DETEKSI *CYBERBULLYING* PADA MEDIA SOSIAL TWITTER MENGGUNAKAN LSTM, BILSTM, GRU, BIGRU, DAN SRU DENGAN *FASTTEXT*

SKRIPSI

ANDI NUR SALSABILA SYAMSU

H071191015

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

DETEKSI *CYBERBULLYING* PADA MEDIA SOSIAL TWITTER MENGGUNAKAN LSTM, BILSTM, GRU, BIGRU, DAN SRU DENGAN *FASTTEXT*

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin

ANDI NUR SALSABILA SYAMSU

H071191015

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

LEMBAR PERNYATAAN KEOTENTIKAN

Yang bertandatangan di bawah ini:

Nama : Andi Nur Salsabila Syamsu

NIM : H071191015

Program Studi : Sistem Informasi

Jenjang : S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

DETEKSI *CYBERBULLYING* PADA MEDIA SOSIAL TWITTER MENGGUNAKAN LSTM, BILSTM, GRU, BIGRU, DAN SRU DENGAN *FASTTEXT*

adalah karya tulisan saya sendiri dan bukan merupakan pengambilan alih tulisan orang lain, dan belum pernah dipublikasikan dalam bentuk apapun.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 3 Januari 2024

Andi Nur Salsabila Syamsu

NIM. H071191015

DETEKSI *CYBERBULLYING* PADA MEDIA SOSIAL TWITTER MENGGUNAKAN LSTM, BILSTM, GRU, BIGRU, DAN SRU DENGAN *FASTTEXT*

Disusun dan diajukan oleh

ANDI NUR SALSABILA SYAMSU

H071191015

Menyetujui,

Pembimbing Utama	Pembimbing Pertama
Dr. Eng. Armin Lawi, S.Si., M.Eng.	**A. Muh. Amil Siddik, S.Si., M.Si**
NIP. 19720423 199512 1001	NIP. 199110032019031015

Kepala Program Studi

Dr. Hendra, S.Si., M.Kom
NIP. 197601022002121001

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh:

Nama : Andi Nur Salsabila Syamsu

NIM : H071191015

Program Studi : Sistem Informasi

Judul Skripsi : Deteksi *Cyberbullying* pada Media Sosial Twitter menggunakan LSTM, BiLSTM, GRU, BiGRU, SRU dengan *FastText*

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi Departemen Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin.

DEWAN PENGUJI

		Tanda Tangan
Ketua	: Dr. Eng. Armin Lawi, S.Si., M.Eng.	(.....)
Sekretaris	: A. Muh. Amil Siddik, S.Si., M.Si	(.....)
Anggota	: Edy Saputra Rusdi, S.Si., M.Si	(.....)
Anggota	: Rozalina Amran, S.T., M.Eng	(.....)

Ditetapkan di : Makassar

Tanggal : 3 Januari 2024

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan karunianya. Sholawat dan salam senantiasa tercurahkan kepada baginda Nabi Muhammad SAW, sebagai Nabi yang telah menjadi suri tauladan bagi seluruh umatnya, sehingga penulis dapat menyelesaikan skripsi ini. Skripsi ini disusun untuk memenuhi salah satu syarat dalam memperoleh gelar Sarjana Komputer pada Program Studi Sistem Informasi, FMIPA, Universitas Hasanuddin.

Penulis menyadari bahwa skripsi ini dapat terwujud berkat bantuan arahan, bimbingan, dan doa dari berbagai pihak. Oleh karena itu, pertama-tama, penulis ingin menyampaikan rasa terima kasih kepada kedua orang tua penulis, ayahanda **Andi Syamsu** dan ibunda **Murniati**, dengan penuh kesabaran membimbing dan mendidik penulis, serta senantiasa mendo'akan sehingga penulis bisa mencapai pencapaian ini dan menyelesaikan pendidikan di perguruan tinggi hingga mendapat gelar yang insya Allah dapat bermanfaat bagi penulis kedepannya. Terima Kasih kepada kakak saya **Jihan** dan adik saya **Rehan** yang telah senantiasa membantu dan mendukung penulis dalam menyelesaikan skripsi ini.

Penulis juga ingin menyampaikan terima kasih kepada semua pihak yang telah menemani perjalanan penulis dari awal hingga akhir masa perkuliahan ini. Dengan tulus hati, penulis ingin menyatakan terima kasih kepada:

1. Bapak **Prof. Dr. Ir. Jamaluddin Jompa, M. Sc.** selaku Rektor Universitas Hasanuddin beserta seluruh jajarannya, serta Bapak **Dr. Eng. Amiruddin** selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin beserta jajarannya.
2. Bapak **Prof. Dr. Nurdin, S. Si., M. Si.** selaku Ketua Departemen Matematika dan Ilmu Pengetahuan Alam Universitas Hasanuddin, Bapak **Dr. Hendra, S.Si., M. Kom.**, selaku Ketua Program Studi Sistem Informasi, serta Bapak dan Ibu **Dosen Departemen Matematika** terutama kepada Bapak dan Ibu **Dosen Program Studi Sistem Informasi** yang telah memberikan banyak ilmu dan pengetahuan kepada penulis selama menempuh pendidikan di program studi sistem informasi, serta para **Staf**

Departemen Matematika yang telah membantu penulis dalam hal administrasi.

3. Bapak **Dr. Eng. Armin Lawi, S. Si., M. Eng.**, selaku dosen pembimbing utama yang telah meluangkan waktunya sehingga bisa membimbing dan mengarahkan dalam menyelesaikan skripsi penulis.
4. Bapak **A. Muh. Amil Siddik, S. Si., M. Si.**, selaku dosen pembimbing pertama skripsi dan dosen pembimbing akademik yang telah meluangkan waktunya dalam membimbing dan memberikan arahan serta pengetahuan selama penyusunan skripsi dan selama perkuliahan.
5. Bapak **Edy Saputra Rusdi, S. Si., M. Si.**, selaku dosen penguji pertama dan Ibu **Rosalina Amran, S. T., M. Eng.**, selaku dosen penguji kedua, yang telah memberikan saran dan kritik yang membangun dalam penulisan skripsi penulis lebih baik.
6. Sahabat Girls Talk yaitu **Rahmi, Sakinah Yunus, Eka Fitri Ramadani, Alika Oktaviani, Anugrah Lestari** yang selalu dan senantiasa membersamai, memberi dukungan, saling membantu dari awal masa studi hingga selesainya penulisan skripsi penulis.
7. Sahabat anggur yaitu **Mutiara** dan **Dila** yang telah membersamai dan senantiasa membantu penulis dari sejak SMA hingga berlanjut menempuh pendidikan di Fakultas yang sama.
8. Sahabat penulis yang senantiasa membersamai penulis selama di kampus, baik di perkuliahan maupun di himpunan yaitu **Indira Septianita Larasati**, **A. Muh. Risqullah, Meiliana Nurul Rahmah, Zidan Nadif Ramadhan**, **Ayu Pratiwi Putri B. L.**
9. **Seluruh teman-teman POL19ON, TPS ME 21-23, Posko 1 PS Maros** yang telah membersamai dan memberikan momen-momen yang sangat berharga kepada penulis selama masa studi sarjana.
10. **Seluruh teman-teman Program Studi Ilmu Komputer 2019** (telah berubah nama menjadi program Studi Sistem Informasi) yang telah membantu penulis selama perkuliahan hingga akhir menyelesaikan skripsi ini.

11. Terakhir, terima kasih pada diriku sendiri untuk ketekunan, ketabahan, dan usaha keras yang telah diberikan. Tanpa dedikasi dan kerja kerasmu, pencapaian ini tidak akan mungkin terjadi. *Keep going*!

Semoga Allah SWT. membalas semua kebaikan yang telah bapak, ibu, dan saudara berikan kepada penulis dengan kebaikan yang lebih besar disertai dengan curahan rahmat dan kasih sayang-Nya. Penulis menyadari bahwa skripsi ini masih belum mencapai kesempurnaan, baik dari segi materi, penulisan, maupun penyajian, karena adanya keterbatasan dan kemampuan penulis. Oleh karena itu, penulis dengan rendah hati mengharapkan saran dan kritik yang konstruktif guna meningkatkan kualitas skripsi ini. Penulis berharap agar skripsi ini dapat memberikan manfaat, tidak hanya bagi penulis sendiri, tetapi juga bagi pembaca umum, serta dapat memberikan kontribusi positif bagi perkembangan ilmu pengetahuan dan dunia pada umumnya. Semoga hasil penelitian ini dapat menjadi suatu sumbangan yang berarti dalam memajukan bidang ini.

Makassar, 4 Desember 2023

Andi Nur Salsabila Syamsu

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Hasanuddin, saya yang bertanda tangan di bawah ini:

Nama : Andi Nur Salsabila Syamsu
NIM : H071191015
Program Studi : Sistem Informasi
Departemen : Matematika
Fakultas : Matematika dan Ilmu Pengetahuan Alam
Jenis Karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Hasanuddin **Hak Bebas Royalti Noneksklusif (*Non-exsclusive Royalti-Free Right*) atas karya ilmiah berjudul:**

Deteksi *Cyberbullying* pada Media Sosial Twitter menggunakan LSTM, BiLSTM, GRU, BiGRU, dan SRU dengan *FastText*

Beserta perangkat yang ada (jika diperlukan). Terkait dengan hal di atas, maka pihak universitas berhak menyimpan, mengalih-media/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.
Dibuat di Makassar. Pada Tanggal 3 Januari 2024
Yang menyatakan

(Andi Nur Salsabila Syamsu)

ABSTRAK

Di era digital saat ini, media sosial menjadi *platform* utama untuk berinteraksi dan berkomunikasi. Kemajuan media sosial yang terjadi dengan cepat, disertai juga dampak negatif, salah satunya peningkatan kasus *cyberbullying*. Fenomena ini merupakan isu serius yang berpotensi merugikan individu dan masyarakat secara luas. Penelitian ini mengembangkan sistem deteksi *cyberbullying* menggunakan model *Recurrent Neural Network* (RNN) seperti *Long Short-Term Memory* (LSTM), *Bidirectional Long Short-Term Memory* (BiLSTM), *Gated Recurrent Unit* (GRU), *Bidirectional Gated Recurrent Unit* (BiGRU), dan *Simple Recurrent Unit* (SRU), yang dikombinasikan dengan *word embedding FastText*. Sumber data penelitian ini dari Twitter, sebanyak 2400 data dikumpulkan yang berdasarkan 24 kata kunci berpotensi *bullying* dan dilabeli secara manual. Pendekatan pelatihan yang digunakan adalah *Stratified* 5-*Fold Cross Validation* di mana model dilatih dan diuji pada berbagai kombinasi subset data. Hasil penelitian menunjukkan bahwa model SRU memiliki kinerja lebih baik dari model lain, dengan akurasi 73.44% dan f1-*score* 70.48%. Sedangkan model LSTM mencapai akurasi 73.35% dan f1-*score* 68.71%, model BiLSTM mencapai akurasi 72.48% dan f1-*score* 69.19%, model GRU mencapai akurasi 70.93% dan f1-*score* 67.58%, model BiGRU mencapai akurasi 72.81% dan f1-*score* 68.44%.

Kata Kunci: Klasifikasi, *Cyberbullying*, Twitter, RNN, *FastText*

ABSTRACT

In today's digital era, social media has become the main platform for interaction and communication. The rapid advancement of social media is followed by negative impacts, one of which is the increase in cyberbullying cases. This phenomenon is a serious issue that has the potential to harm individuals and society at large. This research develops a cyberbullying detection system using Recurrent Neural Network (RNN) models such as Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), Gated Recurrent Unit (GRU), Bidirectional Gated Recurrent Unit (BiGRU), and Simple Recurrent Unit (SRU), combined with FastText word embeddings. The data source of this research is from Twitter, where 2400 data were collected based on 24 potential bullying keywords and labeled manually. The training approach used is Stratified 5-Fold Cross Validation, where the model is trained and tested on various combinations of data subsets. The research results showed that the SRU model performed better than others, with 73.44% accuracy and 70.48% f1-score. While, the LSTM model achieves 73.35% accuracy and 68.71% f1-score, the BiLSTM model achieves 72.48% accuracy and 69.19% f1-score, the GRU model achieves 70.93% accuracy and 67.58% f1-score, the BiGRU model achieves 72.81% accuracy and 68.44% f1-score.

Keywords: Classification, Cyberbullying, Twitter, RNN, FastText

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR RUMUS

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penetrasi pengguna internet di Indonesia terus meningkat setiap tahunnya. Berdasarkan hasil survei Asosiasi Penyelenggara Jasa Internet Indonesia (APJII), pengguna internet di Indonesia pada tahun 2023 telah mencapai 215 juta jiwa atau 78,19% dari total populasi di Indonesia sebesar 275,77 juta jiwa. Dibanding tahun lalu mengalami kenaikan 1,17%. Saat ini, masyarakat menggunakan internet untuk melakukan kegiatan sehari-hari seperti menggunakan aplikasi konferensi video untuk sekolah maupun bekerja, media sosial untuk berkomunikasi dan sebagainya. Menurut survei yang sama menyatakan alasan paling umum menggunakan internet adalah untuk dapat mengakses media sosial seperti Facebook, Whatsapp, Telegram, Line, Twitter, Instagram, Youtube (APJII, 2023). Pengguna sosial media aktif di Indonesia di awal tahun 2023 tercatat mencapai 167 juta jiwa atau 60,4% dari total populasi (Datareportal, 2023).

Twitter merupakan salah satu media sosial yang sering digunakan oleh masyarakat Indonesia. Jumlah pengguna aktif Twitter di Indonesia mencapai 14,8 juta pada April 2023. Angka tersebut menempatkan Indonesia berada di peringkat keenam negara pengguna Twitter terbesar di dunia (Datareportal, 2023). Hal tersebut mengindikasikan bahwa Indonesia memiliki potensi pertumbuhan pengguna media sosial sebagai media komunikasi untuk berbagi informasi. Media sosial seperti Twitter memberikan beberapa opsi bagi orang untuk membangun jaringan mereka, memungkinkan mereka untuk mengobrol atau berinteraksi dengan bebas. Semua orang dapat mengekspresikan ide-ide mereka secara spontan untuk memenuhi kebutuhan mereka akan eksistensi, aktualisasi, dan sosialisasi yang ditransfer melalui kata-kata, gambar dan video. Akan tetapi, seiring berjalannya waktu, kenyamanan eksistensi, aktualisasi dan sosialisasi untuk membangun informasi dan komunikasi telah disalahgunakan oleh beberapa oknum. Mereka menggunakan media sosial untuk mengirimkan kata-kata, gambar, atau video yang menakuti, menghina dan mengintimidasi seseorang. Tindakan tersebut biasanya disebut sebagai *cyberbullying* (Margono dkk, 2014).

Pelaku *cyberbullying* menyakiti perasaan seseorang melalui media sosial dengan menyebarkan kebencian kepada orang-orang lain. Hal ini seringkali tidak dianggap serius karena orang-orang menganggap kejadian tersebut sebagai sebuah lelucon. *Cyberbullying* termasuk jenis pelanggaran hak asasi manusia karena melakukan tindakan penghinaan, ujaran kebencian dan permusuhan, serta pengancaman terhadap seseorang yang dilakukan secara *online* melalui *platform* media sosial (Hafidz, 2021). Tindakan *cyberbullying* mempunyai dampak yang lebih buruk dibandingkan tindakan *bully* yang dilakukan secara langsung oleh pelaku di depan korban. Dampak dari *cyberbullying* dapat bertahan lama dan mempengaruhi seseorang baik secara mental, emosional, maupun fisik. Bahkan dalam kasus yang ekstrim, *cyberbullying* dapat menyebabkan seseorang mengakhiri nyawanya sendiri (UNICEF, 2020). Oleh karena itu, diperlukan adanya tindakan pencegahan atau deteksi agar tidak membahayakan pengguna media sosial.

Bentuk *cyberbullying* di media sosial Twitter dilakukan dengan menulis *tweet* yang mengandung kata-kata hinaan atau kata-kata kasar, bahkan kata-kata yang menjurus kepada penghinaan terhadap SARA. Margono dkk. (2014) menganalisis kata-kata *bullying* bahasa Indonesia pada twitter menggunakan *Association Rules* dan *FP-Growth*. Mereka menggunakan beberapa kata kasar untuk dataset bahasa Indonesia. Namun, kata-kata kasar di Indonesia bersifat dinamis dan tidak semua kata-kata kasar dapat dikategorikan sebagai *bullying*. Deteksi *cyberbullying* pada media sosial dapat menerapkan analisis *text mining* dengan klasifikasi teks. Klasifikasi teks merupakan proses membedakan teks yang telah ditentukan menjadi kelas tertentu atau beberapa kelas tertentu (Zhou & Yifan, 2020). Baru-baru ini, model *deep learning* telah mencapai hasil yang luar biasa di berbagai bidang *Natural Language Processing* (NLP) seperti klasifikasi teks. Penelitian yang dilakukan oleh Zulqarnain dkk (2020) membandingkan beberapa arsitektur dasar *deep learning* untuk klasifikasi teks yaitu *Deep Belief Neural* (DBN), *Convolutional Neural Network* (CNN), dan *Recurrent Neural Network* (RNN), diperoleh hasil terbaik ada pada arsitektur RNN.

Pendekatan untuk klasifikasi teks yang diusulkan adalah dengan metode *deep learning*, khususnya digunakan untuk memproses teks adalah arsitektur

Recurrent Neural Network (RNN). RNN merupakan salah satu arsitektur paling populer dalam *Natural Language Processing* (NLP) karena kemampuannya dalam memproses teks dengan panjang yang bervariabel. Struktur berulang dari RNN memungkinkannya untuk menggunakan representasi kata terdistribusi dengan cara mengubah setiap kata dalam teks menjadi vektor yang membentuk matriks. RNN dapat memahami konteks dan pola dalam teks yang lebih kompleks karena mempertimbangkan urutan kata dalam kalimat. Namun, sulit untuk melatih RNN untuk menangkap ketergantungan jarak jauh antar kata (*long-term dependencies*) karena gradien cenderung *vanish*/menghilang (sebagian besar waktu) atau *explode*/meledak (jarang, tetapi dengan efek yang serius) (Bengio dkk., 1994). Hal tersebut dapat mempengaruhi kinerja RNN dalam memproses teks yang sangat panjang atau rumit. Di sisi lain, pengembangan untuk mengatasi masalah tersebut telah diusulkan, seperti *Long Short-Term Memory* (LSTM), *Gated Recurrent Unit* (GRU) dan *Simple Recurrent Unit* (SRU).

LSTM diusulkan oleh Hochreiter dan Schmidhuber (1997) untuk secara khusus mengatasi masalah *long-term dependencies* dan masalah *exploding vanishing gradient*. LSTM memiliki sel memori terpisah di dalamnya yang diperbarui dan mengungkapkan isinya hanya ketika dianggap perlu (Liu dkk, 2016). Struktur LSTM merupakan sebuah rangkaian yang satu kesatuan utuh atau tidak dapat dipotong, dimana struktur dokumen teks yang apabila dipotong akan mengubah makna kalimatnya. Dengan mempertimbangkan LSTM hanya dapat mengekstraksi informasi semantik teks secara satu arah, *Bidirectional Long-Short Term Memory* (BiLSTM) juga digunakan dalam penelitian ini karena arsitekturnya terdiri dari dua lapisan LSTM yang prosesnya saling berkebalikan arah, sehingga model dapat memahami dan mengambil perspektif dari kata terdahulu dan kata terdepan, menyebabkan proses pembelajaran akan semakin dalam dan lebih memahami konteks pada kalimat tersebut (Fadli & Hidayatullah, 2020).

GRU diusulkan oleh Cho dkk (2014) untuk mengadaptasi setiap unit berulang untuk menangkap ketergantungan dari berbagai skala waktu secara fleksibel. Seperti halnya unit LSTM, GRU juga memiliki unit *gating* yang mengatur aliran informasi di dalamnya, tetapi tidak menggunakan sel memori terpisah

(Chung dkk, 2014). Ketika GRU menganalisis sebuah kata, ia hanya mempertimbangkan konteks linguistik dari arah maju (ke depan), sehingga sangat penting bagi GRU untuk memahami konteks dengan melihat ke belakang (jalur mundur) (Zulqarnain dkk, 2020). Oleh karena itu, *Bidirectional Gated Recurrent Unit* (BiGRU) juga digunakan dalam penelitian ini karena memiliki satu layer lebih banyak dari GRU yang berjalan secara *forward* dan *backward* melalui dua arah yaitu layer pertama *forward* yang melakukan ekstraksi kata depan sampai akhir dan *backward* dari kata belakang sampai awal.

SRU diusulkan oleh Lei dkk (2018) untuk mengatasi masalah ketergantungan jarak jauh dalam pemrosesan teks dengan menggunakan pendekatan yang lebih sederhana dan paralel. SRU memungkinkan pemrosesan data berurutan dengan panjang yang bervariasi dengan biaya komputasi yang lebih rendah dan kemampuan pemrosesan yang lebih cepat. SRU memiliki pendekatan yang lebih sederhana dalam menghitung *gating*, yang mengontrol aliran informasi dalam unit berulang dibandingkan dengan LSTM dan GRU (Lei dkk, 2018). Sehingga model SRU digunakan pada penelitian ini sebagai perbandingan dengan model LSTM dan GRU nantinya.

Penggunaan *word embedding* akan menjadi *input layer* pada model sebelum melakukan klasifikasi teks. Teknik *word embedding* yang sering digunakan yaitu *Word2vec*, *GloVe*, dan *FastText*. Penelitian yang dilakukan oleh Nurdin, dkk (2020) membandingkan teknik *word embedding* tersebut untuk klasifikasi teks, memperoleh kinerja terbaik saat menggunakan teknik *word embedding FastText*, karena dapat menangani masalah *out of vocabulary* yaitu kata yang tidak terdapat saat pelatihan (Nurdin dkk, 2020). Penelitian yang menggunakan teknik *word embedding FastText* dengan *deep learning* pernah dilakukan sebelumnya. Adinda dan Yaya (2022) dalam penelitiannya, membuktikan dengan penambahan *word embedding FastText* sebagai input pada model LSTM dan BiLSTM dapat menambah kinerja sebesar 89%, sedangkan tanpa *FastText* mengalami penurunan akurasi 1-2% (Shalehah & Triana, 2022).

Berdasarkan latar belakang diatas, penelitian ini mengklasifikasikan teks untuk mendeteksi *cyberbullying* pada media sosial Twitter menjadi dua kelas yaitu

cyberbullying dan bukan *cyberbullying*. Metode yang digunakan yaitu algoritma LSTM, BiLSTM, GRU, BiGRU dan SRU, serta melakukan penambahan *word embedding FastText*. Penelitian ini dilakukan untuk mengetahui dan membandingkan kinerja dari masing-masing algoritma serta pengaruh penambahan *word embedding FastText* pada klasifikasi teks untuk melakukan deteksi *cyberbullying* pada media sosial Twitter.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang, rumusan masalah pada penelitian ini sebagai berikut:

1. Bagaimana melakukan klasifikasi dalam mendeteksi *tweets* yang mengandung *cyberbullying* pada Twitter menggunakan model LSTM, BiLSTM, GRU, BiGRU, dan SRU dengan *FastText*?
2. Bagaimana menganalisis performa kinerja model LSTM, BiLSTM, GRU, BiGRU, SRU dengan *FastText* dalam melakukan klasifikasi untuk mendeteksi *tweets* yang mengandung *cyberbullying* pada media sosial Twitter?
3. Bagaimana mendeploy aplikasi *website* menggunakan *best* model diantara model-model yang telah dibuat dalam melakukan deteksi *tweets* yang mengandung *cyberbullying* pada media sosial Twitter?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini sebagai berikut:

1. Untuk melakukan klasifikasi dalam mendeteksi *cyberbullying* pada media sosial Twitter menggunakan model LSTM, BiLSTM, GRU, BiGRU, SRU dengan *FastText*.
2. Untuk menganalisis performa kinerja model dalam melakukan klasifikasi untuk mendeteksi *cyberbullying* pada media sosial Twitter berdasarkan tingkat *accuracy*, *recall*, *precision*, F1-*score*.
3. Untuk mendeploy aplikasi *website* terhadap hasil model terbaik yang telah dibuat dalam melakukan deteksi *tweets* yang mengandung *cyberbullying* pada Twitter.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini sebagai berikut:

1. Data yang digunakan berasal dari hasil *scraping* pada Twitter menggunakan *library snscrape* python.
2. Data yang digunakan berbahasa Indonesia.
3. Data diambil berdasarkan 24 kata kunci yang berpotensi *bully*.
4. Panjang maksimum urutan (*sequence*) untuk *input* adalah 52.
5. Algoritma yang digunakan LSTM, BiLSTM, GRU, BiGRU, dan SRU.
6. Model *word embedding* yang digunakan adalah model *pre-trained FastText*.
7. *Output* yang dihasilkan program yaitu nilai *accuracy*, *recall*, *precision*, *f1-score*, dan memprediksi *tweets* yang mengandung *cyberbullying* atau tidak.

1.5 Manfaat Penelitian

Penelitian ini menghasilkan model yang dapat digunakan untuk mendeteksi *tweets* yang mengandung *cyberbullying* pada media sosial Twitter, memberikan informasi tentang implementasi LSTM, BiLSTM, GRU, BiGRU, dan SRU dengan *FastText* dalam untuk mendeteksi *cyberbullying* pada sebuah *tweets* dan perbandingan antar modelnya, serta implementasi dari metode prediksi diharapkan dapat dijadikan sebagai pembanding dengan penelitian lain dan selanjutnya dalam hal deteksi *cyberbullying* berbahasa Indonesia.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Beberapa penelitian sebelumnya yang digunakan dalam melakukan penelitian ini. Penelitian yang dilakukan oleh Fadli dan Hidayatullah (2020) melakukan deteksi cuitan pada media sosial Twitter dengan memasukkan ke dalam kelas *cyberbullying* dan bukan *cyberbullying*. Deteksi tersebut menggunakan dua algoritma *deep learning* yaitu LSTM dan BiLSTM. Hasil yang didapat pada penelitian tersebut menunjukkan bahwa tingkat *accuracy, precision, recall* dan F1-*score* dari LSTM sebesar 93.77%, 91.59%, 92.45%, 92.02%, dan BiLSTM sebesar 95.24%, 92.29%, 93.40%, 93.84%.

Penelitian yang dilakukan oleh Shalehah (2022) membandingkan kinerja dari tiga varian dari RNN yaitu SimpleRNN, LSTM, dan BiLSTM dengan penambahan *FastText embedding* terhadap data ulasan PeduliLindungi. Hasil yang didapat pada penelitian tersebut menunjukkan bahwa akurasi terbaik didapat oleh model LSTM dan Bi-LSTM dengan penambahan *FastText Embeddings* sebesar 89%, sedangkan SimpleRNN dengan *FastText* sebesar 88%. Model yang diuji tanpa *FastText* mengalami penurunan akurasi sebesar 1-2%, dengan hasil akurasi SimpleRNN sebesar 87%, LSTM sebesar 87% dan BiLSTM sebesar 88%.

Penelitian yang dilakukan oleh Faturrohman dan Rosmala (2022) melakukan klasifikasi polaritas pada data sentimen dari sosial media Twitter menggunakan *Bidirectional* GRU (BiGRU) dengan *word embedding GloVe*. Hasil model terbaik yang didapatkan menggunakan parameter pada *learning rate* 0,001, *batch size* 16, dan *epoch* 50. Nilai *accuracy* terbaik yang didapatkan 97.7 % dengan nilai *precision* 97.8%, *recall* 97.5% dan nilai *loss* 1.3%. Model BiGRU yang dibangun sudah cukup optimal untuk melakukan klasifikasi sentimen dengan akurasi yang baik.

Penelitian yang dilakukan oleh Khasanah (2021) melakukan klasifikasi sentimen dengan menggunakan dua model *deep learning* dengan arsitektur sederhana dan menyelidiki bagaimana penggunaan *embedding FastText*

mempengaruhi kinerja model klasifikasi sentimen. Model pertama adalah BiGRU dengan lapisan tunggal dan menggunakan *FastText embedding*, dan model kedua adalah CNN dengan lapisan tunggal dan menggunakan *embedding FastText*. Kedua model tersebut memberikan hasil yang kompetitif. Penelitian ini menunjukkan bahwa penggunaan *embedding FastText* dapat meningkatkan kinerja model.

Penelitian yang dilakukan oleh Al-Dabet dan Tedmori (2019) melakukan penelitian tentang analysis sentiment bahasa Arab. Mereka menggunakan integrasi antara varian dari RNN yaitu *Simple Recurrent Unit* yang memiliki perhitungan berulang yang ringan, dan mekanisme *attention* yang berkonsentrasi pada bagian penting dari *input text*. Penelitian ini memperoleh akurasi mencapai 94,53% dengan waktu eksekusi yang lebih cepat.

2.2 *Cyberbullying*

Cyberbullying adalah pemanfaatan korespondensi elektronik untuk mengancam seseorang, biasanya dengan mengirimkan pesan yang bersifat mengintimidasi atau membahayakan. Di era media sosial dan jejaring online, penggunaan kata-kata menyinggung dan agresif meningkat secara signifikan. Komentar ini membangun budaya tidak hormat di dunia maya (Ahmed dkk, 2021). *Cyberbullying* biasanya berupa ujaran kebencian, perilaku agresif yang memiliki tujuan untuk menjatuhkan atau memojokan individu biasanya dilakukan oleh kelompok atau individu. Wujud dari *cyberbullying* di media sosial sangatlah beragam, dapat berupa kata-kata umpatan, gosip, cemooh, penghinaan, dan lain-lain. Perilaku *cyberbullying* juga tidak mengenal usia, siapa ataupun jabatan, namun umumnya pesohor lebih mudah mendapat perilaku *cyberbullying* (Rachmayanti & Candrasari, 2022).

Cyberbullying dapat terjadi dalam berbagai bentuk dan terjadi melalui begitu banyak tempat yang berbeda. Willard (Kowalski dkk, 2014) menciptakan taksonomi jenis *cyberbullying*, antara lain:

a. *Flaming*, yaitu mengirim pesan yang kasar, vulgar tentang seseorang ke grup *online* atau ke *cybervictim* melalui *email* atau pesan teks lainnya.
b. *Online harassment*, yaitu berulang kali mengirim pesan *offensive* melalui *email* atau teks lainnya kepada seseorang.

c. *Cyberstalking*, yaitu pelecehan *online* yang mencakup ancaman bahaya atau mem*bullying* dengan memberikan komentar menyakitkan.
d. *Denigration*, yaitu mengirim pernyataan berbahaya, tidak benar, atau pernyataan yang kejam tentang seseorang atau memposting materi *online* semacam itu.
e. *Masquerade*, yaitu berpura-pura menjadi orang lain dan mengirim atau memposting materi yang membuatnya *cybervictim* terlihat buruk.
f. *Outing*, yaitu mengirim atau memposting materi tentang seseorang yang berisi hal atau informasi sensitif, pribadi, atau informasi yang memalukan, termasuk meneruskan pesan atau gambar pribadi.
g. *Exclusion*, yaitu secara kejam mengucilkan, mengabaikan, dan menghapus seseorang dari grup *online*.

2.3 *Natural Language Processing* (NLP)

Natural Language Processing merupakan subbidang dari *Artificial Intelligence* (AI) untuk memproses, menganalisis, memahami, dan menghasilkan bahasa manusia. NLP termasuk subbidang AI karena pemrosesan bahasa dianggap sebagai bagian dari kecerdasan manusia. Penggunaan bahasa merupakan keterampilan paling menonjol yang membedakan manusia dengan hewan lainnya.

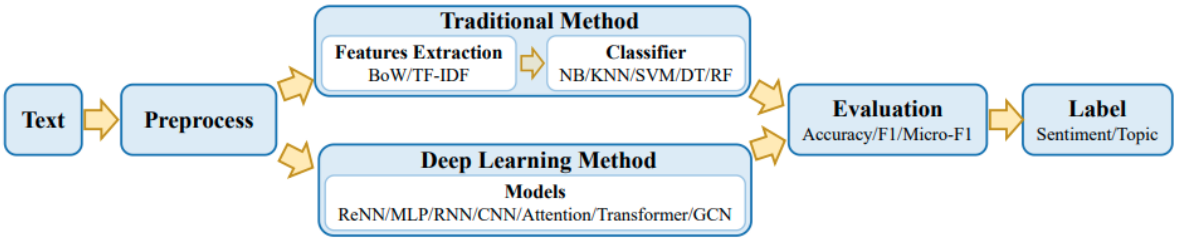
Dalam dekade terakhir, NLP telah menjadi bagian tak terpisahkan dari kehidupan sehari-hari kita. Terjemahan mesin otomatis telah menjadi hal yang umum di web dan media sosial. Klasifikasi teks menjaga kotak masuk email kita agar tidak kewalahan oleh spam. Mesin pencari telah melampaui pencocokan string dan analisis jaringan menjadi tingkat kecanggihan linguistik yang tinggi. Sistem dialog memberikan cara yang semakin umum dan efektif untuk mendapatkan dan berbagi informasi (Einstein, 2019).

Tujuan utama NLP adalah membuat sebuah mesin yang dapat memproses dan menganalisis sejumlah besar data bahasa alami. Untuk mencapai hal tersebut, kita harus mengubah manusia (teks) menjadi bentuk numerik dengan cara memastikan data mesin memiliki semua informasi yang diperlukan dan dapat memprosesnya dengan benar. Cara tradisional dalam merepresentasikan kata-kata adalah menggunakan vektor *one-hot*, di mana vektor dengan hanya satu elemen

target bernilai 1 dan elemen lainnya bernilai 0, dengan panjang vektor sesuai dengan jumlah kosakata unik dalam korpus. Namun, pendekatan ini memiliki masalah utama yaitu ketidakmampuan untuk menghubungkan hubungan antara dua kata berdasarkan representasi *one-hot* mereka. Untuk mengatasi masalah ini, dikembangkanlah *word embedding* (Al-Ansari & Khaled, 2020).

2.4 *Text Classification*

Text classification adalah proses atau teknik untuk mengklasifikasikan atau mengategorikan teks ke dalam kategori atau label yang telah ditentukan sebelumnya. *Text classification* merupakan teknik paling populer di NLP dan digunakan dalam berbagai aplikasi, seperti analisis sentimen, label topik, menjawab pertanyaan, dan *dialog act classification*. Di tengah era ledakan informasi, memproses dan mengklasifikasikan sejumlah besar data teks secara manual memakan waktu dan penuh tantangan. Selain itu, akurasi klasifikasi teks manual dapat dengan mudah dipengaruhi oleh faktor manusia, seperti kelelahan dan tingkat keahlian yang berbeda. Oleh karena itu, penting untuk menggunakan metode *machine learning* guna mengotomatisasi proses klasifikasi, sehingga menghasilkan hasil yang lebih dapat diandalkan dan kurang terpengaruh subjektivitas manusia (Qian Li dkk, 2022).



Gambar 2. 1 Ilustrasi *Flowchart Text Classification*

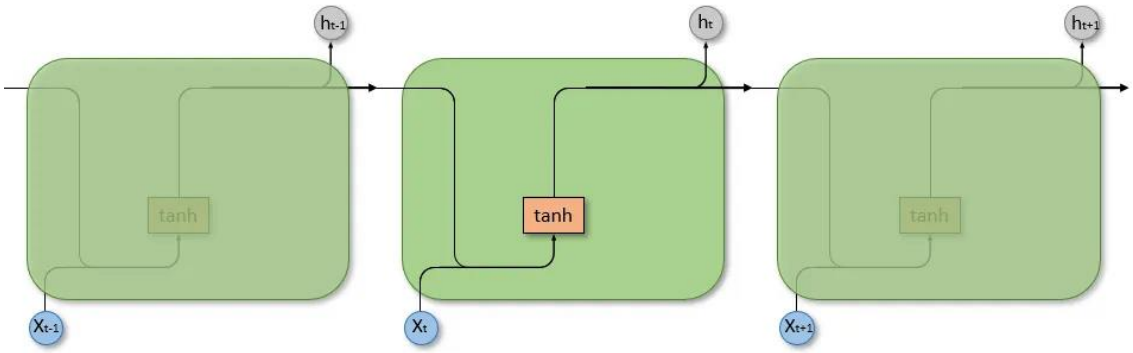
Sumber: (Qian Li dkk, 2022)

Data teks memiliki perbedaan dengan data numerik, gambar, atau sinyal. Untuk memprosesnya dengan tepat, diperlukan teknik NLP yang hati-hati. Langkah penting pertama adalah melakukan *preprocessing* pada data teks agar sesuai dengan model. Model-model tradisional umumnya memerlukan fitur sampel yang baik melalui metode buatan dan kemudian mengklasifikasikannya dengan menggunakan

algoritma *machine learning* klasik, seperti Naïve Bayes, KNN, SVM. Akibatnya, keefektifan metode ini terbatas oleh ekstraksi fitur. Namun, berbeda dengan model-model tradisional, *deep learning* menggabungkan rekayasa fitur ke dalam proses penyesuaian model dengan mempelajari serangkaian transformasi *non-linear* yang berfungsi untuk langsung memetakan fitur ke keluaran (Qian Li dkk, 2022).

2.5 *Recurrent Neural Network* (RNN)

Recurrent Neural Network (RNN) adalah salah satu arsitektur *neural network* yang digunakan untuk memproses data yang berurutan, seperti berupa tulisan tangan, genom, teks atau data *time series*. RNN memiliki siklus dan mentransmisikan informasi kembali ke dirinya sendiri (Robin, 2019). RNN menyimpan informasi dari masa lalu dengan melakukan *looping* di dalam arsitekturnya, sehingga informasi dari masa lalu tetap tersimpan.



Gambar 2. 2 Arsitektur RNN

Sumber: (Dancker, 2022)

Di setiap sel input dari *time step* saat ini x_t, *hidden state* h_{t-1} dari step sebelumnya dan bias digabungkan dan kemudian dibatasi oleh fungsi aktivasi untuk menentukan *hidden state* dari tahap waktu saat ini h_t.

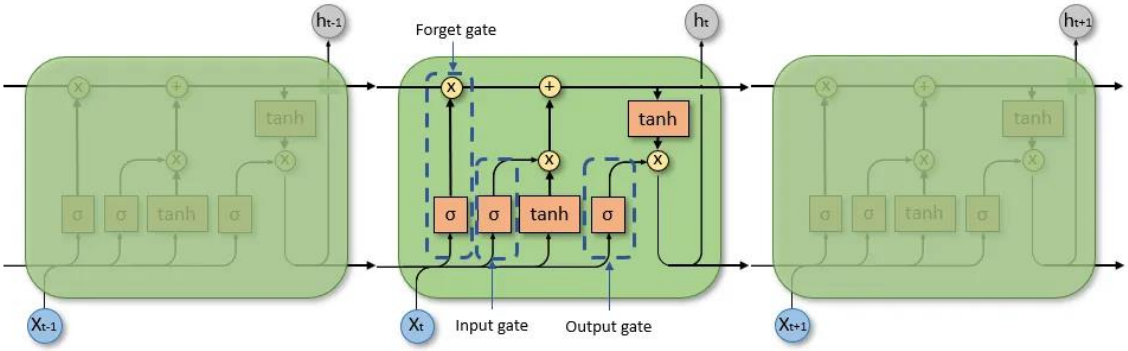
$$h_t = tanh\left(W_x x_t + W_h h_{t-1} + b\right) \quad (2.\ 1)$$

RNN melakukan perhitungan yang sama secara berulang-ulang atas input yang diberikan. RNN mempunyai masalah selama pelatihan, komponen vektor gradien dapat tumbuh secara eksponensial dalam urutan yang panjang. Masalah dengan *exploding* atau *vanishing gradients* ini menyulitkan model RNN untuk

mempelajari korelasi jarak jauh secara berurutan (Liu dkk, 2016). *Vanishing gradient* disebabkan karena gradien yang semakin mengecil hingga *layer* terakhir membuat nilai bobot tidak berubah sehingga menyebabkan tidak pernah memperoleh hasil yang lebih baik atau konvergen. Sebaliknya gradien yang semakin membesar menyebabkan nilai bobot pada beberapa layer juga ikut membesar sehingga algoritma optimasi menjadi divergen atau disebut *exploding gradient*.

2.5.1 *Long Short-Term Memory* (LSTM)

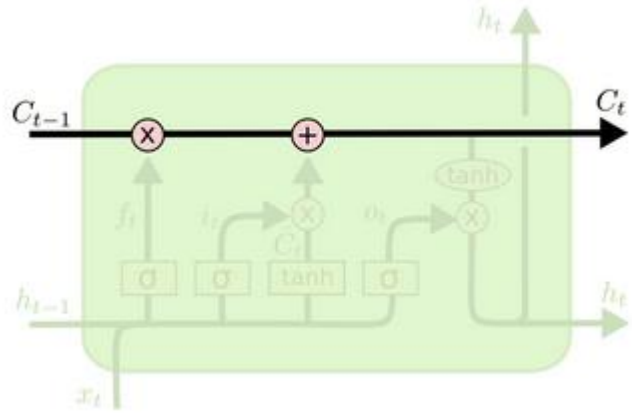
Long Short-Term Memory (LSTM) adalah modifikasi dari arsitektur RNN yang diusulkan oleh Hochreiter and Schmidhuber (1997) untuk secara khusus mengatasi masalah *long-term dependencies* (ketergantungan jangka panjang). LSTM menggunakan mekanisme *gate* untuk menangani masalah *vanishing gradient* pada RNN dan juga untuk mengingat, membaca, dan memperbarui informasi penting dalam jangka waktu yang panjang (Hochreiter dan Schmidhuber, 1997).



Gambar 2. 3 Arsitektur LSTM

Sumber: (Dancker, 2022)

Ide kunci dari LSTM adalah jalur yang menghubungkan konteks lama C_{t-1} ke konteks baru C_t yang terletak di garis horizontal bagian atas modul LSTM seperti pada gambar 2.4.



Gambar 2. 4 Mekanisme Konteks Memori LSTM

Sumber: (Olah, 2015)

Konteks C_t disebut juga *cell state* atau *memory cell*. Dengan jalur tersebut, suatu nilai di konteks yang lama dapat dengan mudah diteruskan ke konteks yang baru dengan modifikasi yang sangat sedikit apabila diperlukan. Konteks merupakan sebuah vektor yang jumlah elemennya ditentukan sendiri. Setiap elemen diharapkan dapat merekam suatu fitur *input* yang fitur-fiturnya ditemukan sendiri oleh LSTM dalam proses latihan.

Gerbang *sigmoid* merupakan ide kunci lain dari LSTM yang mengatur seberapa banyak informasi bisa lewat. *Output* dari gerbang *sigmoid* akan dikalikan dengan suatu nilai lain untuk mengontrol seberapa banyak informasi dari C_{t-1} yang diikutkan menjadi C_t. Fungsi aktivasi *tanh* dapat dijabarkan dalam persamaan berikut:

$$\tanh(x) = \sigma(2x) - 1 \qquad (2.\ 2)$$

2.5.1.1 *Forget Gate*

Forget gate adalah gerbang pertama dalam LSTM. Pada gerbang ini informasi apapun yang tidak dibutuhkan atau kurang memiliki makna akan dibuang menggunakan fungsi *sigmoid*. Ketika nilai dari *forget gate* menghasilkan 1 maka informasi akan diingat, namun apabila nilainya 0 maka informasi tersebut harus dilupakan. Rumus perhitungan nilai pada *forget gate* adalah sebagai berikut:

$$f_t = \sigma\left(W_f \cdot \left[h_{t-1}, x_t\right] + b_f\right) \quad (2.\ 3)$$

2.5.1.2 *Input Gate*

Setelah informasi diolah dalam *forget gate*, tahapan selanjutnya informasi akan diolah di dalam *input gate*. *Input gate* berfungsi untuk menyeleksi informasi tertentu yang akan di update ke bagian *cell state* dengan fungsi aktivasi sigmoid. Selanjutnya tahapan *candidate gate* akan membuat kandidat vektor baru dengan fungsi aktivasi tanh yang ditambahkan pada bagian *cell state*.

Rumus perhitungan pada *input gate* dan *candidate gate* adalah sebagai berikut:

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

$$\bar{C_t} = tanh\left(W_c \cdot \left[h_{t-1}, x_t\right] + b_c\right) \quad (2.\ 5)$$

2.5.1.3 *Cell State*

Kemudian nilai *cell state* yang lama akan diperbarui menjadi nilai *cell state* baru. Dengan menggunakan persamaan sebagai berikut:

$$c_t = f_t * c_{t-1} + i_t * \bar{C_t} \quad (2.\ 6)$$

2.5.1.4 *Output Gate*

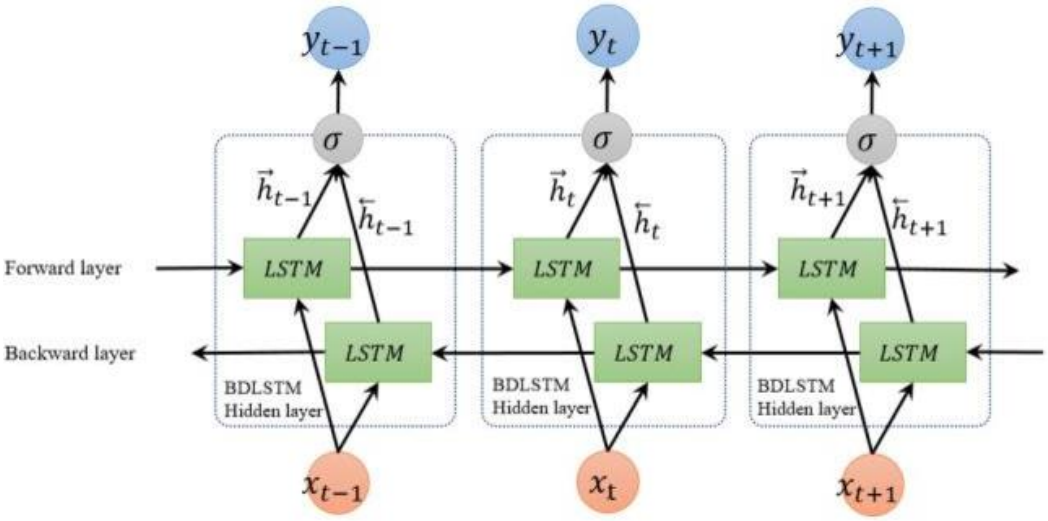
Gerbang terakhir pada arsitektur LSTM yaitu *output gate*. Pada gerbang ini, nilai *hidden state* akan dihasilkan menggunakan fungsi *sigmoid* dan *cell state* menggunakan *tanh* (untuk mendorong nilai menjadi antara -1 dan 1). Setelah itu kedua hasil aktivasi tersebut dikalikan sebelum menuju langkah selanjutnya. Perhitungan *output gate* dan *hidden state* pada persamaan berikut:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.\ 7)$$

$$h_t = o_t * tanh(c_t) \quad (2.\ 8)$$

2.5.2 *Bidirectional Long Short-Term Memory* (BiLSTM)

Bidirectional Long Short-Term Memory (BiLSTM) adalah jaringan saraf LSTM yang terdiri dari dua lapisan LSTM, yaitu lapisan *forward* LSTM untuk memproses dari kata pertama menuju kata terakhir dan lapisan *backward* LSTM untuk memproses dari kata terakhir menuju kata pertama. BiLSTM menghubungkan dua *hidden layers* dari arah yang berlawanan ke *output* yang sama. Dengan bentuk ini, lapisan-lapisan neuron dapat memperoleh informasi dari kondisi masa lalu dan masa depan secara bersamaan (Isnain dkk, 2020).



Gambar 2. 5 Arsitektur BiLSTM

Sumber: (Fadli & Hidayatullah, 2021)

Gambar 2.5 adalah arsitektur dari BiLSTM yang merupakan gabungan dari dua lapisan LSTM, *forward* LSTM *layer* dan *backward* LSTM *layer*. Setiap *hidden unit* pada keluaran lapisan bawah dan atas digabungkan membentuk nilai fitur kata tersebut dengan ukuran lebih panjang daripada LSTM biasa. Dengan nilai fitur yang lebih panjang, informasi akan diproses pada tahapan selanjutnya yaitu *feed forward neural* akan mengklasifikasikan dengan lebih akurat (Fadli & Hidayatullah, 2021).

Perhitungan dari lapisan setiap unit sama dengan LSTM, dan *output* akhir ditentukan oleh keadaan LSTM dari arah depan dan belakang, dapat dijabarkan dalam persamaan berikut (Chen dkk, 2020):

$$\vec{h}_t = LSTM\left(X_t, \vec{h}_{t-1}\right) \quad (2.\ 9)$$

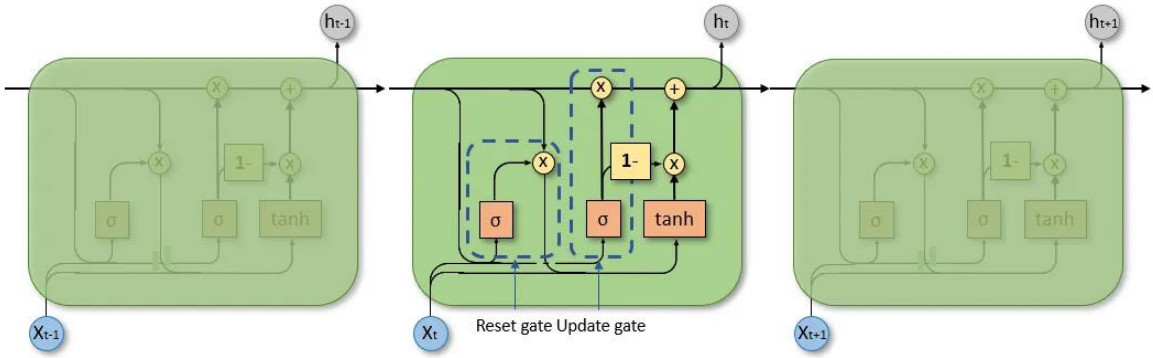
$$\overleftarrow{h}_t = LSTM\left(X_t, \overleftarrow{h}_{t-1}\right) \quad (2.\ 10)$$

$$h_t = w_t\vec{h}_t + v_t\overleftarrow{h}_t + b_t \quad (2.\ 11)$$

dimana $\vec{h}_t$ adalah *forward output* dari LSTM pada waktu *t*, $\overleftarrow{h}_t$ adalah *backward output* dari LSTM pada waktu *t*, h_t adalah *output gate* LSTM dua arah pada waktu *t*, X_t adalah input pada waktu *t*, w_t adalah bobot matrix dari *forward output*, v_t menunjukkan bobot matrix dari *backward output*, dan b_t adalah *offset* pada waktu *t*.

2.5.3 *Gated Recurrent Unit* (GRU)

Gated Recurrent Unit (GRU) adalah jenis dari RNN, dan merupakan varian yang lebih sederhana dari LSTM yang diusulkan oleh Chung dkk (2014). GRU adalah bentuk yang lebih simpel dari LSTM, dan hanya memiliki dua gerbang, yaitu gerbang pembaruan (*update gate*) dan gerbang reset (*reset gate*), yang mengatur aliran informasi di dalam unit, tanpa memiliki sel memori terpisah. GRU menunjukkan kemampuan yang hebat dalam memodelkan dan menangkap ketergantungan jarak jauh antara elemen-elemen dalam urutan. GRU menggunakan perhitungan dua *gate* yang mengelola aliran informasi melalui setiap unit yang tersembunyi (Zulqarnain dkk, 2020).



Gambar 2. 6 Arsitektur GRU

Sumber: (Dancker, 2022)

2.5.3.1 *Reset Gate*

Reset gate bertugas mengatur memori jangka pendek dengan memutuskan seberapa banyak informasi masa lalu yang akan dipertahankan dan diabaikan. Hal ini memungkinkan GRU untuk "me-reset" atau menghapus sebagian dari informasi lama yang dianggap tidak relevan, sehingga hanya mempertahankan informasi penting untuk memori jangka pendek.

$$r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right] + b_r\right) \qquad (2.\ 12)$$

Dalam vektor r, nilai-nilainya dibatasi antara 0 dan 1 melalui fungsi sigmoid dan dipengaruhi oleh *hidden state h* dari langkah waktu sebelumnya dan input saat ini x. Keduanya diberi bobot menggunakan matriks bobot W. Selain itu, ada penambahan bias *b* (Dancker, 2022).

2.5.3.2 *Update Gate*

Update gate berfungsi untuk memori jangka panjang dann memiliki kesamaan dengan *forget gate* pada LSTM. *Update gate* menentukan seberapa banyak informasi yang akan disimpan dalam memori jangka panjang, dengan cara mengambil bagian dari informasi yang baru masuk dan mempertahankan sebagian dari informasi lama yang dianggap relevan.

$$z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right] + b_z\right) \quad (2.\ 13)$$

Perbedaan utama antara *reset gate* dan *update gate* adalah hanya pada bobot W (Dancker, 2022).

2.5.3.3 *Hidden State*

Hidden state pada langkah waktu saat ini ditentukan melalui dua tahap proses. Pertama, sebuat kandidat *hidden state* ditentukan. Keadaan kandidat ini merupakan hasil kombinasi dari input saat ini dan *hidden state* dari langkah waktu sebelumnya dengan menggunakan fungsi aktivasi tertentu, seperti *tanh*. Selanjutnya, pengaruh dari *hidden state* sebelumnya dengan kandidat *hidden state* diatur oleh *reset gate* (Dancker, 2022).

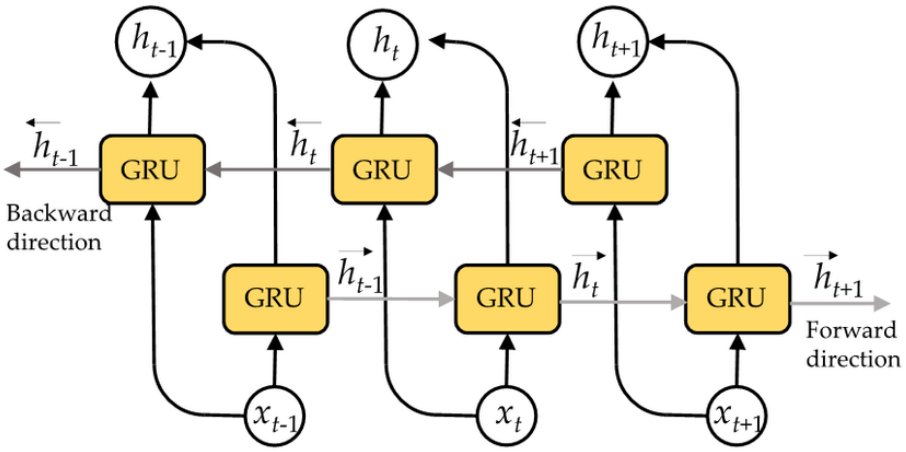
$$\hat{h}_t = tanh\left(W_h \cdot \left[r_t * h_{t-1}, x_t\right] + b_g\right) \quad (2.\ 14)$$

Pada langkah kedua, kandidat *hidden state* bergabung dengan *hidden state* dari langkah waktu sebelumya untuk membentuk *hidden state* saat ini. Proses penggabungan ini ditentukan oleh *update gate* (Dancker, 2022).

$$h_t = z_t * h_{t-1} + (1 - z_t) * \hat{h}_t \quad (2.\ 15)$$

2.5.4 *Bidirectional Gated Recurrent Unit* (BiGRU)

Bidirectional Gated Recurrent Unit (BiGRU) terdiri dari dua unit GRU, yaitu GRU maju (*forward* GRU) yang mengolah informasi dari awal ke akhir, dan GRU mundur (*backward* GRU) yang mengolah informasi dari akhir ke awal. Dengan menggunakan kedua unit GRU ini, BiGRU dapat memahami konteks secara keseluruhan dalam data berurutan, karena dapat memperhitungkan informasi dari kedua arah, baik depan dan belakang (Mekruksavanich & Jitpattanakul, 2022).



Gambar 2. 7 Arsitektur BiGRU

Sumber: (Mekruksavanich & Jitpattanakul, 2022)

Seperti yang ditunjukkan dalam gambar, setiap *input sequence* dimasukkan ke dalam jaringan GRU maju (*forward*) dan jaringan GRU mundur (*backward*), menghasilkan dua vektor *hidden layer state* yang simetris.

$$\overrightarrow{h_t} = GRU\left(x_t, \vec{h}_{t-1}\right) \quad (2.\ 16)$$

$$\overleftarrow{h_t} = GRU\left(x_t, \overleftarrow{h}_{t-1}\right) \quad (2.\ 17)$$

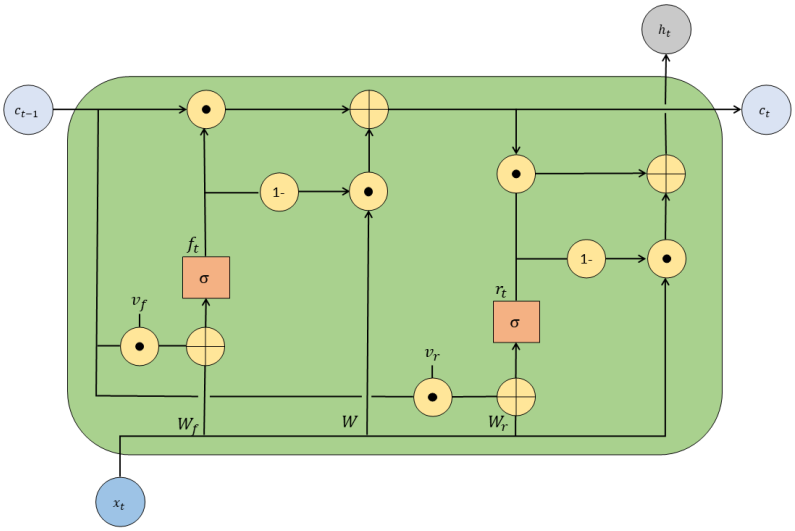
$$h_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right] \quad (2.\ 18)$$

2.5.5 *Simple Recurrent Unit* (SRU)

Simple Recurrent Unit (SRU) merupakan varian dari RNN yang diusulkan oleh Tao Lei dkk (2018). SRU adalah sebuah unit rekurensi ringan (*light recurrent*) yang menyeimbangkan kapasitas dan skalabilitas model. SRU dirancang untuk memberikan rekurensi yang ekspresif, memungkinkan implementasi yang paralel, dan dilengkapi dengan inisialisasi yang hati-hati untuk memfasilitasi pelatihan model-model mendalam (Lei dkk, 2018).

Pada arsitektur LSTM dan GRU, proses komputasi pada setiap langkah dalam jaringan rekurensi harus dilakukan secara berurutan, artinya langkah berikutnya bergantung pada hasil langkah sebelumnya.

Ketergantungan sekuensial ini membuat jaringan rekurensi menjadi lebih lambat dibandingkan dengan operasi lain yang tidak memiliki ketergantungan sekuensial. SRU didesain untuk mengatasi masalah ini dengan memberikan kemampuan rekurensi yang ekspresif dan memungkinkan implementasi yang sangat paralel (Lei dkk, 2018).



Gambar 2. 8 Arsitektur SRU

Sebuah *single layer* SRU melibatkan perhitungan sebagai berikut,

$$f_t = \sigma\left(W_f x_t + v_f \odot c_{t-1} + b_f\right) \quad (2.\ 19)$$

$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot (W x_t) \quad (2.\ 20)$$

$$r_t = \sigma(W_r x_t + v_r \odot c_{t-1} + b_r) \quad (2.\ 21)$$

$$h_t = r_t \odot c_t + (1 - r_t) \odot x_t \quad (2.\ 22)$$

dimana, W_f, W, b_f adalah matriks parameter dan v_f, v_r, b_f adalah vektor parameter yang nilainya akan diatur selama proses pelatihan.

SRU terdiri dari dua komponen penting, yaitu *light recurrence* dan *high network.* Komponen *light recurrence* berfungsi membaca secara berulang vektor masukan x_t dan menghitung urutan state c_t untuk menangkap informasi urutan. Modul ini menangkap informasi sekuensial (Lei dkk, 2018). Komputasi perhitungannya menyerupai dengan jaringan

recurrent lainnya seperti LSTM, GRU, dan RAN. Pada SRU, *forget gate* f_t mengontrol aliran informasi (Persamaan 2.19) dan *state* saat ini c_t ditentukan dengan cara menghitung rata-rata adaptif dari *state* sebelumnya c_{t-1} dan observasi Wx_t saat ini berdasarkan nilai f_t (Persamaan 2.20).

Salah satu perbedaan utama dari arsitektur *gated recurrent* sebelumnya adalah cara c_{t-1} digunakan di gerbang *sigmoid*. Biasanya, c_{t-1} dikalikan dengan matriks parameter untuk menghitung f_t, seperti $f_t = \sigma\left(W_f x_t + V_f c_{t-1} + b_f\right)$. Namun, penggunaan $V_f c_{t-1}$ membuat sulit untuk memparalelkan komputasi *state* karena setiap dimensi dari c_t dan f_t tergantung pada seluruh entri dari c_{t-1}, dan perhitungannya harus menunggu sampai c_{t-1} sepenuhnya dihitung. Komponen *light recurrence* menggunakan operasi *point-wise multiplication* $v_r \odot c_{t-1}$ untuk memungkinkan proses perhitungan menjadi lebih parallel. Dengan penyederhanaan ini, masing-masing dimensi *state vector* menjadi *independent* dan karenanya dapat diparalelkan (Lei dkk, 2018).

Komponen kedua SRU adalah *highway network*, yang berfungsi untuk mendukung pelatihan berbasis gradien pada *deep networks*. Jaringan ini menggunakan sebuah *reset gate* r_t (Persamaan 2.21) untuk menggabungkan input x_t dan *state* saat ini c_t dari komponen *light recurrence* (Persamaan 2.22), dimana $(1 - r_t) \odot x_t$ merupakan sebuah *skip connection* yang memungkinkan gradien untuk langsung di propagasikan ke lapisan sebelumnya, yang telah terbukti meningkatkan skalabilitas.

Dengan menggabungkan kedua komponen ini, arsitektur secara keseluruhan menjadi sederhana namun ekspresif, serta mudah untuk ditingkatkan skalanya karena adanya peningkatan paralelisme dan *gradient propagation* (Lei dkk, 2018).

2.6 *Word Embedding FastText*

Word embedding adalah sebuah cara untuk merepresentasikan kata sebagai vektor bilangan real kontinu dengan panjang yang sudah ditetapkan. Ini memetakan

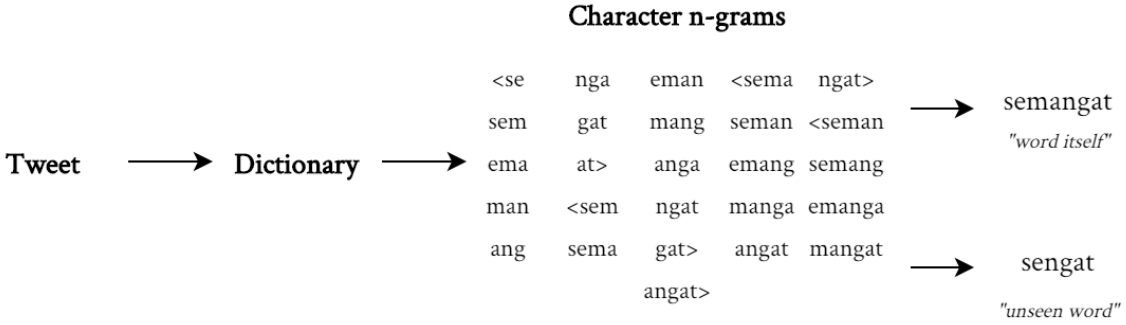
kata dalam kosakata ke *latent vector space*, di mana kata-kata dengan konteks serupa berada dalam jarak yang dekat. *Word embedding* dapat menangkap makna sintaksis dan semantik kata. Sehingga, *word embedding* dianggap sangat cocok untuk digunakan sebagai representasi fitur dalam model *neural network* untuk tugas *natural language processing*, seperti klasifikasi teks, terjemahan mesin, pembelajaran sekuens, dll (Wang dkk, 2020).

Salah satu *word embedding* yang paling populer adalah *FastText*. *FastText* diperkenalkan oleh Bojanowski dkk (2017) adalah metode *word embedding* yang merupakan pengembangan dari model Word2vec. Metode ini mempelajari representasi kata dengan mempertimbangkan informasi *subword*. Setiap kata direpresentasikan sebagai sekumpulan karakter *n-gram*, sehingga dapat membantu menangkap arti kata-kata yang lebih pendek dan memungkinkan *embedding* untuk memahami sufiks dan prefiks dari kata.

Representasi vektor dikaitkan dengan setiap karakter *n-gram*, sedangkan kata-kata direpresentasikan sebagai jumlah dari representasi vektor tersebut (Nurdin dkk, 2020). Contoh implementasi *n-gram* pada kata "makan" dengan *trigram* ($n = 3$) direpresentasikan sebagai:

$$< ma, mak, aka, kan, an >$$

Representasi vektor kata diperoleh dengan menjumlahkan nilai dari setiap *n-gram*. Bahkan kata-kata yang tidak ditemukan dalam korpus dapat direpresentasikan dengan baik karena kemungkinan beberapa *n-gram* pembentuk kata tersebut muncul dalam *n-gram* yang ada dalam korpus (Bojanowski, 2017).



Gambar 2. 9 Ilustrasi Cara Kerja *Character n-grams* pada *FastText*
Sumber: (Alfariqi dkk, 2020)

Misalkan variabel h_w sebagai kata yang ingin dicari vektor representasinya, variabel G_w sebagai himpunan *n-gram* yang muncul pada kata tersebut, variabel z_g sebagai representasi vektor dari setiap *n-gram*, dan v_c sebagai vektor kata dari kata konteks c, maka representasi vektor dari kata tersebut dapat dihitung menggunakan Persamaan (2.23) berikut (Bojanowski, 2017).

$$h_w = \sum_{g \in G_w} z_g^{\intercal} v_c \qquad (2.\ 23)$$

FastText menyediakan dua model untuk menghitung representasi kata yaitu *Skip-gram* dan *Continuous Bag-of-Word* (CBOW). Model *skip-gram* belajar memprediksi target kata berdasarkan kata terdekat. Di sisi lain, model cbow memprediksi target kata sesuai dengan konteksnya. Konteks merupakan representasi sekumpulan kata yang terdapat dalam *fixed size window* di sekitar target kata.

2.7 *Confusion Matrix*

Confusion matrix merupakan sebuah matriks yang berisi informasi untuk mengevaluasi kinerja model klasifikasi yang digunakan, memberikan gambaran tentang bagaimana algoritma klasifikasi bekerja pada tahap evaluasi (Fadli & Hidayatullah, 2021).

Tabel 2.1. *Confusion Matrix*

		Prediksi	
		Positif	**Negatif**
Aktual	**Positif**	*True Positive* (TP)	*False Negative* (FN)
	Negatif	*False Positive* (FP)	*True Negative* (TN)

Dari Tabel 2.1, terdapat beberapa *metrics* yang dapat dihitung untuk mengevaluasi algoritma yaitu *accuracy, precision, recall,* dan *F1-Score*.

a. *Accuracy* mengukur sejumlah prediksi yang tepat berdasarkan keseluruhan data.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.\ 24)$$

b. *Precision* mengukur sejauh mana model benar ketika memprediksi sampel sebagai positif.

$$precision = \frac{TP}{TP+FP} \quad (2.\ 25)$$

c. *Recall* mengukur sejauh mana model dapat mengidentifikasi semua *instance* positif yang seharusnya.

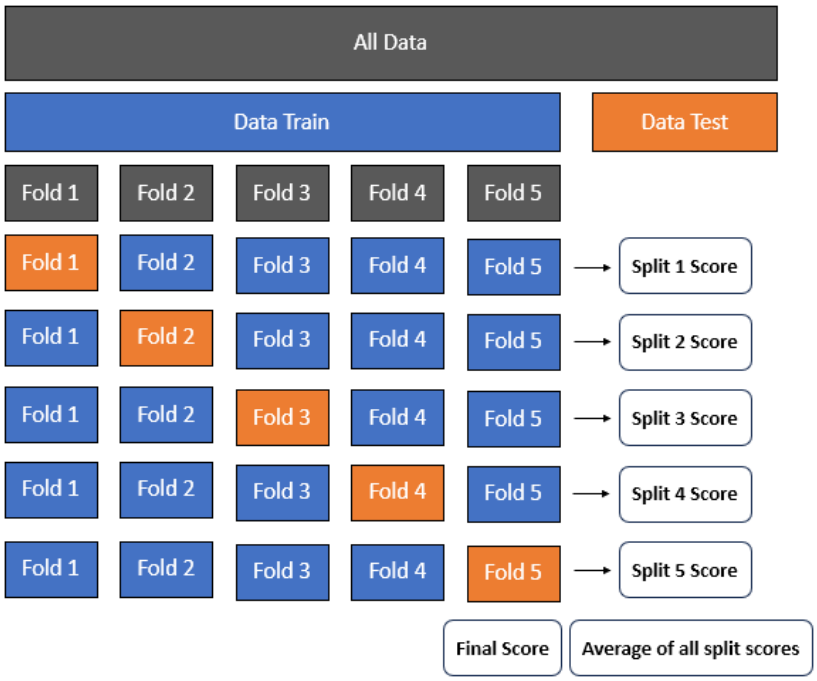
$$recall = \frac{TP}{TP+FN} \quad (2.\ 26)$$

d. *F1-Score* adalah *metric* gabungan yang mengukur keseimbangan antara *precision* dan *recall*.

$$f1 - score = \frac{2*precision*recall}{precision+recall} \quad (2.\ 27)$$

2.8 *Stratified K-Fold Cross Validation*

Cross Validation merupakan sebuah metode statistik yang digunakan untuk mengevaluasi sebuah performa model atau algoritma. Pada *k-fold cross validation*, set pembelajaran yang tersedia dibagi menjadi *k* subset terpisah dengan ukuran yang hampir sama. Di sini, "*fold*" berarti jumlah subset yang dihasilkan. Pembagian ini dilakukan dengan cara mengambil sampel kasus secara acak dari set pembelajaran tanpa penggantian. Model dilatih menggunakan *k-1* subset, ini mewakili set pelatihan. Kemudian, model diterapkan pada subset yang tersisa, yang disebut sebagai set pengujian, dan kinerjanya diukur. Prosedur ini diulang hingga setiap dari *k* subset berfungsi sebagai set pengujian. Performa dari *cross validation* adalah rata-rata dari pengukuran kinerja pada *k* set pengujian. (Berrar, 2019)



Gambar 2. 10 5-*Fold Cross Validation*

Metode *stratified k-fold cross validation* membagi dataset menjadi beberapa lipatan *k* dengan setiap lipatan memiliki (hampir) persentase sampel yang sama dari kelas minoritas dan mayoritas seperti dataset aslinya. Satu lipatan dipilih untuk pengujian, dan sisanya digunakan untuk pelatihan (Szeghalmy & Fazekas, 2023).

2.9 *Hyperparameter*

Hyperparameter adalah parameter dari algoritma *machine learning* (bukan dari model). Oleh karena itu, *hyperparameter* tidak dipengaruhi oleh algoritma pembelajaran itu sendiri, melainkan harus diatur sebelum pelatihan dan tetap konstan selama pelatihan (Géron, 2017).

2.9.1 *Optimizer*

Dalam melatih *deep neural network* yang sangat besar bisa sangat lambat. Untuk meningkatkan kecepatannya dapat menggunakan sebuah *optimizer* yang lebih cepat daripada *gradient descent optimizer* biasa. Beberapa *optimizer* yang paling populer, yaitu *Momentum Optimization*, *Nesterov Accelerated Gradient*, *AdaGrad*, *RMSProp*, dan terakhir *Adam* optimasi.

RMSProp (*Root Mean Square Propagation*) dibuat oleh Tijmen dan Geoffrey Hinton di tahun 2012. *RMSProp* dirancang untuk mengatasi kendala yang dimiliki oleh *AdaGrad* yang dapat melambat terlalu cepat dan tidak konvergen ke optimum global. *RMSProp* mencapai hal ini dengan mengakumulasi hanya gradien dari iterasi paling baru, berbeda dengan metode seperti *AdaGrad* yang mengakumulasi semua gradien sejak awal pelatihan. Untuk mencapai ini, *RMSProp* menggunakan penurunan eksponensial (*exponential decay*) dalam langkah pertama.

2.9.2 *Loss Function*

Loss function mengukur perbedaan antara nilai yang diprediksi oleh model dengan nilai sebenarnya (label atau target). *Binary cross entropy loss*, yang sering disebut *logistic loss*, digunakan pada klasifikasi biner dengan *output* probabilitas kondisional. Dimisalkan sebuah set dua kelas target yang diberi label 0 dan 1, dengan label yang benar $y \in \{0, 1\}$. *Output* klasifikasi $\tilde{y}$ diubah menggunakan fungsi *sigmoid* $\sigma(x) = 1/(1 + e^{-x})$ ke rentang [0, 1], dan diinterpretasikan sebagai probabilitas kondisional $\tilde{y} = \sigma(\tilde{y}) = P(y = 1|x)$. Dengan aturan prediksinya sebagai berikut:

$$prediction = \begin{cases} 0 & \hat{y} < 0.5 \\ 1 & \hat{y} \geq 0.5 \end{cases}$$

Model dilatih untuk memaksimalkan *log* probabilitas $P(y = 1|x)$ bersyarat log untuk setiap contoh pelatihan $(\boldsymbol{x}, y)$. *Logistic loss* didefinisikan sebagai berikut.

$$L_{\text{logistic}}(\hat{y}, y) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \qquad (2.\ 28)$$

Ketika menggunakan *logistic loss*, diasumsikan bahwa *output layer* mengalami transformasi menggunakan fungsi *sigmoid*.

2.9.3 *Learning Rate*

Learning rate adalah parameter yang mengontrol sejauh mana model melakukan penyesuaian berdasarkan gradien hasil dari *loss function*. Dalam konteks optimisasi model, *learning rate* menentukan seberapa besar langkah-langkah yang diambil oleh algoritma selama proses pelatihan.

Pemilihan *learning rate* merupakan hal penting. *Learning rate* yang terlalu besar dapat menghambat model agar tidak konvergen ke solusi yang efektif, atau menyebabkan divergensi. Sebaliknya, *learning rate* yang terlalu kecil akan memerlukan waktu yang sangat lama untuk konvergen. Sebagai aturan praktis, sebaiknya melakukan eksperimen dengan rentang nilai *learning rate* awal dalam rentang [0,1], seperti 0.001, 0.01, 0.1, 1.

2.9.4 *Regularizer*

Regularizer L2 disebut juga sebagai *weight decay* merupakan metode untuk mengendalikan kompleksitas model dengan menambahkan penalti terhadap nilai-nilai besar parameter model. Dalam konteks ini, istilah *weight decay* digunakan untuk menggambarkan pengurangan bobot atau nilai parameter model. Model yang diatur dengan L2 *regularization* akan mendapatkan hukuman yang signifikan untuk bobot parameter yang tinggi. Namun, begitu nilai bobot mendekati nol, pengaruh dari penalti ini menjadi kurang signifikan. Dengan menggunakan L2 *regularization*, model lebih cenderung untuk mengurangi nilai satu parameter dengan bobot tinggi sebanyak 1, dibandingkan dengan mengurangi nilai sepuluh parameter yang sudah memiliki bobot relatif rendah sebanyak 0.1 masing-masing.

2.9.5 *Batch Size*

Batch size adalah jumlah sampel yang diproses dalam *neural network* sebelum model diperbarui. Di akhir *batch*, prediksi dibandingkan dengan variabel *output* yang diharapkan untuk menghitung *error*. Dari *error* ini, dilakukan pembaruan bobot untuk memperbaiki model dengan menggunakan algoritma *backpropagation* yang bergerak mundur dari *layer* terakhir menuju *layer* pertama.

Ketika semua sampel pelatihan digunakan untuk membuat satu *batch*, algoritma pembelajaran disebut *batch gradient descent*. Ketika *batch* adalah ukuran satu sampel, algoritma pembelajaran disebut *stochastic gradient descent*. Ketika *batch size* lebih dari satu sampel dan kurang dari ukuran *dataset* pelatihan, algoritma pembelajaran disebut *mini-batch*

gradient descent. Pada kasus *mini-batch gradient descent*, *batch sizes* yang populer termasuk 32, 64, dan 128 sampel.

2.9.6 *Network Nodes*

Network nodes merujuk pada unit-unit komputasi individual yang membentuk suatu jaringan, seperti *neural network*. Setiap *node* memiliki fungsi untuk menerima *input*, melakukan pengolahan data menggunakan fungsi transfer, dan menghasilkan *output* yang kemudian diteruskan ke *node* atau lapisan berikutnya dalam jaringan. Jika jumlah *node* terlalu sedikit, kemampuan pembelajaran jaringan akan terbatas dan dapat mengakibatkan *underfitting*. Sebaliknya, jika jumlah *node* terlalu besar, kompleksitas meningkat dan model dapat mengalami *overfitting* (Xu dkk, 2019).

2.9.7 *Epochs*

Epochs adalah jumlah iterasi dari set pelatihan. Saat *epochs* meningkat, kemampuan generalisasi model meningkat. Namun, jika jumlah *epochs* terlalu besar, masalah *overfitting* dapat mudah terjadi, dan kemampuan generalisasi model menjadi berkurang. Oleh karena itu, penting dalam jumlah *epochs* yang tepat (Xu dkk, 2019).

2.9.8 *Dropout*

Dropout merupakan salah satu teknik regulasi paling populer untuk *deep neural network.* Teknik ini diusulkan oleh Geoffrey Henton (2012) dan telah terbukti sangat sukses dalam *neural networks* dapat meningkatkan akurasi 1-2% dengan menambahkan *dropout* (Geron, 2017). *Dropout* adalah teknik di mana *neuron* yang dipilih secara acak diabaikan selama pelatihan, artinya aktivasi *neuron* untuk sementara dihapus pada *forward pass* dan setiap pembaruan bobot tidak diterapkan pada *neuoron* pada *backward pass*.

2.10 *Python*

Python adalah bahasa pemrograman serbaguna yang diperkenalkan pada tahun 1991 oleh Guido van Rossum (GvR). *Python* dirancang dengan tujuan sebagai bahasa pemrograman yang mudah dibaca dan dipahami, serta memiliki kemampuan penanganan kesalahan (*exception handling*). *Python* berhasil menjadi bahasa pemrograman yang dapat diterapkan dalam berbagai bidang. Dengan

kemampuannya, *Python* dapat digunakan untuk membangun *website* (*server-side*), melakukan analisis data, hingga untuk pembelajaran mesin (*machine learning*).

Python menduduki peringkat tertinggi di antara bahasa pemrograman yang dipilih oleh para programmer atau pemula untuk memulai belajar kode atau pemrograman. *Python* juga menonjol dalam mendukung berbagai gaya pemrograman, mencakup pendekatan terstruktur, prosedural, berorientasi objek, dan fungsional. Dengan fleksibilitasnya, *python* memberikan ruang yang luas bagi para pengembang untuk mengeksplorasi dan menerapkan berbagai paradigma pemrograman sesuai dengan kebutuhan proyek mereka.

2.11 *Google Colaboratory*

Google Colaboratory, atau disingkat *colab* adalah layanan *cloud computing* yang disediakan oleh *Google*. *Colab* memungkinkan pengguna untuk menulis dan mengeksekusi kode Python di lingkungan *cloud* yang disediakan oleh *Google*, tanpa memerlukan konfigurasi tambahan atau pengaturan. Layanan ini sangat populer di kalangan peneliti, ilmuwan data, dan pengembang *machine learning* karena menyediakan akses gratis ke GPU (*Graphics Processing Unit*) dan TPU (*Tensor Processing Unit*) untuk percepatan komputasi.

2.12 *Snscrape*

Snscrape adalah *library Python* yang digunakan untuk mengekstrak data dari media sosial, terutama Twitter. Dengan *snscrape*, dapat melakukan pencarian, mengekstrak *tweet*, dan mengumpulkan informasi lainnya dari Twitter menggunakan bahasa pemrograma *Python*. Beberapa fungsi utama *snscrape* mencakup kemampuan untuk mengambil *tweet* berdasarkan kueri pencarian tertentu, mengekstrak informasi dari akun pengguna Twitter, dan memperoleh metadata terkait dengan *tweet*, seperti jumlah *retweet* dan favorit.

2.13 *Hugging Face*

Hugging Face adalah *platform open-source* yang berfokus pada pembangunan dan distribusi model bahasa alami (NLP) yang canggih. Mereka terkenal karena menyediakan perpustakaan (*library*) dan repositori besar untuk model bahasa alami, serta alat-alat dan sumber daya yang mendukung pengembangan dalam bidang pemrosesan bahasa alami (*Natural Language*

Processing, NLP). *Hugging face* menyediakan dukungan untuk model *fasttext*, salah satu model populer untuk representasi vektor kata (*word vectors*) dan klasifikasi teks.

Hugging face saat ini menyediakan *hosting* resmi dari vektor kata *fasttext* untuk semua 157 bahasa dan model terbaru untuk identifikasi bahasa (Grave dkk, 2018). Vektor kata yang sudah dilatih (*pre-trained word vectors*) untuk 157 bahasa dilatih pada *Common Crawl* dan Wikipedia menggunakan *FastText*. Model-model ini dilatih menggunakan CBOW dengan *position-weights*, dimensi 300, karakter *n-gram* dengan panjang 5, *window size* ukuran 5 dan 10 negatif (Grave dkk, 2018). Dengan menggunakan *hugging face*, dapat dengan mudah mengunduh dan menggunakan model-model tersebut hanya dengan beberapa perintah.

2.14 *PyTorch*

PyTorch adalah sebuah *framework machine learning open-source* yang dikembangkan oleh *Facebook's* AI *Research lab* (FAIR). *PyTorch* menawarkan pendekatan yang dinamis dan intuitif dalam pembangunan model. Salah satu fitur utama *PyTorch* adalah konsep *tensor*, yang merupakan struktur data mirip *array* multidimensi, memungkinkan representasi data yang efisien.

Model *PyTorch* dapat dibangun dengan mudah berkat fleksibilitasnya, dan *framework* ini mendukung grafik komputasi dinamis, memudahkan eksperimen dan *debug*. *PyTorch* juga menawarkan dukungan untuk komputasi GPU, meningkatkan kinerja pelatihan model pada dataset besar. Kombinasi antarmuka yang ramah pengguna dan kemampuan yang kuat membuat *PyTorch* menjadi pilihan populer di kalangan peneliti dan praktisi dalam dunia kecerdasan buatan.

2.15 *Streamlit*

Streamlit adalah sebuah *framework open-source* untuk membuat aplikasi *web* dengan cepat menggunakan *Python.* Dikembangkan untuk mempermudah proses pembuatan aplikasi data yang interaktif. *Streamlit* memungkinkan pengguna untuk dengan mudah mengonversi skrip *Python* sederhana menjadi aplikasi *web* yang menarik.

2.16 *Pandas*

Pandas adalah *library* populer yang digunakan untuk pengelolaan dan analisis data. *Pandas* bersifat *open-source* dan dibangun dengan bahasa pemrograman *Python*. *Pandas* memungkinkan para pengguna, terutama para analis data dan ilmuwan data, untuk secara efisien melakukan berbagai operasi pada data, seperti pembersihan, transformasi, dan analisis.

2.17 *NumPy*

NumPy (*Numerical Python*) adalah *package* fundamental yang sering digunakan untuk *scientific computing* pada *Python*. Dengan larik N-dimensi yang kuat dan konsep vektorisasi yang efisien, *NumPy* telah menjadi standar *de facto* dalam komputasi larik, menyediakan alat-alat matematika, transformasi Fourier, dan rutinitas aljabar linear yang mendalam. Dengan lisensi *open-source* dan dukungan dari komunitas yang beragam, *NumPy* tetap menjadi pilihan utama bagi para ilmuwan data, peneliti, dan pengembang dalam menjalankan tugas-tugas komputasi ilmiah dengan *Python*.

2.18 *Matplotlib*

Matplotlib adalah *library* yang digunakan untuk melakukan visualisasi data. Dengan *Matplotlib*, pembuatan visualisasi yang sederhana menjadi mudah dilakukan, sementara hal-hal yang kompleks juga dapat diwujudkan. *Matplotlib* memberikan kemudahan dalam menyajikan data secara grafis dan menyediakan berbagai alat untuk menyesuaikan plot sesuai kebutuhan. Dengan keandalan dan keluasan fungsinya, *Matplotlib* menjadi pilihan utama bagi para pengembang dan ilmuwan data yang ingin menghasilkan visualisasi data yang informatif dan estetis dalam lingkungan *Python*.

2.19 *Keras*

Keras adalah sebuah API yang dirancang untuk memudahkan pembuatan dan pelatihan model jaringan saraf tiruan. Penyediaan API yang konsisten dan simpel, pengurangan tindakan yang diperlukan untuk kasus umum, serta penyediaan pesan kesalahan yang jelas dan mudah dimengerti. *Keras* juga memberikan prioritas tinggi pada pembuatan dokumentasi dan panduan pengembang yang berkualitas. Dengan strategi ini, *Keras* bertujuan memberikan

pengalaman pengguna yang intuitif dan efisien dalam pengembangan model jaringan saraf tiruan menggunakan *Python*.

2.20 *Sastrawi*

Sastrawi adalah sebuah proyek *open-source* yang bertujuan untuk mengembangkan alat pemrosesan bahasa alami (*Natural Language Processing*) untuk Bahasa Indonesia. Proyek ini mencakup implementasi algoritma-algoritma pemrosesan bahasa alami seperti *stemming* (penghapusan imbuhan), *lemmatization* (pengembalian ke bentuk dasar), dan beberapa fungsi lainnya yang berguna dalam analisis teks dalam Bahasa Indonesia. *Sastrawi* menerapkan algoritma Nazief dan Adriani dalam proses *stemming*nya.

2.21 *Scikit-Learn*

Scikit-learn adalah *library* dalam bahasa pemrograman *Python* yang sangat populer untuk *machine learning*. *Scikit-learn* menyediakan berbagai algoritma dan fungsi utilitas untuk tugas-tugas seperti klasifikasi, regresi, *clustering*, reduksi dimensi, dan lainnya. Selain itu, *scikit-learn* juga menyediakan utilitas untuk evaluasi model, pemrosesan data, dan pemilihan model yang efektif. Dengan dokumentasi yang baik dan desain yang bersih, *scikit-learn* menjadi salah satu pilihan utama untuk praktisi *machine learning* dan data *scientist* dalam mengembangkan dan menerapkan model machine learning di berbagai proyek.