*”, dll. harus diubah menjadi “*do not*”, “*does not*”, “*did not*”, dll. Setelah kata-kata negasi itu diubah, maka baru bisa dilakukan *transform negation*. Tujuan dilakukan *transform negation* adalah menggantikan kata yang diawali “*not*” menjadi kata antonimnya. Pada tabel 3.4, terlihat contoh dari penjabaran kata negasi dan pada tabel 3.5, terlihat contoh dari penggunaan *transform negation*.

Tabel 3.4 Contoh Penjabaran Kata Negasi pada Teks

Sebelum	Sesudah
<i>i don't like when u come here days ago</i>	<i>i do not like when u come here days ago</i>

Tabel 3.5 Contoh *Transform Negation* pada Teks

Sebelum	Sesudah
<i>i do not like when u come here days ago</i>	<i>i do unlike when u come here days ago</i>

3.4.2.4 *Spellcheck*

Pada tahap ini dilakukan *spellcheck*, dimana proses *spellcheck* adalah mengidentifikasi kata-kata yang mungkin salah eja dan memperbaikinya. Tujuan dilakukan *spellcheck* adalah untuk mengoreksi salah eja dan kesalahan ketik atau membiarkannya seperti dengan asumsi mereka mewakili teks bahasa alami dan kompleksitas yang terkait. Pada tabel 3.6, terlihat contoh dari penggunaan *spellcheck*.

Tabel 3.6 Contoh *Spellcheck* pada Teks

Sebelum	Sesudah
<i>i do unlike when u come here days ago</i>	<i>i do unlike when you come here days ago</i>

3.4.2.5 Remove Stopwords

Pada tahap ini dilakukan *remove stopwords*, dimana *stopwords* adalah menghasilkan kata-kata yang paling umum. Tujuan dilakukannya *remove stopwords* adalah untuk menghilangkan kata-kata umum sehingga ketika melakukan pelatihan pada model diproses lebih cepat. Pada tabel 3.7, terlihat contoh dari penggunaan *remove stopwords*.

Tabel 3.7 Contoh *Stopwords* pada Teks

Sebelum	Sesudah
<i>i do unlike when you come here days ago</i>	<i>unlike days ago</i>

3.4.3 Tokenizer

Pada tahap ini dijelaskan proses *tokenizer* dalam teks. Setiap teks yang sudah melakukan tahap pembersihan dilakukan *tokenizer* untuk membagi teks yang berupa kalimat, paragraf atau dokumen, menjadi token-token/bagian-bagian tertentu seperti pada tabel 3.8. Setelah dibuat menjadi token-token, setiap token dari teks dan label yang diberikan pada teks ini dilakukan proses ke tahap *text to sequence* dan *sequence & padding*.

Tabel 3.8 Contoh Tokenisasi pada Teks

Sebelum	Sesudah
<i>unlike days ago</i>	["unlike", "days", "ago"]

3.4.3.1 Text to Sequence

Pada tahap ini, teks yang telah dilakukan tokenisasi diubah menjadi urutan kata yang dijadikan sebagai indeks kata. Kumpulan kata tersebut memiliki indeks yang unik, sehingga antar kata berbeda indeksnya. Indeks mewakili sebuah kata, sehingga teks berisikan kata yang diubah menjadi urutan indeks yang mewakili kata. Berikut adalah tabel 3.9 sebelum dan sesudah dilakukan *text to sequence*.

Tabel 3.9 Contoh *Text to Sequence* pada Teks

Sebelum	Sesudah
["unlike", "days", "ago"]	[0, 1, 2]

3.4.3.2 Padding & Sequence

Pada tahap ini, dilakukan *padding* yaitu dengan menyesuaikan bentuk panjang *sequence* yang beragam menjadi panjang yang sama. Pada contoh ini, digunakannya panjang *sequence* sebesar 10 dan sisa *sequence* dijadikan 0. Berikut tabel 3.10 sebelum dan sesudah penerapan tahap padding.

Tabel 3.10 Contoh *Padding & Sequence*

Sebelum	Sesudah
[0, 1, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 2]

3.4.4 Word Embedding GloVe

Pada tahap ini, dilakukan pemetaan vektor pada setiap kata yang ada pada indeks kata. Indeks berisikan angka yang mewakili sebuah kata dan setiap indeks memiliki kata yang unik. Digunakannya *word embedding* GloVe untuk penelitian ini dan dengan dimensi 50 untuk setiap vektor katanya. Proses dilakukan dengan melihat indeks terlebih dahulu, kemudian melihat kata yang ada. Dari kata tersebut, dilihat pada *pretrained* GloVe, apakah kata tersebut ada dalam GloVe?. Jika ada, diambil vektor yang ada. Kemudian vektor tersebut disimpan dalam *array embedding* berukuran [total indeks kata x 50] untuk menyimpan vektor yang sesuai dengan yang ada pada indeks kumpulan kata. Berikut contoh tabel 3.11 representasi kata dan vektor yang diambil dari GloVe dari 3 indeks kata.

Tabel 3.11 Contoh *Word Embedding*

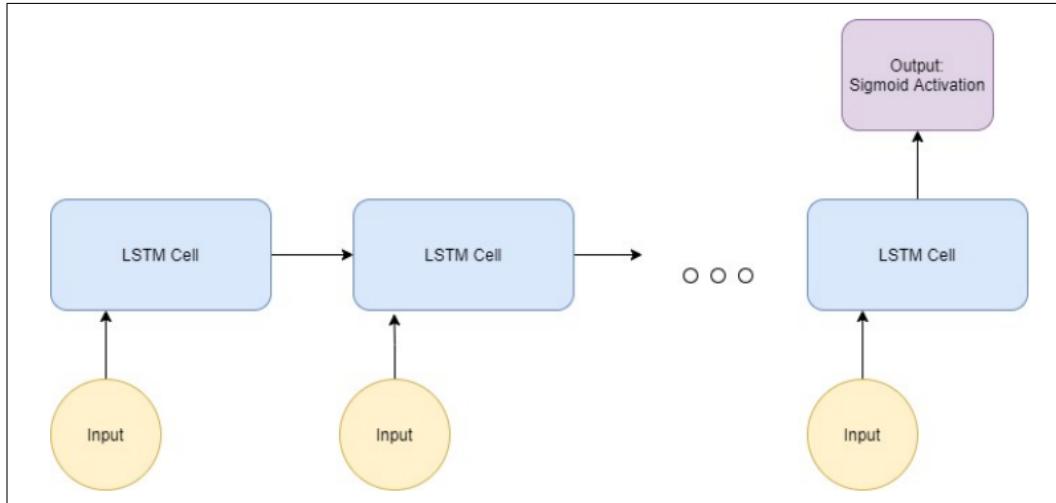
Indeks	Kata	Vektor
0	unlike	[0.40351, 0.25297, 0.058938, 0.016263, -0.16723, 0.49866, -0.46665, -0.39667, -0.42482, 0.22456, 0.17972, 0.31613, 0.17297, 0.25262, 0.21552, 0.4215, 0.092249, 0.26599, -0.079251, -0.50896, -0.30306, -0.42921, 0.1109, 0.4603, -0.072813, -1.1166, -0.6532, -0.16932, 0.0043999, -0.035547, 2.3802, -0.17461, 0.0065762, -0.57749, 0.17811, 0.080492, -0.087587, 0.072499, -0.75795, -0.15779, -0.15288, 0.24441, 0.076898, 0.52795, 0.16373, 0.37204, -0.21652, -0.3913, -0.56267, 0.10644]

Tabel 3.11 Contoh Word Embedding (Lanjutan)

Indeks	Kata	Vektor
1	<i>days</i>	[0.62236, 0.1973, 0.0023326, -0.44924, 0.084905, -0.33628, -0.90642, 0.62154, -0.20657, -0.20043, -0.6269, -0.64429, -0.31616, 0.23314, 1.2348, -0.080113, -0.7382, -0.3497, -0.97589, -0.046964, 0.29777, 0.68288, 0.87876, -0.28921, 0.1456, -1.411, -0.11301, 0.2132, 0.49548, 0.38685, 3.397, 0.40639, -0.54343, -0.00037414, 0.44268, -0.1015, 0.37693, 0.047623, 0.044414, 0.10038, -0.99794, 0.2082, 0.063226, 0.12436, -0.059387, 0.046682, -0.22335, -0.22082, -0.40414, -0.18502]
2	<i>ago</i>	[0.48384, 0.036723, 0.3554, -0.11934, 0.13925, 0.089866, -1.2636, -0.027009, -0.52288, -0.059505, -0.38398, -0.80808, -0.65736, -0.24665, 1.2932, 0.20113, -0.31522, -0.14376, -0.79002, 0.2567, -0.059354, 0.43529, 0.2418, -0.62907, -0.037233, -1.7944, -0.16567, 0.0054654, -0.056066, 0.19838, 2.9616, -0.11965, -0.16888, -0.035133, 0.024751, -0.41839, -0.21409, 0.41672, 0.16242, -0.2483, -0.90225, 0.15443, 0.17426, 0.21366, 0.10285, 0.096885, -0.56849, -0.21438, -0.68166, -0.3792]

3.4.5 Long Short Term Memory (LSTM)

Pada tahap pemodelan, digunakannya metode LSTM untuk membuat model belajar. Analisa secara manual dijelaskan dengan 2 *timestep*, yaitu *input* kata pertama sebagai *timestep* pertama dan kata kedua sebagai *timestep* kedua dari contoh kata. Kata yang digunakan yaitu ”*unlike*” dan ”*days*” dan digunakannya 3 nilai vektor pertama dari 50 vektor. Nilai secara acak dari distribusi normal dengan *mean* adalah 0 dan standar deviasi adalah 1 untuk inisialisasi weight, nilai 0 untuk inisialisasi bias dan jumlah *unit hidden layer* sebanyak 3. Berikut gambar 3.4 ilustrasi LSTM yang digunakan dalam penghitungan manual.



Gambar 3.4 Ilustrasi LSTM Perhitungan Manual

3.4.6 Timestep Pertama

Pada *timestep* pertama dimulai dari kata *"unlike"* sebagai input dengan vektor yang dimilikinya adalah [0.40351, 0.25297, 0.058938] dan inisialisasi nilai *hidden state* sebelumnya dengan nilai 0 karena masih pada posisi perhitungan awal. Nilai vektor digunakan pada gerbang yang ada sebagai *input*.

3.4.6.1 Forget Gate

Penghitungan pertama dimulai dari *forget gate* menggunakan persamaan 2.1. Nilai W_f , U_f diinisialisasi dengan distribusi normal secara acak dan nilai b_f adalah 0. Berikut adalah penghitungannya untuk *forget gate timestep* pertama.

$$f_t = \sigma \left(\begin{bmatrix} -0.36658844 & -1.61106401 & 1.87203336 \\ -0.28343935 & 0.20231672 & -0.4251159 \\ 1.38593374 & -0.21863406 & 1.87334779 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} 1.6002104 & -0.27069961 & -1.08463431 \\ 1.21455431 & 0.3184536 & -0.66009903 \\ 1.33881495 & 0.6323578 & -1.04369611 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari penghitungan diatas, maka didapat matriks vektor hasil *forget gate* sebagai berikut.

$$f_t = \begin{bmatrix} 0.39051712 \\ 0.4779528 \\ 0.64893055 \end{bmatrix}$$

3.4.6.2 Input Gate

Setelah dilakukan penghitungan *forget gate*, penghitungan selanjutnya dilakukan untuk menghitung *input gate* dengan menggunakan persamaan 2.2. Nilai W_i , U_i diinisialisasi dengan distribusi normal secara acak dan nilai b_i adalah 0. Berikut adalah perhitungannya untuk *input gate timestep* pertama.

$$i_t = \sigma \left(\begin{bmatrix} 0.62298068 & 0.44012166 & 0.93798973 \\ 0.38306699 & -0.86468679 & -0.36847324 \\ -0.36003432 & -0.05322914 & -0.38808374 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} 0.18626383 & -0.37952795 & -0.92415426 \\ 0.21628096 & -0.45263405 & -0.7327017 \\ 0.56757731 & 0.29866377 & -0.64779853 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *input gate* sebagai berikut.

$$i_t = \begin{bmatrix} 0.60300451 \\ 0.47854181 \\ 0.45472046 \end{bmatrix}$$

Kemudian dilanjutkan dengan perhitungan untuk *candidate state*. Hal yang membedakan pada penghitungan untuk *candidate state* adalah digunakannya *hyperbolic tangent* sebagai fungsi aktivasi. Persamaan 2.3 digunakan dalam penghitungan *candidate state*. Berikut perhitungan untuk *candidate state*.

$$\tilde{C}_t = \sigma \left(\begin{bmatrix} 1.36162997 & -0.98190943 & -0.85895997 \\ 0.30978415 & -0.14687127 & -3.30831318 \\ 1.23276443 & 0.04983586 & 1.30341836 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \right. \\ \left. \begin{bmatrix} 0.23412451 & -1.16006662 & 1.7361476 \\ -3.3169362 & -1.62590328 & -1.96597113 \\ -1.25348511 & -0.93574191 & 0.10017843 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *candidate state* sebagai berikut.

$$\tilde{C}_t = \begin{bmatrix} 0.24530619 \\ -0.10673032 \\ 0.52763398 \end{bmatrix}$$

3.4.6.3 Output Gate

Perhitungan selanjutnya yaitu *output gate*. *Output gate* menentukan nilai untuk *hidden state* selanjutnya dengan menggunakan persamaan 2.4. Nilai W_o , U_o diinisialisasi dengan distribusi normal secara acak dan nilai b_o adalah 0. Berikut adalah perhitungannya untuk *output gate*.

$$o_t = \sigma \left(\begin{bmatrix} -0.06649506 & -1.3413727 & -1.10056287 \\ 0.89552448 & 0.72447281 & -2.15472494 \\ 1.59390351 & -0.37055401 & 0.38455522 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \right. \\ \left. \begin{bmatrix} -1.86619525 & -0.49338088 & -0.49403357 \\ -1.03749413 & 1.62050985 & 1.13333811 \\ -0.76364195 & -0.44071865 & 0.73512484 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *output gate* sebagai

berikut.

$$o_t = \begin{bmatrix} 0.39388197 \\ 0.60291546 \\ 0.63924342 \end{bmatrix}$$

3.4.6.4 Cell State

Perhitungan selanjutnya adalah memperbarui nilai *memory* yang ada pada *cell state*. Pada *timestep* pertama, nilai *cell state* sebelumnya tidak ada karena masih penghitungan awal, maka diinisialisasikan dengan nilai 0. Persamaan 2.5 digunakan untuk perhitungan *cell state*. Berikut perhitungan *cell state*.

$$C_t = \begin{bmatrix} 0.39051712 \\ 0.4779528 \\ 0.64893055 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.60300451 \\ 0.47854181 \\ 0.45472046 \end{bmatrix} \odot \begin{bmatrix} 0.24530619 \\ -0.10673032 \\ 0.52763398 \end{bmatrix}$$

Dari perhitungan diatas, maka didapat nilai *cell state* yang telah diperbarui, nilai ini diteruskan untuk *cell* selanjutnya dan untuk menentukan nilai *hidden state*. Berikut hasil penghitungan dari *cell state*

$$C_t = \begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix}$$

3.4.6.5 Hidden State

Perhitungan terakhir adalah menentukan nilai *hidden state*. Nilai *hidden state* digunakan untuk penghitungan pada *cell* selanjutnya. Dengan menggunakan persamaan 2.6, didapat nilai *hidden state*. Berikut perhitungan untuk *hidden state*.

$$h_t = \begin{bmatrix} 0.39388197 \\ 0.60291546 \\ 0.63924342 \end{bmatrix} \odot \sigma_h \left(\begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix} \right)$$

Dari perhitungan terakhir diatas, maka didapat nilai *hidden state* dari sebuah *cell*. Berikut hasil dari perhitungan untuk nilai *hidden state*.

$$h_t = \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix}$$

3.4.7 *Timestep Kedua*

Setelah didapatkan nilai *hidden state* pada *timestep* pertama, maka dilanjutkan dengan *input* menggunakan kata kedua. Contoh yang digunakan adalah kata "days" yang memiliki vektor [0.62236, 0.1973, 0.0023326]. Urutan penghitungan sama dengan *timestep* sebelumnya, dengan menggunakan nilai *hidden state* dan *cell state* yang telah diperbaharui sebelumnya.

3.4.7.1 *Forget Gate*

Perhitungan dimulai dengan menggunakan persamaan 2.1 untuk mendapatkan nilai *forget gate*. *Input* yang dimasukan adalah vektor kata dan nilai *hidden state* pada *timestep* pertama. Berikut penghitungan untuk *forget gate* pada *timestep* kedua.

$$f_t = \sigma \left(\begin{bmatrix} -0.04913193 & -0.23109881 & -0.60465429 \\ 1.23493577 & 2.17908225 & 0.44119263 \\ -0.68026136 & 0.15091 & -0.22769326 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} -0.16674495 & -0.7063736 & -0.47867191 \\ 1.26108418 & 0.28889196 & 0.04094385 \\ 0.41756866 & 1.79028567 & 0.93253053 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *forget gate* sebagai berikut.

$$f_t = \begin{bmatrix} 0.46567079 \\ 0.7807006 \\ 0.42929887 \end{bmatrix}$$

3.4.7.2 Input Gate

Kemudian dilakukannya perhitungan untuk *input gate* dengan menggunakan persamaan 2.2. Berikut perhitungan untuk *input gate* pada *timestep* kedua.

$$i_t = \sigma \left(\begin{bmatrix} 1.46662098 & -1.18835177 & -1.1066311 \\ 0.97978437 & -0.67817222 & -1.67490876 \\ 0.73719701 & -0.37583658 & 0.58606196 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} -0.75156608 & -0.24255579 & 0.9542411 \\ -0.51737153 & 1.13295774 & -1.10393987 \\ -0.43572849 & 0.37843361 & 1.54677234 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *input gate* sebagai berikut.

$$i_t = \begin{bmatrix} 0.68639557 \\ 0.55999651 \\ 0.64151604 \end{bmatrix}$$

Perhitungan selanjutnya adalah menemukan nilai *candidate state* dengan menggunakan persamaan 2.3. Berikut penghitungan untuk *candidate state* pada *timestep* kedua.

$$\tilde{C}_t = \sigma \left(\begin{bmatrix} -0.00480373 & 0.65676634 & 0.95335957 \\ 1.77298325 & -1.00736161 & 1.30444274 \\ 0.39506239 & -0.08948894 & 1.28120212 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \right. \\ \left. \begin{bmatrix} 1.003646 & -1.16106992 & -0.59164454 \\ 1.96547564 & 1.10985581 & 1.87020428 \\ 0.38294438 & -0.2390663 & 1.00793169 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *candidate state* sebagai berikut.

$$\tilde{C}_t = \begin{bmatrix} 0.13276231 \\ 0.85345023 \\ 0.39050617 \end{bmatrix}$$

3.4.7.3 Output Gate

Perhitungan selanjutnya adalah *output gate* yang menggunakan persamaan 2.4 untuk menentukan nilai *hidden state* selanjutnya. Berikut perhitungan untuk *output gate*.

$$o_t = \sigma \left(\begin{bmatrix} 1.00549259 & 1.76712501 & -1.27702349 \\ -0.96922206 & -0.33626641 & 1.14818098 \\ 0.78386791 & -1.88373285 & -0.53568864 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \right. \\ \left. \begin{bmatrix} 0.0702497 & -0.52573271 & -0.2914909 \\ 0.81404842 & -0.22791995 & 0.924764 \\ 1.05096884 & 1.40161185 & -0.45755878 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *output gate* sebagai berikut.

$$o_t = \begin{bmatrix} 0.72067741 \\ 0.38376413 \\ 0.51593076 \end{bmatrix}$$

3.4.7.4 Cell State

Perhitungan selanjutnya adalah memberikan pembaharuan untuk nilai *memory cell* sebelumnya dengan menggunakan nilai *forget gate*, *input gate* dan *candidate gate* pada *timestep* kedua. Persamaan 2.5 digunakan untuk memperbaharui nilai *cell state*. Berikut penghitungan untuk *cell state* pada *timestep* kedua.

$$C_t = \begin{bmatrix} 0.46567079 \\ 0.7807006 \\ 0.42929887 \end{bmatrix} \odot \begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix} + \begin{bmatrix} 0.68639557 \\ 0.55999651 \\ 0.64151604 \end{bmatrix} \odot \begin{bmatrix} 0.13276231 \\ 0.85345023 \\ 0.39050617 \end{bmatrix}$$

Dari perhitungan diatas, maka didapat nilai *cell state* yang telah diperbaharui, nilai ini diteruskan untuk *cell* selanjutnya dan untuk menentukan nilai *hidden state*. Berikut hasil penghitungan dari *cell state*

$$C_t = \begin{bmatrix} 0.16000983 \\ 0.43805493 \\ 0.35351592 \end{bmatrix}$$

3.4.7.5 Hidden State

Perhitungan selanjutnya adalah menentukan nilai *hidden state*. Nilai *hidden state* didapatkan dengan menggunakan persamaan 2.6. Berikut penghitungan untuk *hidden state*.

$$h_t = \begin{bmatrix} 0.72067741 \\ 0.38376413 \\ 0.51593076 \end{bmatrix} \odot \sigma_h \begin{pmatrix} 0.16000983 \\ 0.43805493 \\ 0.35351592 \end{pmatrix}$$

Dari perhitungan terakhir diatas, maka didapat nilai *hidden state* dari sebuah cell. Berikut hasil dari penghitungan untuk nilai *hidden state*.

$$h_t = \begin{bmatrix} 0.1143413 \\ 0.15812267 \\ 0.1751533 \end{bmatrix}$$

Proses untuk *timestep* ketiga hingga nilai yang ditentukan memiliki penghitungan yang sama. Setelah penghitungan selesai, nilai *hidden state* yang dihasilkan oleh LSTM dilanjutkan perhitungannya dengan menggunakan *neural network*. Setelah dilakukan perhitungan LSTM, dilakukan perhitungan pada *dropout layer*.

3.4.8 Dropout Layer

Hasil yang sudah dihitung melalui *layer* LSTM, nilai *hidden neuron* dikeluarkan pada *cell* terakhir pada LSTM, dan nilai tersebut dilakukan perhitungan pada *Dropout layer*, dengan menggunakan persamaan 2.7. Hal ini dilakukan untuk menonaktifkan beberapa nilai fitur agar *neural network* yang dibangun tidak terlalu banyak belajar. Parameter probabilitas yang digunakan adalah 0.5, sehingga kemungkinan menghapus setiap fitur adalah 0.5. Berikut adalah contoh perhitungan proses *dropout*.

$$\begin{aligned} O_i &= O_i \times \frac{1}{p} \\ &= 0.1143413 \times \frac{1}{0.5} \\ &= 0.2286826 \end{aligned}$$

Dari perhitungan diatas, maka didapat nilai hasil perhitungan *dropout*. Berikut

hasil dari penghitungan untuk nilai *dropout*.

$$Output\ dropout = \begin{bmatrix} 0.2286826 \\ 0.31624534 \\ 0.3503066 \end{bmatrix}$$

3.4.9 Dense Layer

Pada lapisan *dense*, dilakukan perhitungan sesuai dengan persamaan 2.8. Jumlah kelas pada contoh perhitungan manual ini adalah 4 sehingga menggunakan *dense layer* sebanyak 4 unit. Hasil dari *dropout layer* digunakan untuk menghitung *dense layer*. Berikut adalah contoh perhitungan *dense layer* yang berdasarkan persamaan 2.8.

$$y = f \left(\begin{bmatrix} -0.095457 \\ 0.26964079 \\ 0.62887384 \end{bmatrix} \bullet \begin{bmatrix} 0.2286826 \\ 0.31624534 \\ 0.3503066 \end{bmatrix} + \begin{bmatrix} -0.12877315 \\ -0.29305055 \\ 0.51698025 \\ -0.77540555 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat nilai hasil perhitungan *dense*. Berikut hasil dari penghitungan untuk nilai *dense*.

$$y = f \begin{bmatrix} 0.15496879 \\ -0.00930861 \\ 0.80072219 \\ 0.49166361 \end{bmatrix}$$

3.4.10 Output Layer dengan Aktivasi Softmax

Hasil yang sudah dihitung melalui *dense layer*, dilanjutkan dengan perhitungan selanjutnya dengan menggunakan fungsi aktivasi *softmax*. Perhitungan ini dilakukan dengan mencari probabilitas setiap kelas yang telah didefinisikan sebelumnya. Berikut adalah contoh perhitungan aktivasi *softmax* yang berdasarkan persamaan 2.9.

$$\begin{aligned}\phi_i &= \frac{\exp(z_i)}{\sum_j \exp(z_j)} \\ &= \frac{\exp(0.15496879)}{\exp(0.15496879) + \exp(-0.00930861) + \dots + \exp(0.49166361)} \\ &= 0.23365922\end{aligned}$$

Dari perhitungan terakhir diatas, maka didapat nilai hasil perhitungan *softmax* yang dimana hasil ini merepresentasikan *output* yang diinginkan apakah teks ini masuk dalam kategori emosi *others*, *happy*, *sad*, dan *angry*. Berikut hasil dari penghitungan untuk nilai *softmax*.

$$\phi_i = [0.23365922 \quad 0.19826139 \quad 0.4456871 \quad 0.12239228]$$

Dari hasil nilai yang diatas, menunjukkan hasil prediksi yang didapat. Nilai-nilai tersebut mewakili probabilitas dari jumlah kelas yang ada sesuai dengan indexnya masing-masing dimana indeks 0 mewakili emosi *others*, indeks 1 mewakili *happy*, indeks 2 mewakili *sad*, dan index 3 mewakili *angry*. Indeks yang nilainya paling tinggi mengartikan bahwa teks tersebut termasuk ke dalam kategori emosi sesuai indeksnya dan dari contoh ini nilai paling tinggi ada di indeks 2 yang berarti teks ini masuk ke dalam kategori *sad*.

3.4.11 *Categorical Cross-Entropy*

Setelah menghitung nilai *softmax*, proses selanjutnya adalah dengan menghitung nilai *loss / error* menggunakan *categorical cross-entropy*. *Loss function* merupakan fungsi pengukuran *error* dari sebuah model *neural network* dalam melakukan klasifikasi atau prediksi. *Categorical cross-entropy* digunakan karena klasifikasi yang digunakan sebanyak lebih dari 2 kelas dengan menggunakan persamaan 2.10. Contoh perhitungannya adalah sebagai berikut.

$$\begin{aligned}Loss &= -\log(\phi_i) \\&= -\log(0.23365922) \\&= 0.6314170774\end{aligned}$$

3.4.12 Confusion Matrix

Confusion matrix digunakan untuk mengukur performa dalam klasifikasi. Pada penelitian ini, terdapat 4 buah *class* dalam mendekripsi emosi dan ke-4 buah *class* tersebut diwakili oleh sebuah nilai berbentuk vektor 4×1 yang dimana di setiap vektor tersebut mewakili seberapa besar probabilitas kecenderungan sebuah emosi. Vektor pertama mewakili emosi "others" dalam teks, vektor kedua mewakili emosi "happy" dalam teks, vektor ketiga mewakili emosi "sad" dalam teks, dan vektor keempat mewakili emosi "angry" dalam teks. *Confusion matrix* tercipta dari hasil pengujian dibandingkan dengan hasil yang sebenarnya, sehingga ditentukan apakah pengujian memiliki hasil yang baik atau tidak.

Diambilnya 20 teks dari sebuah dataset yang digunakan untuk pengujian sebagai contoh. Dari 20 teks tersebut, dilakukan *text preprocessing* dan *tokenizer*. Kemudian teks tersebut dilakukan prediksi dengan model yang sudah dilatih. Proses pengujian dilakukan dengan menggunakan metode LSTM. Output dari pengujian adalah nilai dari probabilitas terhadap masing-masing *class*. Berikut tabel 3.12 untuk contoh *output* dari hasil pengujian beserta *class* aslinya.

Tabel 3.12 Contoh *Output Testing* Dibandingkan dengan *Class* Aslinya

No	Output	Predicted	Actual
1	[0.23365922, 0.19826139, 0.4456871, 0.12239228]	2	2
2	[0.13365922, 0.5456871, 0.29826139, 0.02239228]	1	1
3	[0.57214479, 0.15294985, 0.02469444, 0.25021092]	0	0
4	[0.11000595, 0.17890865, 0.1726846, 0.5384008]	3	3
5	[0.146054, 0.56719807, 0.21479072, 0.0719572]	1	1
6	[0.14455366, 0.22312725, 0.56534016, 0.06697893]	2	1
7	[0.27490751, 0.2156633, 0.07040159, 0.4390276]	3	3
8	[0.3348164, 0.16867405, 0.05259408, 0.44391548]	3	3
9	[0.14306156, 0.1052263, 0.51716088, 0.23455126]	2	2
10	[0.38866931, 0.33897824, 0.19337403, 0.07897841]	0	1

Tabel 3.12 Contoh *Output Testing* Dibandingkan dengan *Class* Aslinya (Lanjutan)

No	Output	Predicted	Actual
11	[0.08950106, 0.09899153, 0.73787686, 0.07363056]	2	2
12	[0.26694779, 0.18266468, 0.20047083, 0.34991671]	3	3
13	[0.30817482, 0.20555882, 0.14596202, 0.34030434]	3	0
14	[0.42657658, 0.20337053, 0.20509333, 0.16495956]	0	1
15	[0.06900305, 0.00994723, 0.89049897, 0.03055075]	2	2
16	[0.10074773, 0.85664745, 0.01296517, 0.02963965]	1	1
17	[0.83261179, 0.01383494, 0.06257871, 0.09097456]	0	0
18	[0.19484142, 0.10872772, 0.40056537, 0.2958655]	2	3
19	[0.36294168, 0.09103902, 0.28149261, 0.26452669]	0	2
20	[0.14287557, 0.15106955, 0.25815884, 0.44789605]	3	3

Confusion matrix terbentuk dari tabel 3.12 dengan pembagian hasil *predicted* dibandingkan dengan hasil *actual* tercipta tabel *confusion matrix* untuk *multiclass*. Berikut tabel 3.13 *confusion matrix* yang tercipta.

Tabel 3.13 Contoh *Confusion Matrix* 4x5 dengan 4 *Class*

		Predicted			
		Others	Happy	Sad	Angry
Actual	Others	2	0	0	1
	Happy	2	3	1	0
	Sad	1	0	4	0
	Angry	0	0	1	5

Dari tabel 3.13 bisa didapatkan bahwa untuk kelas *Others* ditemukan nilai *True Positive* sebanyak 2, nilai *True Negative* sebanyak 14, nilai *False Positive* sebanyak 3, dan nilai *False Negative* sebanyak 1. Kelas *Happy* ditemukan nilai *True Positive* sebanyak 3, nilai *True Negative* sebanyak 16, nilai *False Positive* sebanyak 0, dan nilai *False Negative* sebanyak 3. Kelas *Sad* ditemukan nilai *True Positive* sebanyak 4, nilai *True Negative* sebanyak 13, nilai *False Positive* sebanyak 2, dan nilai *False Negative* sebanyak 1. Terakhir kelas *Angry* ditemukan nilai *True Positive* sebanyak 5, nilai *True Negative* sebanyak 13, nilai *False Positive* sebanyak 1, dan nilai *False Negative* sebanyak 1. Dari hasil tersebut, maka ditemukan nilai akurasi dari

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

pengujian. Berikut perhitungannya dengan menggunakan persamaan 2.11 untuk mendapatkan nilai akurasi.

$$\text{Accuracy} = \frac{2+3+4+5}{20} = 0.7$$

Maka ditemukan juga nilai *Precision*, *Recall*, dan *F1 Score* dari hasil tabel *confusion matrix* tersebut. Berikut penghitungan *recall* menggunakan rumus dari 2.12.

$$\text{Recall} = \frac{(2/(2+1)) + (3/(3+3)) + (3/(4+1)) + (5/(5+1))}{4} = 0.65$$

Kemudian perhitungan untuk *precision* dilakukan dengan menggunakan rumus dari 2.13. Berikut perhitungan untuk *precision*.

$$\text{Precision} = \frac{(2/(2+3)) + (3/(3+0)) + (3/(4+2)) + (5/(5+1))}{4} = 0.683$$

Pada perhitungan terakhir, dihitungnya nilai *F1 Score* yang didapat dari penghitungan *precision* dan *recall* di atas menggunakan rumus 2.14. Berikut perhitungan untuk *f1 score*.

$$\text{F1Score} = \frac{2 \times 0.683 \times 0.65}{0.683 + 0.65} = 0.666$$

Dari tabel 3.13 bisa didapatkan hasil *accuracy*, *recall*, *precision*, dan *F1Score* untuk setiap kelasnya. Berikut hasil dari perhitungan *accuracy*, *recall*, *precision*, dan *F1Score* yang dilihat dalam tabel 3.14.

Tabel 3.14 *Classification Report* dengan 4 Class

	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1Score</i>
<i>Others</i>	0.80	0.40	0.67	0.50
<i>Happy</i>	0.85	1	0.50	0.67
<i>Sad</i>	0.85	0.67	0.80	0.73
<i>Angry</i>	0.90	0.83	0.83	0.83

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan mengenai proses implementasi dan pengujian yang dilakukan terhadap sistem yang telah dibangun dalam penelitian ini. Proses implementasi dan pengujian tersebut merupakan bagian dari tujuan akhir dalam proses penelitian tugas akhir ini.

4.1 Lingkungan Implementasi

Bagian ini membahas mengenai perangkat yang digunakan dalam proses pembangunan sistem. Perangkat yang digunakan terdiri dari perangkat keras dan perangkat lunak.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi dari perangkat keras yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. *Laptop* ASUS VivoBook S14 S410U.
2. *Processor* Intel Core i5-8250U CPU @ 3.4GHz.
3. *Solid State Drive* kapasitas 512 GB.
4. RAM dengan kapasitas 16 GB.

4.1.2 Spesifikasi Perangkat Lunak

Lingkungan perangkat lunak yang digunakan dalam pembangunan aplikasi adalah sebagai berikut:

1. Sistem Operasi Windows 10.
2. IDE: Google Colaboratory Pro.
3. *Development Tools* Anaconda 3, 2020.07 (Python 3.8.3 64-bit).

4.2 Implementasi Perangkat Lunak

Bagian ini menjelaskan mengenai seluruh kelas dan metode, penggunaan *Google Colaboratory* dan penggunaan dataset pada implementasi perangkat lunak. Bagian ini juga menjelaskan secara rinci masukan dan kegunaan dari setiap metode di setiap kelas.

4.2.1 Daftar *Class* dan *Method*

Bagian ini menjelaskan seluruh *class* dan *method* yang digunakan pada penelitian ini. *Method* dikelompokkan ke dalam sebuah *class* sesuai dengan fungsinya. Berikut adalah penjelasan dari setiap *method* yang dikelompokkan dalam sebuah *class* yang digunakan dalam penelitian ini.

4.2.1.1 *Class Dataset*

Class Dataset digunakan untuk mengambil dataset dan melihat analisis persebaran data dalam dataset sebelum digunakan untuk proses selanjutnya. Tabel 4.1 menunjukkan *method* yang digunakan dalam *Class Dataset*.

Tabel 4.1 Daftar *method* pada *Class Dataset*

No.	Method	Input	Output	Description
1.	load_dataset_text	-	dataset: dataframe	Digunakan untuk <i>load</i> dataset.
2.	show_histogram_dataset	dataset: dataframe	-	Memberikan visualisasi persebaran dataset setiap <i>class</i> .

4.2.1.2 *Class Preprocessing*

Class Preprocessing digunakan untuk melakukan tahap *preprocessing* pada teks. *Class* ini mengolah data teks yang selanjutnya digunakan untuk diproses pada *Class TokenizerWord*. Tabel 4.2 menunjukkan daftar *method* yang digunakan dalam *Class Preprocessing*.

Tabel 4.2 Daftar *method* pada *Class Preprocessing*

No.	Method	Input	Output	Description
1.	find_antonym_word	word: string word	word: string	Digunakan untuk mencari lawan kata.
2.	replace_negation_word	text: string	new_text: string	Digunakan untuk menggantikan kata negasi di dalam teks.
3.	spell_check	text: string	new_text: string	Digunakan untuk menggantikan kata ke bahasa yang baku di dalam teks.

4.	text_preprocessing	text: string	new_text: string	Digunakan untuk membersihkan teks.
5.	show_wordcloud	text: string	-	Memberikan visualisasi data teks yang sudah dilakukan preprocessing.

4.2.1.3 Class TokenizerWord

Class TokenizerWord digunakan untuk melakukan tahap *tokenizer* pada teks. *Class* ini mengolah data teks hasil proses dari *Class Preprocessing* ke bentuk vektor sehingga dilakukan pelatihan pada model. Tabel 4.3 menunjukkan daftar *method* yang digunakan dalam *Class TokenizerWord*.

Tabel 4.3 Daftar *method* pada *Class TokenizerWord*

No.	Method	Input	Output	Description
1.	get_unique_words	arr_sentences: <i>array of string</i>	unique_words: jumlah kata unik dalam dataset, max_word_length: jumlah maksimal kata yang ada dalam teks	Digunakan untuk mencari jumlah kata yang unik dan jumlah panjang kata terbanyak dalam 1 teks.
2.	n_gram	text: string, n: jumlah kata yang ingin dipecah	token: <i>array of string</i>	Digunakan untuk memisahkan teks menjadi potongan-potongan berupa token.

3.	padded_sequence_tokenizer	X_train: data teks untuk pelatihan, X_test: data teks untuk pengujian, max_word_length: jumlah maksimal kata yang ada dalam teks	padded_train: data teks pelatihan yang diubah menjadi bentuk vektor, padded_test: data teks pengujian yang diubah menjadi bentuk vektor	Digunakan untuk mengubah data teks menjadi bentuk vektor.
----	---------------------------	--	---	---

4.2.1.4 Class EmbeddingGlove

Class EmbeddingGlove digunakan untuk mengambil dan memproses data *embedding*. *Class* ini mengambil data *embedding* dan mengolahnya sehingga bisa digunakan dalam model. Hasil proses dari *Class EmbeddingGlove* digunakan untuk proses pembuatan pada model pada lapisan *Embedding*. Tabel 4.4 menunjukkan daftar *method* yang digunakan dalam *Class EmbeddingGlove*.

Tabel 4.4 Daftar *method* pada *Class EmbeddingGlove*

No.	Method	Input	Output	Description
1.	load_embedding	-	-	Digunakan untuk mengunduh data <i>embedding GloVe</i> dan menyimpannya dalam folder.
2.	read_glove_text	file_url: string path	embedding_index: <i>dictionary of key string & array of float value</i>	Digunakan untuk membaca data <i>embedding GloVe</i> dalam file dan menyimpannya dalam bentuk <i>dictionary</i> .

3.	initialize _embeddings_glove	file_url: string path, embedding _dimension: integer	embedding _matrix: array 2 dimension of float	Digunakan untuk menghasilkan <i>embedding matrix</i> .
----	---------------------------------	--	--	--

4.2.1.5 Class GenerateModel

Class GenerateModel digunakan untuk membuat, melatih, menguji, dan menganalisis model. *Class* ini membuat sebuah model, melatih model, dan menguji coba model yang dibuat. Hasil dari uji coba pada model menghasilkan analisis untuk menghasilkan laporan terhadap model. Tabel 4.5 menunjukkan daftar *method* yang digunakan dalam *Class GenerateModel*.

Tabel 4.5 Daftar *method* pada *Class GenerateModel*

No.	Method	Input	Output	Description
1.	create_model	dimension: integer, embedding_matrix _weight: array 2 dimension of float, dropout_rate: float, learning_rate: float	model: <i>sequential model</i>	Digunakan untuk membuat lapisan pada model.
2.	show_plot _accuracy_model	model: <i>sequential model</i>	-	Digunakan untuk membuat plot akurasi selama proses pelatihan.
3.	show_plot _loss_model	model: <i>sequential model</i>	-	Digunakan untuk membuat plot <i>loss</i> selama proses pelatihan.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

4.	<code>show_confusion_matrix</code>	<code>Y_actual: array of float, Y_predicted: array of float</code>	<code>df_cm: dataframe confusion matrix</code>	Digunakan untuk membuat <i>plot confusion matrix</i> selama proses pengujian.
5.	<code>create_prediction_classification_report</code>	<code>model: sequential model, X_test: array 2 dimension of float, Y_test: array of float</code>	<code>predictions: array of float</code>	Digunakan untuk membuat laporan teks yang menunjukkan metrik klasifikasi utama.
6.	<code>show_evaluate_model</code>	<code>model: sequential model, X_test: array 2 dimension of float, Y_test: array of float</code>	-	Digunakan untuk melihat hasil evaluasi dari model.
7.	<code>get_accuracy_class_others</code>	<code>df_cm: dataframe confusion matrix</code>	<code>accuracy_others: float</code>	Digunakan untuk menghasilkan akurasi dari kelas <i>others</i> .
8.	<code>get_accuracy_class_happy</code>	<code>df_cm: dataframe confusion matrix</code>	<code>accuracy_happy: float</code>	Digunakan untuk menghasilkan akurasi dari kelas <i>happy</i> .
9.	<code>get_accuracy_class_sad</code>	<code>df_cm: dataframe confusion matrix</code>	<code>accuracy_sad: float</code>	Digunakan untuk menghasilkan akurasi dari kelas <i>sad</i> .
10.	<code>get_accuracy_class_angry</code>	<code>df_cm: dataframe confusion matrix</code>	<code>accuracy_angry: float</code>	Digunakan untuk menghasilkan akurasi dari kelas <i>angry</i> .
11.	<code>show_accuracy_each_class</code>	<code>df_cm: dataframe confusion matrix</code>	-	Digunakan untuk melihat akurasi dari setiap kelas.
12.	<code>save_model</code>	<code>model: sequential model, index: integer</code>	-	Digunakan untuk menyimpan model.

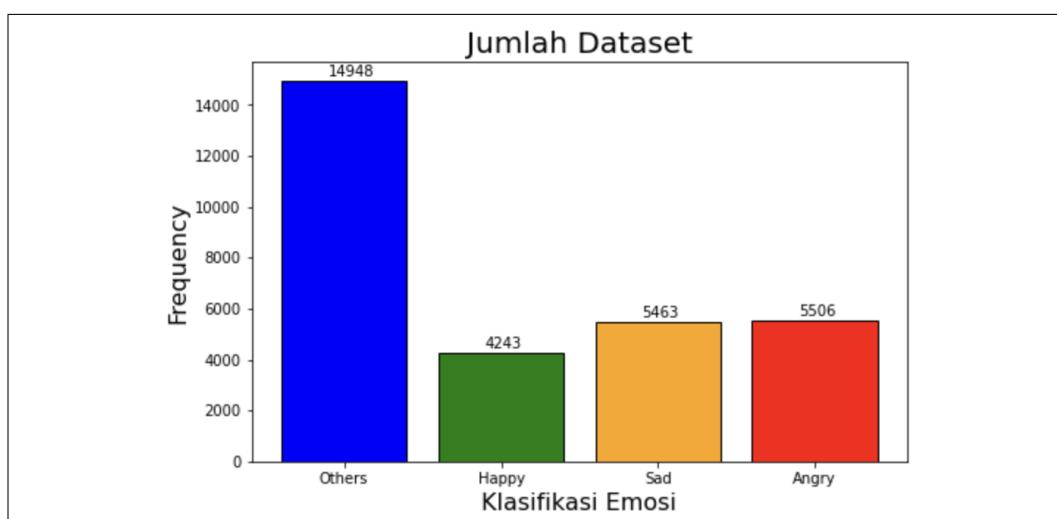
13.	load_model	index: integer	-	Digunakan untuk memuat model yang telah disimpan.
-----	------------	----------------	---	---

4.2.2 Penggunaan Google Colaboratory

Dalam penelitian ini, *tools* yang digunakan untuk melakukan pelatihan dan pengujian adalah Jupyter Notebook. Google Colaboratory merupakan penyedia Jupyter Notebook secara *online* oleh Google yang memiliki *resource* GPU/TPU dan RAM yang besar. Google Colaboratory dipilih sebagai alternatif dibandingkan dengan Jupyter Notebook yang dijalankan pada komputer lokal karena memiliki *resource* yang lebih besar dan jika dijalankan pada komputer lokal, memperlambat proses implementasi aplikasi.

4.2.3 Implementasi Penggunaan Dataset

Pada penelitian ini, digunakannya sebuah dataset yang diambil dari [20] yang merupakan data dari *Computational Linguistics at Concordia* (CLaC) Lab. Seluruh data yang digunakan, dilakukannya *text preprocessing* untuk memproses data yang masih kotor menjadi data yang siap digunakan untuk pelatihan model. Kemudian data tersebut dibagi dengan komposisi 80% untuk proses pelatihan dan 20% untuk proses pengujian. Dari jumlah data proses pelatihan, data diambil 20% untuk dijadikan validasi. Data validasi digunakan untuk menguji model selama proses pelatihan berjalan untuk mengetahui kualitas model. Jumlah pembagian data yang digunakan pada penelitian ini adalah 24.128 untuk pelatihan dan 6.032 untuk pengujian.



Gambar 4.1 Persebaran Data Setiap Kelas

Pada gambar 4.1 dijelaskan persebaran seluruh data untuk setiap kelas yang ada di

dalam dataset. Dalam dataset terdapat 14.948 kelas *others*, 4.243 kelas *happy*, 5.463 kelas *sad*, dan 5.506 kelas *angry*. Seluruh data ini digunakan dalam penelitian ini untuk melakukan pelatihan dan pengujian terhadap model.

4.2.4 Implementasi Penggunaan *Pretrained GloVe*

Implementasi penggunaan *word embedding* dari *pretrained* Glove digunakan untuk mengubah kata menjadi vektor dengan mengambil vektor yang ada pada *pretrained* Glove yang disimpan dalam bentuk matriks *embedding* yang terdiri dari vektor yang mewakili kata. Matriks ini digunakan sebagai *input* untuk *weight* pada *Embedding Layer* yang dimana mengubah *sequence* kata menjadi vektor sebelum dilakukan penghitungan LSTM. Gambar 4.2 adalah contoh implementasi dari penggunaan GloVe di satu kata:

```
array([-0.13128 , -0.45199999,  0.043399 , -0.99798 , -0.21053 ,  
-0.95867997, -0.24608999,  0.48413 ,  0.18178 ,  0.47499999,  
-0.22305 ,  0.30063999,  0.43496001, -0.36050001,  0.20245001,  
-0.52594 , -0.34707999,  0.0075873 , -1.04970002,  0.18673 ,  
0.57369 ,  0.43814 ,  0.098659 ,  0.38769999, -0.22579999,  
0.41911 ,  0.043602 , -0.73519999, -0.53583002,  0.19276001,  
-0.21961001,  0.42515001, -0.19081999,  0.47187001,  0.18826 ,  
0.13357 ,  0.41839001,  1.31379998,  0.35677999, -0.32172 ,  
-1.22570002, -0.26635 ,  0.36715999, -0.27586001, -0.53245997,  
0.16786 , -0.11253 , -0.99958998, -0.60706002, -0.89270997,  
0.65156001, -0.88783997,  0.049233 ,  0.67110997, -0.27553001,  
-2.40050006, -0.36989 ,  0.29135999,  1.34979999,  1.73529994,  
0.27000001,  0.021299 ,  0.14421999,  0.023784 ,  0.33643001,  
-0.35475999,  1.09210002,  1.48450005,  0.49430001,  0.15688001,  
0.34678999, -0.57221001,  0.12093 , -1.26160002,  1.05410004,  
0.064335 , -0.002732 ,  0.19038001, -1.76429999,  0.055068 ,  
1.47370005, -0.41782001, -0.57341999, -0.12129 , -1.31690001,  
-0.73882997,  0.17682 , -0.019991 , -0.49175999, -0.55247003,  
1.06229997, -0.62879002,  0.29098001,  0.13237999, -0.70414001,  
0.67128003, -0.085462 , -0.30526 , -0.045495 ,  0.56509 ])
```

Gambar 4.2 Ilustrasi Vektor dari Penggunaan *Pretrained GloVe*

Gambar 4.2 adalah representasi contoh kata 'said' dalam bentuk vektor yang diambil dari *pretrained* GloVe. Dari implementasi penggunaan *word embedding* dari GloVe, tidak semua kata diambil vektor katanya yang disebabkan karena terdapat beberapa kata yang mengalami salah pengetikan atau kurangnya spasi antar kata yang menyebabkan tidak ditemukannya vektor kata pada Glove. Untuk kata yang tidak ditemukan dalam GloVe, maka diisikan dengan vektor berisikan 0.

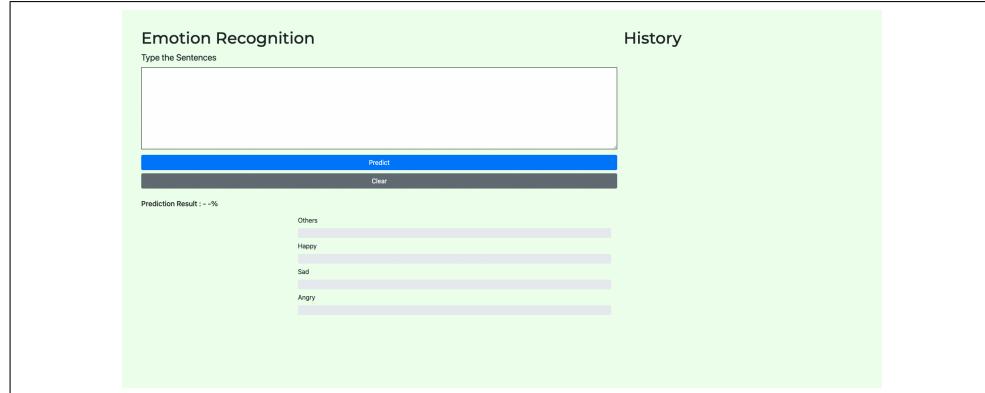
4.2.5 Implementasi Aplikasi

Bagian ini menjelaskan *Graphical User Interface* (GUI) yang terdapat dalam aplikasi deteksi emosi dalam teks. GUI dibuat untuk mempermudah pengujian. Berikut tampilan dari GUI.

1. Tampilan utama aplikasi, dimintanya sebuah *input* teks untuk mengetahui apakah emosi dalam teks tersebut. Adanya tombol *predict* untuk melakukan

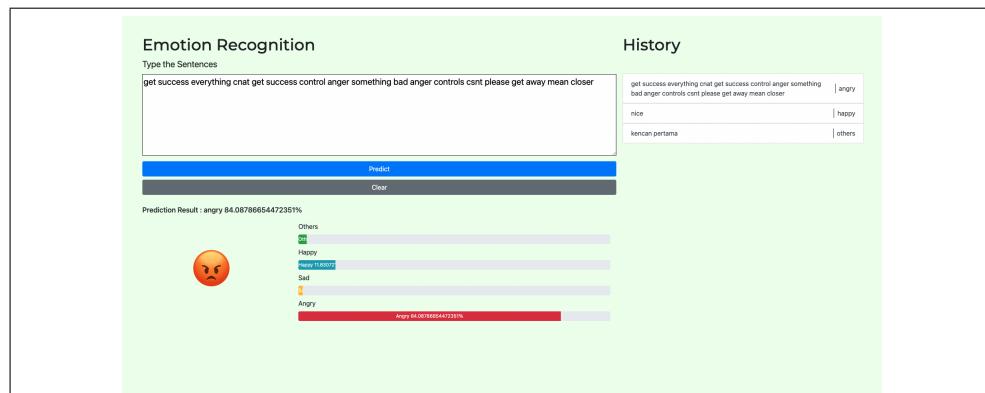
BAB 4 IMPLEMENTASI DAN PENGUJIAN

proses prediksi dan adanya tombol *clear* untuk menghapus isi teks. Berikut Gambar 4.3 untuk tampilan utama aplikasi.



Gambar 4.3 Tampilan Utama Aplikasi

- Setelah teks ditambahkan pada kolom teks dan menekan tombol *predict* sistem memproses hasilnya. Proses yang dilakukan yang berjalan yaitu membersihkan teks, mengubah teks menjadi *sequence*, *padding*, dan melakukan uji dengan *input* yang telah diberikan. Maka *output* menghasilkan performa akurasi yang didapat dan kelas yang dideteksi dalam teks. *History* menunjukkan sejarah prediksi yang sudah pernah dilakukan sebelumnya. Berikut gambar 4.4 untuk tampilan dengan hasil akurasi yang di dapat.



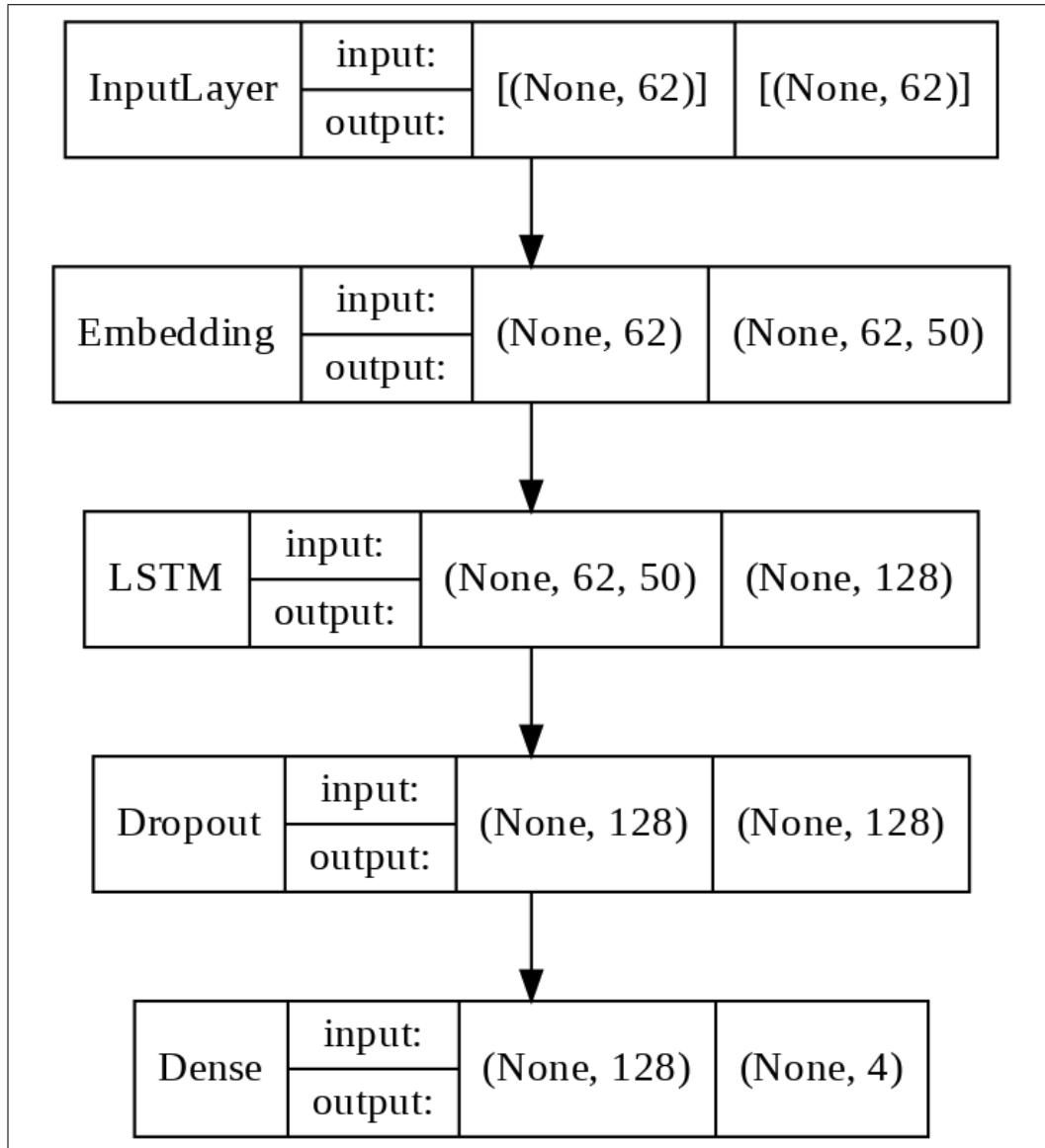
Gambar 4.4 Tampilan Keluaran Aplikasi

4.3 Pengujian

Bagian ini menjelaskan pengujian terhadap aplikasi deteksi emosi dalam teks yang telah dibuat. Pengujian yang dilakukan diantaranya adalah menguji kepengaruan *hyperparameter* terhadap model. Pengujian yang dilakukan dirincikan sebagai berikut.

4.3.1 Pengujian Arsitektur

Penelitian ini terdapat sebuah arsitektur yang diuji dengan sebuah dataset untuk mengetahui pengaruh dari penambahan *layer*. Arsitektur ini digunakan dari [5] dengan ditambahkannya layer *Dropout*. Gambar 4.5 menunjukkan rancangan arsitektur yang digunakan dalam penelitian ini:



Gambar 4.5 Rancangan Arsitektur dalam Penelitian

Gambar 4.5 memperlihatkan bahwa ada 4 lapisan yang digunakan ketika membuat model ini, keempat lapisan itu adalah:

1. *Input Layer* menerima *input* dengan panjang vektor sebanyak 62.
2. *Embedding Layer* dengan parameter *input_dim* yang merupakan jumlah *input dimension*, *output_dim* yang merupakan jumlah *output dimension*, *input_length* yang merupakan panjang dari *sequence*, *weights* yang merupakan matriks

embedding dari GloVe, dan *trainable* diberikan *input False* untuk menjaga nilai vektor tidak berubah. *Embedding layer* menerima *input* dengan panjang vektor 62 dan menghasilkan *output* dengan panjang vektor 62 x 50.

3. *Long Short Term Memory Layer* menerima *input* dengan ukuran panjang *sequence X dimensi GloVe*, yaitu 62 x 50 dan menghasilkan *output* dengan panjang vektor 128.
4. *Dropout Layer* dengan parameter rate yang merupakan fraksi unit *input* untuk dijatuhkan. *dropout layer* menerima *input* dengan panjang vektor 128 dan menghasilkan *output* dengan panjang vektor 128.
5. *Dense Layer* dengan fungsi aktivasi *softmax* yang digunakan sebagai *output layer* untuk deteksi. *Dense layer* menerima *input* dengan ukuran panjang vektor 128 dan menghasilkan *output* dengan panjang vektor 4.

4.3.2 Pengujian *Embedding Output Dimension & Embedding Weights*

Embedding Dimension merupakan panjang m dari ruang embedding vektor tunggal yang digunakan agar merekonstruksi ruang fase berturut-turut dari suatu proses dan *Embedding Weights* merupakan bobot dari sebuah kata yang ditunjukkan dalam bentuk vektor. Penggunaan *Embedding Output Dimension & Embedding Weights* sangat mempengaruhi proses pelatihan model yang disebabkan semakin besar vektor yang digunakan, maka semakin kompleks model pada saat pelatihan. Pengujian dilakukan untuk mengetahui *Embedding Output Dimension & Embedding Weights* yang paling optimal. Nilai *Embedding Output Dimension & Embedding Weights* yang digunakan adalah 50, 100, 200, dan 300.

4.3.3 Pengujian *Dropout Rate*

Dropout Rate merupakan teknik regularisasi jaringan syaraf dimana beberapa neuron dipilih secara acak dan tidak dipakai selama pelatihan. *Dropout* juga merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses pembelajaran. *Dropout* mengacu kepada menghilangkan neuron yang berupa *hidden* mapun lapisan yang terlihat di dalam jaringan. Dengan menghilangkan suatu neuron, berarti menghilangkannya sementara dari jaringan yang ada. Pengujian dilakukan untuk mengetahui nilai *dropout rate* yang optimal. Nilai *dropout rate* yang diuji adalah 0.1, 0.2, 0.3, dan 0.4.

4.3.4 Pengujian Nilai *Learning Rate*

Learning rate merupakan *hyperparameter* yang mempengaruhi kecepatan belajar dalam proses pelatihan. Penggunaan *learning rate* yang besar menyebabkan perubahan performa yang besar dan cepat tetapi berlaku juga sebaliknya dengan pengorbanan memakan waktu yang sangat lama untuk dilatih jika menggunakan *learning rate* yang lebih rendah. Pengujian dilakukan untuk mengetahui nilai *learning rate* yang optimal. Nilai *learning rate* yang diuji adalah 0.1, 0.01, 0.001 dan 0.0001.

4.4 Hasil Pengujian

Pada bagian ini dituliskan hasil pengujian yang sudah dilakukan berdasarkan pemaparan yang sudah dijelaskan pada bagian 4.3. Hasil pengujian yang dilakukan berupa pengujian nilai *Embedding Output Dimension & Embedding Weights*, *dropout rate*, dan *learning rate*. Hasil pengujian ini dikelompokkan berdasarkan *embedding dimension* dan *Embedding Weights*, dimana setiap kelompok tersebut memperlihatkan hasil performanya dengan menggunakan nilai *precision*, *recall*, *F1 Score*, dan akurasi untuk setiap kelas.

4.4.1 Hasil Pengujian 50 *Embedding Dimension & Embedding Weights*

Pengujian dilakukan dengan nilai *Embedding Output Dimension* dan *Embedding Weights* sebesar 50. Pengujian dilakukan dengan pengelompokan setiap kelas. Setiap pengujian menggunakan arsitektur yang sama dan *hyperparameter* yang berbeda-beda. Pengujian mencari hasil yang paling optimal untuk setiap kelas.

4.4.1.1 Hasil Pengujian 50 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

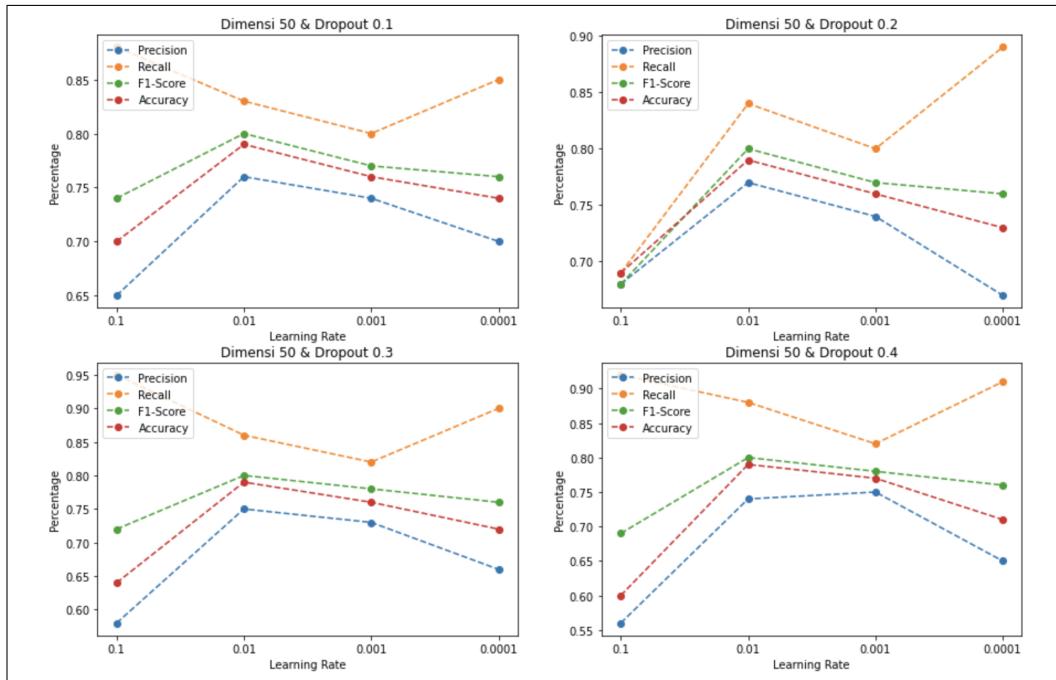
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 50 pada kelas *Others* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 79% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.1, 0.2, 0.3, dan 0.4. Berikut tabel 4.6 untuk rincian pengujianya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.6 Hasil Pengujian 50 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.1	0.1	65%	88%	74%	70%
2			0.01	76%	83%	80%	79%
3			0.001	74%	80%	77%	76%
4			0.0001	70%	85%	76%	74%
5		0.2	0.1	68%	69%	68%	69%
6			0.01	77%	84%	80%	79%
7			0.001	74%	80%	77%	76%
8			0.0001	67%	89%	76%	73%
9		0.3	0.1	58%	95%	72%	64%
10			0.01	75%	86%	80%	79%
11			0.001	73%	82%	78%	76%
12			0.0001	66%	90%	76%	72%
13		0.4	0.1	56%	92%	69%	60%
14			0.01	74%	88%	80%	79%
15			0.001	75%	82%	78%	77%
16			0.0001	65%	91%	76%	71%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.6 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 50 Pada Kelas *Others*

Berdasarkan gambar 4.6, didapatkan hasil pengujian dengan pengelompokan 50 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1, 0.2, 0.3, dan 0.4. Sedangkan nilai akurasi terendah sebesar 60% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 77% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.2. Sedangkan nilai *precision* terendah sebesar 56% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 95% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 69% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate*

sebesar 0.2.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 80% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1, 0.2, 0.3, dan 0.4. Sedangkan nilai *f1 score* terendah sebesar 68% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

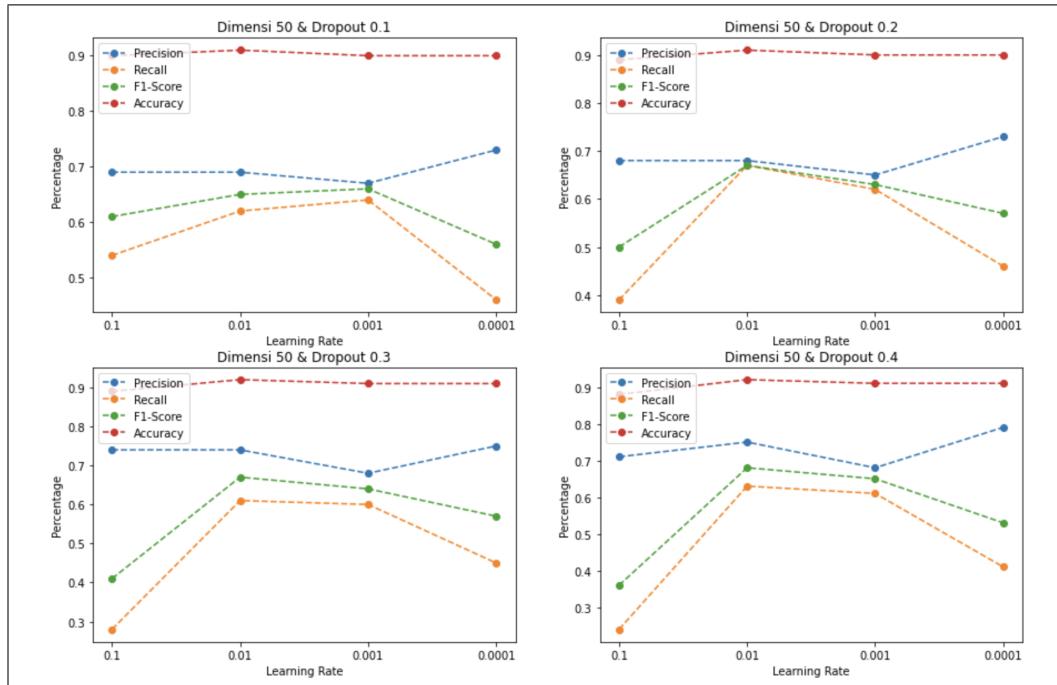
4.4.1.2 Hasil Pengujian 50 Embedding Dimension & Embedding Weights Pada Kelas Happy

Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 50 pada kelas *Happy* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 92% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.4. Berikut tabel 4.7 untuk rincian pengujinya.

Tabel 4.7 Hasil Pengujian 50 Embedding Dimension & Embedding Weights Pada Kelas Happy

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.1	0.1	69%	54%	61%	90%
2			0.01	69%	62%	65%	91%
3			0.001	67%	64%	66%	90%
4			0.0001	73%	46%	56%	90%
5		0.2	0.1	68%	39%	50%	89%
6			0.01	68%	67%	67%	91%
7			0.001	65%	62%	63%	90%
8			0.0001	73%	46%	57%	90%
9		0.3	0.1	74%	28%	41%	89%
10			0.01	74%	61%	67%	92%
11			0.001	68%	60%	64%	91%
12			0.0001	75%	45%	57%	91%
13		0.4	0.1	71%	24%	36%	88%
14			0.01	75%	63%	68%	92%
15			0.001	68%	61%	65%	91%
16			0.0001	79%	41%	53%	91%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.7 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 50 Pada Kelas *Happy*

Berdasarkan gambar 4.7 didapatkan hasil pengujian dengan pengelompokan 50 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 92% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai akurasi terendah sebesar 88% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 65% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 67% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.2. Sedangkan nilai *recall* terendah sebesar 24% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 68% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai *f1 score* terendah sebesar 36% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

4.4.1.3 Hasil Pengujian 50 Embedding Dimension & Embedding Weights Pada Kelas Sad

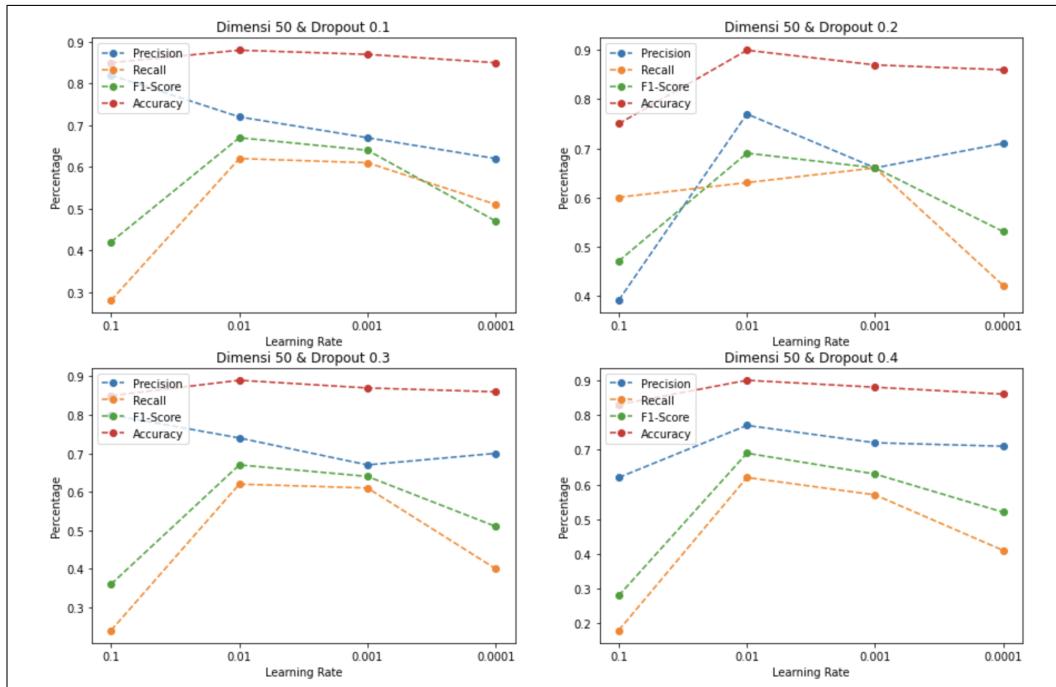
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 50 pada kelas *Others* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 90% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.2 dan 0.4. Berikut tabel 4.9 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.9 Hasil Pengujian 50 Embedding Dimension & Embedding Weights Pada Kelas Sad

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.1	0.1	82%	28%	42%	85%
2			0.01	72%	62%	67%	88%
3			0.001	67%	61%	64%	87%
4			0.0001	62%	51%	56%	85%
5		0.2	0.1	39%	60%	47%	75%
6			0.01	77%	63%	69%	90%
7			0.001	66%	66%	66%	87%
8			0.0001	71%	42%	53%	86%
9		0.3	0.1	80%	24%	36%	85%
10			0.01	74%	62%	67%	89%
11			0.001	67%	61%	64%	87%
12			0.0001	70%	40%	51%	86%
13		0.4	0.1	62%	18%	28%	83%
14			0.01	77%	62%	69%	90%
15			0.001	72%	57%	63%	88%
16			0.0001	71%	41%	52%	86%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.8 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 50 Pada Kelas Sad

Berdasarkan gambar 4.8 didapatkan hasil pengujian dengan pengelompokan 50 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 90% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.2 dan 0.4. Sedangkan nilai akurasi terendah sebesar 75% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 82% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1. Sedangkan nilai *precision* terendah sebesar 39% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 66% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.2. Sedangkan nilai *recall* terendah sebesar 18% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate*

sebesar 0.4.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 69% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.2 dan 0.4. Sedangkan nilai *f1 score* terendah sebesar 28% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

4.4.1.4 Hasil Pengujian 50 Embedding Dimension & Embedding Weights Pada Kelas Angry

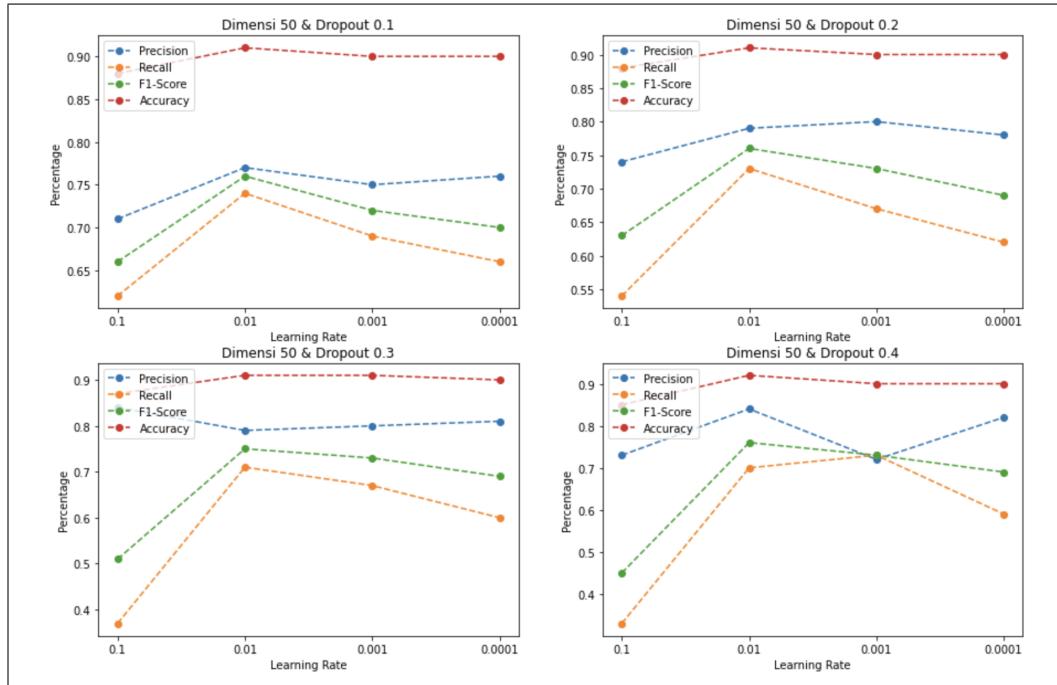
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 50 pada kelas *Angry* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 92% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.4. Berikut tabel 4.11 untuk rincian pengujinya.

Tabel 4.11 Hasil Pengujian 50 *Embedding Dimension & Embedding Weights* Pada Kelas *Angry*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.1	0.1	71%	62%	66%	88%
2			0.01	77%	74%	76%	91%
3			0.001	75%	69%	72%	90%
4			0.0001	76%	66%	70%	90%
5		0.2	0.1	74%	54%	63%	88%
6			0.01	79%	73%	76%	91%
7			0.001	80%	67%	73%	90%
8			0.0001	78%	62%	69%	90%
9		0.3	0.1	84%	37%	51%	87%
10			0.01	79%	71%	75%	91%
11			0.001	80%	67%	73%	91%
12			0.0001	81%	60%	69%	90%
13		0.4	0.1	73%	33%	45%	85%
14			0.01	84%	70%	76%	92%
15			0.001	72%	73%	73%	90%
16			0.0001	82%	59%	69%	90%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.9 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 50 Pada Kelas Angry

Berdasarkan gambar 4.9 didapatkan hasil pengujian dengan pengelompokan 50 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 92% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai akurasi terendah sebesar 85% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 84% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 71% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 74% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1. Sedangkan nilai *recall* terendah sebesar 33% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 76% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1 dan 0.2. Sedangkan nilai *f1 score* terendah sebesar 45% dengan menggunakan nilai parameter dimensi sebesar 50, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

4.4.2 Hasil Pengujian 100 *Embedding Dimension*

Pengujian dilakukan dengan nilai *Embedding Output Dimension* dan *Embedding Weights* sebesar 100. Pengujian dilakukan dengan pengelompokan setiap kelas. Setiap pengujian menggunakan arsitektur yang sama dan *hyperparameter* yang berbeda-beda. Pengujian mencari hasil yang paling optimal untuk setiap kelas.

4.4.2.1 Hasil Pengujian 100 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

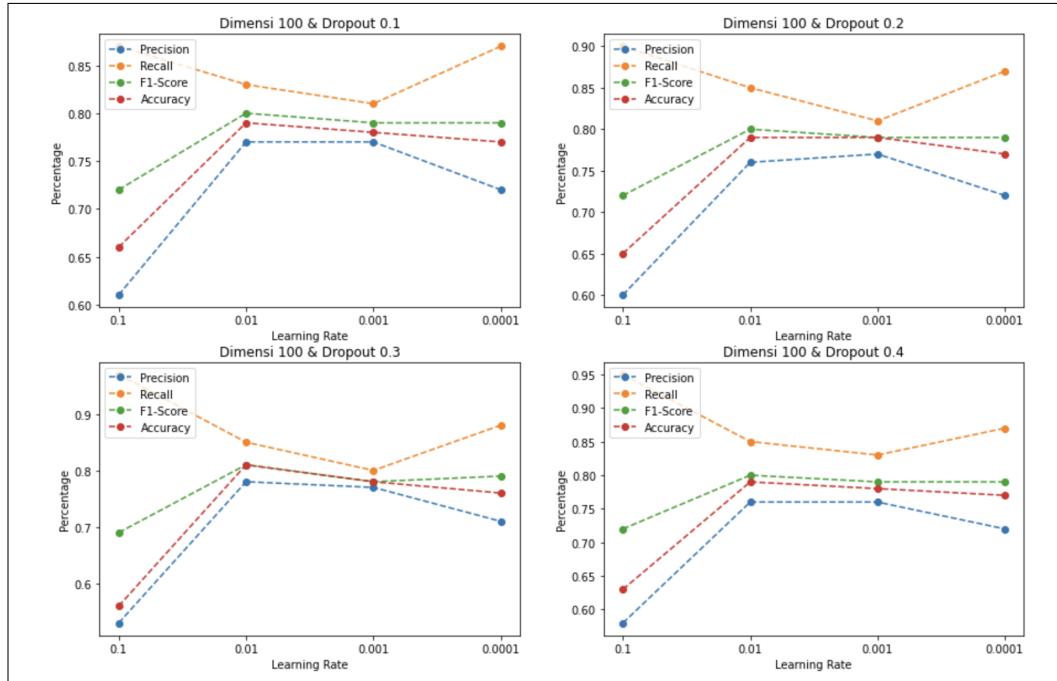
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 100 pada kelas *Others* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 81% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.3. Berikut tabel 4.12 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.12 Hasil Pengujian 100 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	100	0.1	0.1	61%	87%	72%	66%
2			0.01	77%	83%	80%	79%
3			0.001	77%	81%	79%	78%
4			0.0001	72%	87%	79%	77%
5		0.2	0.1	60%	90%	72%	65%
6			0.01	76%	85%	80%	79%
7			0.001	77%	81%	79%	79%
8			0.0001	72%	87%	79%	77%
9		0.3	0.1	53%	97%	69%	56%
10			0.01	78%	85%	81%	81%
11			0.001	77%	80%	78%	78%
12			0.0001	71%	88%	79%	76%
13		0.4	0.1	58%	95%	72%	63%
14			0.01	76%	85%	80%	79%
15			0.001	76%	83%	79%	78%
16			0.0001	72%	87%	79%	77%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.10 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 100 Pada Kelas *Others*

Berdasarkan gambar 4.10 didapatkan hasil pengujian dengan pengelompokan 100 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai akurasi terendah sebesar 56% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 78% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *precision* terendah sebesar 53% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 97% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 80% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.001, dan *dropout rate*

sebesar 0.3.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 69% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

4.4.2.2 Hasil Pengujian 100 Embedding Dimension & Embedding Weights Pada Kelas Happy

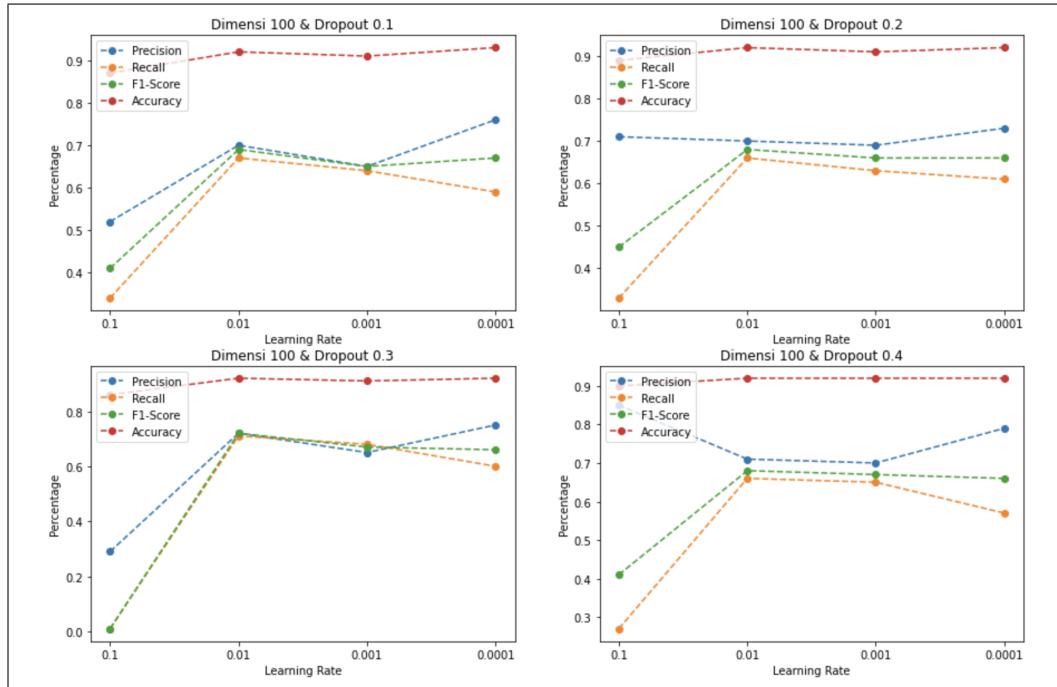
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 100 pada kelas *Happy* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 93% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.1. Berikut tabel 4.13 untuk rincian pengujinya.

Tabel 4.13 Hasil Pengujian 100 Embedding Dimension & Embedding Weights Pada Kelas Happy

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	100	0.1	0.1	52%	34%	41%	87%
2			0.01	70%	67%	69%	92%
3			0.001	65%	64%	65%	91%
4			0.0001	76%	59%	67%	93%
5		0.2	0.1	71%	33%	45%	89%
6			0.01	70%	66%	68%	92%
7			0.001	69%	63%	66%	91%
8			0.0001	73%	61%	66%	92%
9		0.3	0.1	29%	1%	1%	86%
10			0.01	72%	71%	72%	92%
11			0.001	65%	68%	67%	91%
12			0.0001	75%	60%	66%	92%
13		0.4	0.1	85%	27%	41%	90%
14			0.01	71%	66%	68%	92%
15			0.001	70%	65%	67%	92%
16			0.0001	79%	57%	66%	92%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.11 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 100 Pada Kelas *Happy*

Berdasarkan gambar 4.11 didapatkan hasil pengujian dengan pengelompokan 100 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 93% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai akurasi terendah sebesar 86% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 85% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 29% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 71% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 1% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 72% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 1% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

4.4.2.3 Hasil Pengujian 100 Embedding Dimension & Embedding Weights Pada Kelas Sad

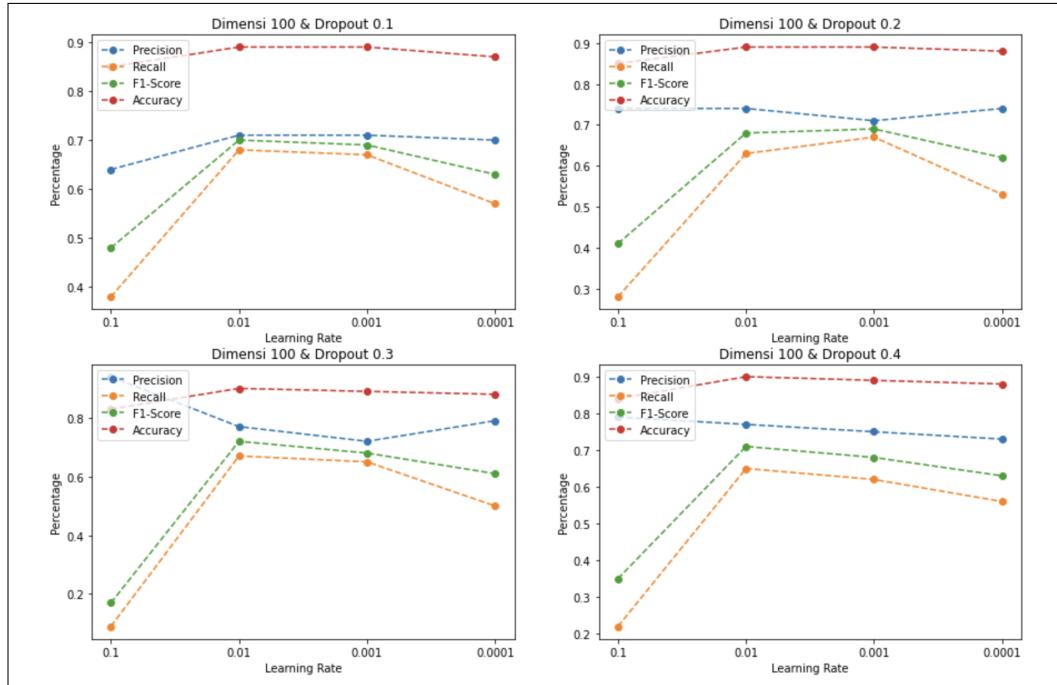
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 100 pada kelas *Sad* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 90% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.3 dan 0.4. Berikut tabel 4.14 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.14 Hasil Pengujian 100 *Embedding Dimension & Embedding Weights* Pada Kelas Sad

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	100	0.1	0.1	64%	38%	48%	85%
2			0.01	71%	68%	70%	89%
3			0.001	71%	67%	69%	89%
4			0.0001	70%	57%	63%	87%
5		0.2	0.1	74%	28%	41%	85%
6			0.01	74%	63%	68%	89%
7			0.001	71%	67%	69%	89%
8			0.0001	74%	53%	62%	88%
9		0.3	0.1	94%	9%	17%	83%
10			0.01	77%	67%	72%	90%
11			0.001	72%	65%	68%	89%
12			0.0001	79%	50%	61%	88%
13		0.4	0.1	79%	22%	35%	84%
14			0.01	77%	65%	71%	90%
15			0.001	75%	62%	68%	89%
16			0.0001	73%	56%	63%	88%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.12 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 100 Pada Kelas Sad

Berdasarkan gambar 4.12 didapatkan hasil pengujian dengan pengelompokan 100 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 90% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3 dan 0.4. Sedangkan nilai akurasi terendah sebesar 83% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 94% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3. Sedangkan nilai *precision* terendah sebesar 64% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 68% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1. Sedangkan nilai *recall* terendah sebesar 9% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate*

sebesar 0.3.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 72% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 17% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

4.4.2.4 Hasil Pengujian 100 Embedding Dimension & Embedding Weights Pada Kelas Angry

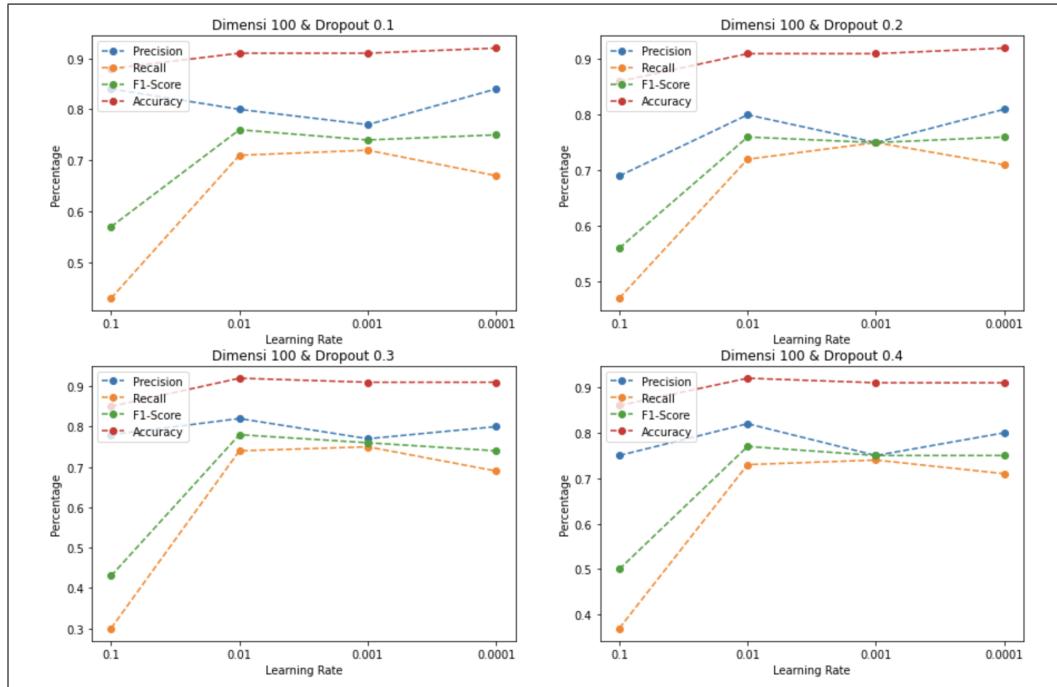
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 100 pada kelas *Angry* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 92% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.1 dan 0.2. Berikut tabel 4.15 untuk rincian pengujiannya.

Tabel 4.15 Hasil Pengujian 100 Embedding Dimension & Embedding Weights Pada Kelas Angry

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	100	0.1	0.1	84%	43%	57%	88%
2			0.01	80%	71%	76%	91%
3			0.001	77%	72%	74%	91%
4			0.0001	84%	67%	75%	92%
5		0.2	0.1	69%	47%	56%	86%
6			0.01	80%	72%	76%	91%
7			0.001	75%	75%	75%	91%
8			0.0001	81%	71%	76%	92%
9		0.3	0.1	78%	30%	43%	85%
10			0.01	82%	74%	78%	92%
11			0.001	77%	75%	76%	91%
12			0.0001	80%	69%	74%	91%
13		0.4	0.1	75%	37%	50%	86%
14			0.01	82%	73%	77%	92%
15			0.001	75%	74%	75%	91%
16			0.0001	80%	71%	75%	91%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.13 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 100 Pada Kelas Angry

Berdasarkan gambar 4.13 didapatkan hasil pengujian dengan pengelompokan 100 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 92% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1 dan 0.2. Sedangkan nilai akurasi terendah sebesar 85% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 84% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai *precision* terendah sebesar 69% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 75% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 30% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 78% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 43% dengan menggunakan nilai parameter dimensi sebesar 100, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.3.

4.4.3 Hasil Pengujian 200 *Embedding Dimension*

Pengujian dilakukan dengan nilai *Embedding Output Dimension* dan *Embedding Weights* sebesar 200. Pengujian dilakukan dengan pengelompokan setiap kelas. Setiap pengujian menggunakan arsitektur yang sama dan *hyperparameter* yang berbeda-beda. Pengujian mencari hasil yang paling optimal untuk setiap kelas.

4.4.3.1 Hasil Pengujian 200 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

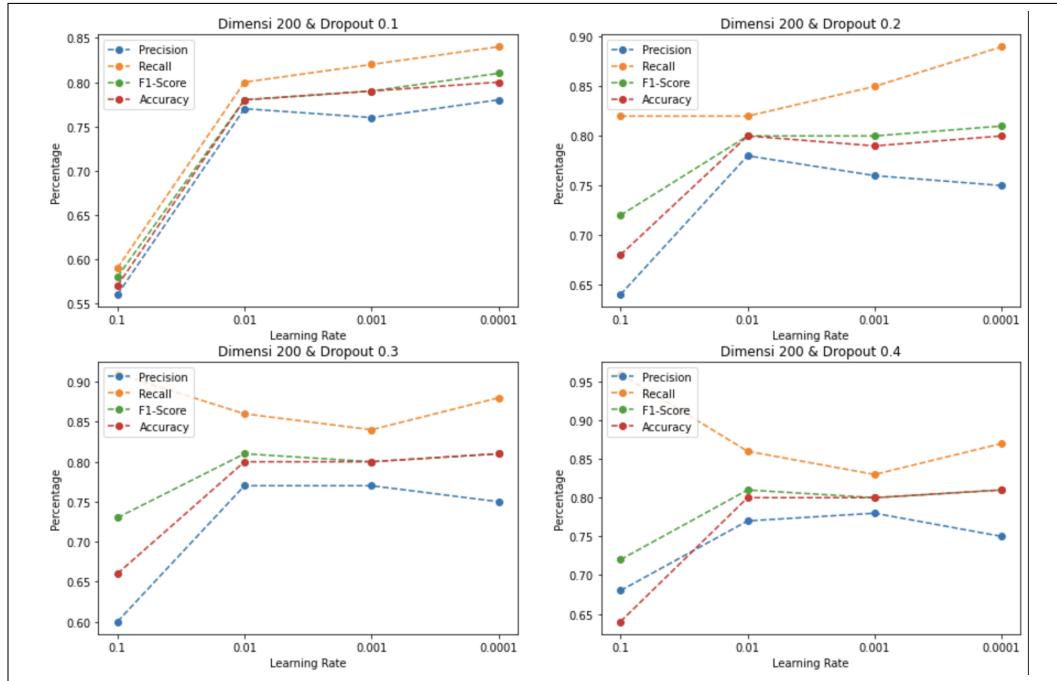
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 200 pada kelas *Others* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 81% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.3 dan 0.4. Berikut tabel 4.16 untuk rincian pengujiannya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.16 Hasil Pengujian 200 *Embedding Dimension & Embedding Weights* Pada Kelas *Others*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	200	0.1	0.1	56%	59%	58%	57%
2			0.01	77%	80%	78%	78%
3			0.001	76%	82%	79%	79%
4			0.0001	78%	84%	81%	80%
5		0.2	0.1	64%	82%	72%	68%
6			0.01	78%	82%	80%	80%
7			0.001	76%	85%	80%	79%
8			0.0001	75%	89%	81%	80%
9		0.3	0.1	60%	91%	73%	66%
10			0.01	77%	86%	81%	80%
11			0.001	77%	84%	80%	80%
12			0.0001	75%	88%	81%	81%
13		0.4	0.1	68%	96%	72%	64%
14			0.01	77%	86%	81%	80%
15			0.001	78%	83%	80%	80%
16			0.0001	75%	87%	81%	81%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.14 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 200 Pada Kelas *Others*

Berdasarkan gambar 4.14 didapatkan hasil pengujian dengan pengelompokan 200 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.3 dan 0.4. Sedangkan nilai akurasi terendah sebesar 57% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 78% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai *precision* terendah sebesar 56% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 96% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 30% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate*

sebesar 0.4.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1, 0.2, 0.3, dan 0.4. Sedangkan nilai *f1 score* terendah sebesar 58% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

4.4.3.2 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Happy

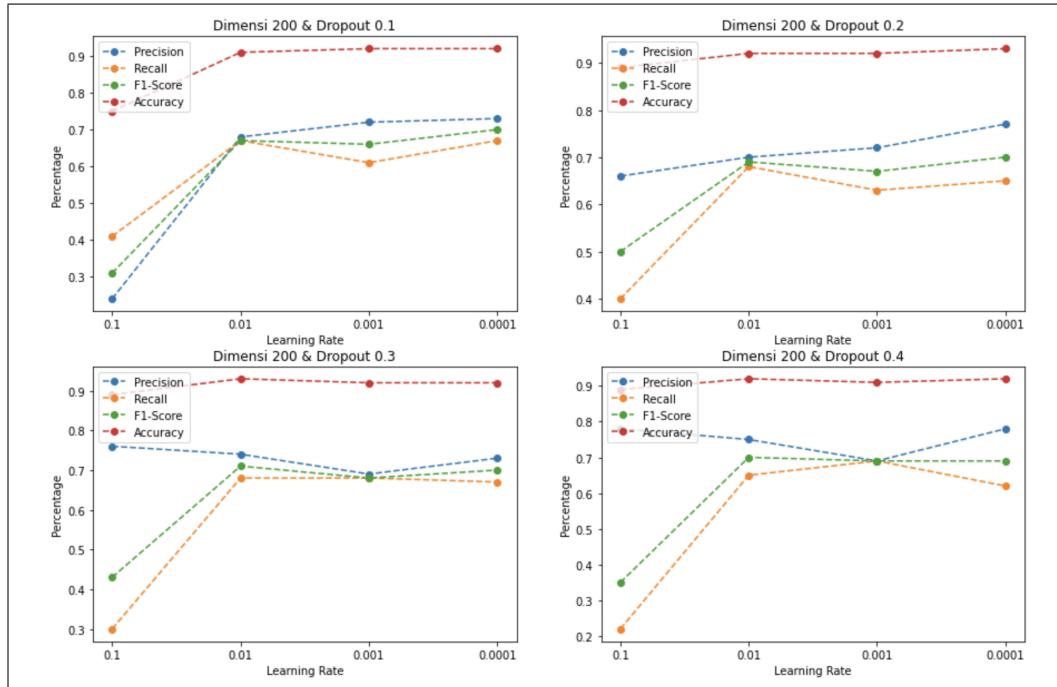
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 200 pada kelas *Happy* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 93% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.3. Berikut tabel 4.17 untuk rincian pengujinya.

Tabel 4.17 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Happy

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	200	0.1	0.1	24%	41%	31%	75%
2			0.01	68%	67%	67%	91%
3			0.001	72%	61%	66%	92%
4			0.0001	73%	67%	70%	92%
5		0.2	0.1	66%	40%	50%	89%
6			0.01	70%	68%	69%	92%
7			0.001	72%	63%	67%	92%
8			0.0001	77%	65%	70%	93%
9		0.3	0.1	76%	30%	43%	89%
10			0.01	74%	68%	71%	93%
11			0.001	69%	68%	68%	92%
12			0.0001	73%	67%	70%	92%
13		0.4	0.1	78%	22%	35%	89%
14			0.01	75%	65%	70%	92%
15			0.001	69%	69%	69%	91%
16			0.0001	78%	62%	69%	92%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.15 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 200 Pada Kelas *Happy*

Berdasarkan gambar 4.15 didapatkan hasil pengujian dengan pengelompokan 200 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 93% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai akurasi terendah sebesar 75% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 78% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 24% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 69% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *recall* terendah sebesar 22% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 71% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 31% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

4.4.3.3 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Sad

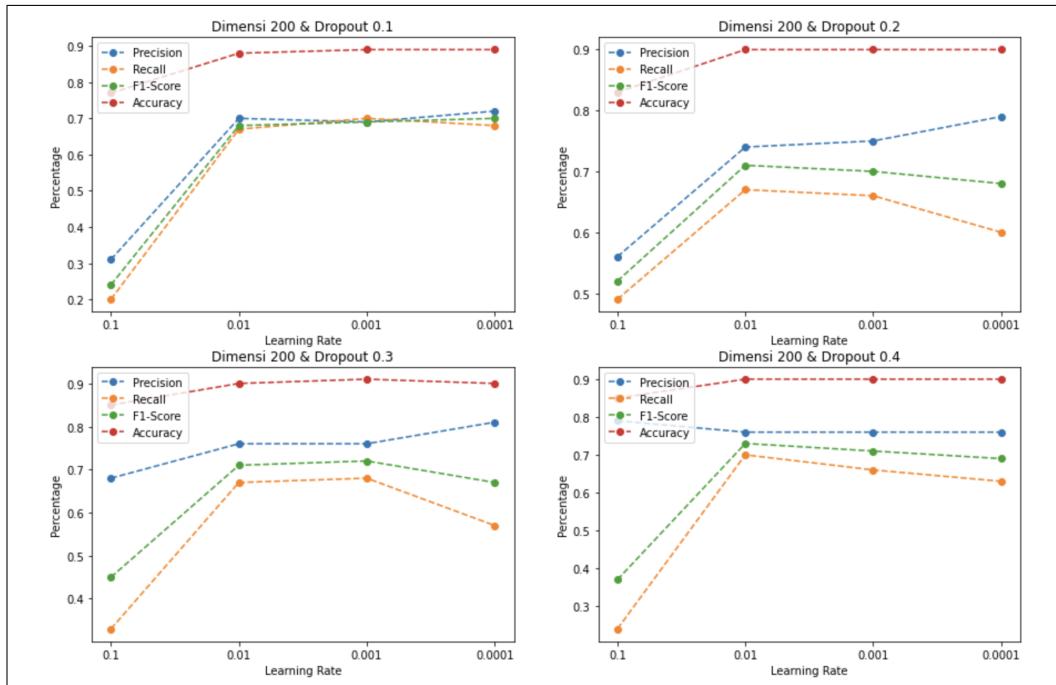
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 200 pada kelas *Sad* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 91% dengan parameter *learning rate* sebesar 0.001 dan *dropout rate* sebesar 0.3. Berikut tabel 4.18 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.18 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Sad

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	200	0.1	0.1	31%	20%	24%	77%
2			0.01	70%	67%	68%	88%
3			0.001	69%	70%	69%	89%
4			0.0001	72%	68%	70%	89%
5		0.2	0.1	56%	49%	52%	83%
6			0.01	74%	67%	71%	90%
7			0.001	75%	66%	70%	90%
8			0.0001	79%	60%	68%	90%
9		0.3	0.1	68%	33%	45%	85%
10			0.01	76%	67%	71%	90%
11			0.001	76%	68%	72%	91%
12			0.0001	81%	57%	67%	90%
13		0.4	0.1	79%	24%	37%	85%
14			0.01	76%	70%	73	90%
15			0.001	76%	66%	71%	90%
16			0.0001	76%	63%	69%	90%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.16 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 200 Pada Kelas Sad

Berdasarkan gambar 4.16 didapatkan hasil pengujian dengan pengelompokan 200 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 91% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.3. Sedangkan nilai akurasi terendah sebesar 77% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.3. Sedangkan nilai *precision* terendah sebesar 31% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 70% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.1 dan 0.4. Sedangkan nilai *recall* terendah sebesar 20% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan

dropout rate sebesar 0.1.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 73% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai *f1 score* terendah sebesar 24% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

4.4.3.4 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Angry

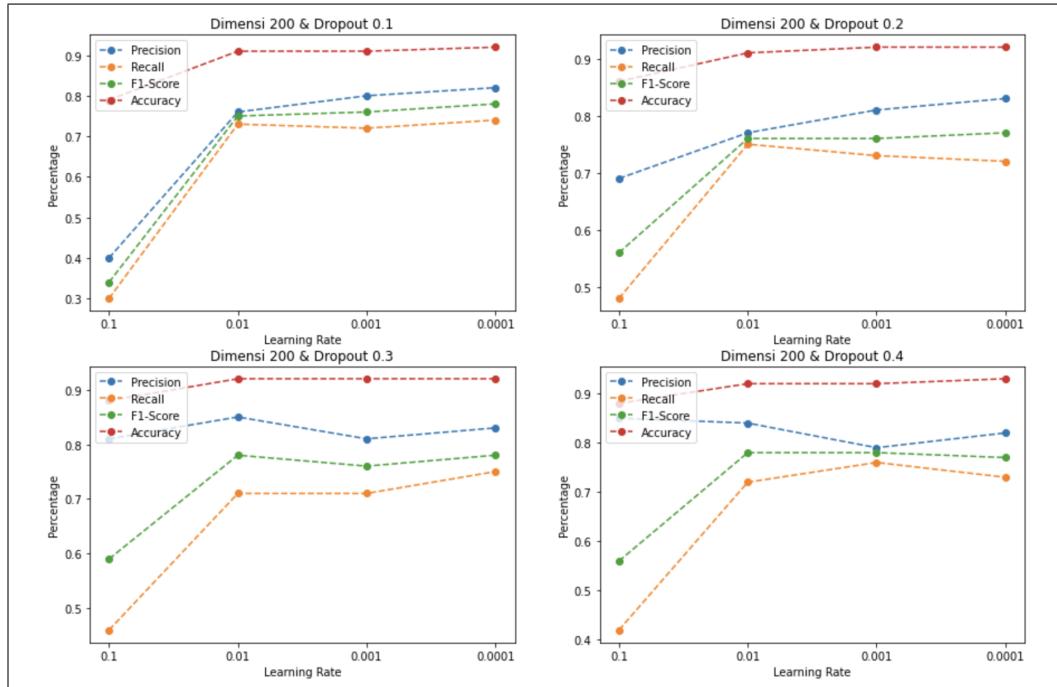
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 200 pada kelas *Angry* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 93% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.4. Berikut tabel 4.19 untuk rincian pengujinya.

Tabel 4.19 Hasil Pengujian 200 Embedding Dimension & Embedding Weights Pada Kelas Angry

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	200	0.1	0.1	40%	30%	34%	79%
2			0.01	76%	73%	75%	91%
3			0.001	80%	72%	76%	91%
4			0.0001	82%	74%	78%	92%
5		0.2	0.1	69%	48%	56%	86%
6			0.01	77%	75%	76%	91%
7			0.001	81%	73%	76%	92%
8			0.0001	83%	72%	77%	92%
9		0.3	0.1	81%	46%	59%	88%
10			0.01	85%	71%	78%	92%
11			0.001	81%	71%	76%	92%
12			0.0001	83%	75%	78%	92%
13		0.4	0.1	85%	42%	56%	88%
14			0.01	84%	72%	78%	92%
15			0.001	79%	76%	78%	92%
16			0.0001	82%	73%	77%	93%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.17 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 200 Pada Kelas *Angry*

Berdasarkan gambar 4.17 didapatkan hasil pengujian dengan pengelompokan 200 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 93% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.4. Sedangkan nilai akurasi terendah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 85% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 40% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 76% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *recall* terendah sebesar 30% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 78% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *f1 score* terendah sebesar 34% dengan menggunakan nilai parameter dimensi sebesar 200, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.1.

4.4.4 Hasil Pengujian 300 Embedding Dimension

Pengujian dilakukan dengan nilai *Embedding Output Dimension* dan *Embedding Weights* sebesar 300. Pengujian dilakukan dengan pengelompokan setiap kelas. Setiap pengujian menggunakan arsitektur yang sama dan *hyperparameter* yang berbeda-beda. Pengujian mencari hasil yang paling optimal untuk setiap kelas.

4.4.4.1 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas *Others*

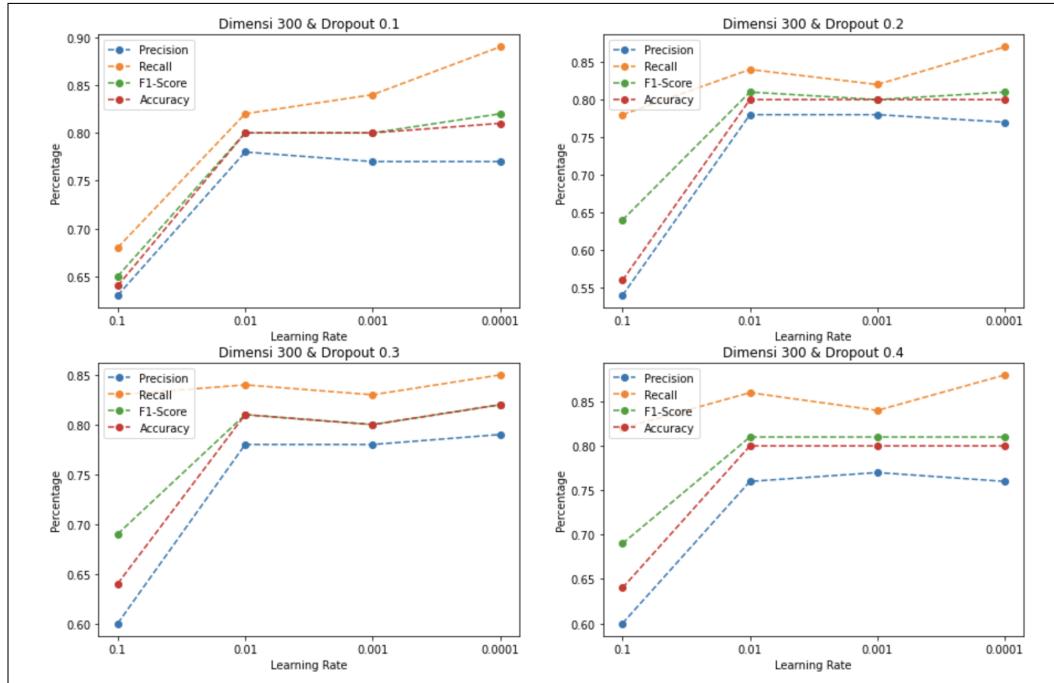
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 300 pada kelas *Others* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 82% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.3. Berikut tabel 4.20 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.20 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Others

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	300	0.1	0.1	63%	68%	65%	64%
2			0.01	78%	82%	80%	80%
3			0.001	77%	84%	80%	80%
4			0.0001	77%	89%	82%	91%
5		0.2	0.1	54%	78%	64%	56%
6			0.01	78%	84%	81%	80%
7			0.001	78%	82%	80%	80%
8			0.0001	77%	87%	81%	80%
9		0.3	0.1	60%	83%	69%	64%
10			0.01	78%	84%	81%	81%
11			0.001	78%	83%	80%	80%
12			0.0001	79%	85%	82%	82%
13		0.4	0.1	60%	82%	69%	64%
14			0.01	76%	86%	81%	80%
15			0.001	77%	84%	81%	80%
16			0.0001	76%	88%	81%	80%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.18 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 300 Pada Kelas *Others*

Berdasarkan gambar 4.18 didapatkan hasil pengujian dengan pengelompokan 300 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 82% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.3. Sedangkan nilai akurasi terendah sebesar 56% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 89% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai *precision* terendah sebesar 68% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 89% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai *recall* terendah sebesar 68% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan

dropout rate sebesar 0.1.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 82% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 64% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

4.4.4.2 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Happy

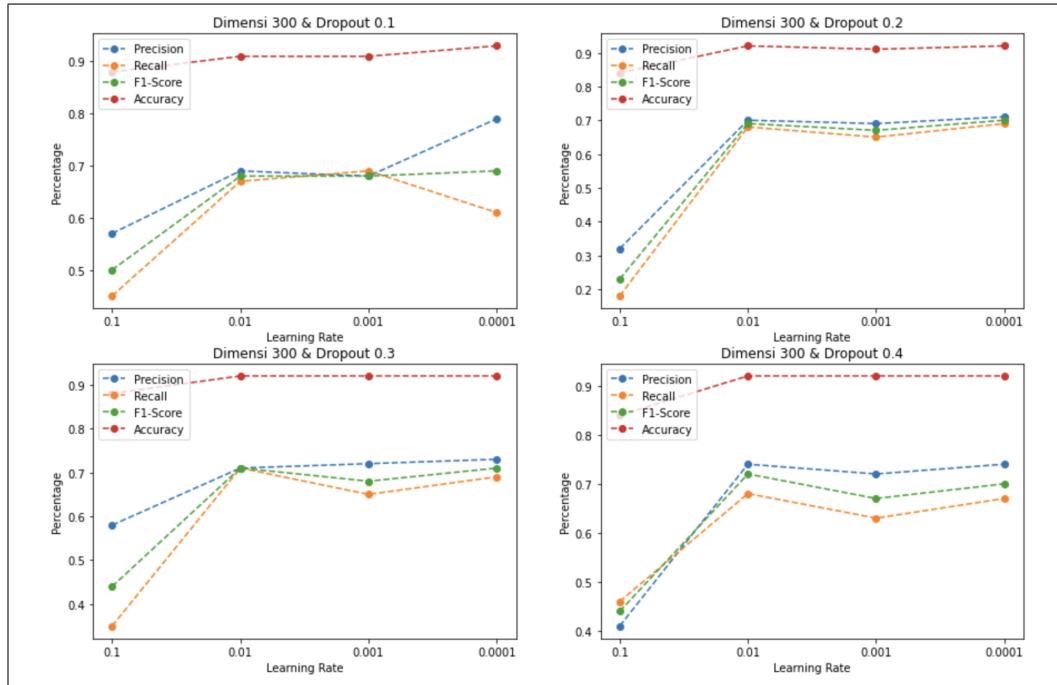
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 300 pada kelas *Happy* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 93% dengan parameter *learning rate* sebesar 0.0001 dan *dropout rate* sebesar 0.1. Berikut tabel 4.21 untuk rincian pengujinya.

Tabel 4.21 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Happy

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	300	0.1	0.1	57%	45%	50%	88%
2			0.01	69%	67%	68%	91%
3			0.001	68%	69%	68%	91%
4			0.0001	79%	61%	69%	93%
5		0.2	0.1	32%	18%	23%	84%
6			0.01	70%	68%	69%	92%
7			0.001	69%	65%	67%	91%
8			0.0001	71%	69%	70%	92%
9		0.3	0.1	58%	35%	44%	88%
10			0.01	71%	71%	71%	92%
11			0.001	72%	65%	68%	92%
12			0.0001	73%	69%	71%	92%
13		0.4	0.1	41%	46%	44%	84%
14			0.01	74%	68%	72%	92%
15			0.001	72%	63%	67%	92%
16			0.0001	74%	67%	70%	92%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.19 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 300 Pada Kelas *Happy*

Berdasarkan gambar 4.19 didapatkan hasil pengujian dengan pengelompokan 300 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 93% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai akurasi terendah sebesar 84% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2 dan 0.4.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1. Sedangkan nilai *precision* terendah sebesar 32% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 71% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 18% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 72% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai *f1 score* terendah sebesar 23% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

4.4.4.3 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Sad

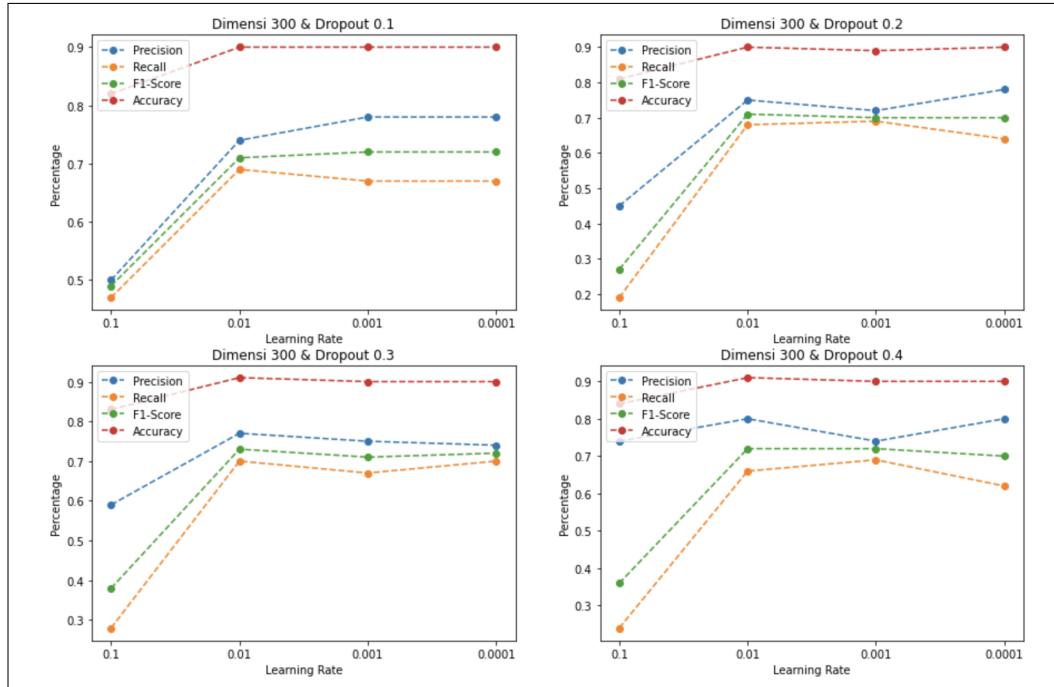
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 300 pada kelas *Sad* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 91% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.3 dan 0.4. Berikut tabel 4.22 untuk rincian pengujinya.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.22 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Sad

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	300	0.1	0.1	50%	47%	49%	82%
2			0.01	74%	69%	71%	90%
3			0.001	78%	67%	72%	90%
4			0.0001	78%	67%	72%	90%
5		0.2	0.1	45%	19%	27%	81%
6			0.01	75%	68%	71%	90%
7			0.001	72%	69%	70%	89%
8			0.0001	78%	64%	70%	90%
9		0.3	0.1	59%	28%	38%	83%
10			0.01	77%	70%	73%	91%
11			0.001	75%	67%	71%	90%
12			0.0001	74%	70%	72%	90%
13		0.4	0.1	72%	24%	36%	84%
14			0.01	80%	66%	72%	91%
15			0.001	74%	69%	72%	90%
16			0.0001	80%	62%	70%	90%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.20 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 300 Pada Kelas Sad

Berdasarkan gambar 4.20 didapatkan hasil pengujian dengan pengelompokan 300 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 91% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3 dan 0.4. Sedangkan nilai akurasi terendah sebesar 81% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 80% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01 dan 0.0001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 45% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 70% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01 dan 0.0001, dan *dropout rate* sebesar 0.3. Sedangkan nilai *recall* terendah sebesar 19% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan

dropout rate sebesar 0.2.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 73% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.3. Sedangkan nilai *f1 score* terendah sebesar 27% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

4.4.4.4 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Angry

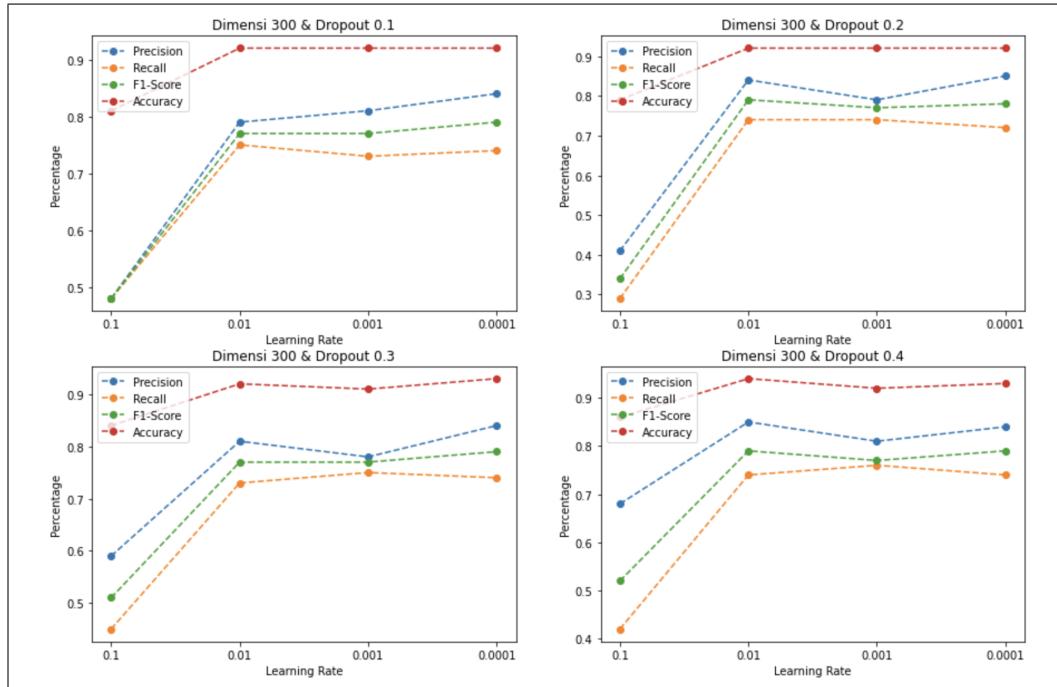
Pengujian nilai *Embedding Dimension* dan *Embedding Weights* sebesar 300 pada kelas *Angry* di dapatkan hasil yang baik dengan hasil akurasi tertinggi yaitu 94% dengan parameter *learning rate* sebesar 0.01 dan *dropout rate* sebesar 0.4. Berikut tabel 4.23 untuk rincian pengujinya.

Tabel 4.23 Hasil Pengujian 300 Embedding Dimension & Embedding Weights Pada Kelas Angry

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	300	0.1	0.1	48%	48%	48%	81%
2			0.01	79%	75%	77%	92%
3			0.001	81%	73%	77%	92%
4			0.0001	84%	74%	79%	92%
5		0.2	0.1	41%	29%	34%	79%
6			0.01	84%	74%	79%	92%
7			0.001	79%	74%	77%	92%
8			0.0001	85%	72%	78%	92%
9		0.3	0.1	59%	45%	51%	84%
10			0.01	81%	73%	77%	92%
11			0.001	78%	75%	77%	91%
12			0.0001	84%	74%	79%	93%
13		0.4	0.1	68%	42%	52%	86%
14			0.01	85%	74%	79%	94%
15			0.001	81%	76%	77%	92%
16			0.0001	84%	74%	79%	93%

Berdasarkan hasil pengujian pada tabel 4.6 dilihat perbandingan setiap model

dengan nilai *precision*, *recall*, *f1 score*, dan akurasinya masing-masing. Dari hasil pengujian di atas ini terdapat beberapa informasi yang didapat, seperti model dengan nilai *precision*, *recall*, *f1 score*, dan akurasi yang tertinggi maupun terendah, serta nilai parameter dimensi, *learning rate*, dan *dropout* yang digunakan pada model tersebut.



Gambar 4.21 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* dengan Dimensi 300 Pada Kelas *Angry*

Berdasarkan gambar 4.21 didapatkan hasil pengujian dengan pengelompokan 300 dimensi di atas ini memberikan hasil yang cukup baik. Dari hasil tersebut disimpulkan bahwa nilai akurasi tertinggi yang diperoleh adalah sebesar 94% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai akurasi terendah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Hasil pengujian menunjukkan nilai *precision* yang menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Untuk nilai *precision* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 85% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.01, dan *dropout rate* sebesar 0.4. Sedangkan nilai *precision* terendah sebesar 41% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Di sisi yang lain, pengujian pada nilai *recall* menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. Untuk nilai *recall* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 76% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.001, dan *dropout rate* sebesar 0.4. Sedangkan nilai *recall* terendah sebesar 29% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

Dan terakhir dari hasil pengujian tersebut dilihat nilai *f1 score* yang menunjukkan *harmonic mean* dari *precision* dan *recall*. Untuk nilai *f1 score* disimpulkan bahwa nilai tertinggi yang diperoleh adalah sebesar 79% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.0001, dan *dropout rate* sebesar 0.1 dan 0.4. Sedangkan nilai *f1 score* terendah sebesar 34% dengan menggunakan nilai parameter dimensi sebesar 300, *learning rate* sebesar 0.1, dan *dropout rate* sebesar 0.2.

4.4.5 Hasil Pengujian Keseluruhan

Pada bagian ini dituliskan hasil pengujian keseluruhan yang sudah dilakukan berdasarkan pemaparan yang sudah dijelaskan pada bagian 4.4. Hasil pengujian keseluruhan ini dilakukan dengan membandingkan setiap model terbaik di setiap kelompok pengujian untuk setiap kelas. Parameter yang dibandingkan adalah nilai *Embedding Output Dimension & Embedding Weights*, *dropout rate*, dan *learning rate*. Hasil pengujian keseluruhan ini memperlihatkan hasil performanya dengan menggunakan nilai *precision*, *recall*, *F1 Score*, dan akurasi.

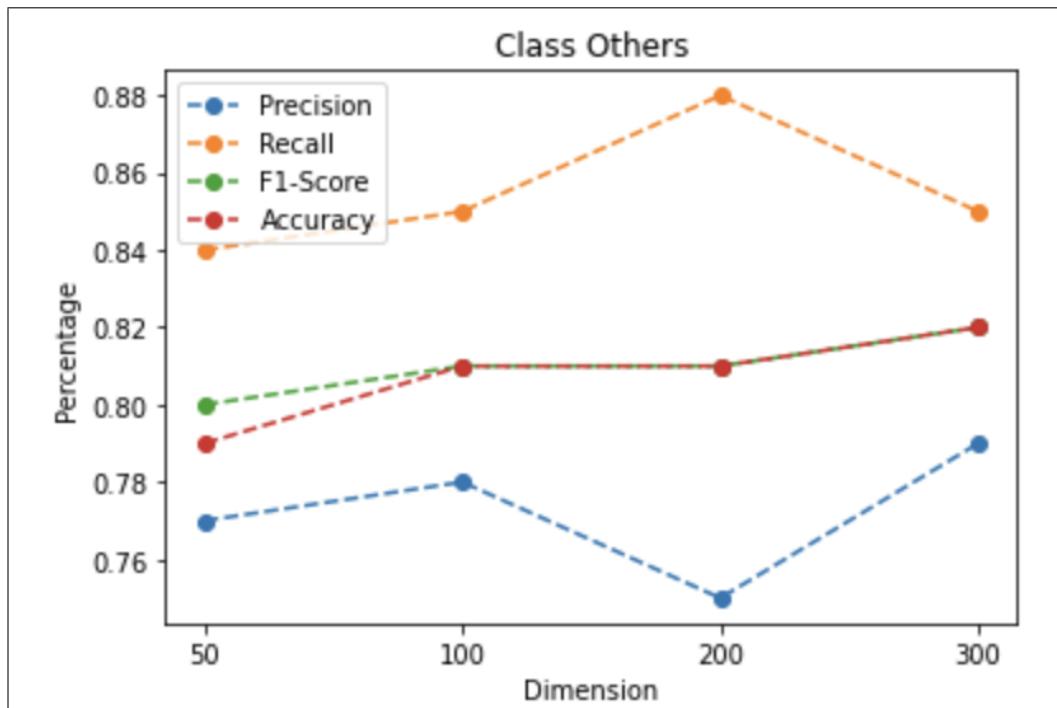
4.4.5.1 Hasil Pengujian Keseluruhan Pada Kelas *Others*

Hasil pengujian keseluruhan pada kelas *others* mendapatkan hasil yang baik dan hasil dari pengujian tersebut memberikan informasi yang penting. Hasil pengujian ini membandingkan seluruh parameter terbaik untuk kelas *others* yang diambil dari setiap kelompok dimensi. Berikut tabel 4.24 untuk rincian pengujiannya.

Tabel 4.24 Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas *Others*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.2	0.01	77%	84%	80%	79%
2	100	0.3	0.01	78%	85%	81%	81%
3	200	0.3	0.0001	75%	88%	81%	81%
4	300	0.3	0.0001	79%	85%	82%	82%

Berdasarkan hasil pengujian pada tabel 4.24 dilihat bahwa setiap model yang ada di atas memiliki perbedaan nilai *precision*, *recall*, *f1 score*, dan akurasi yang sangat tapis. Meskipun dengan nilai tertinggi ada pada model dengan parameter dimensi 300, *learning rate* 0.0001, dan *dropout* 0.3 yang dimana pada model ini memiliki kompleksitas yang tinggi hanya memiliki perbedaan nilai yang sangat kecil dibanding model lainnya.



Gambar 4.22 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Others*

Berdasarkan gambar 4.22 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *others*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *others* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.3, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 79%, nilai *recall* sebesar 85%, nilai *f1 score* sebesar 82%, dan nilai akurasi sebesar 82%.

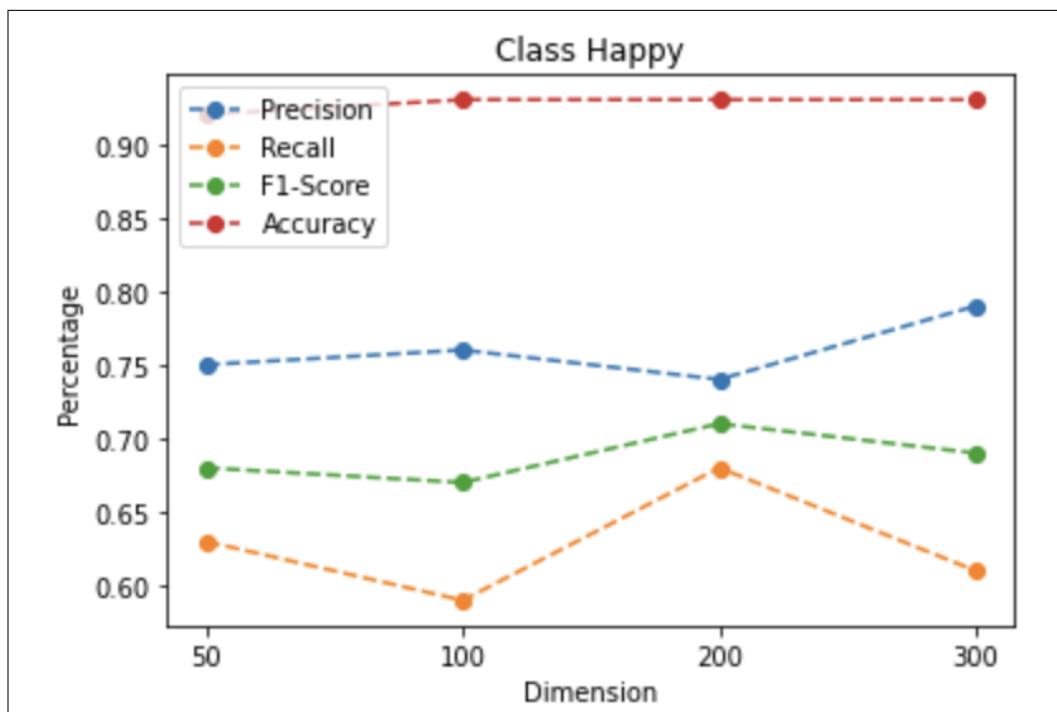
4.4.5.2 Hasil Pengujian Keseluruhan Pada Kelas *Happy*

Bagian ini menjelaskan hasil pengujian keseluruhan pada kelas *happy* mendapatkan hasil yang baik dan hasil dari pengujian tersebut memberikan informasi yang penting. Hasil pengujian ini membandingkan seluruh parameter terbaik untuk kelas *happy* yang diambil dari setiap kelompok dimensi. Berikut tabel 4.25 untuk rincian pengujinya.

Tabel 4.25 Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas *Happy*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.4	0.01	75%	63%	68%	92%
2	100	0.1	0.0001	76%	59%	67%	93%
3	200	0.3	0.01	74%	68%	71%	93%
4	300	0.3	0.0001	79%	61%	69%	93%

Berdasarkan hasil pengujian pada tabel 4.25 dilihat bahwa 3 model (model 2, model 3, dan model 4) di atas memiliki nilai akurasi yang sama, yaitu 93%. Namun hal yang unik ada pada model 2 dan model 3 yang memiliki nilai *learning rate* dan *dropout* yang sama, dimana yang berbeda dari kedua model tersebut adalah nilai dimensi yang digunakan. Hal ini mengartikan bahwa model yang memiliki dimensi yang tinggi belum tentu mendapatkan nilai yang lebih bagus dibandingkan model yang memiliki dimensi yang lebih rendah.



Gambar 4.23 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Happy*

Berdasarkan gambar 4.23 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *happy*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk

kelas *happy* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.1, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 79%, nilai *recall* sebesar 61%, nilai *f1 score* sebesar 69%, dan nilai akurasi sebesar 93%.

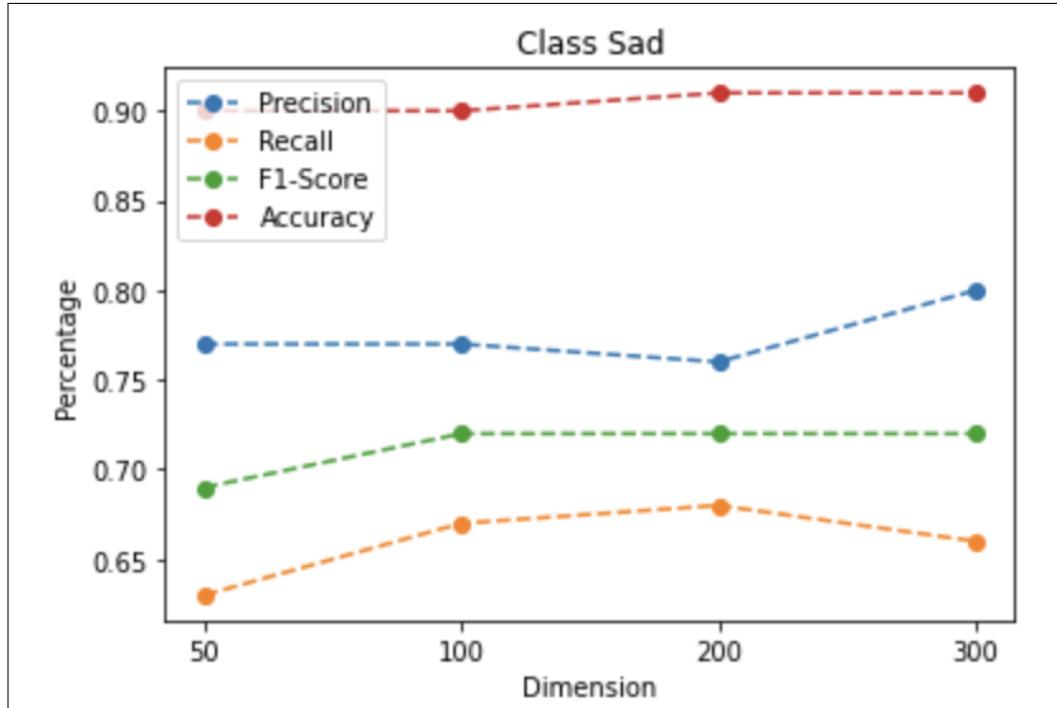
4.4.5.3 Hasil Pengujian Keseluruhan Pada Kelas *Sad*

Hasil pengujian keseluruhan pada kelas *sad* mendapatkan hasil yang baik dan hasil dari pengujian tersebut memberikan informasi yang penting. Hasil pengujian ini membandingkan seluruh parameter terbaik untuk kelas *sad* yang diambil dari setiap kelompok dimensi. Berikut tabel 4.26 untuk rincian pengujinya.

Tabel 4.26 Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas *Sad*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.2	0.01	77%	63%	69%	90%
2	100	0.3	0.01	77%	67%	72%	90%
3	200	0.3	0.001	76%	68%	72%	91%
4	300	0.4	0.001	80%	66%	72%	91%

Berdasarkan hasil pengujian pada tabel 4.26 dilihat bahwa semakin kecil nilai *learning rate* meningkatkan nilai akruasi pada model. Hal ini disebabkan nilai *learning rate* menentukan besarnya perubahan yang diaplikasikan pada nilai bobot dan bias sehingga semakin besar nilai *learning rate* membuat komputasi pada saat pembelajaran menjadi lebih cepat, namun memiliki kekurangan yang menyebabkan model mendapatkan akurasi yang kurang maksimal begitu juga sebaliknya.



Gambar 4.24 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Sad*

Berdasarkan gambar 4.24 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *sad*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *sad* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.4, dan *learning rate* sebesar 0.001. Dari model tersebut didapat nilai *precision* sebesar 80%, nilai *recall* sebesar 66%, nilai *f1 score* sebesar 72%, dan nilai akurasi sebesar 91%.

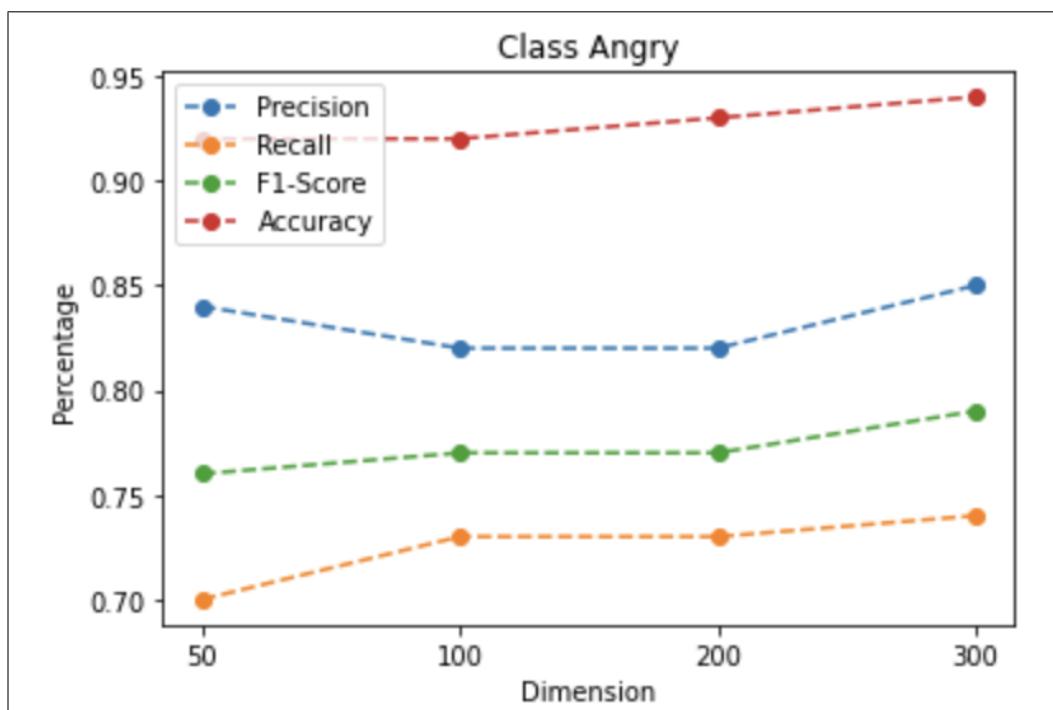
4.4.5.4 Hasil Pengujian Keseluruhan Pada Kelas *Angry*

Hasil pengujian keseluruhan pada kelas *angry* mendapatkan hasil yang baik dan hasil dari pengujian tersebut memberikan informasi yang penting. Hasil pengujian ini membandingkan seluruh parameter terbaik untuk kelas *angry* yang diambil dari setiap kelompok dimensi. Berikut tabel 4.27 untuk rincian pengujinya.

Tabel 4.27 Hasil Pengujian Terbaik Untuk Setiap Dimensi Pada Kelas *Angry*

No.	Dimension	Dropout	Learning Rate	Precision	Recall	F1 Score	Accuracy
1	50	0.4	0.01	84%	70%	76%	92%
2	100	0.4	0.01	82%	73%	77%	92%
3	200	0.4	0.0001	82%	73%	77%	93%
4	300	0.4	0.0001	85%	74%	79%	94%

Berdasarkan hasil pengujian pada tabel 4.27 dilihat bahwa model 3 memiliki akurasi yang lebih rendah dibanding model 4 meskipun nilai *learning rate* yang digunakan pada model 3 lebih kecil dibandingkan pada model 4. Hal ini mengartikan bahwa nilai yang didapatkan pada model tidak selalu tergantung pada *learning rate* dan nilai *dropout* memberikan dampak juga terhadap model yang memiliki kompleksitas yang tinggi.



Gambar 4.25 Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Angry*

Berdasarkan gambar 4.25 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *angry*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *angry* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.4, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 85%, nilai *recall* sebesar 74%, nilai *f1 score* sebesar 79%, dan nilai akurasi sebesar 94%.

4.4.5.5 Hasil Pengujian Keseluruhan *Precision & Recall*

Hasil pengujian keseluruhan yang sudah dilakukan, seperti yang dicantumkan pada tabel 4.25, 4.26, dan 4.27 mendapatkan hasil *precision* dan *recall* yang cukup baik, dimana nilai *precision* pada klasifikasi kelas *happy*, *sad*, dan *angry* lebih tinggi dari pada nilai *recall*. Hal ini disebabkan karena nilai FP yang kecil ketika melakukan prediksi, yang dimana mengartikan jumlah data yang di prediksi benar

padahal sebenarnya salah memiliki jumlah yang lebih sedikit. Sedangkan pada klasifikasi kelas *others* yang dilihat pada tabel 4.24, nilai *recall* lebih besar daripada nilai *precision*. Hal ini disebabkan karena nilai FN yang kecil ketika melakukan prediksi, yang dimana mengartikan jumlah data yang di prediksi salah padahal sebenarnya benar memiliki jumlah yang lebih sedikit.

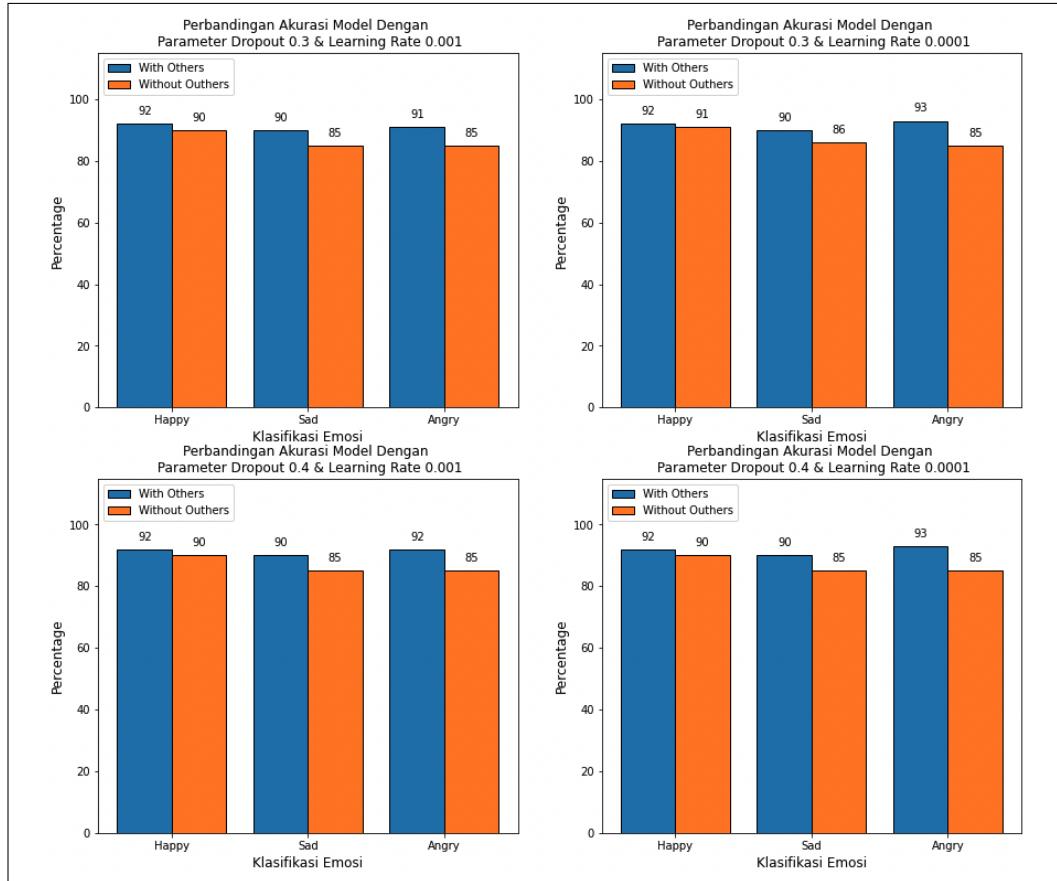
4.4.6 Hasil Pengujian Model Tanpa Kelas *Others*

Hasil pengujian model tanpa menggunakan data dengan label *others* mendapatkan hasil yang cukup baik dan hasil dari pengujian tersebut memberikan informasi yang penting. Hasil pengujian ini membandingkan beberapa paramater terbaik, yaitu dengan menggunakan nilai *Embedding Dimension & Embedding Weight* sebesar 300, *dropout rate* sebesar 0.3 & 0.4, serta *learning rate* sebesar 0.001 & 0.0001. Berikut tabel 4.28 untuk rincian pengujiannya.

Tabel 4.28 Hasil Pengujian Model Tanpa Kelas *Others*

No.	Dimension	Dropout	Learning Rate	Accuracy Happy	Accuracy Sad	Accuracy Angry
1	300	0.3	0.001	90%	85%	85%
2			0.0001	91%	86%	85%
3		0.4	0.001	90%	85%	85%
4			0.0001	90%	85%	85%

Berdasarkan hasil pengujian pada tabel 4.28 dilihat bahwa model yang dilakukan pelatihan tanpa menggunakan data kelas *others* memberikan hasil akurasi yang cukup bagus, dimana pada kelas *happy* memiliki rata - rata akurasi sebesar 90% dan pada kelas *sad* & kelas *angry* memiliki rata - rata akurasi sebesar 85%. Namun akurasi yang didapatkan dengan menggunakan model tanpa menggunakan data pada kelas *others* memiliki nilai yang lebih kecil dibandingkan nilai akurasi yang didapatkan pada model yang menggunakan data kelas *others* dalam pembelajaran. Hal tersebut dilihat pada gambar 4.26 di bawah ini.



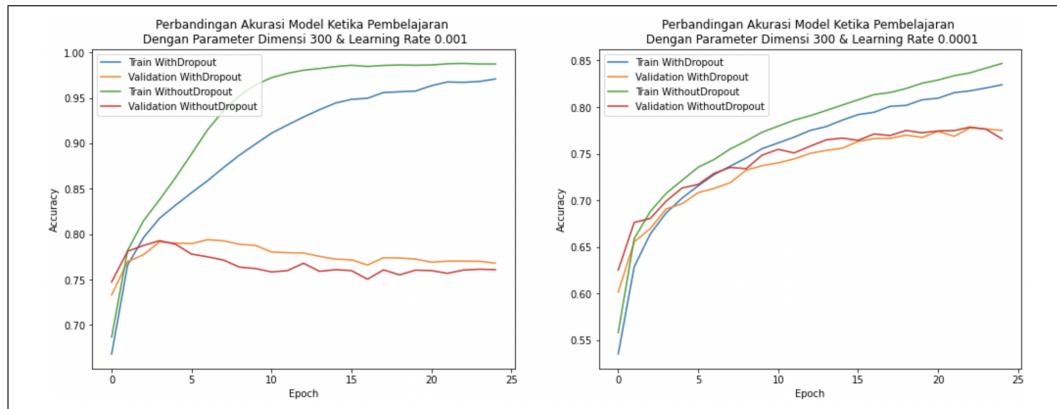
Gambar 4.26 Perbandingan Visualisasi Plot Accuracy Model Dengan Kelas *Others* & Tanpa Kelas *Others*

Berdasarkan gambar 4.26 didapatkan hasil pengujian akurasi antara model yang menggunakan data kelas *others* dan tanpa data kelas *others* dalam melakukan pembelajaran. Hasil yang didapatkan adalah bahwa nilai akurasi model yang menggunakan data kelas *others* pada saat pembelajaran memiliki nilai yang lebih tinggi dibandingkan dengan nilai akurasi model yang tanpa menggunakan data kelas *others*. Hal ini terjadi karena pada saat model melakukan pembelajaran terdapat varian data *input* dan jumlah dimensi data *input* yang lebih sedikit sehingga model tidak mempelajari varian kata yang lebih banyak dan membuat prediksi model menjadi menurun.

4.4.7 Hasil Pengujian Model Tanpa *Dropout*

Hasil pengujian model tanpa menggunakan *dropout* mendapatkan hasil akurasi yang cukup baik, namun ketika pembelajaran model yang dilakukan terjadinya *overfit* pada model. Hasil pengujian ini membandingkan akurasi ketika pembelajaran model dengan menggunakan beberapa parameter terbaik, yaitu dengan menggunakan nilai *Embedding Dimension & Embedding Weight* sebesar 300, *dropout rate* sebesar 0.3 & 0.4, serta *learning rate* sebesar 0.001 & 0.0001.

Berikut gambar 4.27 untuk rincian pengujinya.



Gambar 4.27 Perbandingan Visualisasi Plot Accuracy Model Dengan *Dropout* & Tanpa *Dropout*

Berdasarkan gambar 4.27 didapatkan hasil pengujian akurasi ketika melakukan pembelajaran model. Hasil yang didapatkan adalah bahwa pembelajaran model yang tanpa menggunakan *dropout layer* menghasilkan model yang *overfit*, sedangkan model yang menggunakan *dropout layer* menghindari *overfit* pada saat model melakukan pembelajaran. Dari gambar plot diatas model yang pembelajarannya tanpa menggunakan *dropout layer* dan dengan menggunakan *dropout layer* menghasilkan perbandingan jarak antara akurasi dan validasi akurasi sebesar 4%. yang menyebabkan penggunaan *dropout layer* memberikan dampak yang cukup besar dalam pembelajaran model agar model tidak menjadi *overfit*.

4.4.8 Pembahasan Umum Hasil Pengujian

Bagian ini menjelaskan hasil pengujian arsitektur LSTM yang sudah dilakukan dengan keterkaitan antara rumusan masalah pada bagian 1.2. Berikut dijelaskan pengaruh semua indikator pengujian secara menyeluruh terhadap akurasi sistem.

1. Menetapkan *parameter* merupakan hal yang paling krusial dalam melakukan pemodelan *deep learning*. *Hyperparameter* yang telah diuji diantaranya *Embedding Dimension & Embedding Weights*, *learning rate*, dan *dropout rate*. Parameter yang telah diuji digunakan pada sebuah dataset untuk mengetahui sejauh mana performa model LSTM dalam mengolah dataset untuk sistem deteksi emosi dalam teks dan seberapa besar nilai akurasi yang didapatkan dari model LSTM.
2. Pengujian dari penggunaan arsitektur yang tertera pada bagian 4.3.1 memberikan hasil yang cukup baik untuk dataset yang digunakan. Dataset yang digunakan memiliki jumlah kata yang beragam mulai dari 2 kata hingga yang paling banyak memiliki 62 kata untuk setiap teksnya. Dengan

beragamnya jumlah kata untuk setiap teks dalam dataset membuat model mengartikan jumlah kata yang beragam juga. Selain itu penggunaan *text preprocessing* memiliki dampak yang besar dalam meningkatkan akurasi model, terutama dalam penggunaan *spell check*. Hal ini disebabkan pada dataset terdapat banyak *noise*, seperti banyaknya kata yang disingkat dan salah penulisan kata yang digunakan pada dataset yang membuat model menjadi *overfitting* karena mempelajari data *noise*.

Pengaruh yang paling berdampak selain proses *spell check* adalah proses *transform negation*. Pada tahap *text preprocessing* sebuah dataset melalui proses *stopword* dimana pada proses ini seluruh kata umum pada setiap teks dibuang termasuk kata negasi sehingga pada tahap ini teks yang memiliki arti negatif berubah artinya menjadi positif sehingga memberikan dampak pada model untuk memprediksi kelas negatif pada teks, seperti kelas *sad* dan *angry*. Hal ini menyebabkan nilai akurasi untuk kelas negatif menjadi lebih rendah dibanding nilai akurasi untuk kelas positif.

3. Pengujian nilai *Embedding Dimension & Embedding Weights* pada model merupakan parameter yang berpengaruh dalam melihat performa pembelajaran terhadap akurasi yang didapat. Dari hasil pengujian yang sudah dilakukan, nilai *Embedding Dimension & Embedding Weights* yang kecil mendapatkan akurasi lebih rendah dibandingkan dengan nilai *Embedding Dimension & Embedding Weights* yang lebih banyak. Hal ini disebabkan karena semakin besar jumlah *Embedding Dimension & Embedding Weights* yang digunakan membuat model mempelajari data yang lebih kompleks. Pengujian model untuk setiap kelas menunjukkan nilai akurasi yang lebih besar pada jumlah *Embedding Dimension & Embedding Weights* yang besar. Pengujian model dengan menggunakan nilai *Embedding Dimension & Embedding Weights* yang besar (300) menghasilkan hasil yang paling optimal untuk setiap kelas.

Kesimpulan dari pengujian nilai *Embedding Dimension & Embedding Weights* terhadap deteksi emosi dalam teks adalah dengan menggunakan nilai jumlah *Embedding Dimension & Embedding Weights* yang kecil, maka menghasilkan nilai akurasi untuk deteksi emosi dalam teks yang baik tetapi tidak maksimal. Dengan menggunakan nilai *Embedding Dimension & Embedding Weights* yang besar, maka menghasilkan nilai akurasi untuk deteksi emosi dalam teks yang lebih baik dibandingkan dengan nilai *Embedding Dimension & Embedding Weights* kecil, sehingga perlu dilakukan pengujian untuk mendapatkan nilai *Embedding Dimension &*

Embedding Weights yang optimal.

4. Pengujian nilai *learning rate* pada model merupakan parameter yang berpengaruh dalam melihat performa kecepatan dalam pembelajaran model dan nilai akurasi yang didapat. Hal tersebut terjadi karena *learning rate* mempengaruhi kecepatan dalam pembelajaran model dan *learning rate* mengatur nilai bobot dan bias ketika pembelajaran model. Dari hasil pengujian yang sudah dilakukan, terlihat bahwa nilai *learning rate* yang tinggi saat pembelajaran, lebih cenderung tidak stabil dan hal itu menandakan bahwa model susah beradaptasi dengan nilai *learning rate* yang tinggi. Pada pengujian *learning rate* dengan nilai yang tinggi, model mengalami gagal belajar karena tidak stabil yang berlebihan dan sudah di titik jenuh untuk model belajar dan *gradient* yang terlalu besar perubahannya hingga diluar batas. Kasus penggunaan nilai *learning rate* rendah lebih baik dibandingkan penggunaan nilai *learning rate* yang tinggi. Hal ini disebabkan oleh model memerlukan langkah yang kecil sehingga model tetap stabil walaupun mengalami *overfitting* yang tidak terlalu parah dibandingkan dengan pengujian menggunakan nilai *learning rate* lainnya. Penggunaan nilai *learning rate* yang besar mempercepat proses pembelajaran tetapi memungkinkan model untuk lebih cepat mengalami *overfitting* dan sulit untuk menerima data baru untuk dilakukan prediksi sehingga nilai akurasi menurun karena *overfitting*. Dengan menggunakan nilai *learning rate* yang rendah, dalam proses pembelajaran memungkinkan model untuk lambat dalam pembelajaran sehingga model tidak cepat mengalami *overfitting* sehingga nilai akurasi meningkat karena model mencapai konvergen. disimpulkan bahwa pemilihan *learning rate* harus dilakukan pengujian *trial* dan *error* untuk mengetahui performa model terhadap akurasi.
5. Pengujian nilai *dropout rate* merupakan parameter yang berpengaruh pada pelatihan model agar model tidak menjadi *overfitting*. *Dropout* merupakan teknik regularisasi yang membuang unit (bersama dengan koneksinya) dari jaringan saraf selama pelatihan. Dari hasil pengujian yang sudah dilakukan, terlihat nilai *dropout rate* yang tinggi membantu model yang memiliki kompleksitas yang tinggi (nilai *learning rate* yang kecil dan nilai *Embedding Dimension & Embedding Weights* yang besar) agar tidak menjadi *overfitting*. Namun penggunaan nilai *dropout rate* tergantung pada nilai *Embedding Dimension & Embedding Weights* dan nilai *learning rate* yang digunakan.

Semakin tinggi kompleksitas parameter yang digunakan pada model, membuat nilai *dropout rate* yang digunakan juga menjadi semakin tinggi untuk membuat model menjadi stabil. disimpulkan bahwa penggunaan nilai *dropout rate* tergantung pada parameter nilai *Embedding Dimension & Embedding Weights* dan nilai *learning rate* yang digunakan pada model.

6. Pengujian nilai *precision* dan *recall* merupakan nilai yang penting dan memiliki pengertian *trade-off* sesuai dengan kasus yang dilakukan. Pada penelitian ini nilai *precision* dan *recall* seimbang yang dimana, dataset yang digunakan merupakan teks - teks yang ada di kolom komentar, seperti komentar terhadap suatu produk yang mengartikan nilai *precision* yang berbicara mengenai faktor yang menunjukkan ketepatan model dan nilai *recall* yang digunakan untuk menemukan kembali sebuah informasi dari opini customer yang dijadikan sebagai konteks atau penentu dari suatu produk dan banyaknya minat dari pembeli menjadi kesuksesan penjualan suatu produk. disimpulkan bahwa pengujian nilai *precision* dan *recall* sama pentingnya untuk mengetahui kesuksesan penjualan suatu produk.

4.5 Analisis Kesalahan

Pada bagian ini dibahas mengenai analisis kesalahan yang terjadi selama proses pengujian deteksi emosi dalam teks. Kesalahan ini terjadi karena adanya faktor yang mempengaruhi model selama proses pengujian. Berikut adalah kesalahan yang dibahas.

1. Pada saat dibangunnya indeks *corpus* kata, masih terdapat beberapa *noise* kata. Hal ini disebabkan teks dalam data terdapat kesalahan pengetikan atau kurangnya spasi antar kata yang menyebabkan *noise*. Hal ini juga berdampak pada pengambilan vektor dari GloVe. Jika kata-kata pada indeks *corpus* kata ditemukan di GloVe, maka nilai vektor diambil dan disimpan dalam matriks *embedding*, jika tidak ditemukan maka nilai vektor menjadi tetap 0, seperti contohnya kata *tearsjoy* yang seharusnya jika ada spasi menjadi *tears joy*. Kesalahan disini adalah ketika terdapat dua kata yang tergabung secara tidak sengaja dan pada GloVe dipastikan tidak ditemukan karena GloVe kata hanya mewakili satu kata.
2. Pada saat membuat model, nilai *learning rate* yang besar tidak memberikan hasil yang baik pada saat pelatihan model terutama ketika dipasangkan dengan nilai *dropout rate* yang tinggi dan nilai dimensi yang kecil, seperti contohnya penggunaan *learning rate* sebesar 0.1, penggunaan nilai *dropout*

rate sebesar 0.3 dan 0.4, dan penggunaan nilai dimensi sebesar 50 dan 100 menghasilkan nilai akurasi yang kecil. Hal ini disebabkan karena model cepat mengalami *overfit* dan model tidak bisa mempelajari data secara menyeluruh sehingga akurasi pelatihan model menjadi kecil. Hal ini juga berdampak pada saat melakukan validasi model yang mengakibatkan nilai akurasi validasi menjadi sangat kecil dan nilai *precision*, *recall*, dan *f1 score* yang kecil.

3. Pada saat validasi model, nilai *recall* pada kelas *others* lebih tinggi dibandingkan nilai *recall* pada kelas lainnya. Hal ini disebabkan dataset dengan kelas *others* memiliki jumlah data yang lebih banyak dibandingkan kelas lainnya yang dimana pada kelas *others* terdapat data sebanyak 14.948 dari jumlah total data sebanyak 30.160 yang mengartikan hampir 50% data memiliki kelas *others* dan ini mengakibatkan model lebih condong mempelajari kelas *others* lebih banyak dibandingkan kelas lainnya.

BAB 5 KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada bagian ini dijelaskan kesimpulan dari pembuatan sistem deteksi emosi dalam teks dengan menggunakan metode *Long Short Term Memory* melalui pengujian yang telah dilakukan. Berikut adalah rincian dari kesimpulan setelah dilakukannya pengujian:

1. Pada pengujian ini didapatkan nilai akurasi terbaik untuk setiap kelas yang ada pada penelitian deteksi emosi dalam teks. Pada kelas *others* didapatkan nilai akurasi tertinggi sebesar 82%, pada kelas *happy* didapatkan nilai akurasi tertinggi sebesar 93%, pada kelas *sad* didapatkan nilai akurasi tertinggi sebesar 91%, dan pada kelas *angry* didapatkan nilai akurasi tertinggi sebesar 94%. Disimpulkan bahwa metode LSTM di menghasilkan performa akurasi yang sangat baik.
2. *Parameter* yang optimal yang digunakan pada pengujian untuk setiap kelas adalah sebagai berikut pada kelas *others* menggunakan nilai *Embedding Dimension & Embedding Weights* sebesar 300, nilai *learning rate* sebesar 0.0001, dan nilai *dropout rate* sebesar 0.3. Pada kelas *happy* menggunakan nilai *Embedding Dimension & Embedding Weights* sebesar 300, nilai *learning rate* sebesar 0.0001, dan nilai *dropout rate* sebesar 0.3. Pada kelas *sad* menggunakan nilai *Embedding Dimension & Embedding Weights* sebesar 300, nilai *learning rate* sebesar 0.001, dan nilai *dropout rate* sebesar 0.4. Terakhir pada kelas *angry* menggunakan nilai *Embedding Dimension & Embedding Weights* sebesar 300, nilai *learning rate* sebesar 0.0001, dan nilai *dropout rate* sebesar 0.4.
3. Pengujian nilai *Embedding Dimension & Embedding Weights* terhadap deteksi emosi dalam teks berpengaruh terhadap akurasi. Dengan menggunakan nilai *Embedding Dimension & Embedding Weights* yang kecil menghasilkan nilai akurasi yang baik tetapi tidak optimal dibandingkan dengan nilai *Embedding Dimension & Embedding Weights* yang besar dengan hasil akurasi yang optimal.
4. Nilai *learning rate* mempengaruhi hasil akurasi pada deteksi emosi dalam teks. Nilai *learning rate* yang besar mempercepat hasil belajar tetapi menghasilkan model yang *overfit* dan menghasilkan akurasi yang tidak

optimal dibandingkan dengan nilai *learning rate* kecil yang memperlambat model belajar, namun menghasilkan nilai akurasi yang optimal.

5. Pengaruh nilai *dropout rate* terhadap akurasi tergantung pada nilai parameter lainnya yang dimana semakin tinggi kompleksitas (nilai *learning rate* yang kecil dan nilai *Embedding Dimension & Embedding Weights* yang besar) suatu model ketika pelatihan membutuhkan nilai *dropout rate* yang semakin besar agar model tidak menjadi *overfit* dan mengimbangi model yang memiliki kompleksitas tinggi sedangkan semakin kecil kompleksitas model ketika pelatihan membutuhkan nilai *dropout rate* yang tidak terlalu besar.
6. Pengaruh nilai *precision* dan *recall* memiliki *trade-off* sesuai dengan kasus yang dilakukan. Pada penelitian ini nilai *precision* lebih diutamakan yang mengartikan ketepatan suatu model yang dibuat, dimana target pemasaran dan minat pembeli yang diprediksi benar padahal sebenarnya salah memiliki nilai yang kecil. Hal ini karena target pemasaran dan minat pembeli yang dicari adalah orang yang belum memiliki minat sehingga memungkinkan memperluas target pemasaran karena orang yang sudah memiliki minat pasti membeli produk itu.

5.2 Saran

Saran untuk pengembangan yang dilakukan untuk sistem deteksi emosi dalam teks adalah sebagai berikut.

1. Dataset yang digunakan disarankan memiliki jumlah kata-kata yang beragam di setiap datanya dan memiliki jumlah data yang banyak dikarenakan jumlah data yang beragam ini membuat model bisa memprediksi berbagai macam panjang kata dengan lebih akurat.
2. Dataset yang digunakan disarankan untuk diproses lebih jauh terutama dalam mengkoreksi kata - kata yang menjadi *noise*, seperti kata - kata yang salah penulisan, kata - kata yang disingkat, dan kata - kata yang tergabung karena pada penelitian ini hanya dilakukan pengkoreksian kata yang paling banyak muncul saja bila di gambar plotnya menggunakan *word cloud*.
3. Mengembangkan dan mengoptimisasi metode dengan penerapan *Bidirectional LSTM* untuk penelitian selanjutnya.

DAFTAR REFERENSI

- [1] J. Marín-Morales, C. Llinares, J. Guixeres, and M. Alcañiz, “Emotion Recognition in Immersive Virtual Reality: From Statistics to Affective Computing”, *Sensors*, vol. 20, no. 18, p. 5163, 2020.
- [2] M.-H. Su, C.-H. Wu, K.-Y. Huang, and Q.-B. Hong, “LSTM-based Text Emotion Recognition Using Semantic and Emotional Word Vectors”, *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, May 2018.
- [3] M. Krommyda, A. Rigos, K. Bouklas, and A. Amditis, “An Experimental Analysis of Data Annotation Methodologies for Emotion Detection in Short Text Posted on Social Media”, *Informatics*, vol. 8, no. 1, p. 19, 2021.
- [4] F. A. Acheampong, C. Wenyu, and H. Nunoo-Mensah, “Text-based Emotion Detection: Advances, Challenges, and Opportunities”, *Engineering Reports*, vol. 2, no. 7, 2020.
- [5] N. Murthy, S. Rao Allu, B. Andhavarapu, M. Bagadi, and M. Belusonti, “Text Based Sentiment Analysis Using LSTM”, *International Journal of Engineering Research and Technology*, vol. 9, no. 05, May 2020.
- [6] W. K. Sari, D. P. Rini, and R. F. Malik, “Text Classification Using Long Short Term Memory with GloVe Features”, *Scientific Journal of Computer Electrical Engineering and Informatics*, vol. 5, no. 2, p. 85, 2020.
- [7] H. Park and K. Kim, “Impact of Word Embedding Methods on Performance of Sentiment Analysis with Machine Learning Techniques,” *Journal of The Korea Society of Computer and Information*, vol. 25, no. 8, pp. 181–188, Aug. 2020.
- [8] Pramod Singh, *Machine Learning with PySpark With Natural Language Processing and Recommender Systems*, 1st ed. Reading, MA: Apress, 2019. [E-book] Available: z-library.
- [9] B. Bengfort, *Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning*, O'Reilly Media, Incorporated, 2018. [E-book] Available: z-library.

DAFTAR REFERENSI

- [10] M. T. Pilehvar and J. Comacho-Collados, *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning*, Morgan and Claypool, 2020. [E-book] Available: z-library.
- [11] Pennington, J., Socher, R. and Manning, C.D., "Glove: Global Vectors for Word Representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP), pp. 1532-1543, 2014.
- [12] S. Kostadinov, *Recurrent Neural Networks with Python Quick Start Guide: Sequential Learning and Language Modeling with Tensorflow*, PACKT Publishing Limited, 2018. [E-book] Available: z-library.
- [13] J. Brownlee, *Long Short-Term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*, Machine Learning Mastery, 2017. [E-book] Available: z-library.
- [14] K. Sailunaz, M. Dhaliwal, J. Rokne, and R. Alhajj, "Emotion Detection From Text and Speech: A Survey", *Social Network Analysis and Mining*, vol. 8, no. 1, 2018.
- [15] N. Srivastava, G. Hinton, A. Krizhevsky, and I. Sutskever, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
- [16] Kingma, Diederik P., and Jimmy Ba., "Adam: A Method for Stochastic Optimization.", *International Conference for Learning Representations*, San Diego, 2015.
- [17] J. Heaton, *Artificial Intelligence for humans, volume 3: Deep Learning and Neural Networks*, vol. 3. s.l.: Heaton Research, Inc., 2015. [E-book] Available: z-library.
- [18] Y. Ho and S. Wookey, "The Real World Weight Cross-Entropy Loss Function: Modeling the costs of Mislabeling", *IEEE Access*, vol. 8, pp. 4806–4813, 2020.
- [19] N. Babanejad, A. Agrawal, A. An, and M. Papagelis, "A Comprehensive Analysis of Preprocessing for Word Representation Learning in Affective Tasks", *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

DAFTAR REFERENSI

- [20] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal, “Semeval-2019 task 3: Emocontext Contextual Emotion Detection in Text”, *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019.