

**ANALISIS SENTIMEN PENGGUNA MEDIA SOSIAL TERHADAP
KEBIJAKAN PEMERINTAH DALAM PROGRAM VAKSINASI COVID-19
DI INDONESIA MENGGUNAKAN METODE *BIDIRECTIONAL GATED
RECURRENT UNIT* (BiGRU)**

SKRIPSI

Diajukan untuk Menempuh Ujian Sarjana Program Studi Statistika
Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran

Oleh:

Dimas Ananda

140610180039



**PROGRAM STUDI STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS PADJADJARAN**

2022

LEMBAR PENGESAHAN
ANALISIS SENTIMEN PENGGUNA MEDIA SOSIAL TERHADAP KEBIJAKAN
PEMERINTAH DALAM PROGRAM VAKSINASI COVID-19 DI INDONESIA
MENGGUNAKAN METODE *BIDIRECTIONAL GATED RECURRENT UNIT*
(BiGRU)

Oleh :

Dimas Ananda

140610180039

Setelah membaca skripsi ini dengan seksama, menurut pertimbangan telah

memenuhi persyaratan ilmiah sebagai suatu skripsi

Jatinangor, Februari 2022

Menyetujui,

Tim Pembimbing

Pembimbing



Dr. Anindya Apriliyanti Pravitasari, S.Si., M.Si.

NIP. 19840416 200812 2 004

Pembimbing Pendamping



Dr. Intan Nurma Yulita, M.T.

NIP. 19850704 201504 2 003

KATA PENGANTAR



Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas rahmat, karunia, dan izin-Nya penulis dapat menyelesaikan skripsi dengan judul “**ANALISIS SENTIMEN PENGGUNA MEDIA SOSIAL TERHADAP KEBIJAKAN PEMERINTAH DALAM PROGRAM VAKSINASI COVID-19 DI INDONESIA MENGGUNAKAN METODE *BIDIRECTIONAL GATED RECURRENT UNIT (BiGRU)***”. Skripsi ini dibuat untuk memenuhi mata kuliah Skripsi sebanyak enam satuan kredit semester (sks) serta memenuhi rangkaian proses penulis untuk menempuh ujian sarjana Strata-1 (S1) di Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran

Dalam penulisan skripsi ini penulis mengucapkan terima kasih kepada semua pihak yang telah mendukung dan memberi saran yang bermanfaat bagi penulis. Penulis menyadari masih banyak kekurangan dalam penulisan skripsi ini. Oleh karena itu penulis mengharapkan kritik dan saran bagi berbagai pihak guna perbaikan selanjutnya. Semoga proposal penelitian ini dapat bermanfaat dan diterima bagi pembaca.

Jatinangor, Februari 2022

Penulis,

Dimas Ananda

NPM. 140610180039

ABSTRAK

Judul : Analisis Sentimen Pengguna Media Sosial terhadap Kebijakan Pemerintah dalam Program Vaksinasi COVID-19 di Indonesia Menggunakan Metode *Bidirectional Gated Recurrent Unit* (BiGRU)

Nama : Dimas Ananda

NPM : 140610180039

Pembimbing : Dr. Anindya Apriliyanti Pravitasari, S.Si., M.Si.

Co-Pembimbing : Dr. Intan Nurma Yulita, M.T.

Salah satu kebijakan pemerintah dalam penanggulangan COVID-19 yaitu program vaksinasi yang diberikan kepada masyarakat, mendapatkan banyak komentar dari masyarakat melalui media sosial. Penelitian ini menjelaskan bagaimana *Bidirectional Gated Recurrent Unit* (BiGRU) digunakan untuk analisis sentimen dalam mengklasifikasikan opini positif, netral, dan negatif pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19. Model BiGRU dengan 1 *embedding layer*, lapisan BiGRU dengan unit sebanyak 20 unit, 1 lapisan *dense* dengan 30 *neuron*, menggunakan *dropout* dengan ukuran 0.3, *batch size* sebesar 64, menggunakan *loss function categorical cross entropy*, fungsi optimasi menggunakan Adam, *epoch* sebanyak 27 iterasi, *word embedding* dengan Word2Vec, dan tanpa penanganan *imbalanced classes* menghasilkan *accuracy* 63%, *weighted average f1-score* 62%, rata-rata *precision* 61 % dan rata-rata *recall* sebesar 56 %.

Kata kunci: Analisis Sentimen, COVID-19, Vaksinasi, Media Sosial, BiGRU

ABSTRACT

Title : Sentiment Analysis of Social Media Users to Government Policies in the COVID-19 Vaccination Program in Indonesia Using the Bidirectional Gated Recurrent Unit (BiGRU) Method

Name : Dimas Ananda

Student Number : 140610180039

Advisor : Dr. Anindya Apriliyanti Pravitasari, S.Si., M.Si.

Co-Advisor : Dr. Intan Nurma Yulita, M.T.

One of the government's policies in dealing with COVID-19, namely the vaccination program given to the public, received a lot of responses from the public through social media. This study explains how the Bidirectional Gated Recurrent Unit (BiGRU) is used for sentiment analysis in classifying positive, neutral, and negative opinions of social media users on government policies in the COVID-19 vaccination program. BiGRU model with 1 embedding layer, BiGRU layer with 20 units, 1 dense layer with 30 neurons, using dropout size 0.3, batch size 64, using loss function categorical cross-entropy, optimization function using Adam, epoch 27 iterations, word embedding with Word2Vec, and without addressing imbalanced classes, achieved 63% accuracy, 62% weighted average f1-score, 61% average precision, and 56% average recall.

Keywords: Sentiment Analysis, COVID-19, Vaccination, Social Media, BiGRU

DAFTAR ISI

LEMBAR PENGESAHAN	ii
KATA PENGANTAR.....	iii
ABSTRAK	iv
ABSTRACT	v
DAFTAR ISI.....	vi
DAFTAR GAMBAR.....	viii
DAFTAR TABEL	ix
BAB I PENDAHULUAN.....	10
1.1 Latar Belakang	10
1.2 Identifikasi Masalah	13
1.3 Maksud dan Tujuan Penelitian	13
1.4 Manfaat Penelitian.....	13
1.5 Batasan Masalah	13
BAB II TINJAUAN PUSTAKA.....	15
2.1 Pendahuluan	15
2.2 Analisis Sentimen	15
2.3 Klasifikasi Teks	16
2.4 Regresi Logistik	16
2.5 <i>Naïve Bayes Classifier</i>	17
2.6 <i>Support Vector Machine (SVM)</i>	18
2.7 <i>Artificial Neural Network (ANN)</i>	19
2.8 <i>Recurrent Neural Network (RNN)</i>	21
2.8.1 <i>Long Short-Term Memory (LSTM)</i>	22
2.8.2 <i>Gated Recurrent Unit (GRU)</i>	24
2.8.3 <i>Bidirectional Gated Recurrent Unit (BiGRU)</i>	26
BAB III ANALISIS SENTIMEN PENGGUNA MEDIA SOSIAL TERHADAP KEBIJAKAN PEMERINTAH DALAM PROGRAM VAKSINASI COVID-19 DI	

INDONESIA MENGGUNAKAN METODE BIDIRECTIONAL GATED RECURRENT UNIT (BiGRU)	31
3.1 Pendahuluan	31
3.2 Data Penelitian	31
3.3 Langkah-langkah Penelitian	34
3.3.1 <i>Data Preprocessing</i>	35
3.3.2 Pembagian Data	37
3.3.3 <i>Word Embedding</i>	37
3.3.4 Penanganan <i>Imbalanced Classes</i>	38
3.3.5 Pemodelan dengan <i>Bidirectional Gated Recurrent Unit (BiGRU)</i>	39
3.3.6 Evaluasi Model	47
BAB IV HASIL DAN PEMBAHASAN	51
4.1 Pendahuluan	51
4.2 <i>Data Preprocessing</i>	51
4.2.1 <i>Case Folding</i>	51
4.2.2 <i>Tokenizing</i>	51
4.2.3 <i>Filtering</i>	52
4.2.4 <i>Stemming</i>	52
4.3 Pembagian Data	53
4.4 <i>Word Embedding</i>	54
4.5 Penanganan <i>Imbalanced Classes</i>	57
4.6 Pemodelan dengan <i>Bidirectional Gated Recurrent Unit (BiGRU)</i>	58
4.6.1 Arsitektur Jaringan	58
4.6.2 Proses Pelatihan dan Validasi Model	59
4.7 Evaluasi Model	69
BAB V KESIMPULAN DAN SARAN	73
5.1 Kesimpulan	73
5.2 Saran	74
DAFTAR PUSTAKA	75
LAMPIRAN	82

DAFTAR GAMBAR

Gambar 1. 1 Persentase Platform yang Sering Digunakan Pengguna Internet Berusia 16 hingga 64 Tahun di Indonesia pada Januari 2021	11
Gambar 2. 1 Grafik Regresi Logistik	17
Gambar 2. 2 Arsitektur <i>Artificial Neural Network</i>	20
Gambar 2. 3 Arsitektur <i>Recurrent Neural Network</i>	21
Gambar 2. 4 Unit LSTM	23
Gambar 2. 5 Unit <i>Gated Recurrent Unit</i>	24
Gambar 2. 6 Arsitektur <i>Bidirectional GRU</i>	27
Gambar 2. 7 <i>Bidirectional Recurrent Neural Network</i>	28
Gambar 3. 1 Grafik Frekuensi Data Penelitian Berdasarkan Sumbernya	32
Gambar 3. 2 Grafik Presentase Hasil Pelabelan Data	34
Gambar 3. 3 <i>Flowchart</i> Penelitian.....	34
Gambar 3. 4 Fungsi Aktivasi Sigmoid	42
Gambar 3. 5 Fungsi Aktivasi ReLU	43
Gambar 3. 6 Fungsi Aktivasi TanH.....	44
Gambar 4. 1 Grafik Hasil Pembagian Data Pelatihan dan Pengujian	54
Gambar 4. 2 Grafik Hasil Penanganan <i>Imbalanced Classes</i> Menggunakan ROS dan RUS	57
Gambar 4. 3 Ilustrasi <i>5-fold Cross Validation</i>	60
Gambar 4. 4 Grafik Perbandingan Hasil Pengujian Penggunaan Word2Vec	64
Gambar 4. 5 Hasil Loss Pelatihan dengan Penanganan <i>Imbalanced Classes</i> dan Tanpa Penanganan <i>Imbalanced Classes</i>	68
Gambar 4. 6 Hasil Akurasi Pelatihan dengan Penanganan <i>Imbalanced Classes</i> dan Tanpa Penanganan <i>Imbalanced Classes</i>	68

DAFTAR TABEL

Tabel 2. 1 Perbandingan Literatur	29
Tabel 3. 1 Contoh Pelabelan Data	33
Tabel 3. 2 <i>Confusion Matrix</i>	48
Tabel 4. 1 Contoh Hasil <i>Case Folding</i>	51
Tabel 4. 2 Contoh Hasil <i>Tokenizing</i>	52
Tabel 4. 3 Contoh Hasil <i>Filtering</i>	52
Tabel 4. 4 Contoh Hasil <i>Stemming</i>	53
Tabel 4. 5 Tabel Contoh Hasil <i>Word Sequences</i>	55
Tabel 4. 6 Contoh Hasil Urutan Kata	56
Tabel 4. 7 Contoh Hasil <i>Word Embedding</i>	56
Tabel 4. 8 Hasil Pengujian Unit BiGRU	61
Tabel 4. 9 Hasil Pengujian <i>Dense Layer Neuron</i>	62
Tabel 4. 10 Hasil Pengujian <i>Dropout</i>	63
Tabel 4. 11 Hasil Pengujian <i>Batch Size</i>	63
Tabel 4. 12 Hasil Pengujian Penggunaan Word2Vec	64
Tabel 4. 13 Proses <i>5-fold Cross Validation</i>	65
Tabel 4. 14 Hasil <i>5-fold Cross Validation</i>	67
Tabel 4. 15 Hasil Pengujian Penanganan <i>Imbalanced Classes</i>	68
Tabel 4. 16 Hasil <i>Confusion Matrix</i>	70
Tabel 4. 17 <i>Performance Metrics</i>	70
Tabel 4. 18 Contoh Hasil Prediksi Sentimen dengan BiGRU	71
Tabel 4. 19 Hasil Perbandingan Metode dengan <i>K-fold</i>	72

BAB I

PENDAHULUAN

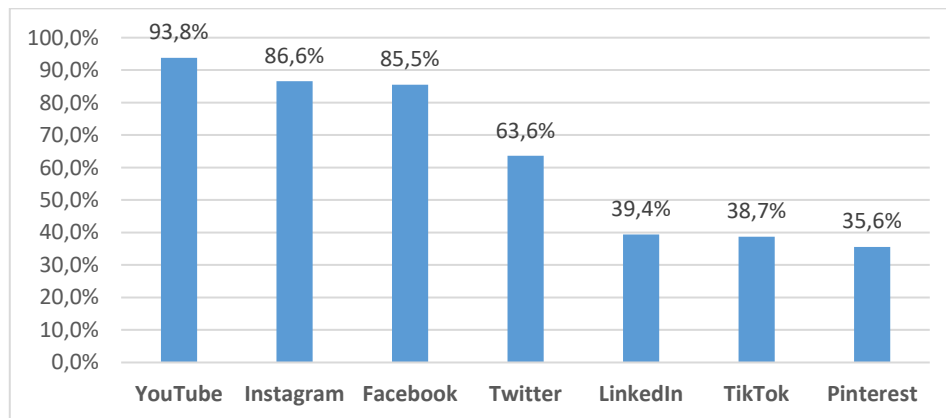
1.1 Latar Belakang

Tipe baru virus corona ditemui pada akhir bulan Desember 2019 di Kota Wuhan, China. Gejala umum dari penderita yang terpapar ditanda dengan indikasi demam, batuk serta keletihan (Huang *et al.*, 2020). Wabah penyakit baru yang diakibatkan oleh virus korona (2019-nCoV) ataupun yang biasa disebut dengan COVID-19 diresmikan secara formal sebagai pandemi global oleh *World Health Organization* pada 11 Maret 2020 (WHO, 2020).

Penyebaran COVID-19 yang cepat menimbulkan risiko terkena penyakit jika tidak segera ditangani. Salah satu cara yang paling mungkin untuk menghentikan penyebaran virus ini adalah dengan mengembangkan vaksin (C. Liu *et al.*, 2020). Menyikapi hal tersebut, pemerintah Indonesia juga terlibat aktif dalam merencanakan kegiatan vaksinasi yang akan diberikan kepada masyarakat. Presiden Jokowi pada tanggal 5 Oktober 2020 lalu meresmikan Peraturan Presiden (Perpres) Republik Indonesia Nomor 99 Tahun 2020 Tentang Pengadaan Vaksin dan Pelaksanaan Vaksinasi dalam Rangka Penanggulangan Pandemi Corona Virus Disease 2019 (COVID-19).

Program vaksinasi COVID-19 harus mempertimbangkan berbagai aspirasi, seperti melihat bagaimana respon dan opini masyarakat terhadap program vaksinasi

COVID-19 (Rachman & Pramana, 2020). Salah satu sarana yang sering digunakan oleh masyarakat untuk berpendapat adalah media sosial. Media sosial adalah sumber paling populer yang digunakan untuk berkomunikasi dengan banyak komunitas dan berbagi dokumen maupun data (Mustakim *et al.*, 2019). Pada bulan Januari 2021, tercatat dari total 274,9 juta penduduk di Indonesia, 170 juta di antaranya telah menggunakan media sosial sehingga angka penetrasinya sekitar 61,8% (We Are Social & Hootsuite, 2021). YouTube, Facebook, dan Instagram termasuk *platform* yang paling sering digunakan pengguna media sosial di Indonesia yang berusia 16 hingga 64 tahun pada Januari 2021. Berikut ini merupakan grafik yang menunjukkan media sosial yang paling sering digunakan di Indonesia:



Gambar 1. 1 Persentase Platform yang Sering Digunakan Pengguna Internet Berusia 16 hingga 64 Tahun di Indonesia pada Januari 2021
(Sumber: We Are Social & Hootsuite, 2021)

Analisis sentimen adalah studi tentang bagaimana suatu entitas dapat mengatasi dan memecahkan masalah berdasarkan opini publik, sikap, dan emosi suatu entitas,

dimana entitas tersebut dapat merepresentatifkan individu (Wati, 2016). Tujuan dari analisis sentimen ini adalah untuk melihat apakah pandangan atau pendapat seseorang tentang suatu isu atau objek cenderung positif, negatif, atau netral. (Rozi *et al.*, 2012). Analisis sentimen ini dapat melihat bagaimana pandangan publik dari kolom komentar yang disampaikan melalui YouTube, Facebook, dan Instagram terhadap kebijakan pemerintah.

Salah satu tantangan terbesar dari analisis sentimen adalah perlunya pakar atau ahli yang memiliki spesialisasi dibidang linguistik. Dibutuhkannya linguist untuk menganalisis dan mengelompokan pandangan yang terdapat dalam tulisan seorang individu karena kompleksnya sebuah bahasa sehingga sulit dipahami dari mata orang awam. Kehadiran *big data* menjadi tantangan bagi pakar dalam melakukan analisis sentimen karena melimpahnya data membutuhkan pengolahan data yang tepat untuk digunakan secara komprehensif (Yudistira, 2021).

Termotivasi mendukung program vaksinasi COVID-19, peneliti memodelkan klasifikasi teks menggunakan pendekatan *machine learning*. Digunakannya *machine learning* atau pembelajaran mesin untuk mengatasi permasalahan jumlah data yang besar dan pengembangan sistem secara otomatis. Dengan model ini diharapkan dapat diperoleh prediksi sentimen masyarakat terhadap program vaksiansi yang nanti akan berkontribusi dalam menyukseskan program tersebut guna melawan pandemi COVID-19.

1.2 Identifikasi Masalah

Dalam penelitian ini, permasalahan utamanya adalah bagaimana membuat model yang dapat mengklasifikasikan sentimen positif, netral, dan negatif pandangan masyarakat Indonesia terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia.

1.3 Maksud dan Tujuan Penelitian

Maksud dan tujuan dari penelitian ini adalah mengimplementasikan *Bidirectional Gated Recurrent Unit* (BiGRU) pada analisis sentimen untuk mendapatkan model klasifikasi opini positif, netral, dan negatif pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19.

1.4 Manfaat Penelitian

Hasil penelitian ini diharapkan dapat bermanfaat untuk melihat pandangan masyarakat terhadap program vaksinasi COVID-19 di Indonesia, sehingga dapat dijadikan bahan evaluasi dan rekomendasi dalam menjalankan program vaksinasi COVID-19 di Indonesia selanjutnya. Manfaat lain dari penelitian ini adalah untuk menambah pemahaman dalam pengaplikasian statistika, khususnya mengenai *text mining*, dan metode *Bidirectional Gated Recurrent Unit* (BiGRU).

1.5 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian ini yaitu adalah data komentar masyarakat dalam unggahan akun YouTube, Facebook, dan Instagram

Kementerian Kesehatan (Kemenkes) Republik Indonesia yang membahas terkait program vaksinasi COVID-19 periode Oktober 2020 sampai November 2021.

BAB II

TINJAUAN PUSTAKA

2.1 Pendahuluan

Seperti yang sudah dipaparkan di BAB I, penelitian ini memanfaatkan metode klasifikasi untuk melihat sentimen positif, netral, dan negatif masyarakat Indonesia terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia. Menimbang tipe data yang berupa teks, maka dibutuhkan metode klasifikasi yang sesuai agar mendapatkan model yang akurat dan dapat digunakan

2.2 Analisis Sentimen

Analisis sentimen merupakan teknologi penambangan dan analisis teks subjektif yang digunakan untuk mendapatkan pengetahuan dan informasi yang bermanfaat dari teks (Y. Zhang *et al.*, 2019). Menurut (B. Liu, 2012) *sentiment analysis* atau *opinion mining* adalah bidang ilmu yang luas dari pengolahan bahasa alami, komputasi linguistik dan *text mining* yang memiliki tujuan menganalisis opini, sentimen, evaluasi, sikap, dan emosi seseorang. Dilakukannya analisis sentimen ini bertujuan untuk melihat pendapat atau kecenderungan pendapat terhadap sebuah masalah, apa memiliki kecenderungan positif, negatif, atau netral (Rozi *et al.*, 2012)

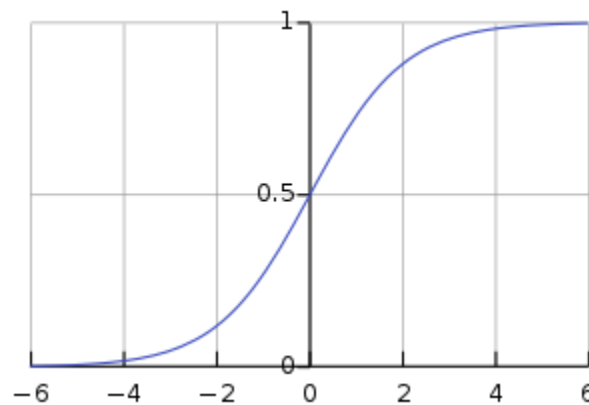
2.3 Klasifikasi Teks

Ada dua cara dalam penggolongan teks, yaitu klasterisasi teks dan klasifikasi teks. klasterisasi teks berhubungan dengan mencari struktur kelompok yang belum terlihat (*unsupervised*) dari *dataset* dokumen. Sedangkan pengklasifikasian teks merupakan proses membentuk kelas dari dokumen berdasarkan kelas kelompok yang sudah diketahui sebelumnya (*supervised*) (Darujati & Gumelar, 2012).

Klasifikasi teks adalah proses yang secara otomatis menempatkan dokumen teks ke dalam kategori berdasarkan isi dari teks tersebut (X. F. Zhang *et al.*, 2009). Klasifikasi teks ini adalah teknik dari *data mining* dan penambangan teks yang dimanfaatkan untuk menemukan atau mengatur kelas dengan menggunakan model pembelajaran mesin (Wijaya & Santoso, 2016).

2.4 Regresi Logistik

Regresi Logistik merupakan salah satu teknik klasifikasi yang umum digunakan. Regresi logistik mengasumsikan bahwa terdapat hubungan yang linear antara variabel prediktor dan respon (Tampil *et al.*, 2017). Walaupun mengasumsikan hubungan linear antara prediktor dan respon, grafik yang dihasilkan dari regresi logistik bukan merupakan garis lurus, melainkan seperti yang diilustrasikan gambar berikut:



Gambar 2. 1 Grafik Regresi Logistik

(Sumber: Hilbe, 2009)

Karena mengasumsikan hubungan linear antara prediktor dan respon, maka regresi logistik tidak tepat digunakan apabila tidak terdapat hubungan linear antara prediktor dan respon. Dalam klasifikasi teks dengan input kalimat, tiap kata pada kalimat dijadikan sebagai variabel independen untuk memprediksi respon. Oleh karena itu sebelum melakukan klasifikasi dengan regresi logistik terdapat variabel yang terlebih dahulu harus dicek linearitasnya. Pada penelitian ini terdapat 9070 variabel yang harus dicek terlebih linearitasnya. Oleh karena itu regresi logistik tidak cocok untuk digunakan dalam mengklasifikasi teks karena tidak efisien.

2.5 *Naïve Bayes Classifier*

Klasifikasi Bayes merupakan klasifikasi statistik yang digunakan untuk melakukan prediksi pada kelas suatu anggota probabilitas. Ide awal dari *Naive Bayes* ini yaitu dengan *join* probabilitas kata dan kategori yang diberikan oleh sebuah

dokumen (Darujati & Gumelar, 2012). Teknik klasifikasi ini didasari oleh *Bayes' Theorem* yang secara sederhana dapat diinterpretasikan bahwa peluang suatu kejadian $P(A|B)$ dapat dihitung dengan menggunakan data yang sudah ada yang menggambarkan kondisi $P(B|A)$. Kejadian $P(A|B)$ disebut posterior sedangkan kejadian yang sudah diketahui yaitu $P(B|A)$ disebut dengan prior.

Naïve Bayes Classifier mengasumsikan bahwa tiap fitur-fiturnya bersifat independen satu sama lain (Pang *et al.*, 2002). Hal ini yang menyebabkan *Naïve Bayes Classifier* tidak cocok untuk klasifikasi teks pada penelitian karena tiap kata pada kalimat memiliki hubungan satu sama lain sehingga menghasilkan suatu kesatuan (kalimat). Selanjutnya, masalah lain dengan teknik ini adalah, jika beberapa nilai fitur, yang tidak ditemukan dalam data pelatihan, terlihat pada data input, probabilitas yang sesuai akan ditetapkan ke 0 (Singhal & Bhattacharyya, 2016).

2.6 *Support Vector Machine (SVM)*

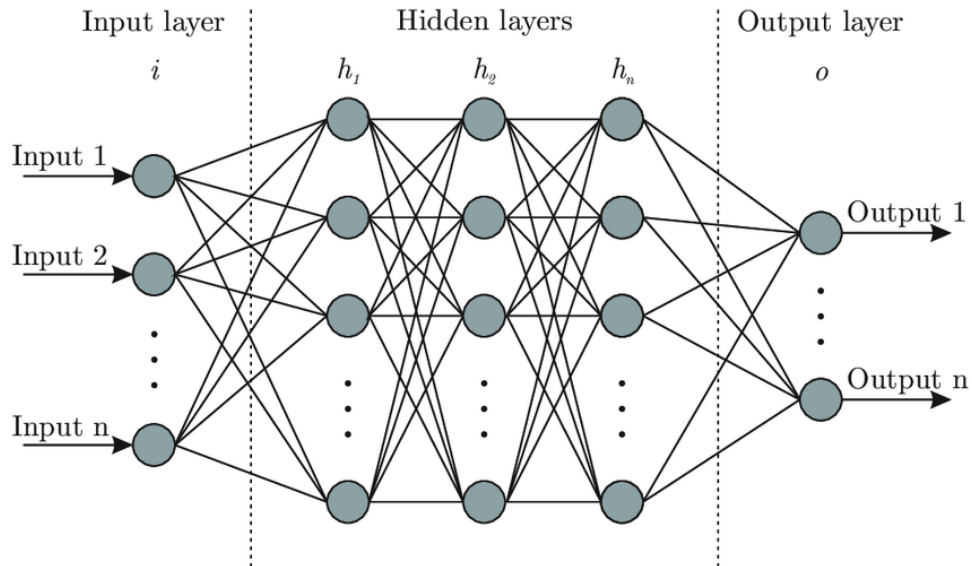
Support Vector Machine (SVM) pertama kali diperkenalkan pada akhir 1979 oleh Vapnik yang selanjutnya mendapatkan perhatian yang lebih pada tahun 1992 bersama dengan Boser (Härdle *et al.*, 2014). *Support Vector Machine* adalah salah satu metode dalam machine learning yang digunakan untuk melakukan suatu prediksi dan pengklasifikasian (Sari *et al.*, 2020). Konsep klasifikasi dengan menggunakan metode *Support Vector Machine* memiliki ide dasar bahwa menemukan fungsi pemisah (*hyperlane*) yang nantinya dapat memisahkan dua kelas secara optimal (Tan *et al.*,

2006) Metode klasifikasi SVM bertujuan untuk mencari *hyperplane* pemisah dengan nilai margin terbesar sehingga dapat memisahkan observasi dengan optimal.

SVM merupakan salah satu teknik klasifikasi yang cukup banyak digunakan karena pada umumnya memiliki performa yang baik. Akan tetapi, SVM tidak berfungsi dengan baik ketika memiliki kumpulan data yang besar karena waktu *training* yang dibutuhkan cukup tinggi (Akkaya, 2019).

2.7 *Artificial Neural Network (ANN)*

Artificial Neural Network (ANN) atau disebut juga Jaringan Syaraf Tiruan, adalah suatu teknik komputasi yang terinspirasi dari sistem saraf pada hewan. Pada dasarnya, ANN terdiri dari neuron-neuron (node) yang tersambung satu sama lain (Pangaribuan & Sagala, 2017). Jika pada hewan neuron-neuron tersebut dapat mentransmisikan sinyal, pada ANN neuron-neuron tersebut terhubung dan memiliki konstanta (bobot) masing-masing. Bobot-bobot tersebut awalnya diinisiasi dengan angka acak dan kemudian diperbaharui dengan mencari nilai yang dapat memperkecil *error/loss*. Selain itu, di setiap neuron juga terdapat fungsi yang disebut fungsi aktivasi. Dilihat dari susunan lapisannya, ANN terdiri dari input, hidden, dan output layer. Ilustrasi dari ANN dapat dilihat pada Gambar 2.2



Gambar 2. 2 Arsitektur Artificial Neural Network

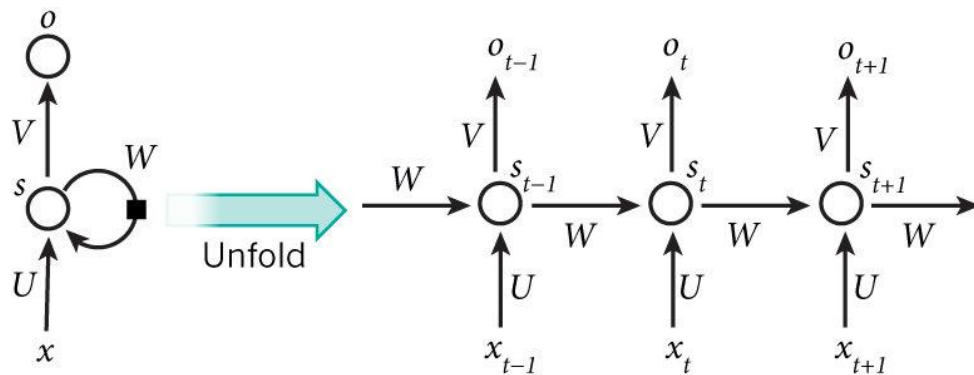
(Sumber: Bre *et al.*, 2018)

Model ANN memiliki kelebihan jika dibandingkan dengan metode-metode sebelumnya. Salah satu kelebihan ANN adalah dapat memproses data tanpa memerlukan asumsi-asumsi tertentu pada data. Selain itu, ANN merupakan metode teknik klasifikasi yang iteratif sehingga diperoleh performa model yang secara teori lebih baik apabila dibandingkan dengan metode yang tidak memiliki proses iteratif. Akan tetapi, ANN memiliki *computational cost* yang sangat tinggi khususnya apabila melakukan klasifikasi teks (Sakunthala *et al.*, 2018). Meskipun memiliki kekurangan yakni biaya komputasi tinggi, ANN *robust* terhadap *dataset* yang memiliki banyak *noise* dan juga efisien untuk dataset besar. Kompleksitas dan ukuran input pada ANN tergantung pada jenis dan arsitektur model yang digunakan, sehingga biaya komputasi

dapat ditangani dengan memodifikasi arsitektur meskipun terdapat *trade off* antara biaya komputasi dan akurasi model.

2.8 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) merupakan salah satu metode deep learning yang meniru cara berpikir manusia dalam pengambilan keputusan. Proses pelatihan pada RNN sangat mirip dengan pelatihan pada *Neural Network* menggunakan algoritma *Backpropagation* tetapi dengan sedikit putaran. Karena parameter yang dibagikan secara merata pada setiap *time step*, maka *gradient* untuk setiap *output* tergantung tidak hanya pada kalkulasi dari *time step* saat ini, tetapi juga pada *time step* sebelumnya. Pada RNN terdapat beberapa unit gate seperti *Gated Recurrent Unit (GRU)*, *Backpropagation Through Time (BPTT)* dan *Long Short-Term Memory (LSTM)* (Putra *et al.*, 2018).



Gambar 2. 3 Arsitektur *Recurrent Neural Network*

(Sumber: Kumar *et al.*, 2021)

Gambar 2.3 merupakan arsitektur RNN, dimana simbol x_t adalah *input* pada setiap langkah atau disebut juga dengan *time step*. Sedangkan untuk simbol S_t adalah *hidden state* pada setiap *time step* t . *Hidden state* bisa disebut juga sebagai *memory* pada sebuah jaringan yang berfungsi menyimpan hasil kalkulasi dan rekaman yang telah dilakukan. Untuk menghitung *hidden state* menggunakan persamaan ini.

$$s_t = \tanh (U . x_t + W . s_{t-1})$$

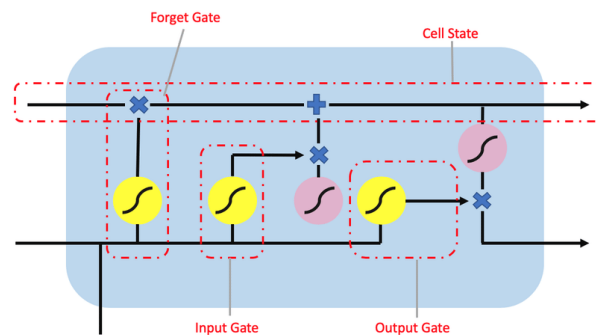
s_{t-1} digunakan untuk menghitung *hidden state* yang pertama, biasanya pada inisialisasi diawali dari 0 (nol). O_t adalah *output* untuk setiap step t . *Output* tahap O_t dihitung disetiap waktu t di dalam memori. Rekaman RNN pada setiap waktunya sangatlah rumit terlebih lagi s_t tidak bisa merekam informasi terlalu banyak pada setiap *time step*-nya. Untuk mengatasi masalah tersebut muncullah pengembangan dari RNN yaitu *Gated Recurrent Unit* (GRU) dan *Long-Short Term Memory*. LSTM dan GRU dapat digunakan untuk mengatasi permasalahan dalam mengingat dependensi yang lama, dan juga berguna untuk teks berukuran besar.

2.8.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) merupakan pengembangan dari arsitektur RNN yang sama-sama memiliki jaringan berulang. LSTM diusulkan pada tahun 1997 oleh Sepp Hochreiter dan Jurgen Schmidhuber (Hochreiter & Schmidhuber, 1997). Sebuah unit komputasi dari jaringan LSTM disebut *memory cell* atau *cell state*. Hal

inilah yang menjadi pembeda LSTM dengan RNN. Pada RNN perulangan jaringan hanya menggunakan satu lapisan sederhana, yaitu lapisan *tanh*.

Sel LSTM terdiri dari bobot (*weight*) dan gerbang (*gate*). LSTM terdiri dari satu set blok yang terhubung berulang, yang dikenal sebagai blok memori. Blok-blok ini dapat dianggap sebagai versi yang berbeda dari *chip* memori dalam komputer digital. Masing-masing berisi satu atau lebih sel memori yang terhubung berulang dan tiga unit multiplikasi yaitu adanya *input*, *output* dan *forget gate*. Jaringan hanya dapat berinteraksi dengan sel melalui gerbang atau *gate* (Brownlee, 2017).



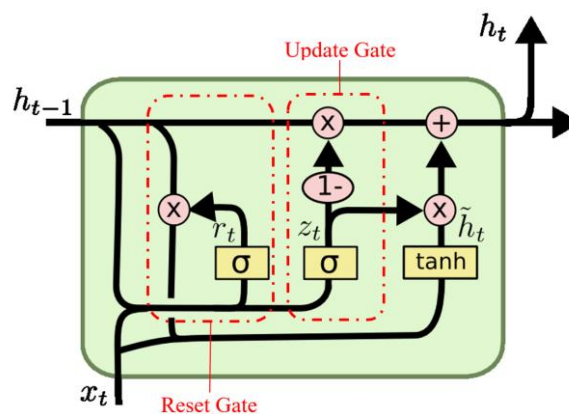
Gambar 2. 4 Unit LSTM
(Sumber: Rengasamy *et al.*, 2020)

Semenjak LSTM diperkenalkan pada tahun 1997 sudah banyak peneliti-peneliti yang menggunakan metode ini untuk berbagai kasus, seperti analisis deret waktu dan analisis sentimen. Akan tetapi pada penerapannya LSTM memiliki perhitungan yang cukup kompleks dan komputasi yang tinggi. Oleh karena itu, pada penelitian ini peneliti mencari metode yang memiliki tingkat komputasi yang lebih sederhana, namun memiliki kinerja yang sebanding.

2.8.2 Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU) merupakan pengembangan dari arsitektur RNN. GRU adalah pengembangan yang cukup baru yang diusulkan oleh K. Cho pada tahun 2014. Sama seperti *Long Short-Term Memory*, *Gated Recurrent Unit* juga dapat mengurangi masalah *Vanishing Gradient* dari *Recurrent Neural Network* (Ghods & Cook, 2019). GRU memiliki unit gerbang yang memodulasi aliran informasi di dalam unit, namun tanpa memiliki sel memori terpisah (Biswas *et al.*, 2015).

Gated Recurrent Unit (GRU) memiliki dua *gate* yang disebut dengan *reset gate* dan *update gate*. *Reset gate* sama seperti *forget gate* dan *input gate* pada LSTM, yang akan memilih informasi mana yang harus disimpan atau dibuang dan bagaimana cara menggabungkan *input* yang baru dengan *memory* sebelumnya. Sedangkan *update gate* digunakan untuk menentukan seberapa banyak data yang perlu dibuang (Giarsyani, 2020).



Gambar 2. 5 Unit *Gated Recurrent Unit*

(Sumber: Abdulwahab, 2017)

Gated Recurrent Unit (GRU) menghitung dua gerbang yang disebut *update* dan *reset gate* yang mengontrol aliran informasi melalui setiap unit tersembunyi atau *hidden state*. Setiap *hidden state* pada langkah waktu t dihitung menggunakan persamaan berikut:

$$z_t = \sigma(W_z \cdot x_t + U_z h_{t-1} + b_z) \quad (2.1)$$

$$r_t = \sigma(W_r \cdot x_t + U_r h_{t-1} + b_r) \quad (2.2)$$

$$\hat{h}_t = \tanh(W_h \cdot x_t + r_t \odot U_h h_{t-1} + b_h) \quad (2.3)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t \quad (2.4)$$

Dalam rumus ini z_t dan r_t adalah *update* dan *reset gate*. h_{t-1} merupakan *hidden state*, \hat{h}_t adalah *candidate hidden state*, h_t adalah *final hidden state*. $W_z, U_z, W_r, U_r, W_h, U_h$ adalah parameter bobot dan b merupakan parameter bias. Untuk simbol \odot adalah perkalian elemen dan σ merupakan fungsi sigmoid.

GRU tergolong kedalam metode yang cukup baru, namun GRU sudah terbukti menunjukkan kinerja yang lebih baik pada kumpulan data tertentu (Gruber & Jockisch, 2020). *Gated Recurrent Unit* atau GRU merupakan alternatif yang lebih sederhana dibandingkan *Long Short-Term Memory* (LSTM) dan juga cukup populer (DiPietro & Hager, 2019). GRU efektif dalam tugas sentimen analisis karena kemampuannya untuk mengingat lama dependensi dan sangat berguna untuk teks berukuran besar (Biswas *et al.*, 2015). GRU adalah metode yang cukup tepat dalam penelitian kali ini, melihat kelebihan dan performanya yang baik. Namun, GRU masih bisa ditingkatkan

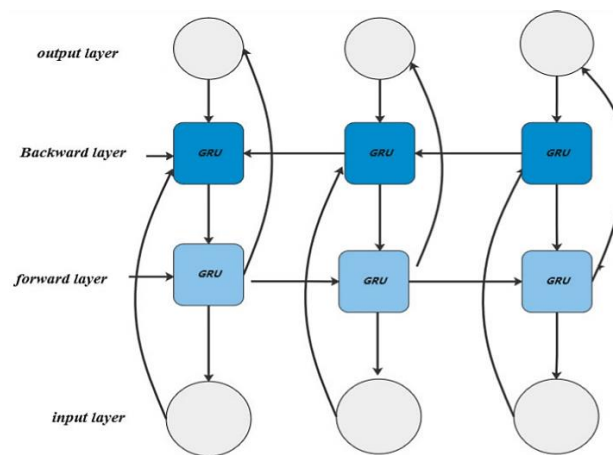
kinerjanya dengan mengembangkan arsitekturnya menggunakan *Bidirectional Gated Recurrent Unit*

2.8.3 *Bidirectional Gated Recurrent Unit (BiGRU)*

Salah satu kelemahan jaringan GRU adalah hanya dapat menggunakan konteks sebelumnya tanpa mempertimbangkan konteks masa depan, sehingga GRU hanya dapat mengatur urutan dari depan ke belakang yang mengakibatkan hilangnya informasi (Yu *et al.*, 2019). GRU dapat dikembangkan menjadi *Bidirectional Gated Recurrent Unit* (BiGRU) yang juga memiliki komputasi lebih sederhana dibandingkan dengan LSTM, namun tetap mempertahankan kinerja yang sebanding.

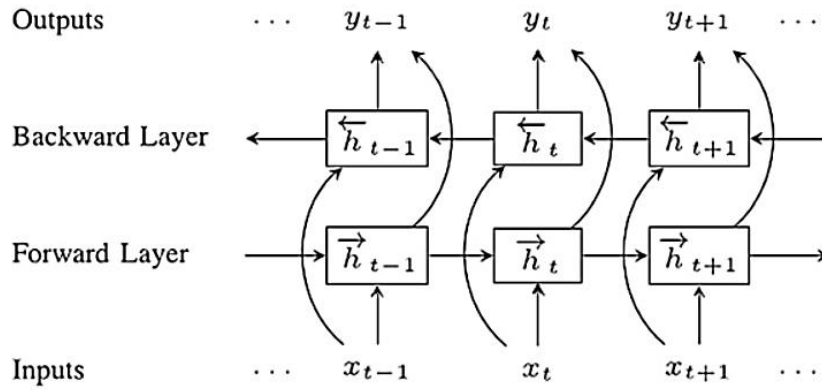
Bidirectional Gated Recurrent Unit atau BiGRU adalah model pemrosesan urutan yang terdiri dari dua GRU, GRU pertama mengambil input dalam arah maju, dan yang lainnya dalam arah mundur. BiGRU adalah model *Recurrent Neural Network* dua arah dengan hanya gerbang *input* dan *forget* (Rana, 2016). Karena BiGRU memiliki struktur yang tidak terlalu rumit, BiGRU bisa berlatih lebih cepat dibandingkan dengan LSTM. Sejak diperkenalkan, banyak peneliti menggunakan BiGRU untuk banyak tugas yang berbeda, termasuk analisis sentimen dan variannya (Setiawan *et al.*, 2020). Begitu banyak peneliti telah memanfaatkan *Bidirectional GRU* karena mampu memproses data di kedua arah dan informasi dari kedua *hidden layer* yang terpisah, kemudian dikumpulkan di *output layer* (Abdelgwad *et al.*, 2021).

Arsitektur dasar jaringan *Bidirectional* GRU sebenarnya hanya menyatukan dua GRU yang terpisah. Urutan *input* disediakan untuk satu jaringan dalam urutan waktu normal dan untuk jaringan lain dalam urutan waktu terbalik. Pada setiap langkah, *output* dari kedua jaringan biasanya digabungkan. Struktur seperti itu dapat memberikan informasi konteks yang lengkap (Abdelgwad *et al.*, 2021). Ilustrasi arsitektur BiGRU dapat dilihat pada Gambar 2.6.



Gambar 2. 6 Arsitektur *Bidirectional* GRU
(Sumber: Ali *et al.*, 2021)

Model BiGRU (*Bidirectional Gated Recurrent Unit*) dapat diperoleh dengan mengganti *hidden layer neurons* di *Bidirectional Recurrent Neural Network* dengan unit memori GRU, yang strukturnya ditunjukkan pada Gambar 2.7.



Gambar 2. 7 *Bidirectional Recurrent Neural Network*
(Sumber: Wang *et al.*, 2015)

Dari Gambar 2.6 mengilustrasikan arsitektur *Bidirectional Recurrent Neural Network* yang terdiri dari dua RNN yaitu RNN maju dan mundur. Yang pertama membaca urutan input dalam arah maju (x_1, \dots, x_n) dan menghasilkan urutan *forward hidden state* ($\vec{h}_1, \dots, \vec{h}_n$), sedangkan yang kedua membaca urutan input dalam urutan terbalik (x_n, \dots, x_1) yang menghasilkan urutan *backward hidden state* ($\overleftarrow{h}_n, \dots, \overleftarrow{h}_1$). Untuk input n-dimensi yang diberikan (x_1, x_2, \dots, x_n). Pada waktu t , *hidden layer* BiGRU menghasilkan h_t . Proses perhitungannya adalah sebagai berikut:

$$\vec{h}_t = \sigma(W_{x\vec{h}} x_t + W_{\vec{h}\vec{h}} \vec{h}_{t-1} + b_{\vec{h}}) \quad (2.5)$$

$$\overleftarrow{h}_t = \sigma(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \quad (2.6)$$

$$h_t = \vec{h}_t \oplus \overleftarrow{h}_t \quad (2.7)$$

Dimana W adalah matriks bobot yang menghubungkan dua lapisan, b adalah vektor bias, σ adalah fungsi aktivasi, \vec{h}_t dan \overleftarrow{h}_t masing-masing adalah output dari GRU maju dan mundur, serta \oplus adalah penjumlahan elemen.

Melihat kelebihan dan performa yang dimiliki BiGRU, pada penelitian analisis sentimen pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia ini menggunakan metode *Bidirectional Gated Recurrent Unit* atau BiGRU. Selanjutnya terdapat studi kasus perbandingan metode untuk analisis sentimen dengan pendekatan *machine learning* dari beberapa publikasi yang dijabarkan pada tabel perbandingan literatur berikut.

Tabel 2. 1 Perbandingan Literatur

Penulis/ Tanggal	Judul	Metodologi	Hasil dan Pembahasan	Keterangan
Merinda Lestandy, Abdurrahim, & Lailis Syafa'ah/ 2021	Analisis Sentimen Tweet Vaksin COVID-19 Menggunakan <i>Recurrent Neural Network</i> dan <i>Naïve Bayes</i>	RNN dan <i>Naïve Bayes Classifier</i>	Diperoleh hasil akurasi keseluruhan 97,7% untuk RNN, dan untuk <i>Naïve Bayes Classifier</i> 80%	Metode berbasis RNN memperoleh hasil akurasi lebih besar dengan perbedaan 17,7% dengan <i>Naïve Bayes Classifier</i>
Liang Zhou & Xiaoyong Bian/ 2019	<i>Improved text sentiment classification methodbased on BiGRU-Attention</i>	RNN, LSTM, GRU, BiLSTM, BiGRU, BiLSTM + Attention, BiGRU+ Attentiton	BiGRU memperoleh akurasi sebesar 88,56%, RNN 80,74%, LSTM 87,23%, GRU 88,56%, BiLSTM 88,23%, BiLSTM - Attention 89,07%, BiGRU-Attentiton 90,45%.	Metode berbasis Gated Recurrent Unit (GRU) selalu memperoleh akurasi lebih besar dibandingkan <i>simple</i> RNN dan LSTM, baik dengan <i>Bidirectional</i> maupun <i>Attention</i>

Penulis/ Tanggal	Judul	Metodologi	Hasil dan Pembahasan	Keterangan
Xing Yin, Changhui Liu, & Xiaodong Fang/ 2021	<i>Sentiment analysis based on BiGRU information enhancement</i>	CNN-BiGRU, CNN- BiLSTM-Attention, BERT, BiGRU	BiGRU memperoleh akurasi sebesar 82,6%, CNN-BiGRU 73,6%, CNN- BiLSTM-Attention 74,3%, dan BERT 82,1%	Metode BiGRU dengan hyperparameter terbaik memperoleh hasil akurasi lebih besar dibandingkan BERT, dan metode Hybrid lainnya
Xuanzhen Feng1 & Xiaohong Liu / 2019	<i>Sentiment Classification of Reviews Based on BiGRU Neural Network and Fine-grained Attention</i>	SVM, Mixsupervised+BOW, MBCNN-LSTM, RNN, BiLSTM, BiGRU, Atten-BiGRU, FGAtten-BiGRU, FGANN-BiGRU	Diperoleh hasil akurasi dari SVM sebesar 81,4%, Mixsupervised+BOW 88,9%, MBCNN- LSTM 89,5%, RNN 84,9%, BiLSTM 86.7%, BiGRU 87,9%, Atten-BiGRU 88,7%, FGAtten- BiGRU 90,5%, FGANN-BiGRU 90,9%	Metode dengan unit BiGRU secara keseluruhah mendapatkan rata- rata akurasi sebesar 89,5% dan paling tinggi mencapai 90,9%. Dimana hasil tersebut lebih baik dibandingkan metode-metode lain yang telah diuji

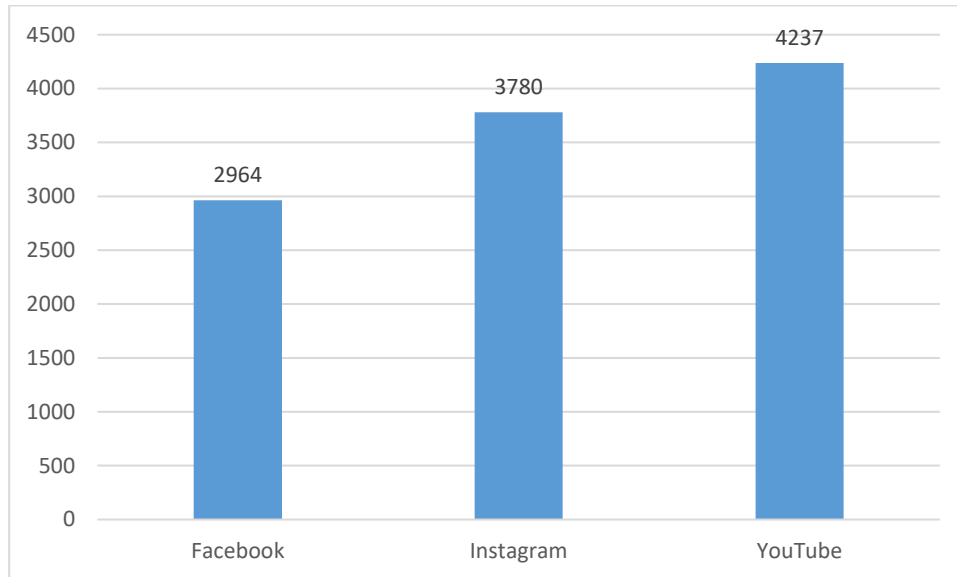
BAB III
ANALISIS SENTIMEN PENGGUNA MEDIA SOSIAL TERHADAP
KEBIJAKAN PEMERINTAH DALAM PROGRAM VAKSINASI COVID-19
DI INDONESIA MENGGUNAKAN METODE *BIDIRECTIONAL GATED*
RECURRENT UNIT (BiGRU)

3.1 Pendahuluan

Pada bab sebelumnya telah dijelaskan tinjauan pustaka yang berkaitan dengan penelitian ini. Metode yang terpilih untuk digunakan dalam penelitian ini adalah metode *Bidirectional Gated Recurrent Unit*. Selanjutnya akan dilakukan proses analisis yang akan dijelaskan mulai dari data yang digunakan, langkah-langkah penelitian dengan metode terpilih, sampai evaluasi model.

3.2 Data Penelitian

Data yang digunakan pada penelitian ini adalah data yang dikumpulkan dari akun YouTube, Facebook, dan Instagram Kementerian Kesehatan (Kemenkes) Republik Indonesia. Data berupa teks komentar masyarakat dalam unggahan akun YouTube, Facebook, dan Instagram Kementerian Kesehatan Republik Indonesia yang membahas terkait program vaksinasi COVID-19 pada periode Oktober 2020 sampai November 2021. Data yang berhasil dikumpulkan untuk penelitian ini sebanyak 10161 komentar. Berikut adalah grafik banyaknya komentar yang dikumpulkan dalam penelitian ini berdasarkan sumber *platform* media sosialnya:



Gambar 3. 1 Grafik Frekuensi Data Penelitian Berdasarkan Sumbernya

Pada Gambar 3.1 terlihat bahwa banyak data teks yang digunakan dalam penelitian ini yang bersumber dari Facebook sebanyak 2964 komentar, Instagram sebanyak 3780 komentar, dan YouTube sejumlah 4237 data komentar.

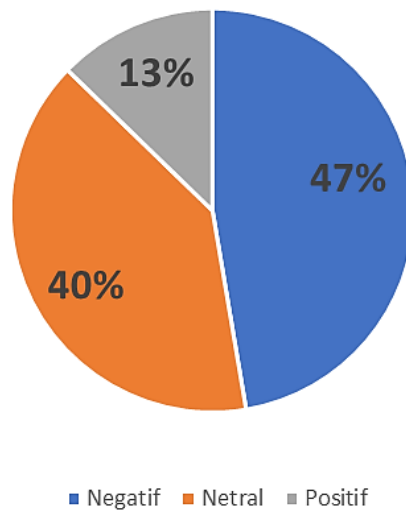
Data komentar yang sudah dikumpulkan pada penelitian ini selanjutnya dilakukan pelabelan data yaitu dengan memberikan tag yang bermakna ke setiap data. Pelabelan secara manual menghasilkan data yang akurat karena manusia dapat membedakan dengan tepat suatu kalimat termasuk ke dalam suatu sentiment (Tama, 2018). Pelabelan dilakukan oleh tim pelabelan dari Pusat Riset Kecerdasan Artifisial dan Big Data Universitas Padjadjaran yang dipimpin oleh Indra Sarathan, M. Hum., dosen Program Studi Sastra Indonesia Fakultas Ilmu Budaya (FIB) Universitas

Padjadjaran. Data komentar dilabelkan ke dalam kelas sentimen positif, netral, dan negatif. Label yang digunakan ada 3 yaitu “Negatif”, “Netral”, dan “Positif”. Label “Negatif” adalah komentar yang mengandung kata-kata buruk, ejekan, atau menyatakan kontra. Label “Netral” merupakan komentar yang memiliki kata yang bermakna informasi, dan tidak memihak pihak pro maupun kontra. Label “Positif” adalah data komentar mengandung kata-kata baik, pujian, pernyataan setuju, atau dukungan. Pada Tabel 4.1 adalah contoh dari pelabelan data pada dataset penelitian ini.

Tabel 3. 1 Contoh Pelabelan Data

Data	Label
pemerintah sudah berusaha semaksimal mungkin memutus mata rantai virus bahkan sampai geram menangani pandemi ini tinggal kitanya sebagai masyarakat mari kita bantu pemerintah kita jangan saling menyalahkan dimulai dari diri kita sendiri jaga kesehatan	Positif
Vaksin adalah produk biologi yang berisi antigen	Netral
jangan mau divaksin	Negatif

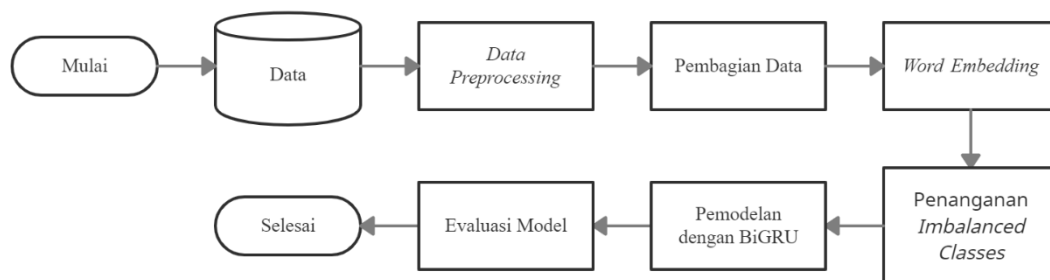
Didapatkan label “Negatif” sebanyak 4810 atau sekitar 47% dari dataset, label “Netral” sebanyak 4057 atau sekitar 40% dari dataset, dan Label “Positif” sebanyak 1294 atau sekitar 13%. Gambar 3.2 adalah grafik presentase hasil pelabelan data.



Gambar 3. 2 Grafik Presentase Hasil Pelabelan Data

3.3 Langkah-langkah Penelitian

Penelitian ini dilakukan dengan tahapan-tahapan penelitian yang meliputi *data preprocessing*, pembagian data, *word embedding*, penanganan *imbalanced classes*, pemodelan dengan *Bidirectional Gated Recurrent Unit* (BiGRU), sampai evaluasi model. Pada Gambar 3.2 merupakan alur atau tahapan penelitian yang dilakukan.



Gambar 3. 3 Flowchart Penelitian

3.3.1 *Data Preprocessing*

Tahapan ini merupakan tahapan yang sangat memiliki pengaruh besar terhadap hasil kualitas data yang akan diolah pada tahapan selanjutnya. Setiap kalimat dan kata akan diproses ke dalam beberapa tahap yaitu *case folding*, *tokenizing*, *filtering*, *stemming* (Hermawan & Ismiati, 2020).

A. *Case Folding*

Tahapan yang pertama adalah *case folding* yaitu merubah kalimat teks yang mengandung huruf kapital menjadi huruf kecil. Karena dengan adanya *case folding* bisa mengurangi redundansi data. Misalnya huruf “B” menjadi huruf “b”.

B. *Tokenizing*

Tahap kedua adalah *tokenizing* yaitu proses memisahkan atau memotong tweet yang berupa frasa atau kalimat menjadi kata per kata atau yang dikenal sebagai token. *Tokenizing* ini dilakukan agar nantinya setiap kata menjadi bentuk *array*.

C. *Filtering*

Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil *tokenizing*. *filtering* bisa dibantu menggunakan *stopword*, *stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang karena tidak terlalu bermakna. Contoh *stopwords* adalah “yang”, “dan”, “di”, “dari” dan seterusnya. Kata-kata seperti “dari”, “yang”, “di”, dan “ke” adalah beberapa contoh kata-kata yang berfrekuensi tinggi dan dapat ditemukan hampir dalam setiap dokumen.

Pada tahap ini juga dilakukan pembersihan dan penormalisasian data teks. Pembersihan data teks merupakan tahap pembersihan atribut yang tidak berpengaruh sama sekali terhadap hasil klasifikasi sentimen. Contoh dari atribut yang tidak penting tersebut adalah yaitu komentar yang diawali dengan atribut ('@'), *hashtag* dengan atribut ('#'), dan karakter simbol (~!@#\$%^&*()_+?<>,.?:{ }[]|). Atribut yang tidak berpengaruh tersebut juga akan dihilangkan dari dokumen. Selanjutnya akan dilakukan normalisasi data teks. Normalisasi data teks dilakukan dengan mengubah kata yang tidak sesuai dengan EYD. Pada tahap ini dilakukan konversi kata singkatan, konversi kata baku, dan konversi kata Inggris. Tahap *filtering* ini bisa mengurangi ukuran indeks dan waktu pemrosesan. Selain itu, juga dapat mengurangi *level noise*.

D. Stemming

Teknik *Stemming* diperlukan selain memperkecil jumlah indeks yang berbeda dari suatu dokumen, juga sebagai melakukan pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa namun memiliki bentuk atau *form* yang berbeda karena mendapatkan imbuhan yang berbeda.

Proses *stemming* pada teks berbahasa Indonesia berbeda dengan *stemming* pada teks berbahasa Inggris. Pada teks berbahasa Inggris, proses yang diperlukan hanya proses menghilangkan sufiks. Sedangkan pada teks berbahasa Indonesia semua kata imbuhan baik itu sufiks dan prefiks juga dihilangkan.

3.3.2 Pembagian Data

Data pada penelitian ini akan dikelompokkan ke dalam data *training* (pelatihan), dan data *testing* (pengujian). Data *training* digunakan untuk membuat dan melatih model yang akan digunakan. Data *testing* digunakan untuk menguji model yang sudah dibuat.

3.3.3 Word Embedding

Word embedding atau penyisipan kata adalah istilah yang digunakan untuk representasi kata untuk analisis teks, biasanya dalam bentuk vektor bernilai nyata yang mengkodekan arti kata sedemikian rupa sehingga kata-kata yang lebih dekat dalam vektor ruang diharapkan memiliki makna yang sama. Metode word embedding yang digunakan dalam penelitian ini adalah *Word2Vec*.

Algoritma Word2vec ini diciptakan oleh Mikolov dkk. pada tahun 2013. Word2Vec ini banyak digunakan dalam penelitian NLP. Model ini merupakan salah satu aplikasi *unsupervised learning* menggunakan neural network yang terdiri dari *hidden layer* dan *fully connected layer*. Word2Vec mengandalkan informasi lokal dari bahasa semantik yang dipelajari dari kata tertentu dipengaruhi oleh kata - kata sekitarnya. Terdapat dua algoritma Word2vec yaitu *Continuous Bag-of-Word* (CBOW) dan *Skip-Gram* (Nurdin *et al.*, 2020). Pada penelitian ini menggunakan algoritma *Skip-Gram*, karena algoritma *Skip-Gram* bertujuan untuk memprediksi kata target yang ada disekitar kata yang ada. Pembentukan model *Skip-Gram* dipengaruhi oleh beberapa

parameter, diantaranya jumlah dimensi dan jumlah *window* kata yang digunakan (Rahutomo *et al.*, 2019).

3.3.4 Penanganan *Imbalanced Classes*

Imbalanced Classes merupakan sebuah kondisi dimana terjadi ketidakseimbangan pada dataset terhadap fitur targetnya. Cara yang paling umum digunakan untuk menangani permasalahan ketidakseimbangan kelas adalah dengan melakukan *oversampling* atau *undersampling* untuk mengubah jumlah data pada kelas mayoritas atau minoritas sehingga membentuk dataset yang seimbang (Indrawati *et al.*, 2020).

Random Over Sampling (ROS) adalah salah satu teknik *oversampling*. ROS menduplikasi sampel data di kelas minoritas dan menambahkannya ke set data latih tanpa menambah variasi data kelas (Hayaty *et al.*, 2021). ROS meningkatkan ukuran kumpulan data pelatihan melalui pengulangan sampel asli hingga distribusi kelas seimbang. Adapun *Random Under Sampling* (RUS) salah satu teknik *undersampling*. RUS melakukan kebalikan dari ROS, ia menghilangkan beberapa sampel dari kelas mayoritas untuk menyeimbangkannya dengan kelas minoritas (Fernández *et al.*, 2018). Penanganan *Imbalanced Classes* tepat digunakan sebagai salah satu metode pengoptimalan BiGRU, teknik ROS dan RUS ini memungkinkan untuk meningkatkan atau menurunkan performa dari model klasifikasi.

3.3.5 Pemodelan dengan *Bidirectional Gated Recurrent Unit (BiGRU)*

Setelah melalui *data preprocessing*, data sudah siap untuk digunakan dalam proses pemodelan. Pada penelitian ini pemodelan dilakukan dengan menggunakan BiGRU. Berikut ini merupakan komponen-komponen yang terdapat pada BiGRU:

A. Lapisan *Embedding*

Lapisan *Embedding* didefinisikan sebagai lapisan tersembunyi pertama dari sebuah jaringan. Lapisan *Embedding* diinisialisasi dengan bobot acak dan akan mempelajari *embedding* untuk semua kata dalam dataset pelatihan. Lapisan *Embedding* dapat digunakan sebagai bagian dari model di mana *embedding* dipelajari bersama dengan model itu sendiri.

B. Lapisan *BiGRU*

BiGRU ini dilakukan dengan terlebih dahulu melakukan *reset gate* kemudian dilakukan *update* pada *gate*, mencari *candidate hidden state* dan terakhir akan menghasilkan memory akhir yang akan digunakan sebagai *output*.

- *Update Gate*

Tujuan dari *update gate* adalah untuk menentukan seberapa banyak data yang perlu dibuang, dimana hasil dari *update gate* ini adalah antara 0 hingga 1 karena menggunakan fungsi sigmoid. Oleh karena itu, semakin mendekati 1 nilai maka informasi itu akan digabungkan dengan informasi di masa lalu, namun jika nilai yang

didapat mendekati 0 maka hanya informasi baru yang akan disimpan. Perhitungan untuk *update gate* menggunakan persamaan (2.1) pada bab sebelumnya.

- *Reset Gate*

Reset gate memungkinkan model untuk mengabaikan informasi masa lalu yang mungkin tidak relevan. *Reset gate* akan mengevaluasi kembali kinerja gabungan dari input sebelumnya dan input baru serta mengatur ulang sesuai kebutuhan untuk input baru. *Reset gate* juga menggunakan fungsi sigmoid sehingga nilai yang mendekati 1 akan disimpan sedangkan nilai yang mendekati 0 akan diabaikan. Untuk menghitung nilai *reset gate* menggunakan persamaan (2.2) pada bab sebelumnya.

- *Candidate Hidden State*

Candidate hidden state adalah menggabungkan informasi dari status tersembunyi sebelumnya dengan input yang ada. Pada perhitungan *candidate hidden state* ini menggunakan tanh. Untuk menghitung *candidate hidden state* menggunakan persamaan (2.3) yang terdapat bab sebelumnya.

- *Final Hidden State*

Final hidden state atau juga bisa disebut sebagai *output* yang akan diteruskan pada *timestep* berikutnya. Persamaan (2.4) pada bab sebelumnya digunakan untuk menghitung nilai *final hidden state*.

- *Bidirectional*

Bidirectional GRU merupakan penyatuan dua GRU yang terpisah. Urutan *input* disediakan untuk satu jaringan dalam urutan waktu normal dan untuk jaringan lain dalam urutan waktu terbalik. Pada setiap langkah, output dari kedua jaringan akan digabungkan. Persamaan (2.5), (2.6), dan (2.7) pada bab sebelumnya digunakan untuk membangun *bidirectional* pada GRU.

C. Fungsi Aktivasi

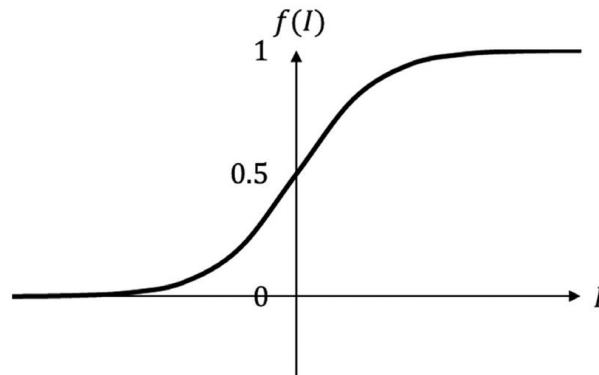
Fungsi aktivasi adalah fungsi yang digunakan dalam BiGRU untuk menghitung jumlah bobot dan bias, yang digunakan untuk memutuskan apakah *neuron* harus diaktifkan atau tidak. Fungsi aktivasi juga membantu proses normalisasi *output* pada setiap *neuron* untuk memiliki nilai dalam rentang antara 1 dan 0, atau antara -1 dan 1. Penelitian ini akan membahas fungsi aktivasi Sigmoid, Softmax, ReLU dan TanH.

- Sigmoid

Dalam beberapa literatur fungsi sigmoid disebut juga sebagai fungsi *logistic* atau *squashing*. Fungsi ini merupakan fungsi *non linear*. Bagian masukannya berupa bilangan *real* yang memiliki *output* dengan rentang nilai 0 sampai 1 (Enyinna Nwankpa *et al.*, 2018). Fungsi aktivasinya dapat didefinisikan oleh persamaan rumus (3.1).

$$f(x) = \sigma = \left(\frac{1}{1 + \exp^{-x}} \right) \quad (3.1)$$

Fungsi aktivasi sigmoid menghasilkan kurva dengan bentuk “S” yang dapat dilihat pada Gambar 3.4



Gambar 3. 4 Fungsi Aktivasi Sigmoid

(Sumber: Matsubara *et al.*, 2019)

- Softmax

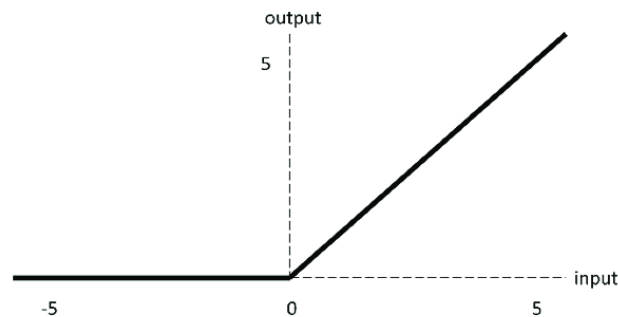
Fungsi aktivasi lainnya adalah Softmax. Fungsi ini digunakan dalam menghitung probabilitas dari setiap kelas target atas semua kelas target yang memungkinkan dan akan membantu untuk menentukan kelas target untuk input yang diberikan. Keluaran dari fungsi aktivasi ini menghasilkan nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1. Berbeda dengan sigmoid yang hanya digunakan untuk klasifikasi biner, softmax dapat digunakan untuk multi klasifikasi (Enyinna Nwankpa *et al.*, 2018). Berikut ini merupakan persamaan dari fungsi aktivasi softmax:

$$f(x_i) = \left(\frac{\exp(x_i)}{\sum_j \exp(x_j)} \right) \quad (3.2)$$

- *Rectified Linear Activation Function (ReLU)*

ReLU pertama kali dipublikasikan oleh (Nair & Hinton, 2010) dan menjadi fungsi aktivasi yang banyak digunakan pada model *Neural Network*. ReLU merupakan fungsi aktivasi terpopuler dan sangat banyak digunakan (Ramachandran *et al.*, 2017). ReLU menghasilkan performa dan generalisasi yang lebih baik dibandingkan fungsi sigmoid dan Tanh (Dahl *et al.*, 2013). Fungsi ReLU dapat diformulasikan sebagai berikut:

$$f(x) = \max(0, x) = \begin{cases} x_i, & \text{jika } x_i \geq 0 \\ 0, & \text{jika } x_i < 0 \end{cases} \quad (3.3)$$



Gambar 3. 5 Fungsi Aktivasi ReLU

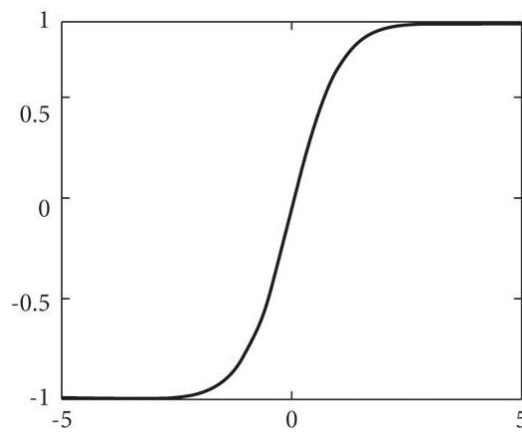
(Sumber: Sultan *et al.*, 2019)

- *Hyperbolic Tangent Function (TanH)*

Hyperbolic tangent function atau yang sering disebut TanH ini sangat mirip dengan fungsi aktivasi sigmoid dan bahkan memiliki bentuk S yang sama. Fungsi ini

mengambil nilai *real* sebagai nilai *input* dan *output* dalam rentang -1 hingga 1. Semakin besar input (semakin positif), semakin dekat nilai outputnya dengan 1, sedangkan semakin kecil input (semakin negatif), semakin dekat outputnya akan menjadi -1.0. Performa yang ditawarkan oleh fungsi aktivasi TanH hampir sama dengan performa klasifikasi yang dihasilkan oleh fungsi aktivasi ReLU (Wibawa, 2017). Fungsi aktivasi TanH dapat diformulasikan sebagai berikut:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (3.4)$$



Gambar 3. 6 Fungsi Aktivasi TanH

(Sumber: B. Wang & Liu, 2021)

D. Lapisan *Dropout*

Lapisan *dropout* merupakan teknik regularisasi untuk menurunkan *overfitting* pada *neural network* yang menghalangi ko-adaptasi kompleks pada data latih. Dalam

dropout layer dilakukan proses dropout yaitu menghilangkan koneksi secara acak pada unit neural network pada proses pelatihan (Baldi & Sadowski, 2013).

E. *Loss Function*

Loss function adalah fungsi untuk menghitung kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh suatu model dengan menghitung perbedaan keluaran aktual dan prediksi (Janocha & Czarnecki, 2017). Jadi *loss function* digunakan untuk mengukur jarak antara output model dan nilai target.

Pada penelitian ini menggunakan *loss function categorical cross entropy*. Semakin besar nilai *loss function* maka model tidak mampu menangkap pola dengan baik. Pada analisis *multi-class* klasifikasi biasanya digunakan fungsi *cross entropy* sebagai *loss function* dengan formula sebagai berikut:

$$CE = - \sum_i^c t_i \log (x_i) \quad (3.5)$$

Dimana t_i dan x_i adalah *groundtruth* dan skor model untuk setiap kelas i di dalam C .

Dan fungsi aktivasi (sigmoid / softmax) diterapkan pada skor sebelum perhitungan *CE Loss*

F. *Optimizer*

Optimizer adalah algoritma atau metode yang digunakan untuk mengubah atribut jaringan saraf seperti bobot dan learning rate untuk mengurangi kerugian atau *loss* dari proses training (Choi *et al.*, 2019). Adapun beberapa jenis *optimizer* yang

sering digunakan dalam percobaan penelitian diantaranya adalah Adam, SGD, Nadam, RMSprop, Adagrad, dan Adadelata.

Adam atau *Adaptive Moment Estimation* merupakan suatu *optimizer* yang mempertahankan *learning rates* setiap bobot dan rata-rata gradien secara eksponensial. Adam merupakan metode stokastik efisien yang hanya membutuhkan gradien orde pertama dengan kebutuhan memori yang sedikit (Kingma & Ba, 2014).

G. *Batch Size*

Batch size digunakan untuk menentukan jumlah observasi atau *sample* yang dilakukan sebelum melakukan perubahan bobot, yang akan disebarkan dalam sebuah *neural network* dan ditentukan relatif berdasarkan spesifikasi komputer (Wirtjes, 2019). Misalnya terdapat jumlah data sebanyak 10.000 data dan nilai *batch size* 10, maka akan membutuhkan 1.000 iterasi agar seluruh data menjadi tersebar dalam *neural network*. *Batch size* yang besar memerlukan memori yang lebih besar dan waktu pelatihan menjadi lebih cepat. *Batch size* yang besar juga membuat update parameter bobot lebih sedikit terjadi.

H. *Epoch*

Satuan iterasi yang sering digunakan adalah *epoch*. Satu *epoch* dapat diartikan sebagai banyaknya iterasi yang dilakukan untuk melatih *neural network* dalam satu kali putaran semua data *training*. Misalnya, memiliki data *training* sebanyak 200 baris data

dan menggunakan ukuran *batch* 10. Maka 1 epoch setara dengan 20 iterasi, yaitu banyaknya data sejumlah 200 dibagi dengan ukuran *batch* (Santosa & Umam, 2018).

Dalam *neural network*, proses pembelajaran yang berulang-ulang bertujuan untuk mencapai konvergensi nilai bobot. Karena nilai *epoch* yang sesuai tidak bisa diketahui, maka pada penelitian ini diujikan beberapa nilai *epoch* untuk mencapai nilai akurasi yang optimum.

3.3.6 Evaluasi Model

Untuk mengevaluasi model digunakan beberapa ukuran untuk mengevaluasi performa model. Berikut akan dijabarkan metode yang digunakan untuk evaluasi dan memahami performa model

A. *Confusion matrix*

Confusion matrix merupakan suatu metode evaluasi kinerja dari model klasifikasi dalam melakukan prediksi (Ting, 2017). Matriks ini berisikan informasi yang merepresentasikan label yang diprediksi oleh model, dan label yang sebenarnya. Dalam penelitian ini digunakan tabel confusion matrix sebagai berikut:

Tabel 3. 2 *Confusion Matrix*

<i>Confusion Matrix</i>		Prediksi		
		a	b	c
Aktual	a	T_{aa}	F_{ba}	F_{ca}
	b	F_{ab}	T_{bb}	F_{cb}
	c	F_{ac}	F_{bc}	T_{cc}

Matriks yang diatas sedikit berbeda dengan *confusion matrix* yang hanya menggunakan dua kelas. Sehingga terdapat perubahan pada penghitungan *True Positive*, *True Negative*, *False Positive*, dan *False Negative*. Dalam konteks kelas a pada tabel, perhitungan nilai tersebut dihitung sebagai berikut:

$$True\ Positive = T_{aa} \quad (3.6)$$

$$True\ Negative = T_{bb} + T_{cc} + F_{bc} + F_{cb} \quad (3.7)$$

$$False\ Positive = F_{ab} + F_{ac} \quad (3.8)$$

$$False\ Negative = F_{ba} + F_{ca} \quad (3.9)$$

Confusion matrix dapat digunakan untuk menghitung matrik matrik kinerja seperti *accuracy*, *precision*, *recall*, dan *F1-score* (Burkov, 2019).

- *Accuracy*

Accuracy merupakan salah satu metrik evaluasi yang dapat dihitung menggunakan hasil positif dan negatif dari prediksi sebagai berikut:

$$accuracy = \frac{TP}{TP + TN + FP + FN} \times 100\% \quad (3.10)$$

- *Precision*

Precision atau ketepatan merupakan rasio prediksi positif yang benar terhadap keseluruhan hasil yang diprediksi positif, yang dapat dihitung dengan persamaan:

$$precision = \frac{TP}{TP + FP} \quad (3.11)$$

- *Recall*

Recall merupakan merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Secara matematis, recall dapat diekspresikan sebagai berikut:

$$recall = \frac{TP}{TP + FN} \quad (3.12)$$

- *F1-Score*

F1-score merupakan ukuran akurasi suatu tes yang dihitung dari nilai precision dan recall yang sudah dijelaskan sebelumnya. F1-score merupakan rata-rata harmonis dari precision dan recall. Nilai tertinggi dari F1-score adalah 1.0 yang mengindikasikan

precision dan recall yang sempurna. Secara matematis, F1-Score dapat dihitung dengan rumus berikut:

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (3.13)$$

$$= \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

B. Cross Validation

Cross validation adalah metode statistik untuk mengevaluasi dan membandingkan algoritma pembelajaran dengan membagi data menjadi dua segmen, satu digunakan untuk mempelajari atau melatih suatu model dan yang lainnya digunakan untuk memvalidasi model (Refaeilzadeh *et al.*, 2009). Salah satu teknik dari *cross validation* adalah *k-fold cross validation*, yang mana memecah data menjadi *k* bagian set data dengan ukuran yang sama.

Penggunaan *k-fold cross validation* untuk menghilangkan bias pada data. Pada *k-fold cross validation* data dibagi kedalam *k* buah segmen yang memiliki rasio yang sama atau hampir sama. Data tersebut dilakukan pelatihan dan validasi sebanyak *k* kali dengan tiap perulangannya mengambil satu segmen berbeda sebagai data tes atau validasi dan *k* – 1 segmen sisanya sebagai data latih untuk kemudian diambil nilai rata-rata dari hasil setiap iterasi. (Refaeilzadeh *et al.*, 2009).

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pendahuluan

Pada bab ini akan membahas proses analisis sentimen pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia. Metode yang digunakan adalah *Bidirectional Gated Recurrent Unit* atau BiGRU.

4.2 Data Preprocessing

4.2.1 Case Folding

Tahapan pertama *data preprocessing* adalah *case folding* yaitu merubah kalimat teks yang mengandung huruf kapital menjadi huruf kecil. Pada Tabel 4.1 merupakan contoh hasil data mentah yang sudah dilakukan *preprocessing* pada tahapan *case folding*.

Tabel 4. 1 Contoh Hasil *Case Folding*

Data Mentah	Hasil <i>Case Folding</i>
Pemerintah daerah harus MENDUKUNG agar vaksinasi covid di Indonesia bisa cepat !!!	pemerintah daerah harus mendukung agar vaksinasi covid di indonesia bisa cepat !!!

4.2.2 Tokenizing

Tahap kedua adalah tokenizing yaitu proses memisahkan atau memotong tweet yang berupa frasa atau kalimat menjadi kata per kata atau yang dikenal sebagai token.

Pada Tabel 4.2 merupakan hasil data komentar yang sudah dilakukan *preprocessing* pada tahapan *tokenizing*.

Tabel 4. 2 Contoh Hasil *Tokenizing*

Hasil <i>Case Folding</i>	Hasil <i>Tokenizing</i>
pemerintah daerah harus mendukung agar vaksinasi covid di indonesia bisa cepat !!!	"pemerintah", "daerah", "harus", "mendukung", "agar", "vaksinasi", "covid", "di", "indonesia", "bisa", "cepat", "!!!"

4.2.3 *Filtering*

Tahap *filtering* adalah tahap mengambil kata-kata penting dari hasil *tokenizing*. Pada tahap ini dilakukan pembersihan dan penormalisasiaaan data teks. Pada Tabel 3.3 merupakan contoh hasil data komentar yang sudah dilakukan *preprocessing* pada tahapan *filtering*.

Tabel 4. 3 Contoh Hasil *Filtering*

Hasil <i>Tokenizing</i>	Hasil <i>Filtering</i>
"pemerintah", "daerah", "harus", "mendukung", " agar ", "vaksinasi", "covid", " di ", "indonesia", "bisa", "cepat", "!!!"	"pemerintah", "daerah", "harus", "mendukung", "vaksinasi", "covid", "indonesia", "bisa", "cepat"

4.2.4 *Stemming*

Tahap terakhir pada *data preprocessing* adalah *stemming*, Tahap ini diperlukan untuk mengubah kata-kata pada dataset menjadi kata. Pada Tabel 4.4 merupakan

contoh hasil data komentar yang sudah dilakukan *preprocessing* pada tahapan *stemming*.

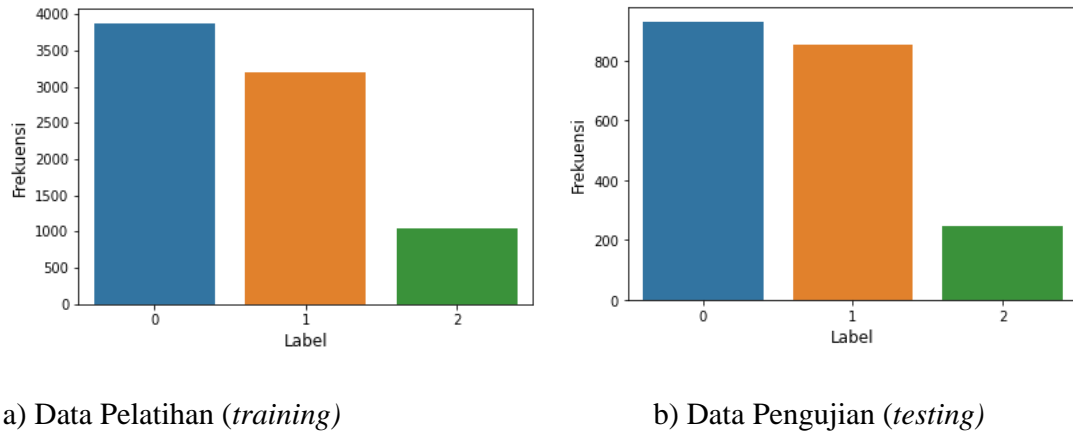
Tabel 4. 4 Contoh Hasil *Stemming*

Hasil <i>Filtering</i>	Hasil <i>Stemming</i>
"pemerintah", "daerah", "harus", "mendukung", "vaksinasi", "covid", "indonesia", "bisa", "cepat"	"pemerintah", "daerah", "harus", "dukung", "vaksinasi", "covid", "indonesia", "bisa", "cepat"

4.3 Pembagian Data

Langkah selanjutnya yang dilakukan dalam analisis sentimen menggunakan metode BiGRU adalah membagi data menjadi data pelatihan (*training*) dan data pengujian (*testing*). Pada penelitian ini, proporsi pembagian data yang akan digunakan adalah 80% untuk data pelatihan dan 20% untuk data pengujian.

Didapatkan data pelatihan sejumlah 8128 data dengan kelas negatif 3878 data, kelas netral 3204 data, dan kelas positif sebanyak 1046 data. Untuk data pengujian terdapat 2033 data dengan kelas negatif 932 data, kelas netral 853 data, kelas positif sejumlah 248 data. Berikut adalah grafik hasil pembagian data pelatihan dan pengujian dengan label 0 adalah kelas "Negatif", label 1 adalah kelas "Netral", dan label 2 adalah kelas "Positif".



Gambar 4. 1 Grafik Hasil Pembagian Data Pelatihan dan Pengujian

4.4 Word Embedding

Data komentar yang telah dibersihkan, kemudian diubah formatnya menjadi tensor. Pada arsitektur *Neural Network*, lapisan *embedding* adalah lapisan yang menerima masukan berupa tensor *integer* dua dimensi. Dalam penelitian ini digunakan 9070 kata unik dalam dataset, kemudian 9070 kata tersebut diubah menjadi urutan kata dalam bentuk *integer*.

Semua urutan yang akan menjadi input di lapisan *embedding* harus mempunyai panjang yang sama. Kalimat terpanjang dalam penelitian ini terdapat kata sebanyak 391 kata, untuk itu perlu dilakukan pemampatan *sequences* apabila sebuah kalimat kurang dari 391 maka akan ditambahkan nol, dan apabila *sequences* lebih dari 391 maka akan dipotong dengan batas maksimal 391. Berikut adalah contoh hasil dari *word sequences*

Tabel 4. 6 Contoh Hasil Urutan Kata

Kata	Urutan
'vaksin'	2
'tidak'	3
'saya'	4
'mau'	5
'covid'	6
'bisa'	7
'orang'	8
'iya'	9
'apa'	10

Data training akan dibuat menjadi sebuah matriks, dimana tiap kata akan dicari nilai hubungannya sesuai dengan ukuran dimensi yang digunakan. Berikut ini contoh matriks *word embedding* dengan dimensi 100.

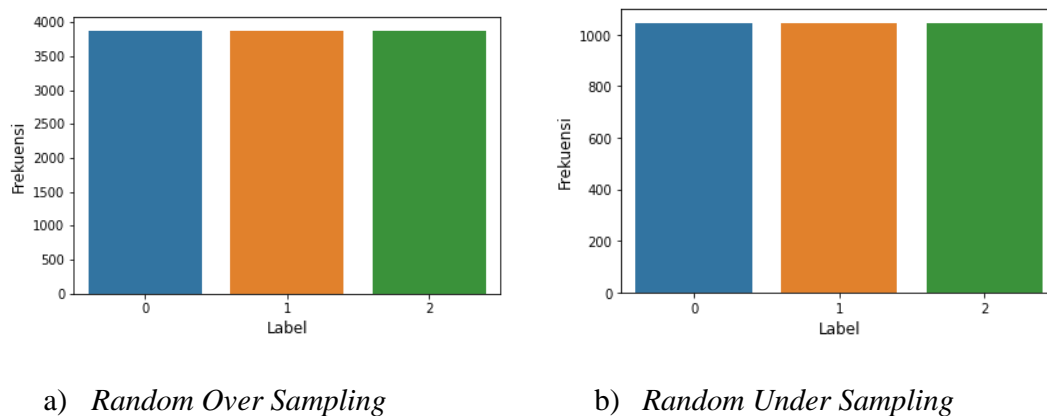
Tabel 4. 7 Contoh Hasil *Word Embedding*

Kata	Hasil <i>Word Embedding</i>
"vaksin"	[0.32555312, 2.20113969, 1.86161757, 0.11609357, 0.09981212, -1.59191442, -0.87172258, 0.98469561, 1.8132658 , 0.37077466, 1.38482702, -1.34788299, 1.058972 , -1.2072655 , 1.17485297, -0.19143499, -0.76071149, 2.04693961, -0.54945904, 1.34990978, 0.21096653, 1.32424688, 0.52040136, -0.94540018, -0.64748842, 0.14086676, 1.89417946, -0.57447088, -0.79195738, -0.75454324, 0.55132121, -1.09418249, 0.93774635, -0.38326579, 0.23500338, -0.09666333, 1.64390588, 0.23061845, 0.78423017, -0.19391261, -0.72707921, -1.40383136, -1.99640942, 1.43834972, -0.49876463, 1.81714451, -1.07643378, 0.78988308, 0.49289203, 0.64455307, 0.21508938, -0.33211794, -0.6580056 , 1.35378754, 1.00200725, -0.18672414, 1.42233741, 0.18829156, -2.21056175, -0.73322088, 0.80204332, -0.94072932, 1.11711812, 0.18568549, -0.24749182, -0.86281496, -0.61951083, -0.22027926, 0.15413691, 1.29747033, -1.712852 , -0.58447599, -1.23901963, -0.21180028, -0.79970992, -0.6917966 , 0.84771246, 0.12553646, -0.67599386, -0.99397844, -0.52750623, -0.15780908, -2.22391462, 0.68348807, -0.17894202, -0.02037326, -0.27537301, 0.46927413, -0.0476853 , 0.51402116, 0.91585982, 1.16396987, -0.96088129, 1.68524587, -1.83091724, 0.61670738, 1.25971484, -1.16232967, 0.13996142, -0.20932198]

Dengan matriks tersebut akan dijadikan bobot dalam *embedding layer* yang digunakan oleh model BiGRU untuk melakukan training terhadap data latih.

4.5 Penanganan *Imbalanced Classes*

Pada Gambar 4.1 terlihat data pelatihan dalam penelitian ini mengalami *imbalanced classes* dimana kelas negatif, netral, dan positif mempunyai proporsi yang berbeda. *Random Over Sampling* (ROS) dan *Random Under Sampling* (RUS) digunakan pada penelitian ini untuk menangani *imbalanced classes* pada dataset sentimen pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia. Berikut adalah hasil penanganan *imbalanced classes* menggunakan ROS dan RUS.



Gambar 4. 2 Grafik Hasil Penanganan *Imbalanced Classes* Menggunakan ROS dan RUS

Berdasarkan Gambar 4.2 dapat dilihat bahwa penerapan ROS dan RUS menangani proporsi kelas yang tidak seimbang pada dataset penelitian ini.

Menggunakan ROS didapatkan total jumlah data pelatihan menjadi 11634 data dengan kelas dengan kelas negatif 3878 data, kelas netral 3878 data, dan kelas positif sebanyak 3878 data. Ketika menggunakan RUS mengubah total jumlah data pelatihan menjadi 3139 data dengan kelas negatif 1046 data, kelas netral 1046 data, dan kelas positif sebanyak 1046 data.

4.6 Pemodelan dengan *Bidirectional Gated Recurrent Unit* (BiGRU)

Pada tahap ini berlangsung proses pelatihan model BiGRU untuk memperoleh model yang dapat mengklasifikasikan sentimen secara akurat.

4.6.1 Arsitektur Jaringan

Arsitektur jaringan dibentuk untuk menghasilkan akurasi yang optimal. Secara umum, model pelatihan dapat memiliki berbagai jumlah lapisan, tetapi dalam penelitian ini terdiri dari empat lapisan yaitu lapisan *Embedding*, lapisan BiGRU, satu lapisan *Dense*, dan *output layer*. input. Berikut ini rincian arsitektur jaringan BiGRU yang digunakan, yaitu:

1. Lapisan pertama yang dibuat adalah *Embedding Layer* yang menggunakan vektor dengan panjang dimensi 100 kata, untuk merepresentasikan masing-masing kata.
2. Lapisan BiGRU
3. *Dense layer* dengan dengan dropout dan fungsi aktivasi ReLU
4. Lapisan terakhir adalah *Output Layer* dengan 3 neuron dan fungsi aktivasi menggunakan softmax.

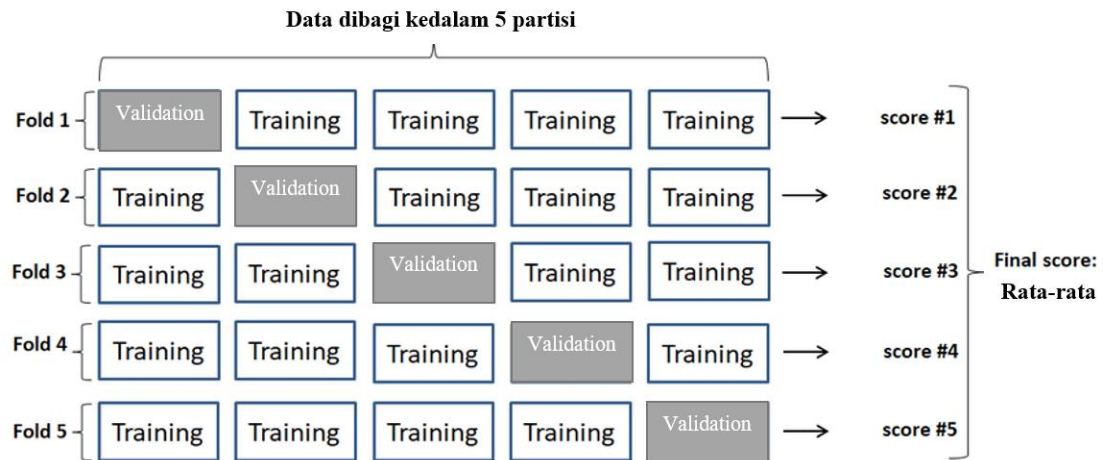
Dikarenakan penelitian merupakan *multiple classification* yaitu sentimen positif, netral, dan negatif, maka digunakan *loss function categorical cross entropy* dan dengan fungsi optimasi menggunakan ‘Adam’. Untuk menghindari masalah utama dalam proses pelatihan yaitu *overfitting*, pada penelitian ini menggunakan fungsi *Early Stopping* yang menghentikan proses *training* apabila akan terjadi *overfitting* pada model yang dibuat.

4.6.2 Proses Pelatihan dan Validasi Model

Subbab ini memaparkan hasil dari pelatihan dan validasi model, yaitu hasil *training* data dan hasil *validation* data dari klasifikasi dengan *cross validation*. *Cross validation* adalah metode statistik untuk mengevaluasi dan membandingkan algoritma pembelajaran, metode *cross validation* yang digunakan adalah *5-fold cross validation*. *5-fold cross validation* pada penelitian ini digunakan untuk mencari hyperparameter terbaik untuk pemodelan BiGRU yang optimal, seperti yang pernah dilakukan (Elgeldawi *et al.*, 2021).

Dengan melatih dan menguji model dengan, beberapa kali pada himpunan bagian yang berbeda dari data pelatihan yang sama, dapat representasi yang lebih akurat tentang seberapa baik model BiGRU tampil pada data yang belum pernah dilihat sebelumnya. *5-fold cross validation* menilai model setelah setiap iterasi dan menghitung rata-rata semua *score* untuk mendapatkan representasi yang lebih baik tentang kinerja model dibandingkan dengan hanya menggunakan satu set pelatihan dan validasi. *Score* yang digunakan dalam mencari *hyperparameter* terbaik pada penelitian

ini adalah *loss*, *accuracy*, dan *f1-score*. Metode *5-fold* pada penelitian ini dapat diilustrasikan pada Gambar 4.8



Gambar 4. 3 Ilustrasi 5-fold *Cross Validation*

Dalam proses pelatihan model BiGRU digunakan *epoch* maksimal sebesar 100 dan menggunakan beberapa skema pengujian yaitu:

1. Membandingkan penggunaan unit BiGRU, banyak *neuron* pada *dense layer*, *dropout*, dan *batch size*.
2. Menggunakan Word2Vec dan tanpa menggunakan Word2Vec.
3. Menerapkan penanganan dan tanpa penanganan *Imbalanced Classes*.

Pengujian pertama adalah pada *hyperparameter* unit BiGRU. Pada Tabel 4.8 terdapat hasil pengujian terhadap beberapa unit BiGRU.

Tabel 4. 8 Hasil Pengujian Unit BiGRU

Pengujian Ke-	Unit	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
1	4	0.844	0.606	0.591
2	10	0.849	0.604	0.588
3	20	0.830	0.621	0.610
4	43	0.835	0.616	0.601
5	130	0.835	0.618	0.605
6	391	0.840	0.616	0.602

Berdasarkan hasil pengujian unit BiGRU, ukuran unit 20 yang memberikan hasil yang lebih baik dibandingkan ukuran unit yang lain yaitu dengan *loss* sebesar 0.830 dengan akurasi mencapai 0.621, *f1-score* sebesar 0.6105, sedangkan jika unitnya lebih kecil dari 20 atau lebih besar dari 20 memberikan nilai *loss* yang lebih besar. Selanjutnya adalah hasil dari pengujian banyak *neurons* pada *dense layer* yang dapat dilihat pada Tabel 4.9

Tabel 4. 9 Hasil Pengujian *Dense Layer Neuron*

Pengujian Ke-	<i>Neurons</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
1	5	0.852	0.610	0.585
2	10	0.831	0.615	0.598
3	15	0.826	0.613	0.601
4	20	0.837	0.618	0.605
5	30	0.830	0.621	0.610
6	60	0.832	0.614	0.603

Berdasarkan Tabel 4.9 dapat disimpulkan bahwa jumlah *neurons* yang paling optimal pada *dense layer* sebanyak 30 *neurons*. Didapatkan *loss* sebesar 0.830, akurasi 0.621, dan *f1-score* 0.610, hasil ini lebih baik jika banyak *neurons* lebih sedikit atau lebih besar dari 30. Pada penelitian ini juga dilakukan pengujian *hyperparameter dropout* dan *batch size*. Berikut adalah hasil dari pengujian *dropout* dan *batch size* pada pemodelan BiGRU.

Tabel 4. 10 Hasil Pengujian *Dropout*

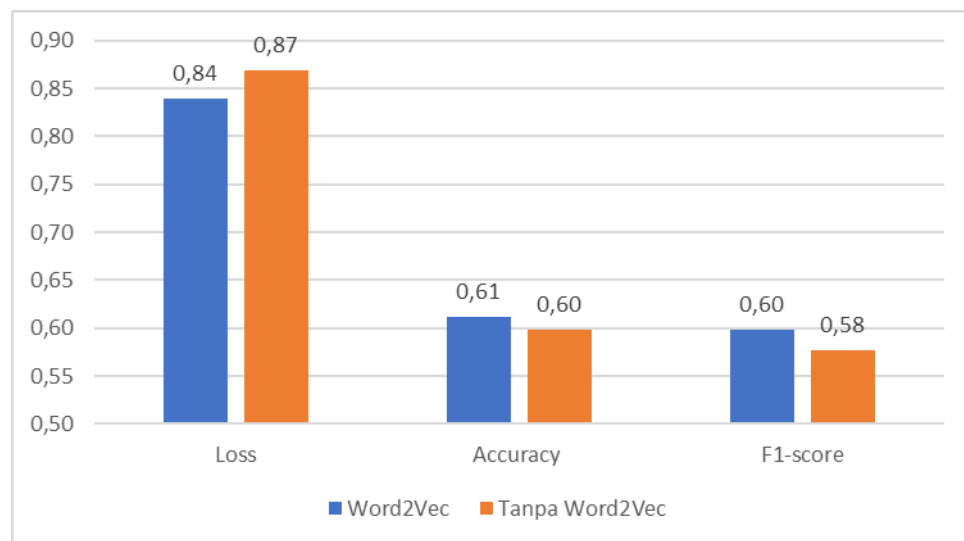
Pengujian Ke-	<i>Dropout</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
1	0.1	0.866	0.605	0.591
2	0.2	0.840	0.616	0.604
3	0.3	0.830	0.621	0.610
4	0.4	0.826	0.611	0.597
5	0.5	0.834	0.612	0.593
6	0.6	0.845	0.605	0.582

Tabel 4. 11 Hasil Pengujian *Batch Size*

Pengujian Ke-	<i>Batch Size</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
1	16	0,833	0,617	0,604
2	32	0,842	0,610	0,597
3	64	0,829	0,621	0,609
4	128	0,830	0,618	0,605
5	256	0,838	0,613	0,599
6	512	0,840	0,610	0,592

Berdasarkan pengujian *hyperparameter* unit BiGRU, banyak *neuron* pada *dense layer*, *dropout*, dan *batch size*. Didapatkan Unit BiGRU paling optimal sebanyak 20 unit, 30 *neuron* pada 1 *dense layer*, ukuran *dropout* 0.3, dan *batch size* sebesar 64.

Selanjutnya proses *training* dengan metode BiGRU dibandingkan dengan menggunakan Word2Vec dan tanpa menggunakan Word2Vec. Hasil *error training* (*loss*), akurasi, *f1-score* menggunakan metode BiGRU dengan menggunakan Word2Vec dan tanpa menggunakan Word2Vec dapat dilihat pada Gambar 4.4 dan Tabel 4.12 dibawah ini.



Gambar 4. 4 Grafik Perbandingan Hasil Pengujian Penggunaan Word2Vec

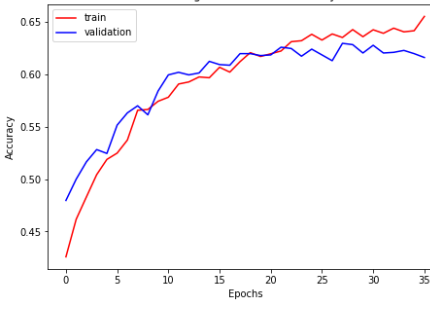
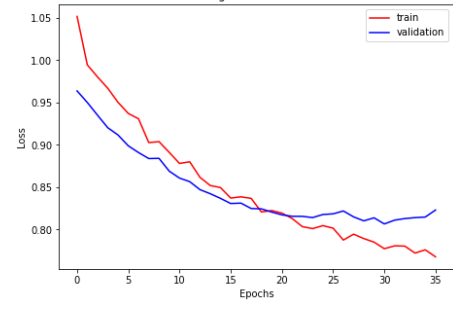
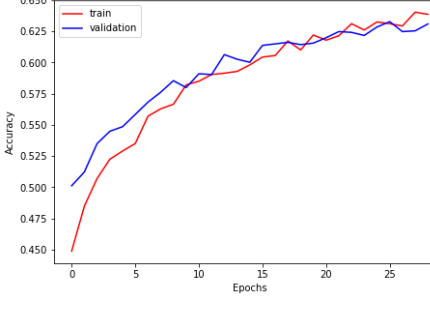
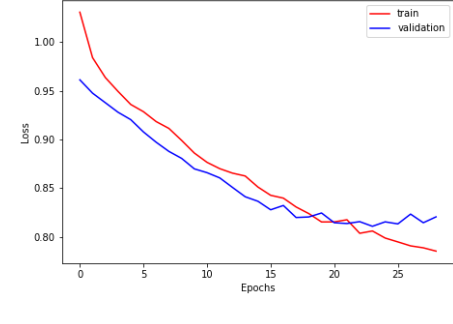
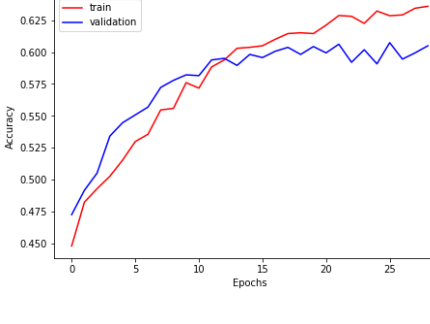
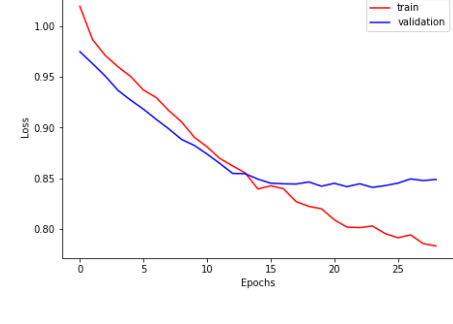
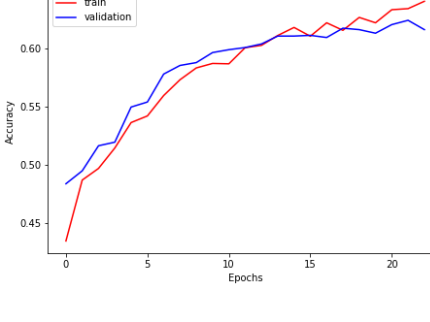
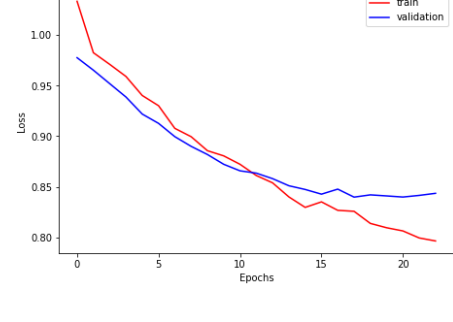
Tabel 4. 12 Hasil Pengujian Penggunaan Word2Vec

<i>Word Embedding</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
Word2Vec	0.839	0.611	0.598
Tanpa Word2Vec	0.869	0.598	0.577

Berdasarkan Tabel 4.12 pemodelan BiGRU menggunakan *Word Embedding* Word2Vec lebih baik dibandingkan tanpa Word2Vec, dengan nilai loss sebesar 0.839, akurasi mencapai 0.611, dan *f1-score* 0.598. Sedangkan ketika tidak menggunakan Word2Vec mendapat *loss* lebih besar yaitu 0.869, akurasi hanya 0.598, dan *f1-score* 0.577. Berdasarkan hasil pengujian pemodelan BiGRU dengan berbagai *hyperparameter* didapatkan pemodelan arsitektur BiGRU yang paling optimal yaitu 1 *embedding layer*, lapisan BiGRU dengan unit sebanyak 20 unit, 1 lapisan *dense layer* dengan 30 *neuron*, menggunakan *dropout* dengan ukuran 0.3, *batch size* sebesar 64, menggunakan *loss function categorical cross entropy*, fungsi optimasi menggunakan ‘Adam’, dan menggunakan *word embedding* Word2Vec. Berikut adalah proses dan hasil validasi model BiGRU dengan *hyperparameter* optimal, menggunakan *5-fold cross validation*.

Tabel 4. 13 Proses *5-fold Cross Validation*

Fold ke-	Grafik Accuracy dan Loss Proses <i>5-fold cross validation</i> .
1	<div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> <p>Training vs Validation Accuracy</p> </div> <div style="text-align: center;"> <p>Training vs Validation Loss</p> </div> </div>

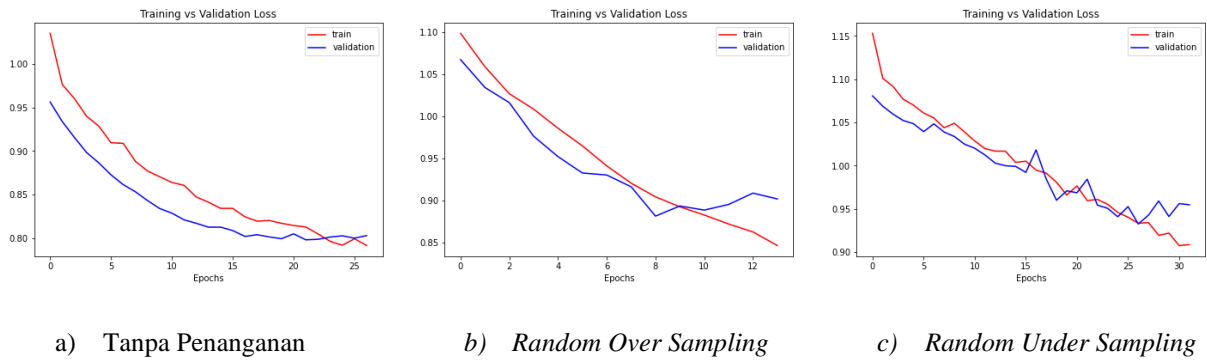
Fold ke-	Grafik Accuracy dan Loss Proses 5-fold cross validation.	
2	 <p>Training vs Validation Accuracy for Fold 2. The x-axis represents Epochs (0 to 35), and the y-axis represents Accuracy (0.45 to 0.65). The training accuracy (red line) starts at approximately 0.43 and increases to about 0.64. The validation accuracy (blue line) starts at approximately 0.48 and increases to about 0.62.</p>	 <p>Training vs Validation Loss for Fold 2. The x-axis represents Epochs (0 to 35), and the y-axis represents Loss (0.80 to 1.05). The training loss (red line) starts at approximately 1.02 and decreases to about 0.78. The validation loss (blue line) starts at approximately 0.96 and decreases to about 0.82.</p>
3	 <p>Training vs Validation Accuracy for Fold 3. The x-axis represents Epochs (0 to 25), and the y-axis represents Accuracy (0.450 to 0.650). The training accuracy (red line) starts at approximately 0.45 and increases to about 0.63. The validation accuracy (blue line) starts at approximately 0.50 and increases to about 0.62.</p>	 <p>Training vs Validation Loss for Fold 3. The x-axis represents Epochs (0 to 25), and the y-axis represents Loss (0.80 to 1.00). The training loss (red line) starts at approximately 1.02 and decreases to about 0.78. The validation loss (blue line) starts at approximately 0.96 and decreases to about 0.82.</p>
4	 <p>Training vs Validation Accuracy for Fold 4. The x-axis represents Epochs (0 to 25), and the y-axis represents Accuracy (0.450 to 0.625). The training accuracy (red line) starts at approximately 0.45 and increases to about 0.63. The validation accuracy (blue line) starts at approximately 0.48 and increases to about 0.61.</p>	 <p>Training vs Validation Loss for Fold 4. The x-axis represents Epochs (0 to 25), and the y-axis represents Loss (0.80 to 1.00). The training loss (red line) starts at approximately 1.02 and decreases to about 0.78. The validation loss (blue line) starts at approximately 0.96 and decreases to about 0.82.</p>
5	 <p>Training vs Validation Accuracy for Fold 5. The x-axis represents Epochs (0 to 20), and the y-axis represents Accuracy (0.45 to 0.65). The training accuracy (red line) starts at approximately 0.43 and increases to about 0.64. The validation accuracy (blue line) starts at approximately 0.48 and increases to about 0.62.</p>	 <p>Training vs Validation Loss for Fold 5. The x-axis represents Epochs (0 to 20), and the y-axis represents Loss (0.80 to 1.00). The training loss (red line) starts at approximately 1.02 and decreases to about 0.78. The validation loss (blue line) starts at approximately 0.96 and decreases to about 0.82.</p>

Berdasarkan Tabel 4.13 didapatkan hasil *loss*, *accuracy*, *f1-score* dengan *epoch* paling *optimal* . Hasil *score 5-fold Cross Validation* dapat dilihat pada tabel 4.14.

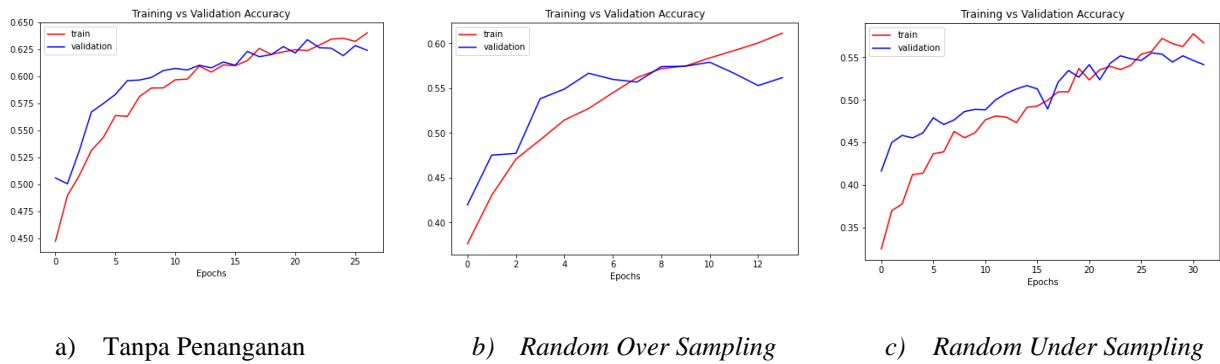
Tabel 4. 14 Hasil *5-fold Cross Validation*

<i>Fold ke-</i>	<i>Epoch</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
1	32	0.843	0.606	0.597
2	36	0.807	0.628	0.623
3	29	0.811	0.622	0.617
4	29	0.841	0.602	0.593
5	23	0.840	0.617	0.607
Rata-rata (<i>Score</i>)		0,828	0,615	0,607

Pengujian selanjutnya adalah membandingkan pemodelan BiGRU dengan penanganan *imbalanced classes* dan tanpa penanganan *imbalanced classes*. Pengujian ini dilakukan dengan melatih data *training* keseluruhan dan data *testing* sebagai validasi. Berikut adalah hasil pelatihan model BiGRU dengan tanpa penanganan *imbalanced classes*, menggunakan *Random Over Sampling*, dan menggunakan *Random Under Sampling*.



Gambar 4. 5 Hasil Loss Pelatihan dengan Penanganan *Imbalanced Classes* dan Tanpa Penanganan *Imbalanced Classes*



Gambar 4. 6 Hasil Akurasi Pelatihan dengan Penanganan *Imbalanced Classes* dan Tanpa Penanganan *Imbalanced Classes*

Dari hasil proses pelatihan tersebut didapatkan nilai *loss*, *accuracy*, dan *f1-score* terbaik, seperti pada Tabel 4.15.

Tabel 4. 15 Hasil Pengujian Penanganan *Imbalanced Classes*

<i>Imbalanced Classes</i>	<i>Loss</i>	<i>Accuracy</i>	<i>F1-score</i>
Tanpa Penanganan	0.798	0.634	0.623
ROS	0.881	0.574	0.599
RUS	0.933	0.555	0.487

Berdasarkan hasil pengujian pemodelan BiGRU dengan berbagai *hyperparameter*, perbandingan menggunakan Word2Vec dan tanpa menggunakan Word2Vec, perbandingan penanganan dan tanpa penanganan *Imbalanced Classes*, didapatkan pemodelan arsitektur BiGRU yang paling optimal yaitu 1 *embedding layer*, lapisan BiGRU dengan unit sebanyak 20 unit, 1 lapisan *dense layer* dengan 30 *neuron*, menggunakan *dropout* dengan ukuran 0.3, *batch size* sebesar 64, menggunakan *loss function categorical cross entropy*, fungsi optimasi menggunakan ‘Adam’, *epoch* sebanyak 27 iterasi, *word embedding* dengan Word2Vec, dan tanpa penanganan *imbalanced classes*. Setelah mendapatkan model BiGRU optimal, dilakukan *training* terhadap model dan menyimpan model tersebut agar dapat digunakan untuk dilakukan pengujian terhadap data *testing* atau pengujian.

4.7 Evaluasi Model

Berdasarkan rancangan penelitian data ini dibagi menjadi dua yaitu data *training* dan *testing*. Data *training* adalah yang digunakan untuk membangun model, sedangkan data *testing* atau data pengujian merupakan data yang digunakan untuk menguji model. Jumlah data pengujian model ini adalah 2033 data dengan kelas negatif 932 data, kelas netral 853 data, kelas positif sejumlah 248 data.

Tabel 4. 16 Hasil *Confusion Matrix*

<i>Confusion Matrix</i>		Prediksi		
		Negatif	Netral	Positif
Aktual	Negatif	716	207	9
	Netral	309	494	50
	Positif	58	111	79

Pada Tabel 4.16 dapat dilihat hasil pengujian model pada data pengujian. Pengujian ini menghasilkan data yang diprediksi benar dengan kelas negatif berjumlah 716 data, diprediksi benar dengan kelas netral sebanyak 494 data, sedangkan data dengan kelas positif yang diprediksi benar berjumlah 79 data. Dari Tabel 4.16 dapat diperoleh nilai *accuracy*, *precision*, *recall* dan *F1-score* dengan hasilnya dapat dilihat pada Tabel 4.17 berikut.

Tabel 4. 17 *Performance Metrics*

	Negatif	Netral	Positif
<i>Precision</i>	0.66	0.61	0.57
<i>Recall</i>	0.77	0.58	0.32
<i>F1-score</i>	0.71	0.59	0.41
<i>Accuracy</i>	0.63		

Tabel 4.17 menunjukan evaluasi klasifikasi sentimen pada penelitian ini. Pengujian pada data ini menghasilkan skor akurasi 63%. Untuk nilai *f1score* antara kelas negatif dengan netral dan positif memiliki perbedaan skor yang cukup jauh, kelas *f1-score* negatif sebesar 71 % dibandingkan *f1-score* kelas netral sebesar 59% dan *f1-score* kelas positif 41% yang berarti model dapat memprediksi dengan lebih akurat untuk kategori kalimat sentimen negatif jika dibandingkan dengan kategori kalimat sentimen netral dan positif. Berikut adalah contoh hasil prediksi sentimen dengan BiGRU.

Tabel 4. 18 Contoh Hasil Prediksi Sentimen dengan BiGRU

Komentar	Hasil Prediksi	Hasil Label
kita hanya di bodohi bukanya virus yg mati tapi masyarakat yg mati kelaparan anda hanya bisa berbicara tanpa tau keadaan kita pribadi	Negatif: 64% Netral: 27% Positif: 9%	Negatif
Go semangat vaksinasi. Semoga jadi jalan keluar. Tapi abis divaksin hati2 jgn tiba2 sok pede ga pake masker keluyuran ya	Negatif: 12% Netral: 42% Positif: 46%	Positif
vaksin sudah masuk ke daerah selain jabodetabek belum ya	Negatif: 19% Netral: 74% Positif: 7%	Netral

Untuk mengetahui *Bidirectional Gated Recurrent Unit* atau BiGRU adalah metode terbaik pada penelitian ini, peneliti melakukan perbandingan metode-metode klasifikasi lainnya seperti yang pernah dilakukan (Darapureddy *et al.*, 2019). Perbandingan metode ini menggunakan *5-fold cross validation* dan menggunakan *accuracy* dan *f1-score* sebagai evaluasi. Hasil perbandingan metode dengan *k-fold* dapat dilihat pada Tabel 4.19.

Tabel 4. 19 Hasil Perbandingan Metode dengan K-fold

Metode	<i>Accuracy</i>	<i>F1-score</i>
<i>Support Vector Machine</i> (SVM)	0.569	0.529
<i>Naïve Bayes Classifier</i>	0.538	0.520
<i>Random Forest Classifier</i>	0.571	0.534
<i>Multi Layer Perceptron</i> (MLP)	0.533	0,500
<i>Simple Recurrent Neural Network</i> (Simple RNN)	0.518	0,498
<i>Long Short-Term Memory</i> (LSTM)	0.589	0,578
<i>Bidirectional Long Short-Term Memory</i> (BiLSTM)	0.591	0,578
<i>Bidirectional Gated Recurrent Unit</i> (BiGRU)	0,615	0,607

Berdasarkan Tabel 4.19 dapat disimpulkan bahwa metode *Bidirectional Gated Recurrent Unit* adalah metode paling akurat pada penelitian ini dengan *score accuracy* sebesar 61.5% dan *score f1-score* 60,7% dibandingkan metode SVM, *Naïve Bayes Classifier*, *Random Forest Classifier*, MLP, *Simple RNN*, LSTM, dan BiLSTM

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan oleh penulis dalam melakukan analisis sentimen pengguna media sosial terhadap kebijakan pemerintah dalam program vaksinasi COVID-19 di Indonesia menggunakan BiGRU, dapat diambil simpulan sebagai berikut:

1. Didapatkan pemodelan arsitektur BiGRU yang paling optimal yaitu 1 *embedding layer*, lapisan BiGRU dengan unit sebanyak 20 unit, 1 lapisan *dense layer* dengan 30 *neuron*, menggunakan *dropout* dengan ukuran 0.3, *batch size* sebesar 64, menggunakan *loss function categorical cross entropy*, fungsi optimasi menggunakan ‘Adam’, *epoch* sebanyak 27 iterasi, *word embedding* dengan Word2Vec, dan tanpa penanganan *imbalanced classes*
2. Secara keseluruhan model BiGRU memperoleh nilai akurasi 63% dari data uji, rata-rata *precision* 61 %, dan rata-rata *recall* sebesar 56 %. Selain itu model mendapatkan *weighted average f1-score* 62%, dengan f1-score kelas negatif sebesar 71 %, kelas netral sebesar 59%, dan f1-score kelas positif 41% yang berarti model cenderung memprediksi dengan lebih akurat untuk kategori kalimat sentimen negatif jika dibandingkan dengan kategori kalimat sentimen netral dan positif

3. Metode BiGRU mendapatkan nilai tertinggi apabila dibandingkan dengan metode SVM, *Naïve Bayes Classifier*, *Random Forest Classifier*, MLP, *Simple RNN*, LSTM, dan BiLSTM

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, saran yang dapat diimplementasikan untuk pengembangan selanjutnya, diantaranya sebagai berikut:

1. Pada penelitian selanjutnya diharapkan dapat menambah jumlah *dataset* agar menghasilkan model yang lebih baik. Penambahan jumlah dataset dapat dilakukan dengan memperbanyak sumber media sosial.
2. Dalam penelitian selanjutnya diharapkan menambah dan berkolaborasi lebih banyak dengan ahli atau pakar khususnya spesialis di bidang bahasa, karena kontribusi pakar sangat penting pada penelitian analisis sentimen khususnya pada tahap pelabelan data.
3. Model BiGRU pada penelitian ini diharapkan dapat dilanjutkan ketahap *deployment* atau tahap pengembangan menjadi sebuah aplikasi agar dapat digunakan oleh pemerintah dalam mengevaluasi kebijakan dalam program vaksinasi COVID-19
4. Dalam penelitian selanjutnya diharapkan tetap dilakukan pengembangan pada pemodelan *Bidirectional Gated Recurrent Unit* atau BiGRU agar model BiGRU mendapatkan hasil performa yang lebih optimal.

DAFTAR PUSTAKA

- Abdelgwad, M. M., Soliman, H. A., Taloba, A. I., & Farghaly, M. F. (2021). *Arabic aspect based sentiment analysis using bidirectional GRU based models*.
- Abdulwahab, S. (2017). *Deep Learning Models for Paraphrases Identification*.
- Akkaya, B. (2019). Comparison of Multi-class Classification Algorithms on Early Diagnosis of Heart Diseases. *Y-BIS 2019 Conference: Recent Advances in Data Science and Business Analytics*.
https://www.academia.edu/41940316/Comparison_of_Multi_class_Classification_Algorithms_on_Early_Diagnosis_of_Heart_Diseases
- Ali, W., Yang, Y., Qiu, X., Ke, Y., & Wang, Y. (2021). Aspect-Level Sentiment Analysis Based on Bidirectional-GRU in SIoT. *IEEE Access*, 9, 69938–69950.
<https://doi.org/10.1109/ACCESS.2021.3078114>
- Baldi, P., & Sadowski, P. J. (2013). Understanding Dropout. *Advances in Neural Information Processing Systems*, 26.
- Biswas, S., Chadda, E., & Ahmad, F. (2015). Sentiment Analysis with Gated Recurrent Units. *Advances in Computer Science and Information Technology (ACSIT)*, 2(11), 59–63. <http://www.krishisanskriti.org/acsit.html>
- Bre, F., Gimenez, J. M., & Fachinotti, V. D. (2018). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 1429–1441. <https://doi.org/10.1016/J.ENBUILD.2017.11.045>
- Brownlee, J. (2017). *Long Short-Term Memory Networks With Python: Develop Sequence Prediction Model With Deep Learning*. Machine Learning Mastery.
- Burkov, A. (2019). *The Hundred-Page Machine Learning*.
- Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., & Dahl, G. E. (2019). *On Empirical Comparisons of Optimizers for Deep Learning*.
<https://arxiv.org/abs/1910.05446v3>
- Dahl, G. E., Sainath, T. N., & Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 8609–8613. <https://doi.org/10.1109/ICASSP.2013.6639346>

- Darapureddy, N., Karatapu, N., & Battula, T. K. (2019). Research of machine learning algorithms using k-fold cross validation. *International Journal of Engineering and Advanced Technology*, 8(6 Special issue), 215–218. <https://doi.org/10.35940/IJEAT.F1043.0886S19>
- Darujati, C., & Gumelar, A. B. (2012). *PEMANFAATAN TEKNIK SUPERVISED UNTUK KLASIFIKASI TEKS BAHASA INDONESIA*.
- DiPietro, R., & Hager, G. D. (2019). Deep learning: RNNs and LSTM. *Handbook of Medical Image Computing and Computer Assisted Intervention*, 503–519. <https://doi.org/10.1016/B978-0-12-816176-0.00026-0>
- Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis. *Informatics 2021*, Vol. 8, Page 79, 8(4), 79. <https://doi.org/10.3390/INFORMATICS8040079>
- Enyinna Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*.
- Fernández, A., García, S., Herrera, F., & Chawla, N. v. (2018). SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary. *Journal of Artificial Intelligence Research*, 61, 863–905. <https://doi.org/10.1613/JAIR.1.11192>
- Ghods, A., & Cook, D. J. (2019). *A Survey of Techniques All Classifiers Can Learn from Deep Networks: Models, Optimizations, and Regularization*. <https://arxiv.org/abs/1909.04791v2>
- Giarsyani, N. (2020). Komparasi Algoritma Machine Learning dan Deep Learning untuk Named Entity Recognition : Studi Kasus Data Kebencanaan. *Indonesian Journal of Applied Informatics*, 4(2), 138–144. <https://doi.org/10.20961/IJAI.V4I2.41317>
- Gruber, N., & Jockisch, A. (2020). Are GRU Cells More Specific and LSTM Cells More Sensitive in Motive Classification of Text? *Frontiers in Artificial Intelligence*, 0, 40. <https://doi.org/10.3389/FRAI.2020.00040>
- Härdle, W. K., Prastyo, D. D., & Hafner, C. M. (2014). Support Vector Machines with Evolutionary Model Selection for Default Prediction. *The Oxford Handbook of Applied Nonparametric and Semiparametric Econometrics and Statistics*. <https://doi.org/10.1093/OXFORDHB/9780199857944.013.011>

- Hayaty, M., Muthmainah, S., & Ghufran, S. M. (2021). Random and Synthetic Over-Sampling Approach to Resolve Data Imbalance in Classification. *International Journal of Artificial Intelligence Research*, 4(2), 86. <https://doi.org/10.29099/ijair.v4i2.152>
- Hermawan, L., & Ismiati, M. B. (2020). Pembelajaran Text Preprocessing berbasis Simulator Untuk Mata Kuliah Information Retrieval. *Jurnal Transformatika*, 17(2), 188–199. <https://doi.org/10.26623/TRANSFORMATIKA.V17I2.1705>
- Hilbe, J. M. (2009). *Logistic Regression Models*. Chapman & Hall/CRC Press.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/NECO.1997.9.8.1735>
- Huang, Wang, Li, Ren, Zhao, Hu, Zhang, Fan, Xu, Gu, Cheng, Yu, Xia, Wei, Wu, Xie, Yin, Li, Liu, ... Cao. (2020). Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. *Lancet (London, England)*, 395(10223), 497–506. [https://doi.org/10.1016/S0140-6736\(20\)30183-5](https://doi.org/10.1016/S0140-6736(20)30183-5)
- Indrawati, A., Subagyo, H., Sihombing, A., Wagiyah, W., & Afandi, S. (2020). ANALYZING THE IMPACT OF RESAMPLING METHOD FOR IMBALANCED DATA TEXT IN INDONESIAN SCIENTIFIC ARTICLES CATEGORIZATION. *BACA: JURNAL DOKUMENTASI DAN INFORMASI*, 41(2), 133–141. <https://jurnalbaca.pdii.lipi.go.id/baca/article/view/702>
- Janocha, K., & Czarnecki, W. M. (2017). On Loss Functions for Deep Neural Networks in Classification. *Schedae Informaticae*, 25, 49–59. <https://arxiv.org/abs/1702.05659v1>
- Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. <https://arxiv.org/abs/1412.6980v9>
- Kumar, A., Purohit, K., & Kumar, K. (2021). Stock Price Prediction Using Recurrent Neural Network and Long Short-Term Memory. *Lecture Notes in Networks and Systems*, 175, 153–160. https://doi.org/10.1007/978-3-030-67187-7_17
- Liu, B. (2012). *Sentiment Analysis and Opinion Mining*.
- Liu, C., Zhou, Q., Li, Y., Garner, L. v., Watkins, S. P., Carter, L. J., Smoot, J., Gregg, A. C., Daniels, A. D., Jerve, S., & Albaiu, D. (2020). Research and Development on Therapeutic Agents and Vaccines for COVID-19 and Related Human Coronavirus Diseases. *ACS Central Science*, 6(3), 315. <https://doi.org/10.1021/ACSCENTSCI.0C00272>

- Matsubara, N., Saito, K., Teramoto, A., & Fujita, H. (2019). *Generation of Pseudo Chest X-ray Images from Computed Tomographic Images by Nonlinear Transformation and Bone Enhancement*. https://www.researchgate.net/publication/336275807_Generation_of_Pseudo_Chest_X-ray_Images_from_Computed_Tomographic_Images_by_Nonlinear_Transformation_and_Bone_Enhancement
- Mustakim, Indah, R. N. G., Novita, R., Kharisma, O. B., Vebrianto, R., Sanjaya, S., Hasbullah, Andriani, T., Sari, W. P., Novita, Y., & Rahim, R. (2019). DBSCAN algorithm: twitter text clustering of trend topic pilkada pekanbaru. *Journal of Physics: Conference Series*, 1363(1), 012001. <https://doi.org/10.1088/1742-6596/1363/1/012001>
- Nair, V., & Hinton, G. E. (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines*.
- Nurdin, A., Anggo, B., Aji, S., Bustamin, A., & Abidin, Z. (2020). PERBANDINGAN KINERJA WORD EMBEDDING WORD2VEC, GLOVE, DAN FASTTEXT PADA KLASIFIKASI TEKS. *Jurnal Tekno Kompak*, 14(2), 74–79. <https://doi.org/10.33365/JTK.V14I2.732>
- Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques. *Undefined*, 79–86. <https://doi.org/10.3115/1118693.1118704>
- Pangaribuan, Y., & Sagala, M. (2017). *Menerapkan Jaringan Saraf Tiruan untuk Mengenali Pola Huruf Menggunakan Metode Perceptron*. 2548–1916.
- Putra, M. R. A., Djamal, E. C., & Ilyas, R. (2018). Brain Computer Interface untuk Menggerakkan Robot Menggunakan Recurrent Neural Network. *Prosiding Seminar Nasional Rekayasa Teknologi Informasi / SNARTISI*, 1. <https://ejurnal.lppmunsera.org/index.php/snartisi/article/view/829>
- Rachman, F. F., & Pramana, S. (2020). Analisis Sentimen Pro dan Kontra Masyarakat Indonesia tentang Vaksin COVID-19 pada Media Sosial Twitter. *Indonesian of Health Information Management Journal (INOHIM)*, 8(2), 100–109. <https://inohim.esaunggul.ac.id/index.php/INO/article/view/223>
- Rahutomo, F., Sandhya, D., Ikawati, E., Rohman, O. A., Informatika, T., Informasi, T., & Malang, P. N. (2019). EVALUASI FITUR WORD2VEC PADA SISTEM UJIAN ESAI ONLINE. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, 4(1), 36–45. <https://doi.org/10.29100/JIPI.V4I1.1098>

- Ramachandran, P., Zoph, B., & le Google Brain, Q. v. (2017). *SEARCHING FOR ACTIVATION FUNCTIONS*.
- Rana, R. (2016). *Gated Recurrent Unit (GRU) for Emotion Classification from Noisy Speech*. <https://arxiv.org/abs/1612.07778v1>
- Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-Validation. *Encyclopedia of Database Systems*, 532–538. https://doi.org/10.1007/978-0-387-39940-9_565
- Rengasamy, D., Jafari, M., Rothwell, B., Chen, X., & Figueredo, G. P. (2020). Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors (Switzerland)*, 20(3). <https://doi.org/10.3390/S20030723>
- Rozi, I. F., Pramono, S. H., & Dahlan, E. A. (2012). Implementasi Opinion Mining (Analisis Sentimen) untuk Ekstraksi Data Opini Publik pada Perguruan Tinggi. *Jurnal EECCIS*, 6. <https://jurnaleeccis.ub.ac.id/index.php/eccis/article/view/164>
- Sakunthala, S., Kiranmayi, R., & Mandadi, P. N. (2018). A review on artificial intelligence techniques in electrical drives: Neural networks, fuzzy logic, and genetic algorithm. *Proceedings of the 2017 International Conference On Smart Technology for Smart Nation, SmartTechCon 2017*, 11–16. <https://doi.org/10.1109/SMARTTECHCON.2017.8358335>
- Sari, E. D. N., Sari, E. D. N., & Irhamah, I. (2020). Analisis Sentimen Nasabah pada Layanan Perbankan Menggunakan Metode Regresi Logistik Biner, Naïve Bayes Classifier (NBC), dan Support Vector Machine (SVM). *Jurnal Sains Dan Seni ITS*, 8(2), D177–D184. <https://doi.org/10.12962/j23373520.v8i2.44565>
- Setiawan, E. I., Ferry, F., Santoso, J., Sumpeno, S., Fujisawa, K., & Purnomo, M. H. (2020). Bidirectional GRU for Targeted Aspect-Based Sentiment Analysis Based on Character-Enhanced Token-Embedding and Multi-Level Attention. *International Journal of Intelligent Engineering and Systems*, 13(5). <https://doi.org/10.22266/ijies2020.1031.35>
- Singhal, P., & Bhattacharyya, P. (2016). *Sentiment Analysis And Deep Learning A Survey*.
- Sultan, H. H., Salem, N. M., & Al-Atabany, W. (2019). Multi-Classification of Brain Tumor Images Using Deep Neural Network. *IEEE Access*, 7, 69215–69225. <https://doi.org/10.1109/ACCESS.2019.2919122>
- Tama, V. O. (2018). *Analisis Pelabelan dalam Klasifikasi Sentimen Ulasan Produk dengan Menggunakan Algoritma Multinomial Naïve Bayes*. Universitas Telkom.

<https://openlibrary.telkomuniversity.ac.id/home/catalog/id/146126/slug/analisis-pelabelan-dalam-klasifikasi-sentimen-ulasan-produk-dengan-menggunakan-algoritma-multinomial-na-ve-bayes.html>

- Tampil, Y., Komaliq, H., & Langi, Y. (2017). Analisis Regresi Logistik Untuk Menentukan Faktor-Faktor Yang Mempengaruhi Indeks Prestasi Kumulatif (IPK) Mahasiswa FMIPA Universitas Sam Ratulangi Manado. *D’CARTESIAN: Jurnal Matematika Dan Aplikasi*, 6(2), 56–62. <https://doi.org/10.35799/DC.6.2.2017.17023>
- Tan, P.-N., Steinbach, M., & Kumar, V. (2006). *Introduction to Data Mining*.
- Ting, K. M. (2017). Confusion Matrix. *Encyclopedia of Machine Learning and Data Mining*, 260–260. https://doi.org/10.1007/978-1-4899-7687-1_50
- Wang, B., & Liu, S. (2021). Prediction Method of College Students’ Psychological Pressure Based on Deep Neural Network. *Scientific Programming*, 2021. <https://doi.org/10.1155/2021/2943678>
- Wang, P., Qian, Y., Soong, F. K., He, L., & Zhao, H. (2015). A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding. *Undefined*.
- Wati, R. (2016). Penerapan Algoritma Genetika Untuk Seleksi Fitur Pada Analisis Sentimen Review Jasa Maskapai Penerbangan Menggunakan Naive Bayes. *EVOLUSI: Jurnal Sains Dan Manajemen*, 4(1). <https://ejournal.bsi.ac.id/ejurnal/index.php/evolusi/article/view/604>
- We Are Social, & Hootsuite. (2021, January). *Digital 2021: The Latest Insights Into The State of Digital*. <https://datareportal.com/reports/digital-2021-indonesia>
- WHO. (2020). *WHO Director-General’s opening remarks at the media briefing on COVID-19 - 11 March 2020*. <https://www.who.int/director-general/speeches/detail/who-director-general-s-opening-remarks-at-the-media-briefing-on-covid-19---11-march-2020>
- Wibawa, M. S. (2017). Pengaruh Fungsi Aktivasi, Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan. *Jurnal Sistem Dan Informatika (JSI)*, 11(2), 167–174. <https://jsi.stikom-bali.ac.id/index.php/jsi/article/view/129>
- Wijaya, A. P., & Santoso, H. A. (2016). Naive Bayes Classification pada Klasifikasi Dokumen Untuk Identifikasi Konten E-Government. *Journal of Applied Intelligent System*, 1(1), 48–55. <http://publikasi.dinus.ac.id/index.php/jais/article/view/1032>

- Yu, Q., Zhao, H., & Wang, Z. (2019). Attention-based bidirectional gated recurrent unit neural networks for sentiment analysis. *ACM International Conference Proceeding Series*, 116–119. <https://doi.org/10.1145/3357254.3357262>
- Yudistira, N., & Yudistira, N. (2021). Peran Big Data dan Deep Learning untuk Menyelesaikan Permasalahan Secara Komprehensif. *EXPERT: Jurnal Manajemen Sistem Informasi Dan Teknologi*, 11(2), 78–89. <https://doi.org/10.36448/expert.v11i2.2063>
- Zhang, X. F., Huang, H. Y., & Zhang, K. L. (2009). KNN text categorization algorithm based on semantic centre. *Proceedings - 2009 International Conference on Information Technology and Computer Science, ITCS 2009, 1*, 249–252. <https://doi.org/10.1109/ITCS.2009.57>
- Zhang, Y., Ren, W., Zhu, T., & Faith, E. (2019). MoSa: A modeling and sentiment analysis system for mobile application big data. *Symmetry*, 11(1). <https://doi.org/10.3390/SYM11010115>

LAMPIRAN

Lampiran 1 Source Code Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import warnings
from wordcloud import WordCloud
import string, re
from Sastrawi.StopWordRemover.StopWordRemoverFactory import S
topWordRemoverFactory
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
warnings.filterwarnings("ignore")
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
import re #regex library
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
nltk.download('punkt')
import ast
import tensorflow as tf
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from keras.utils.np_utils import to_categorical
import array
import numpy as np
import pandas as pd
import os
```

```

from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras import Sequential
from keras.models import Model
from keras.layers import Dense, Embedding, LSTM, Input, GRU,
Bidirectional, Dropout
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re
from keras.callbacks import EarlyStopping
from sklearn.metrics import confusion_matrix, f1_score, precision_score, recall_score
import keras.backend as K
from sklearn.metrics import accuracy_score
%matplotlib inline
import matplotlib.pyplot as plt
import itertools
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from gensim.corpora import WikiCorpus
from gensim.models.word2vec import Word2Vec
from urllib.request import urlopen
from collections import Counter
import gzip
import numpy as np
import word2vec
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter

```

Lampiran 2 Source Code Data Preprocessing

```
#Case Folding
df['komentar'] = df['komentar'].str.lower()
print('Case Folding Result : \n')
print(df['komentar'].head(5))
print('\n\n\n')
#Tokenizing
def remove_comments_special(text):
    # remove tab, new line, ans back slice
    text = text.replace('\t', " ").replace('\n', " ").replace(
        '\u', " ").replace('\\', " ").replace('.', " ")
    # remove non ASCII (emoticon, chinese word, .etc)
    text = text.encode('ascii', 'replace').decode('ascii')
    # remove mention, link, hashtag
    text = ' '.join(re.sub("([@#] [A-Za-z0-9
9]+)|(\w+:\/\/\S+)", " ", text).split())
    # remove incomplete URL
    return text.replace("http://", " ").replace("https://", "
")
df['komentar'] = df['komentar'].apply(remove_comments_special
)
def remove_number(text):
    return re.sub(r"\d+", " ", text)

df['komentar'] = df['komentar'].apply(remove_number)
def remove_punctuation(text):
    return text.translate(str.maketrans("", "", string.punctuat
ion))
df['komentar'] = df['komentar'].apply(remove_punctuation)

def remove_whitespace_LT(text):
    return text.strip()
df['komentar'] = df['komentar'].apply(remove_whitespace_LT)

def remove_whitespace_multiple(text):
    return re.sub('\s+', ' ', text)
df['komentar'] = df['komentar'].apply(remove_whitespace_multi
ple)

def remove_singl_char(text):
```

```

        return re.sub(r"\b[a-zA-Z]\b", " ", text)
df['komentar'] = df['komentar'].apply(remove_singl_char)

def word_tokenize_wrapper(text):
    return word_tokenize(text)
df['comments_tokens'] = df['komentar'].apply(word_tokenize_wr
apper)
print('Tokenizing Result : \n')
print(df['comments_tokens'].head())
print('\n\n\n')

#Filtering
normalizad_word = pd.read_csv("https://raw.githubusercontent.
com/meisaputri21/Indonesian-Twitter-Emotion-
Dataset/master/kamus_singkatan.csv", sep=";", header=None)
normalizad_word_dict = {}
for index, row in normalizad_word.iterrows():
    if row[0] not in normalizad_word_dict:
        normalizad_word_dict[row[0]] = row[1]
def normalized_term(document):
    return [normalizad_word_dict[term] if term in normalizad_
word_dict else term for term in document]
df['comments_normalized'] = df['comments_tokens'].apply(norma
lized_term)
normalizad_word2 = pd.read_csv("https://docs.google.com/sprea
dsheets/d/e/2PACX-1vRQS3tlUL5EcXyqbbYzFLHmHaqm2npjY-
DLyz0dzwMIcUVhfoVWKuhR52P9YCqBAyY9zCgT66JVutWA/pub?output=csv
",header=None)
normalizad_word_dict2 = {}
for index, row in normalizad_word2.iterrows():
    if row[0] not in normalizad_word_dict2:
        normalizad_word_dict2[row[0]] = row[1]
def normalized_term2(document):
    return [normalizad_word_dict2[term] if term in normalizad_
_word_dict2 else term for term in document]
df['comments_normalized'] = df['comments_normalized'].apply(n
ormalized_term2)
list_stopwords=(['yang', 'untuk', 'pada', 'ke', 'para', 'namu
n', 'menurut',
                'antara', 'seperti', 'jika', 'jika', 'sehingg
a', 'mungkin',

```

```

        'kembali', 'dan', 'ini', 'karena', 'oleh', 's
aat', 'sekitar', 'bagi',
        'serta', 'di', 'dari', 'sebagai', 'hal', 'ket
ika', 'adalah',
        'itu', 'dalam', 'bahwa', 'atau', 'dengan', 'a
kan', 'juga', 'kalau',
        'ada', 'terhadap', 'secara', 'agar', 'lain', 'j
adi', 'yang', 'sudah', 'sudah',
        'begitu', 'mengapa', 'kenapa', 'yaitu', 'yakn
i', 'daripada', 'itulah', 'lagi', 'maka',
        'tentang', 'demi', 'dimana', 'kemana', 'pula'
, 'sambil', 'sebelum', 'sesudah', 'supaya',
        'guna', 'kah', 'pun', 'sampai', 'sedangkan',
'selagi', 'sementara', 'tetapi', 'apakah',
        'sebab', 'selain', 'seolah', 'seraya', 'seter
usnya', 'dsb', 'dst', 'dll',
        'dahulu', 'dulunya', 'anu', 'demikian', 'tapi
', 'juga', 'mari', 'nantinya', 'melainkan', 'oh', 'ok',
        'sebetulnya', 'setiap', 'sesuatu', 'pasti', 's
aja', 'toh', 'ya', 'walau', 'apalagi', 'bagaimanapun', 'yg', '
dg', 'rt', 'dgn', 'ny', 'd', 'klo',
        'kalo', 'amp', 'biar', 'bikin', 'bilang', 'kr
n', 'nya', 'nih', 'sih', 'ah', 'ssh', 'om', 'ah', 'si', 'tau', 'tu
h', 'utk', 'ya', 'cek',
        'jd', 'aja', 't', 'nyg', 'hehe', 'pen', 'nan
', 'loh', 'rt',
        '&', 'yah', 'ni', 'ret', 'za', 'nak', 'haa'
, 'zaa', 'maa', 'lg', 'eh', 'hmm', 'kali'])
list_stopwords = set(list_stopwords)
def stopwords_removal(words):
    return [word for word in words if word not in list_stopwo
rds]
df['comments_tokens_sw'] = df['comments_normalized'].apply(st
opwords_removal)
# Spell Checker
def join_text_list(texts):
    texts = ast.literal_eval(texts)
    return ' '.join([text for text in texts])
df["comments_tokens_sw"] = df["comments_tokens_sw"].apply(joi
n_text_list)
from sympellpy import SymSpell

```

```

sym_spell = SymSpell()
corpus_path = "/content/drive/MyDrive/Final IFest Ngenkonst/w
iki-id-formatted.txt"
sym_spell.create_dictionary(corpus_path)
from symspellpy import SymSpell, Verbosity
input_term = "maksuddnya siaapa kam adlh seekr gjha"
suggestions = sym_spell.lookup_compound(input_term, max_edit_
distance=2)
for suggestion in suggestions:
    print(suggestion.term)
Comments = []
for index, row in df.iterrows():
    suggestions = sym_spell.lookup_compound(row["comments_tok
ens_sw"], max_edit_distance=2)
    Comments.append(suggestions[0].term)
df["Comments"] = Comments
df['Comments'] = df['Comments'].apply(word_tokenize_wrapper)
df.head()
#Stemming
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()
def stemmed_wrapper(term):
    return stemmer.stem(term)
term_dict = {}
for document in df['Comments']:
    for term in document:
        if term not in term_dict:
            term_dict[term] = ' '
print(len(term_dict))
print("-----")
for term in term_dict:
    term_dict[term] = stemmed_wrapper(term)
    print(term, ":" ,term_dict[term])
print(term_dict)
print("-----")
def get_stemmed_term(document):
    return [term_dict[term] for term in document]
df['comments_tokens_stemmed'] = df['Comments'].apply(get_stem
med_term)
print(df['comments_tokens_stemmed'])

```

Lampiran 3 Source Code Word Embedding

```
df = pd.read_csv('https://docs.google.com/spreadsheets/d/e/2P
ACX-
1vTbpuC4sfBTZWhW0QFK0mjjUV7wzthziWDMCM91ifTXQc5emKHFJVicgM4wI
2cQULwmq0y2Crxnblee/pub?gid=850839976&single=true&output=csv'
, sep=',')
df
df['Content'] = df['Content_Clean']
id_w2v_2 = Word2Vec.load('/content/drive/MyDrive/Dimas Ananda
, S.Stat/SKRIPSI/idwiki_word2vec_100.model')
print('Vocab length:', len(w2v_model.wv.key_to_index))
print('Vector size:', (w2v_model.wv.vector_size))
EXP_EMBED_DIM = w2v_model.wv.vector_size
df.Content = df.Content.astype(str)
texts = df.Content
tokenizer = Tokenizer(oov_token=('unknown'))
tokenizer.fit_on_texts(texts)
sequences_train = tokenizer.texts_to_sequences(df.Content)
sample_oov_token = 'aneh'
print(sample_oov_token in tokenizer.word_index)
print(tokenizer.word_index['unknown'])
tokenizer.texts_to_sequences([sample_oov_token])
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
EXP_VOCAB_SIZE = len(word_index)+1
EXP_MAX_FEATURES = EXP_VOCAB_SIZE

print('Word Index List :\n', word_index)
# Print sample of tokenized text
try:
    for i, tokenized in texts.sample(5).items():
        print(i, tokenized)
        print(sequences_train[i])
        print('\n')

    for i, tokenized in texts.sample(5).items():
        print(i, tokenized)
        print(sequences_test[i])
        print('\n')
except:
```



```

    print('-')
# Print sample of tokenized text
sample_tokenized = texts.sample(5)
for i, tokenized in sample_tokenized.items():
    print(tokenized)
    print(sequences_train[i])
    print('\n')
len(sequences_train)
embedding_matrix = np.zeros((EXP_VOCAB_SIZE, EXP_EMBED_DIM))

OOV = [] #out of vocabulary word

for word, i in word_index.items():
    if word in w2v_model.wv:
        embedding_matrix[i] = w2v_model.wv[word]
    else:
        OOV.append(word)
print('NUMBER OF OOV:', len(OOV), OOV)
print('SHAPE OF EMBEDDING MATRIX:', embedding_matrix.shape)
word_index['covid']
print(f"Embedding matrix for word '{list(word_index.keys())[42]}' : \n")
embedding_matrix[42]
# Splitting words in every tweets (similar with tokenizing)
Bigger_list = []
for i in df.Content:
    li = list(i.split(" "))
    Bigger_list.append(li)

print(Bigger_list[0:2])
def text_to_int(df, word_index, max_len):
    X = np.zeros((df.shape[0], max_len)) #initialising the n
d-array
    for i, tweet in enumerate(df):
        words = list(tweet.split(" "))
        j = 0
        for word in reversed(words): # reversed -
> right aligned
            if word in word_index.keys(): #if present in ou
r vocab
                X[i, max_len-1-j] = word_index[word]

```

```

        j += 1

    return X

#finding the longest word of tweet
max_len = 0
for list_ in Bigger_list:
    if len(list_) > max_len:
        max_len = len(list_)
# cari rata rata
print('Length of longest tweet is', max_len, 'words')
#converting train_data tweets to integer from word_index
X = text_to_int(df.Content, word_index, max_len)
print(X.shape)
print(df.Content[7], '\n mapped to \n', X[60])
print(df.Content[20], '\n mapped to \n', X[60])
df.shape[0]

```

Lampiran 4 Source Code Penanganan Imbalanced Classes

```
from collections import Counter
Counter(y_train.argmax(axis=1))
from imblearn.over_sampling import RandomOverSampler
y_train.argmax(axis=1)
oversample = RandomOverSampler(sampling_strategy='not majorit
y')
X_over, y_over = oversample.fit_resample(X_train, y_train.arg
max(axis=1))
counter = Counter(y_over)
print(counter)
sns.countplot(y_over, label='count')
plt.ylabel('Frekuensi', fontsize=12)
plt.xlabel('Label', fontsize=12)
plt.show()
from keras.utils.np_utils import to_categorical
y_over = to_categorical(y_over, num_classes=3)
y_over
from collections import Counter
Counter(y_train.argmax(axis=1))
y_train.argmax(axis=1)
from imblearn.under_sampling import RandomUnderSampler
# define oversampling strategy
undersample = RandomUnderSampler(sampling_strategy='not minor
ity')
# fit and apply the transform
X_under, y_under = undersample.fit_resample(X_train, y_train.
argmax(axis=1))
counter = Counter(y_under)
print(counter)
sns.countplot(y_under, label='count')
plt.ylabel('Frekuensi', fontsize=12)
plt.xlabel('Label', fontsize=12)
plt.show()
from keras.utils.np_utils import to_categorical
y_under = to_categorical(y_under, num_classes=3)
y_under
```

Lampiran 5 Source Code Model BiGRU

```
kfold = KFold(n_splits=5, shuffle=True, random_state=14)

fold_var = 1
scores = []
for train, test in kfold.split(X_train, y_train):
    model = Sequential()
    model.add(Embedding(len(word_index)+1, EMBEDDING_DIM, input_length = MAX_SEQUENCE_LENGTH, weights=[embedding_matrix], trainable=False))
    model.add(Bidirectional(GRU(20, dropout=0.3)))
    model.add(Dense(30, activation = 'relu'))
    model.add(Dropout(0.3))
    model.add(Dense(3, activation = 'softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics = ['accuracy'])
    es = EarlyStopping(restore_best_weights=True, monitor='val_loss', mode='min', verbose=1, patience=5)
    print(model.summary())
    history = model.fit(X_train[train], y_train[train], epochs=100, batch_size=64, verbose = 1, validation_data =(X_train[test], y_train[test]), callbacks=[es])
    plt.figure(figsize=(16,5))
    plt.subplot(1,2,1)
    plt.plot(history.history['accuracy'], color='red')
    plt.plot(history.history['val_accuracy'], color='blue')
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.title("Training vs Validation Accuracy")
    plt.legend(['train', 'validation'], loc='best')
    plt.subplot(1,2,2)
    plt.plot(history.history['loss'], color='red')
    plt.plot(history.history['val_loss'], color='blue')
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.title("Training vs Validation Loss")
    plt.legend(['train', 'validation'])
    plt.show()
    score = model.evaluate(X_train[test], y_train[test], verbose = 1)
    print('Hasil pengujian model ke '+str(fold_var))
```

```

        print(str(model.metrics_names[0]) + ": " + str(score[0]) +
              " | " + str(model.metrics_names[1]) + ": " + str(score[1]*100)
        )
        akurasi = score[1]
        scores.append(akurasi)
        y_pred = model.predict(X_train[test])
        y_pred = y_pred.argmax(axis=1)
        cm = confusion_matrix(y_train[test].argmax(axis=1), y_pre
d)
        print(cm)
        print(classification_report(y_train[test].argmax(axis=1),
y_pred))
        fold_var += 1
def Average(lst):
    return sum(lst) / len(lst)
print(Average(scores))

```

RIWAYAT HIDUP

Data Pribadi

Nama : Dimas Ananda
Tempat, Tanggal Lahir : Jakarta, 5 Januari 2001
Jenis Kelamin : Laki-laki
Agama : Islam
Alamat : Jl. Mangga Kweni Blok A4/55-56, Kota Bekasi
No. HP : 081395815105
Email : dimasananda0501@gmail.com



Riwayat Pendidikan

TK Islam Al-Husna	(2005 - 2006)
SDI Al-Husna	(2006 - 2012)
SMP Islam Al-Azhar 6 Jakapermai	(2012 - 2015)
SMA Negeri 2 Kota Bekasi	(2015 - 2018)
Statistika FMIPA Universitas Padjadjaran	(2018 - 2022)