ANALISIS DAN PERBANDINGAN KLASIFIKASI *HATE SPEECH* MENGGUNAKAN METODE FITUR MANUAL *SVM* DAN *BERT*

SKRIPSI

Diajukan Oleh :

HUSNUL MAWADDAH
NIM. 190212035

Bidang Peminatan: Teknik Komputer dan Jaringan

Mahasiswa Fakultas Tarbiyah dan Keguruan

Program Studi Pendidikan Teknologi Informasi

UNIVERSITAS ISLAM NEGERI AR-RANIRY
FAKULTAS TARBIYAH DAN KEGURUAN
PROGRAM STUDI PENDIDIKAN TEKNOLOGI INFORMASI
BANDA ACEH
2023 M / 1445 H

ANALISIS DAN PERBANDINGAN KLASIFIKASI *HATE SPEECH* MENGGUNAKAN METODE FITUR MANUAL *SVM* DAN *BERT*

SKRIPSI

Diajukan Kepada Fakultas Tarbiyah dan Keguruan (FTK)
Universitas Islam Negeri Ar-Raniry Darussalam Banda Aceh
Sebagai Beban Studi Untuk Memperoleh Gelar Sarjana
Dalam Pendidikan Teknologi Informasi

Oleh

HUSNUL MAWADDAH
NIM. 190212035
Mahasiswa Fakultas Tarbiyah dan Keguruan
Prodi Pendidikan Teknologi Informasi

Disetujui Oleh :

Pembimbing I,	Pembimbing II,
(Hendri Ahmadian, M.I.M)	(Ridwan, S.ST. M.T)
NIP/NIDN: 198301042014031002	NIP/NIDN: 198402242019031004

ANALISIS DAN PERBANDINGAN KLASIFIKASI *HATE SPEECH* MENGGUNAKAN METODE FITUR MANUAL *SVM* DAN *BERT*

SKRIPSI

Telah diuji oleh Panitia Ujian Munaqasyah Skripsi Fakultas Tarbiyah dan Keguruan UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus Serta diterima Sebagai salah satu beban studi Program Sarjana (S-1) dalam Pendidikan Teknologi Informasi

Pada:

18 Desember 2023
Darussalam – Banda Aceh
Panitia Ujian Munaqasyah Skripsi

Ketua

(Hendri Ahmadian, S.Si., M.I.M)
NIP/NIDN. 198301042014031002

Sekretaris

(Ridwan, S.ST., M.T.)
NIP/NIDN. 198402242019031004

Penguji 1

(Aulia Syarif Aziz, S.Kom., M.Sc.)
NIP/NIDN. 199305212022031001

Penguji 2

(Baihaqi, M.T.)
NIP/NIDN. 1321028801

Mengetahui,
Dekan Fakultas Tarbiyah dan Keguruan UIN Ar-Raniry
Darussalam Banda Aceh

Prof. Safrul Muluk, S.Ag, M.A., M.Ed., Ph.D
NIP. 197302102199703100[illegible]

LEMBAR PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini:

Nama : Husnul Mawaddah

NIM : 190212035

Program Studi : Pendidikan Teknologi Informasi

Fakultas : Tarbiyah dan Keguruan

Judul Skripsi : Analisis dan Perbandingan Klasifikasi *Hate Speech* Menggunakan Metode Fitur Manual *SVM* dan *BERT*

Dengan ini menyatakan bahwa dalam penulisan skripsi ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggung jawabkan.
2. Tidak melakukan plagiat terhadap naskah karya orang lain
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya
4. Tidak memanipulasi dan memalsukan data
5. Mengerjakan sendiri karya ini dan mampu bertanggung jawab atas karya ini

Bila dikemudian hari ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat dipertanggung jawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenai sanksi berdasarkan aturan yang berlaku di Fakultas Tarbiyah dan Keguruan UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya.

Banda Aceh, 30 November 2023

Yang menyatakan

Husnul Mawaddah

190212035

ABSTRAK

Nama	: Husnul Mawaddah
NIM	: 190212035
Fakultas/Prodi	: Tarbiyah dan Keguruan/Pendidikan Teknologi Informasi
Judul	: Analisis Dan Perbandingan Klasifikasi *Hate Speech* Menggunakan Metode Fitur Manual *SVM* Dan *BERT*
Bidang Peminatan	: Teknik Komputer dan Jaringan
Jumlah Halaman	: 86
Pembimbing I	: Hendri Ahmadian, S.Si., M.I.M
Pembimbing II	: Ridwan, S.ST., M.T
Kata Kunci	: *SVM*, *BERT*, klasifikasi, *hate speech*, media sosial *X (Twitter)*

Media sosial menjadi komunikasi utama seiring pertumbuhan populasi global. Di Indonesia, *X (Twitter)* menjadi *platform* aktif dengan tantangan serius terkait *hate speech*. Dalam penelitian ini, metode klasifikasi seperti *Support Vector Machine (SVM)* dan *Bidirectional Encoder Representations from Transformers (BERT)* diterapkan untuk mengatasi permasalahan yang terjadi. Penelitian ini menggunakan pendekatan kuantitatif untuk membandingkan kinerja metode klasifikasi *SVM* dan *BERT*. Evaluasi yang dilakukan menggunakan metrik-metrik seperti akurasi, presisi, *recall*, dan *F1-Score*, yang merupakan alat untuk mengukur seberapa baik kedua metode tersebut dalam mengidentifikasi konten ujaran kebencian di *platform* media sosial *X (Twitter)*. Penelitian ini bertujuan membandingkan performa *SVM* dengan fitur manual dan *BERT* dengan pemrosesan bahasa alami *end-to-end* menggunakan data yang dihasilkan oleh evaluasi, sehingga dapat disimpulkan metode mana yang memiliki kinerja lebih baik. Data yang digunakan mencakup 13169 data ujaran kebencian dari *X (Twitter)*. Hasil menunjukkan perbedaan signifikan, diketahui bahwa *BERT* mencapai akurasi 98%, sedangkan *SVM* hanya 81%. *BERT* juga menunjukkan presisi, *recall*, dan *F1-Score* signifikan masing-masing 96%, 100%, dan 98%. Ini menandakan metode *BERT* memiliki performa lebih baik dari *SVM* dalam mengenali dan membedakan ujaran kebencian.

KATA PENGANTAR

Dengan limpahan rahmat dan petunjuk Allah SWT, skripsi berjudul **"Analisis Dan Perbandingan Klasifikasi *Hate Speech* Menggunakan Metode Fitur Manual *SVM* Dan *BERT*"** telah berhasil diselesaikan. Shalawat serta salam senantiasa tercurahkan kepada Nabi Muhammad SAW, kepada keluarganya, serta sahabatnya yang telah berjuang mewujudkan kemajuan agama Islam.

Skripsi ini merupakan bagian dari persyaratan untuk menyelesaikan studi pada jenjang Strata 1 dan meraih gelar Sarjana Pendidikan (S.Pd) pada Program Studi Pendidikan Teknologi Informasi, Fakultas Tarbiyah dan Keguruan, Universitas Islam Negeri Ar-Raniry Darussalam Banda Aceh. Bimbingan serta dukungan dari berbagai pihak sangat berarti dalam penulisan skripsi ini. Dengan tulus, penghargaan dan terima kasih disampaikan kepada semua yang turut membantu dalam proses penyusunan skripsi, terutama kepada:

1. Yang teristimewa kepada kedua orang tua yaitu Ayahanda Syarifuddin dan Ibunda tercinta Nurmaila Wardati, penulis ucapkan beribu terima kasih telah mengasuh, mendidik, membimbing, membina dan memberikan semangat serta doa yang tulus setiap saat. Terima kasih atas motivasi dan semangat tiada henti hingga penulis dapat menyelesaikan studi hingga sarjana.
2. Bapak Hendri Ahmadian S.Si., M.I.M, Bapak Ridwan, S.ST., M.T. dan Bapak Bustami, M.Sc selaku dosen pembimbing yang telah meluangkan

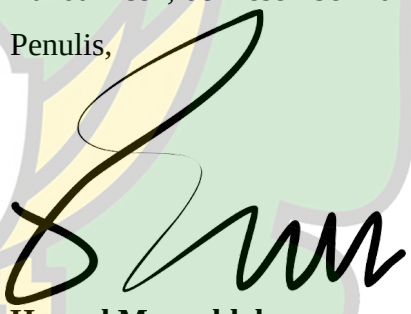
waktu dan mengarahkan penulis untuk membimbing dalam menyelesaikan skripsi.

3. Prof. Safrul Muluk, S.Ag. M.A., Ph.D selaku Dekan Fakultas Tarbiyah dan Keguruan Universitas Islam Negeri Ar-Raniry Banda Aceh.
4. Ibu Mira Maisura, M.Sc selaku Ketua Program Studi Pendidikan Teknologi Informasi dan Bapak Ridwan, S.ST., M.T selaku Sekretaris Prodi Pendidikan Teknologi Informasi.
5. Bapak Ibu Dosen dan Staf pengajar Prodi Pendidikan Teknik Informasi yang telah mau membagi ilmu dan membekali dalam berbagai ilmu pengetahuan sehingga penulis dapat menyelesaikan skripsi ini.
6. Kepada dua adik saya Nanda dan Mafaza, sahabat-sahabat saya Reyy, Susi, Najimah, Tiara, Hanim, Kakek, Pon, Isra, Kholil, Nanat dan Teman-teman mahasiswa Prodi Pendidikan Teknologi Informasi angkatan 2019 yang telah memotivasikan penulis selama penyusunan skripsi.
7. Kepada grup chat Tempat Berbagi, Panitia Sembilan, dan Ppkpm Keude Aceh yang banyak membagi informasi dan kebaikannya dalam membantu penyusunan skripsi penulis.
8. Terimakasih juga kepada karakter yang selalu memotivasi Luffy, Naruto, Asta, Geto dan Gojo yang juga telah membuat penulis bersemangat dalam menyelesaikan skripsi ini.
9. Untuk diri sendiri terimakasih juga telah berjuang dan menyelesaikan semuanya hingga sampai ke tahap ini, terimakasih banyak.

Meskipun telah berusaha menyelesaikan skripsi ini sebaik mungkin, penulis menyadari bahwa skripsi ini masih ada kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun dari para pembaca guna menyempurnakan segala kekurangan dalam penyusunan proposal skripsi ini. Akhir kata, penulis berharap semoga skripsi ini berguna bagi para pembaca dan pihak-pihak lain yang berkepentingan. Semoga Allah SWT meridhai penulisan ini dan senantiasa memberikan Rahmat dan hidayah-Nya kepada kita semua. Aamiin ya rabbal ‘alamin.

Banda Aceh, 08 Desember 2023

Penulis,



Husnul Mawaddah

DAFTAR ISI

HALAMAN SAMPUL JUDUL

LEMBAR PENGESAHAN PEMBIMBING

LEMBAR PENGESAHAN PENGUJI SIDANG SKRIPSI

LEMBAR PERNYATAAN KEASLIAN

ABSTRAK ..... **iii**

KATA PENGANTAR ..... **iv**

DAFTAR ISI ..... **vii**

DAFTAR TABEL ..... **ix**

DAFTAR GAMBAR ..... **x**

BAB I PENDAHULUAN **1**

1.1 Latar Belakang 1

1.2 Rumusan Masalah 5

1.3 Tujuan Penelitian 5

1.4 Batasan Penelitian 6

1.5 Manfaat Penelitian 6

1.6 Relevansi Penelitian Terdahulu 7

1.7 Sistematika Penulisan 8

BAB II LANDASAN TEORITIS **10**

2.1 Media Sosial *X (Twitter)* 10

2.2 Definisi *Hate speech* 11

2.3 *Artficial Inteligence (AI)* 11

2.4 *Text Mining* 12

2.5 *Text Preprocessing* 13

2.6 *Support Vector Machine (SVM)* 14

2.7 *Deep Learning* 15

2.8 *Transformer* 16

2.9 *Bidirectional Encoder Representations from Transformers (BERT)* 17

2.10 *Confusion Matrix* 17

2.11 *Tools* 18

BAB III METODE PENELITIAN 21
3.1 Jenis Penelitian 21
3.2 Tahapan Penelitian 21
3.3 Metode Pengumpulan Data 22
3.4 Pre-processing 23
3.5 Pengembangan Model 24
3.6 Implementasi Model 25
3.7 Rancangan Penelitian 25
3.8 Teknik Analisis Data 26
3.9 Evaluasi Peforma 27
3.10 Alat dan Bahan Penelitian 27
BAB IV HASIL DAN PEMBAHASAN 29
4.1 Pengumpulan Dataset 29
4.2 *Pre-processing* Data 31
4.3 Analisis Hasil dari Implementasi 35
4.3.1 Klasifikasi dengan *SVM* 35
4.3.2 Klasifikasi dengan *BERT* 39
4.4 Perbandingan Hasil dan Performa antara *SVM* dan *BERT* 43
BAB V PENUTUP 45
5.1 Kesimpulan 45
5.2 Saran 46
DAFTAR PUSTAKA 47
LAMPIRAN 50
RIWAYAT HIDUP PENULIS 74

DAFTAR TABEL

Tabel 1. 1 Penelitian Terdahulu 7
Tabel 3. 1 Sub Kategori HS 24
Tabel 4. 1 Jumlah Post (Tweet) Berdasarkan Label 31
Tabel 4. 2 Langkah-langkah Konversi Teks ke Representasi TF-IDF..... 34
Tabel 4. 3 Contoh Tokenized Data 35
Tabel 4. 4 Hasil Akurasi SVM..... 37
Tabel 4. 5 Hasil Akurasi BERT 41
Tabel 4. 6 Hasil Accuracy, Precision, Recall dan F-1 dari Kedua Model 43

DAFTAR GAMBAR

Gambar 2. 1 Hyperplane 14
Gambar 2. 2 Model Transformer 16
Gambar 2. 3 Rumus Confusion Matrix 18
Gambar 3. 1 Flowchart Alur Penelitian 22
Gambar 3. 2 Flowchart Proses Pembangunan Model 26
Gambar 4. 1 Contoh Dataset 29
Gambar 4. 2 Plot HS 30
Gambar 4. 3 Tahap Pre-processing dengan SVM 31
Gambar 4. 4 Hasil dari Apply Pre-processing 32
Gambar 4. 5 Data Column Dataset 32
Gambar 4. 6 Algoritma TF-IDF pada SVM 33
Gambar 4. 7 Hasil Akurasi Menggunakan Metrik 36
Gambar 4. 8 Hasil Akurasi dengan Best Tuning 36
Gambar 4. 9 Confusion Matrix SVM 38
Gambar 4. 10 Pie chart Hasil Klasifikasi SVM 38
Gambar 4. 11 Parameter Training loss Uji Coba Pertama 40
Gambar 4. 12 Parameter Training loss Uji Coba Kedua 40
Gambar 4. 13 Confusion Matrix BERT 41
Gambar 4. 14 Pie chart Hasil Klasifikasi BERT 42

BAB I
PENDAHULUAN

1.1 Latar Belakang

Menurut survei yang dilakukan oleh *Hootsuite*, populasi/jumlah penduduk di seluruh dunia mencapai 8,01 miliar menuju tahun 2023. Pengguna Internet pun ikut meningkat pada tahun 2022 dari 4,95 miliar menjadi 5,16 miliar di tahun 2023. Umumnya aktivitas yang dilakukan adalah pada media *social*. Beberapa media sosial yang paling sering digunakan adalah *WhatsApp, Instagram, Facebook, X (Twitter), Youtube, TikTok,* dan *Telegram*. Jejaring sosial memungkinkan komunikasi instan tanpa terbatas oleh jarak dan waktu, contohnya seperti *X (Twitter)* [1].

Berdasarkan Direktur Jenderal Sumber Daya Perangkat Pos dan Informatika (SDPP) Kementerian Komunikasi dan Informatika menyatakan bahwa Indonesia tercatat sebagai negara kelima yang paling banyak menggunakan *X (Twitter)* [2], yakni sebanyak 24 juta pengguna [3]. Kini seseorang dalam menyampaikan pendapat di dunia maya sangatlah mudah, akibatnya permasalahan terkait ujaran kebencian pun sangat rentan terjadi. Dalam mengatasi permasalahan ini, pemerintah menerbitkan Undang-Undang Informasi dan Transaksi Elektronik (UU ITE) tentang tindakan ujaran kebencian.

Menurut laporan dari Badan Reserse Kriminal Kepolisian Negara Republik Indonesia (Bareskrim Polri), pada tahun 2016, tercatat setidaknya ada 150 kasus ujaran kebencian yang dilaporkan setiap bulannya [2]. Dilansir oleh HotGrid.ID pada Minggu, 11 Juni 2023, salah satu kasus pada tahun 2019 dengan *hashtag*

#JusticeForAudrey sempat viral dan mendapatkan perhatian luas di *platform* media sosial *X (Twitter)*. Kasus ini berhubungan dengan Audrey, seorang remaja perempuan Indonesia yang menjadi korban pelecehan seksual daring. Kasus tersebut memicu tanggapan warganet yang menggunakan *hashtag #JusticeForAudrey* untuk mengecam tindakan pelecehan tersebut. Namun, pada perkembangan selanjutnya, terungkap bahwa ada ketidakselarasan antara pengakuan Audrey dan hasil visum yang menyatakan kondisi jasmaninya sehat. Hasil persidangan memvonis tiga pelaku bersalah. Nabila Echa awalnya tertuduh sebagai pem-*bully* dalam kasus *#JusticeForAudrey*, yang mengakibatkan dia mengalami cibiran dan hujatan dari masyarakat di *platform X (Twitter)* pada tahun 2019. Dampak dari situasi ini sangat terasa pada kesejahteraan mentalnya dan opini publik terhadap dirinya, merusak nama baik, dan memiliki efek jangka panjang. Setelah fakta-fakta terungkap, opini publik mengalami perubahan, yang secara tegas menyoroti bagaimana *hate speech* memengaruhi pandangan awal masyarakat terhadap suatu peristiwa dan dampaknya pada individu serta lingkungan mereka.

Dalam upaya mengatasi masalah ini agar tindakan ujaran kebencian lainnya tidak kembali terjadi, masyarakat dapat mengambil beberapa cara. Pertama, melalui edukasi nilai-nilai toleransi dan etika berkomunikasi di sekolah serta melalui kampanye kesadaran publik tentang dampak buruk ujaran kebencian. Kedua, dengan penguatan hukum yang menjaga keseimbangan antara tindakan tegas terhadap pelaku ujaran kebencian dan perlindungan terhadap kebebasan berbicara. Ketiga, melalui kerjasama dengan *platform* media sosial untuk mengembangkan algoritma deteksi dan pelaporan konten ujaran kebencian.

Bekerjasama dengan *platform* media sosial untuk mengatasi ujaran kebencian menjadi fokus penting karena luasnya skala penggunaan media sosial, keterbatasan sumber daya internal lembaga penegak hukum, serta kemampuan teknologi dan keahlian *platform* dalam mengimplementasikan algoritma cerdas untuk pendeteksian konten berbahaya. Algoritma cerdas seperti Klasifikasi *Natural Language Processing (NLP),* dapat digunakan untuk mendeteksi otomatis ujaran kebencian. *NLP* memahami konteks kalimat, membedakan bahasa kasar, dan mengidentifikasi ancaman sehingga konten berbahaya dapat dihapus secara efisien. Pengembangan algoritma *NLP* dapat terus ditingkatkan melalui pembelajaran mesin dengan pertumbuhan data, membantu menganalisis tren ujaran kebencian, juga memberi peluang untuk pencegahan yang lebih baik di masa depan [4].

Penelitian mengenai *NLP* pernah dilakukan Kikye Martiwi Sukiakhy dkk, di mana dalam penelitiannya tentang aspek Klasifikasi dan Visualisasi dalam konteks *Cyberbullying* peneliti menggunakan metode *SVM* untuk keperluan klasifikasi. Hasil penelitian menunjukkan bahwa sistem klasifikasi data mencapai akurasi rata-rata sebesar 98.04%, presisi rata-rata 90.33%, *recall* rata-rata 90.73%, dan *f-score* rata-rata 89.41% [5]. Penelitian lain oleh Karimah dkk dengan metode yang sama membahas tentang klasifikasi ujaran kebencian yang ditulis dalam bahasa Indonesia di *platform X (Twitter)*. Hasil eksperimen menunjukkan bahwa pengklasifikasian menggunakan *SVM* dan *Classifier Chains* tanpa *stemming*, penghapusan *stopword*, dan terjemahan memberikan akurasi sebesar 74.88%, nilai parameter regularisasi sebesar 10, dan nilai *gamma* sebesar 0.1 [6]. Penelitian-penelitian tersebut menunjukkan potensi dan hasil yang positif dalam tugas-tugas

yang diteliti. Akan tetapi, dalam metode ini waktu yang dibutuhkan untuk melakukan ekstraksi fitur secara manual bisa sangat memakan waktu dan melelahkan.

Metode lain yang digunakan untuk *NLP* pernah dibahas oleh Alan Tusa Bagus W dan Dhomas Hatta Fudholi, yaitu menggunakan metode *BERT* untuk klasifikasi emosi pada teks yang diekspresikan oleh manusia di media *social X (Twitter)*. Penelitian tersebut menghasilkan nilai akurasi sebesar 89.2% dengan sembilan kelas emosi pada data *training*, dan menghasilkan akurasi 76 % [7]. Riset lainnya dengan pendekatan metodologi yang serupa juga pernah disajikan oleh Hind Saleh dkk, yang meneliti tentang deteksi ujaran kebencian di *platform* media sosial daring. Hasil eksperimen menunjukkan representasi kata domain-spesifik menggunakan model berbasis *LSTM* mencapai skor f1 93%, sedangkan *BERT* mencapai 96% pada dataset ujaran kebencian yang seimbang dari berbagai sumber data [8]. Dari kedua penelitian tersebut menunjukkan bahwa *BERT* memiliki keunggulan dalam pemahaman konteks dan mendeteksi ujaran kebencian akan tetapi, waktu yang dibutuhkan untuk melatih model *BERT* bisa cukup lama. Model ini adalah model yang kompleks dan membutuhkan sumber daya komputasi yang cukup besar untuk melatihnya dengan data yang besar.

Oleh karena itu, penelitian ini bertujuan untuk mengevaluasi serta membandingkan kinerja metode klasifikasi *hate speech* menggunakan fitur manual *SVM* dengan model *BERT* yang menerapkan pemrosesan bahasa alami secara *end-to-end*. Fokusnya adalah pada analisis akurasi, presisi, *recall*, dan *F1-Score* dari kedua metode ini. Selain itu, penelitian ini juga bertujuan untuk mengidentifikasi

perbedaan yang signifikan dalam kemampuan metode *SVM* manual dan metode *BERT* dalam mengklasifikasikan *hate speech* di *platform* media sosial *X (Twitter)*.

1.2 Rumusan Masalah

Bedasarkan dari latar belakang masalah diatas, maka masalah dalam penelitian ini dirumuskan sebagai berikut:

1. Bagaimana kinerja metode klasifikasi *hate speech* dengan menggunakan fitur manual dan *SVM* dalam hal akurasi, presisi, dan *recall* dibandingkan dengan metode *BERT* yang menggunakan pemrosesan bahasa alami secara *end-to-end*?
2. Apakah terdapat perbedaan signifikan dalam kemampuan metode fitur manual *SVM* dan metode *BERT* dalam mengklasifikasikan *hate speech* di *platform* media sosial *X (Twitter)*?

1.3 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah:

1. Menganalisis dan membandingkan kinerja metode klasifikasi *hate speech* menggunakan fitur manual *SVM* dengan metode *BERT* yang menggunakan pemrosesan bahasa alami secara *end-to-end* dalam hal akurasi, presisi, *recall* dan *F1-Score*.
2. Mengetahui apakah terdapat perbedaan signifikan dalam kemampuan metode fitur manual *SVM* dan metode *BERT* dalam mengklasifikasikan *hate speech* di *platform* media sosial *X (Twitter)*.

1.4 Batasan Penelitian

Adapun batasan masalah pada penelitian ini, yaitu:

1. Bahasa pemrograman yang dipakai pada penelitian ini berupa *Python*
2. Menggunakan *Google Colab* sebagai media untuk menjalankan program.
3. Data yang digunakan merupakan dataset *X (Twitter)* yang diambil dari https://*GitHub*.com/okkyibrohim/id-multi-label-hate-speech-and-abusive-language-detection.git
4. Penelitian ini memfokuskan perbandingan antara metode klasifikasi *hate speech* menggunakan fitur manual dan *SVM* dengan metode *BERT*. Metode lainnya tidak dieksplorasi dalam penelitian ini, dan pembandingan hanya berfokus pada metode yang telah disebutkan.

1.5 Manfaat Penelitian

Adapun manfaat dari penelitian ini adalah:

1. Dengan membandingkan fitur manual *SVM* dan model *BERT*, penelitian ini memberikan pemahaman yang mendalam tentang sejauh mana metode klasifikasi *hate speech* menggunakan fitur manual dan *SVM* dapat bersaing dengan metode *BERT* yang menggunakan pemrosesan bahasa alami secara *end-to-end.*
2. Hasil penelitian ini memiliki potensi untuk berkontribusi pada penanganan ujaran kebencian di media sosial, membantu *platform* dan pengguna mengidentifikasi konten berbahaya dengan lebih efisien.

3. Memberikan pemahaman yang lebih mendalam tentang seberapa efektif metode-metode tersebut dalam menghadapi konten berbobot kebencian yang khususnya tersebar di *platform* media sosial *X (Twitter)*.
4. Memberikan pandangan penting tentang peran dan relevansi metode klasifikasi dalam mengatasi *hate speech* di *platform* sosial khusus.

1.6 Relevansi Penelitian Terdahulu

Tabel 1. 1 Penelitian Terdahulu

PENELITI	METODE	KASUS	AKURASI
Hind Saleh, Areej Alhothali & Kawthar Moria (2023)	*BERT*	Detection of *Hate speech* using *BERT* and *HateSpeech Word Embedding with Deep Model*	96%
Adine Nayla, Casi Setianingsih & Burhanuddin Dirgantoro (2023)	*BERT*	Deteksi *Hate speech* Pada *Twitter* Menggunakan Algoritma *BERT*	78.77%
Alan Tusa Bagus W & Dhomas Hatta Fudholi (2023)	*BERT*	Klasifikasi Emosi Pada Teks Dengan Menggunakan Metode *Deep Learning*	89.2%

PENELITI	METODE	KASUS	AKURASI
Kikye Martiwi Sukiakhy, Junidar & Sri Azizah Nazhifah (2023)	*SVM*	Penggunaan Data Pada *Twitter* Dalam Klasifikasi Dan Visualisasi *Cyberbullying* Dengan Algoritma *SVM* (*Support Vector Machine*)	98.04%
Brian Wijaya & Viny Christanti Mawardi (2023)	*SVM*	Pendeteksi Ujaran Kebencian Pada *Platform* Media Sosial *Twitter* Menggunakan *Support Vector Machine*	95%
Karimah Mutisari Hana, Adiwijaya, Said Al Faraby & Arif Bramantoro (2021)	*SVM*	Multi-label *Classification* of *Indonesian Hate speech* on *Twitter* Using *Support Vector Machine*s	74.88%

1.7 Sistematika Penulisan

Penelitian ini mengikuti struktur umum dalam penelitian ilmiah dengan mengikuti tahapan yang terorganisir. Penelitian ini akan terdiri dari 5 bab sebagai berikut:

BAB I PENDAHULUAN

Bagian ini menjelaskan latar belakang, perumusan masalah, tujuan, manfaat penelitian, dan struktur penulisan yang menjadi dasar bagi penelitian ini.

BAB II LANDASAN TEORI

Bagian ini secara rinci menjelaskan penelitian terkait serta teori-teori yang menjadi dasar acuan dalam penelitian ini.

BAB III METODE PENELITIAN

Bagian ini merangkum teknik-teknik dalam merancang metode, termasuk alur penelitian, sumber dan jenis data yang digunakan, serta metode analisis data untuk mencapai hasil yang diinginkan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini merangkum bagaimana masalah diselesaikan berdasarkan metodologi yang dipilih dan proses implementasi metode tersebut, serta mencakup hasil pengujian dan evaluasi dari penelitian ini.

BAB V PENUTUP

Bab ini mencakup rangkuman hasil dan kesimpulan dari penelitian, serta memberikan saran untuk perbaikan jika penelitian ini akan dikembangkan lebih lanjut.

BAB II
LANDASAN TEORITIS

2.1 Media Sosial *X (Twitter)*

Mengutip dari halaman resmi *X (Twitter)*, *platform* ini dijelaskan sebagai "*an open service that's home to a world of diverse people, perspectives, ideas, and information."* Ini dirancang untuk memungkinkan koneksi dan komunikasi antara teman, keluarga, dan rekan kerja melalui pertukaran pesan yang instan. Pengguna *X (Twitter)* memiliki kemampuan untuk mengirim pesan yang disebut "*Post (Tweet)*" atau "cuitan," yang dapat berisi teks, gambar, video, dan tautan, yang diterima oleh para pengikut mereka. Sebagai bentuk media sosial dalam format *microblogging*, *X (Twitter)* memberikan layanan kepada penggunanya untuk membagikan aktivitas dan pandangan mereka dalam batasan 280 karakter [9]. Alasan *X (Twitter)* sering digunakan untuk analisis sentimen adalah karena informasi menyebar dengan sangat cepat di *X (Twitter)*, sehingga memudahkan untuk mengetahui pendapat orang tentang produk dan peristiwa di sekitarnya [10].

Aktor digital adalah program komputer otomatis yang beroperasi di *X (Twitter)*. Tiga jenis utama dari aktor digital ini adalah pengguna spam, pengguna berita, dan layanan viral/pemasaran. Pengguna spam adalah *spammer* yang secara otomatis memposting *Post (Tweet)* jahat dengan tujuan memperoleh pengikut. Pengguna berita merupakan layanan otomatis yang memposting informasi dari sumber berita atau RSS *feed*. Sedangkan layanan viral/pemasaran merujuk pada penggunaan bot pintar untuk menyebarluaskan informasi dan melakukan tugas pemasaran. Secara keseluruhan, teks ini mengilustrasikan perbedaan antara pengguna nyata dan aktor

digital dalam ekosistem *X (Twitter)*, serta menggambarkan tipe-tipe pengguna di kedua kategori tersebut [11].

2.2 Definisi *Hate speech*

Menurut Pranowo (2009), ujaran kebencian didefinisikan sebagai tindakan seseorang yang tidak sopan, dan selalu disebabkan oleh beberapa faktor. Artinya, (1) tuturannya selalu didorong emosi, dan (2) pendengarnya selalu emosional. (3) selalu berprasangka buruk dan buruk terhadap lawan bicara, dan (4) selalu membela pendapat [12]. Ujaran kebencian adalah ujaran yang menyerang individu atau kelompok karena jenis kelamin, suku, agama, ras, disabilitas, atau orientasi seksual. Perkataan yang mendorong kebencian juga membawa ancaman pada tingkat tertentu. Semakin tinggi tingkat ancaman ujaran kebencian maka semakin cepat penyebarannya, mulai dari konflik antar individu hingga konflik antar kelompok [13].

2.3 *Artficial Inteligence (AI)*

Kecerdasan buatan (AI) adalah kecerdasan buatan yang meniru perilaku manusia untuk membantu dalam bekerja. Kaplan dan Haenlein mendefinisikan *AI* sebagai sistem yang bisa menganalisis data eksternal, belajar darinya, dan mencapai tujuan melalui penyesuaian fleksibel. *AI* menggabungkan data, pemrosesan, dan algoritma cerdas untuk memahami pola data secara otomatis. Bidang *AI* mencakup pembelajaran mesin, jaringan saraf tiruan, komputasi kognitif, visi komputer, dan pemrosesan bahasa alami. Ini membentuk pemahaman tentang kecerdasan buatan dalam berbagai konteks [14].

2.4 Text Mining

Text Mining diartikan sebagai penambangan data berupa teks yang sumber datanya biasanya diambil dari dokumen dan tujuannya adalah untuk menemukan kata-kata yang dapat mewakili isi dokumen sehingga dapat dilakukan analisis hubungan antar dokumen. Tujuan dari *Text Mining* adalah mengubah teks menjadi atribut numerik yang kemudian dapat digunakan dalam algoritma Data Mining. Gupta dan Lehal (2009) dalam penelitiannya menyelidiki pentingnya penggunaan *Text Mining*. Dalam penelitian itu disebutkan bahwa sebagian besar informasi (lebih dari 80%) disimpan dalam bentuk teks dan *Text Mining* dianggap memiliki potensi nilai bisnis yang tinggi. Pengetahuan dapat diperoleh dari banyak sumber informasi, namun teks tidak terstruktur masih menjadi sumber pengetahuan terbesar yang tersedia. Allahyar dkk. (2017) dalam penelitiannya juga membahas teks tersebut pertambangan. Dalam penelitian ini ditemukan bahwa jumlah dokumen yang dihasilkan per hari meningkat secara signifikan. Artinya, sebagian besar teks tidak terstruktur tidak dapat diproses dan dikenali dengan mudah oleh komputer. Oleh karena itu, diperlukan teknik dan algoritma yang efisien dan efektif untuk menemukan pola yang berguna. Penelitian ini juga menjelaskan beberapa teknik *Text Mining*, termasuk klasifikasi, *clustering* dan ekstraksi informasi [15].

2.5 Text Preprocessing

Data set harus melalui tahap *Text Preprocessing* terlebih dahulu karena data set tidak bisa digunakan tanpa melalui tahap pengelolaan data. *Text Preprocessing* adalah suatu proses pengelolaan data set sebelum data tersebut diproses. Pada kenyataannya, masih banyak data set yang tidak bersih seperti kesalahan sistem saat pencatatatan sehingga terjadinya data duplikat. Data yang belum diolah atau data tidak bersih kategorinya seperti format data yang tidak beraturan, adanya data kosong, tipe data yang berbeda-beda, adanya atribut yang tidak penting, dan lain sebagainya. Semakin bersih pra proses yang dilakukan, maka kemungkinan besar hasil data tersebut semakin akurat [16]. Adapun tahapan *Text Preprocessing* adalah sebagai berikut:

- *Transform Cases*

  Pertama-tama, dilakukan langkah Transformasi Kasus untuk mengubah semua huruf menjadi huruf kecil (*lowercase*). Ini bertujuan untuk mengurangi variasi dalam penulisan teks.

- *Punctuation Removal*

  Langkah selanjutnya adalah penghapusan tanda baca, yang bertujuan untuk menghilangkan semua tanda baca dalam teks, seperti titik, koma, tanda tanya, dan tanda seru. Hal ini dilakukan untuk menghapus karakter yang tidak relevan dan mengurangi variasi dalam penulisan.

- *Tokenize*

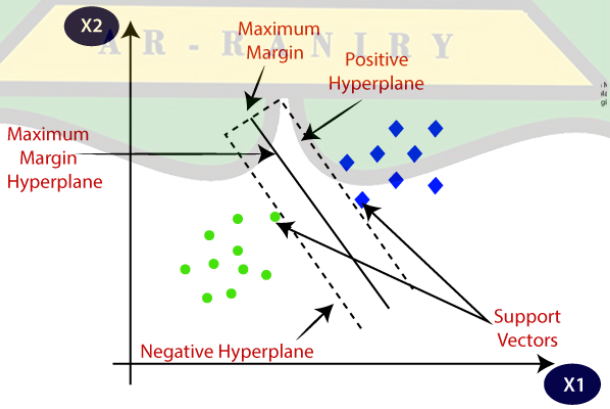
Tokenisasi adalah proses membagi teks dalam setiap dokumen menjadi rangkaian token/*term*. Hasilnya adalah setiap kata dalam teks direpresentasikan sebagai token/*term*.

- *Stopword Removal*

  Penghapusan *Stopword* dilakukan dengan memeriksa setiap token dan menghapusnya jika token tersebut ada dalam daftar *stopword*. Daftar *stopwords* yang digunakan dalam penelitian ini adalah daftar *stopwords* bahasa Indonesia [15].

2.6 Support Vector Machine (SVM)

Support Vector Machine merupakan metode yang bekerja berdasarkan prinsip minimalisasi risiko struktural (SRM) dan bertujuan untuk menemukan *Hyperplane* yang memisahkan dua kelas pada ruang input. *Hyperplane* ditentukan dengan mencari titik maksimum dan menghitung tepi *Hyperplane*. *Margin* adalah jarak dari *Hyperplane* ke data terdekat di setiap kelas, dan data yang paling dekat dengan *Hyperplane* disebut *support vector* [17].



Gambar 2. 1 *Hyperplane*

Gambar 2.1 menunjukkan bagaimana *Hyperplane* membagi data menjadi dua kelas. Jika ada data yang masuk dan berada di sebelah kiri baris, semua data yang masuk akan ditempatkan di grup hijau. Jika data berada di sebelah kanan garis, maka data akan ditempatkan di grup biru. Secara teori, *SVM* digunakan saat melakukan klasifikasi dua kelas. Klasifikasi yang ada ke dalam banyak kelas dapat dikembangkan lebih lanjut dengan menggunakan dua strategi, strategi pertama "satu lawan satu" dan strategi kedua "satu lawan semua" [5].

2.7 Deep Learning

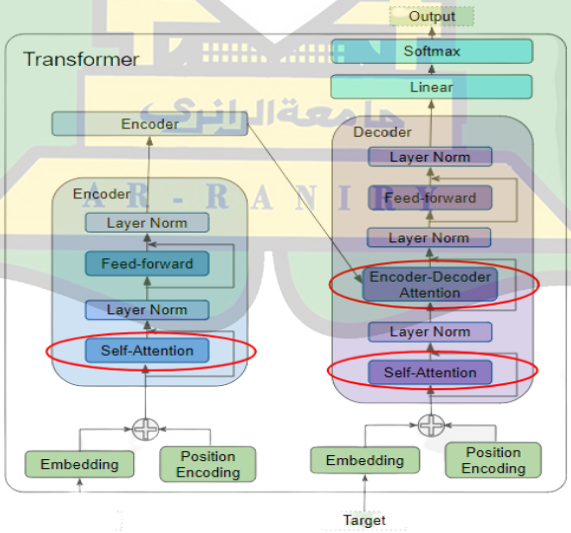
Deep Learning adalah cabang dari *machine learning* yang mengambil inspirasi dari konsep struktur otak manusia. Konsep ini diterapkan dalam bentuk *Artificial Neural Networks (ANN)*, yang merupakan jaringan saraf tiruan dengan tiga lapisan atau lebih. Kemampuannya meliputi pembelajaran dari data besar dan penyesuaian terhadap situasi, serta penerapan pada tantangan-tantangan kompleks yang sulit diatasi oleh algoritma *machine learning* konvensional [4].

Sementara itu, arsitektur pembelajaran mendalam seperti jaringan saraf dalam, jaringan kepercayaan mendalam, pembelajaran penguatan mendalam, jaringan saraf berulang, jaringan saraf konvolusional dan transformator memiliki aplikasi di berbagai bidang seperti visi komputer, pengenalan suara, pemrosesan bahasa alami, terjemahan mesin, dan bioinformatika. Kedokteran desain, analisis citra medis, ilmu iklim, pengujian material, program permainan papan, dan banyak lagi telah mencapai hasil yang sesuai dan terkadang melebihi kinerja manusia. Ada banyak jenis jaringan saraf dalam pembelajaran mendalam. Dan arsitektur pembelajaran mendalam didasarkan

pada jaringan ini. Yang paling populer di antaranya adalah *Recurrent Neural Networks (RNN)*, *Long Short Term Memory* (*LSTM*), dan *Coconvolutional Neural Networks (CNN)* [18].

2.8 Transformer

Awalnya, *Transformer* adalah model yang diusulkan untuk pemrosesan bahasa alami. Model ini diperkenalkan pada tahun 2017 dan telah berkembang menjadi model canggih untuk pemrosesan bahasa alami. Model transformator mengikuti konsep *encoder/decoder*. Lapisan *encoder* bertindak sebagai penerima masukan berkelanjutan. Masing-masing langkah ini bersifat autoregresif dan simbol yang dibuat sebelumnya digunakan sebagai masukan untuk iterasi berikutnya. *Transformer* adalah model pembelajaran berurutan atau mendalam yang menggunakan metode kesadaran diri dan titik waktu yang berumpuk. Setiap *blok self-aware* kemudian dihubungkan ke lapisan yang terhubung sepenuhnya dari setiap jalur *encoder* dan *decoder* [19].



Gambar 2. 2 Model Transformer

Dapat dilihat dari Gambar 2. 2 *Transformer* memiliki dua komponen utama yaitu *encoder* dan *decoder*. Tugas *Encoder* adalah membaca input dan memahami konteks dari input, sedangkan *decoder* bertugas untuk memproduksi hasil prediksi.

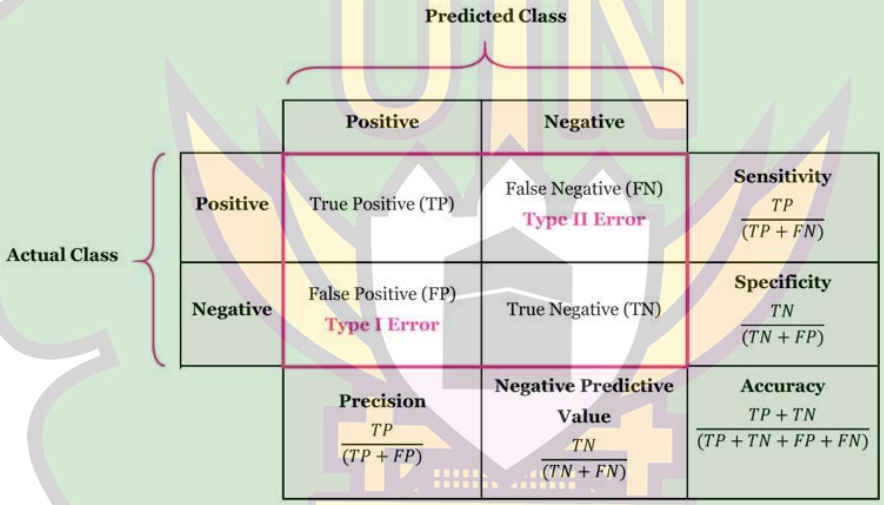
2.9 Bidirectional Encoder Representations from Transformers (BERT)

Representasi *encoder* dua arah dari *Transformer*s (biasa disebut *BERT*) adalah algoritma pembelajaran mendalam yang dikembangkan oleh Google dan masih relevan dengan *NLP* (*Natural Language Processing*). Algoritme ini merupakan intrusi ke dalam model *Transformer*, yang memproses kata dalam kalimat berdasarkan apakah ada hubungan antara kata dan keseluruhan kalimat. Algoritma *BERT* bekerja secara berbeda dibandingkan algoritma pemrosesan bahasa lainnya. *BERT* memproses kata dengan memeriksa konteks kata tersebut jika ada. Algoritma *BERT* menangani semua konteks dengan melihat pola yang muncul sebelum dan sesudah kata. *BERT* dilatih menggunakan 2,5 miliar kata dari Wikipedia dan 800 juta kata dari buku. Pelatihan *BERT* terbagi dalam dua teknik pemodelannya: *Masked Language Models (MLM)* dan *Next Sentence Prediction (NSP)*. Dalam *MLM*, kata-kata dalam korpus disembunyikan dan dilatih. Di sisi lain, *BERT NSP* memprediksi kalimat berikutnya yang cocok dengan konteks kalimat sebelumnya [20].

2.10 Confusion Matrix

Pada bidang *machine learning* dan khususnya masalah klasifikasi statistik, *Confusion Matrix,* (juga dikenal sebagai matriks kesalahan), adalah tata letak tabel khusus yang memungkinkan visualisasi kinerja suatu algoritma, yang biasanya

digunakan pada tugas *supervised learning*. Setiap baris matriks mewakili *instance* di kelas aktual sementara setiap kolom mewakili instance di kelas yang diprediksi, atau sebaliknya [18]. Untuk melihat seberapa baik pembelajaran mesin bekerja seperti yang diharapkan, *confusion matrix* berisi berbagai kinerja yang dapat diukur seperti akurasi, presisi, perolehan, spesifikasi, dan skor F1, untuk melihat seberapa baik pemodelan telah bekerja sebelumnya [16]. Dalam menghitung nilai performa model akurasi, presisi, rekal dan F1 *Score* dengan menggunakan *confusion matrix*, rumus *confusion matrix* dapat dilihat pada gambar 2.3



Gambar 2. 3 Rumus Confusion Matrix

2.11 Tools

1. Python

Python adalah bahasa pemrograman yang ditafsirkan tingkat tinggi. Filosofi desain *Python* menekankan keterbacaan kode dan banyak menggunakan indentasi. *Python* diketik secara dinamis dan dikumpulkan. Ini mendukung berbagai paradigma pemrograman, termasuk pemrograman terstruktur (terutama

prosedural), berorientasi objek, dan fungsional. Karena perpustakaan standarnya yang luas, bahasa ini sering disebut sebagai bahasa "baterai dalam baterai". *Python* adalah bahasa populer untuk pemrograman *ML* dan kecerdasan buatan. Keunggulan yang menjadikan *Python* ideal untuk pembelajaran mesin dan proyek berbasis *AI* mencakup kesederhanaan dan konsistensi, akses ke pustaka dan kerangka kerja *AI* dan pembelajaran mesin (*ML*) terkemuka, fleksibilitas, kemandirian *platform*, dan komunitas luas. termasuk. Hal ini berkontribusi pada popularitas bahasa secara keseluruhan [18].

2. Google Colaboratory

Google Colab atau *Google Collaborator* adalah produk riset Google berbasis *cloud* yang bebas digunakan. Pembuatan *Google Colab* bertujuan untuk mendorong pekerjaan yang berkaitan dengan ilmu data dan pembelajaran mesin. *Google Colab* biasanya digunakan untuk memenuhi kebutuhan *programmer* dan profesional data dengan spesifikasi tinggi. Selain itu, *Google Colab* dapat dibagikan, yang sangat mendukung kebutuhan kolaborasi antar anggota tim.

Google Colab memungkinkan pengguna menulis dan menjalankan kode *Python* tanpa instalasi perangkat lunak di komputer lokal. Menggunakan format *Jupyter Notebook*, pengguna bisa menggabungkan kode, teks, dan visualisasi dalam satu dokumen interaktif. Dengan akses gratis ke sumber daya komputasi yang meliputi *CPU, GPU, dan TPU, Colab* mempercepat komputasi dalam tugas yang kompleks. Integrasi dengan Google Drive memudahkan penyimpanan dan berbagi catatan kerja. *Platform* ini berguna bagi pengembang, ilmuwan data, dan praktisi *machine learning*. *Colab* menyediakan lingkungan nyaman untuk eksplorasi data,

pengembangan model, dan analisis data. Pengguna dapat mengimpor pustaka *Python*, menginstal dependensi, serta berkolaborasi secara *real-time*. Dukungan *GPU* dan *TPU* mempercepat komputasi dalam tugas sulit. *Google Colab* memberikan alternatif praktis untuk bekerja dengan kode *Python* dan melakukan analisis data efisien di lingkungan berbasis *cloud* [21].

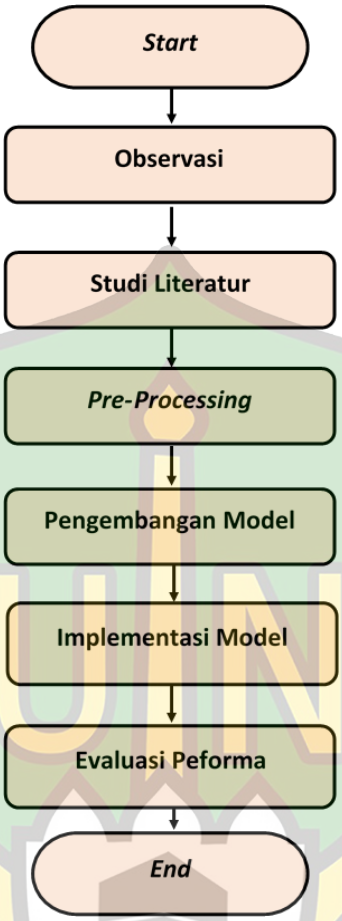
BAB III
METODE PENELITIAN

3.1 Jenis Penelitian

Dalam penelitian ini, penulis menggunakan pendekatan kuantitatif untuk membandingkan kinerja metode untuk klasifikasi *hate speech*, yaitu *SVM* dengan fitur manual dan *BERT* dengan pengolahan bahasa alami *end-to-end*. Penulis menggunakan evaluasi kuantitatif, seperti akurasi, presisi, *recall*, dan *F1-Score*, sebagai alat untuk mengukur dan membandingkan efektivitas kedua metode dalam mengidentifikasi *hate speech* di *platform* media sosial *X (Twitter)*.

3.2 Tahapan Penelitian

Bagian ini menjelaskan urutan langkah-langkah yang dilakukan oleh penulis dalam penelitian ini. Berbagai percobaan algoritmik dan tahap *Preprocessing* dilakukan selama proses pembangunan model untuk memastikan bahwa model yang dihasilkan memenuhi harapan. Berikut rincian langkah-langkah yang harus dilakukan, dapat dilihat pada gambar 3.1



Gambar 3. 1 *Flowchart* Alur Penelitian

3.3 Metode Pengumpulan Data

Penulis memperoleh metode pengumpulan data dalam penelitian ini melalui observasi, penulis melakukan identifikasi masalah penelitian yang berkaitan dengan deteksi kalimat kasar di *platform* media sosial *X (Twitter)*. Selain itu, dalam studi literatur, penulis menggali referensi-referensi yang mendukung pembentukan metode yang akan diimplementasikan dalam penelitian ini. Pengumpulan data dilakukan dengan mengambil dataset yang dibutuhkan dari *platform GitHub*. Peneliti menggali kembali data dari penelitian sebelumnya yang telah dilakukan oleh Ibrohim & Budi pada tahun 2019 dalam makalah berjudul *"Multi-label Hate speech and Abusive*

Language Detection in Indonesian X (Twitter)" yang disajikan di *Proceedings of the Third Workshop on Abusive Language Online*. Dataset yang digunakan, terdiri dari 13.169 *volume Post (Tweet),* diunduh melalui link *GitHub*.com yang tertera dalam publikasi tersebut.

3.4 Pre-processing

Setelah data terkumpul, langkah berikutnya dalam menggunakan *SVM (Support Vector Machine)* dan *BERT (Bidirectional Encoder Representations from Transformers)* adalah *pre-processing* data dan pengaturan label dataset. Pada *SVM*, *pre-processing* melibatkan langkah-langkah seperti standarisasi teks, pembuatan vektor kata, serta pemilihan fitur yang cocok untuk mewakili data yang akan dimasukkan ke dalam model *SVM*. Sementara itu, dalam konteks *BERT*, *pre-processing* melibatkan tokenisasi teks, transformasi teks menjadi token, dan penyesuaian teks agar sesuai dengan struktur input yang diinginkan oleh model *BERT*, seperti token IDs atau *attention masks*.

Setelah dataset siap, langkah berikutnya adalah melakukan proses pelabelan data. Dalam dataset yang digunakan untuk penelitian ini, label-label sudah tersedia. Untuk pemberian label, data yang kita import pada penelitian ini telah terlabeli dengan baik. Istilah-istilah yang digunakan dalam dataset adalah sebagai berikut, dapat dilihat pada table 3.1 yang mana *Hate speech* terdiri dari beberapa subkategori atau jenis ujaran kebencian yang berbeda.

Tabel 3. 1 Sub Kategori *HS*

No	Label *HS*
1	Ujaran Kebencian *(HS)*
2	Kasar
3	Ujaran Kebencian Individu (*HS_Individual)*
4	Ujaran Kebencian Kelompok *(HS_Group)*
5	Ujaran Kebencian Agama *(HS_Religion)*
6	Ujaran Kebencian Ras *(HS_Race)*
7	Ujaran Kebencian Fisik *(HS_Physical)*
8	Ujaran Kebencian Gender *(HS_Gender)*
9	Ujaran Kebencian Lainnya *(HS_Other)*
10	Ujaran Kebencian yang Lemah (*HS*_Lemah)
11	Ujaran Kebencian Sedang *(HS_Moderate)*
12	Ujaran Kebencian yang Kuat *(HS_Strong)*

Pada tabel 3.1 di atas menunjukkan daftar label-label yang digunakan untuk mengkategorikan berbagai jenis ujaran kebencian dalam konteks analisis atau klasifikasi teks.

3.5 Pengembangan Model

Operasi ini dilakukan dengan menggunakan dataset yang diambil sebelumnya dari *GitHub*. Selanjutnya, proses latih model dapat diuraikan sebagai berikut:

1. Pelatihan *SVM*

   Model *SVM* dilatih menggunakan fitur manual yang sudah dipilih. Data yang telah dipersiapkan digunakan untuk melatih model *SVM* dalam tugas klasifikasi ujaran kebencian. Proses ini melibatkan penentuan parameter *SVM* yang optimal dan pembentukan *Hyperplane* terbaik untuk memisahkan kelas ujaran kebencian dan bukan ujaran kebencian.

2. Model *BERT*

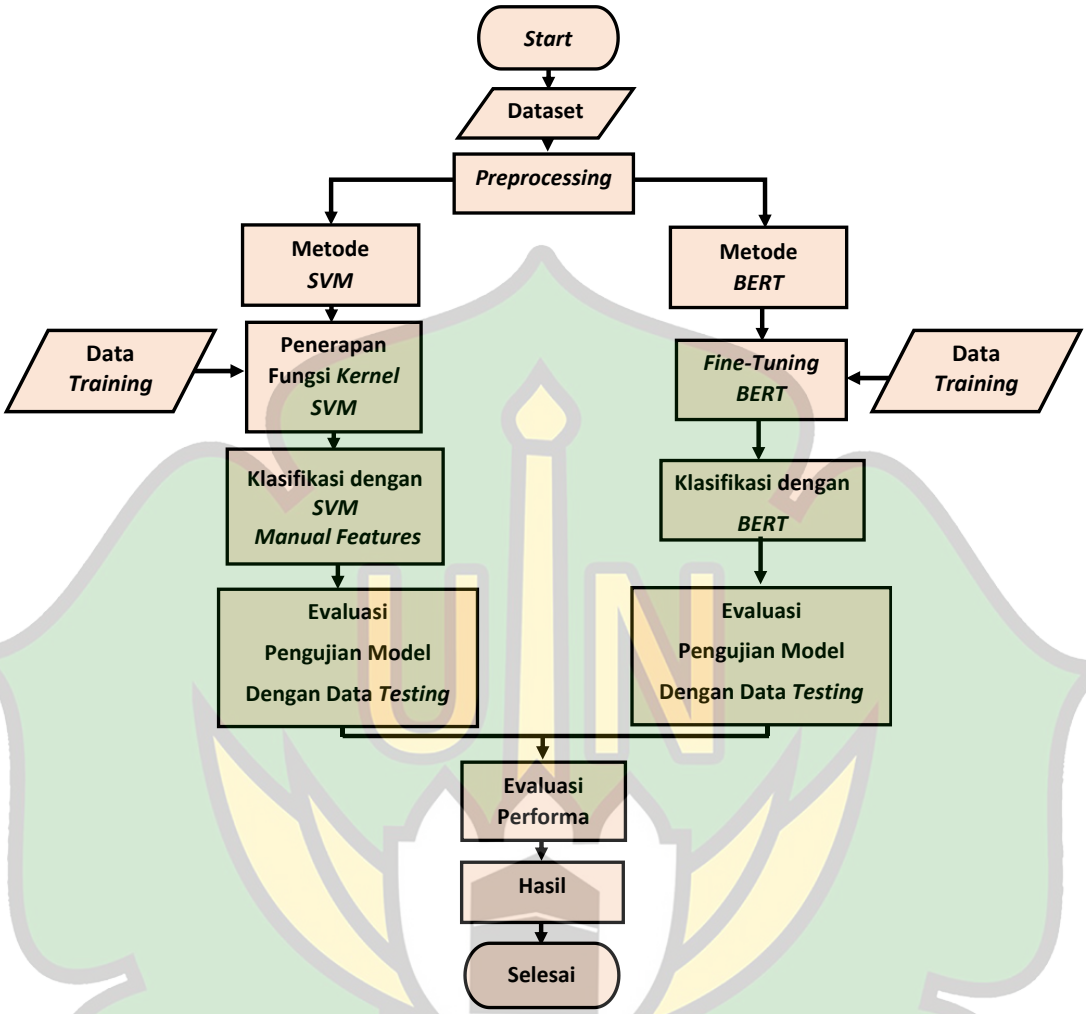
Model ini dilatih menggunakan data teks *Post (Tweet)* yang sudah diproses. Proses ini memanfaatkan mekanisme pre-*training* dan *fine-Tuning BERT* untuk menghasilkan model yang mampu memahami konteks dan nuansa dalam ujaran.

3.6 Implementasi Model

Membandingkan hasil keluaran fitur manual *SVM* dengan model *BERT* untuk klasifikasi. Proses perbandingan ini dapat melibatkan analisis performa masing-masing model secara terpisah, termasuk metode penilaian seperti akurasi, presisi, *Recall*, dan *F1-Score*. Tujuan utama adalah untuk memahami keunggulan dan kelemahan masing-masing model dalam konteks tugas klasifikasi yang diberikan.

3.7 Rancangan Penelitian

Keseluruhan proses penelitian, mulai dari tahap studi pendahuluan hingga pengujian, telah dirinci dalam kerangka penelitian yang dapat dilihat melalui gambar 3.2



Gambar 3. 2 *Flowchart* Proses Pembangunan Model

3.8 Teknik Analisis Data

Penelitian terkait klasifikasi ujaran kebencian dan analisis emosi dalam teks di *platform* media sosial melibatkan sejumlah teknik analisis data. Salah satu pendekatan umum adalah ekstraksi fitur dari teks, di mana fitur-fitur penting seperti kata kunci dan frasa diekstraksi untuk mendukung proses klasifikasi. Beberapa penelitian menerapkan metode klasifikasi seperti *SVM* dan *BERT* untuk memahami pola dan tren dalam teks yang rumit. Performa model dinilai dengan metrik seperti akurasi, presisi,

Recall, dan *F1-Score*, yang memberikan wawasan tentang kemampuan model dalam mengklasifikasikan dengan benar.

3.9 Evaluasi Peforma

Evaluasi akan dilakukan dengan hasil pengujian dari kedua model, oleh karena itu akan digunakan sejumlah metrik evaluasi seperti akurasi, presisi, *Recall*, dan *F1-Score* untuk menghitung nilai peforma. Evaluasi ini memungkinkan peneliti untuk membandingkan kinerja kedua model pada data uji independen yang belum pernah mereka lihat sebelumnya. Dengan demikian, hasil pengujian tersebut akan membantu dalam membuat keputusan akhir terkait model mana yang lebih cocok untuk tugas klasifikasi ujaran kebencian yang dijalankan oleh penulis.

3.10 Alat dan Bahan Penelitian

Peralatan Komputasi yang digunakan dalam penelitian ini adalah sebagai berikut:

- *Operating System Windows* 10, 64-bit
- *Version* 21H2 (OS 10.0.19045 Build 19045)
- *Processor* AMD A4-9120e Radeon R3
- 4 *Compute Cores* 2C+2G/2 CPUs, ~1.5GHz
- RAM 4,00 GB (3,88 GB *usable*)

Software yang digunakan penulis dalam penelitian ini adalah sebagai berikut:

- *Google Colaboratory*
- *Python* 3 type T4 *Google Compute Engine backend* (GPU)
- RAM Sistem 3.0 / 12.7 GB
- RAM GPU 9.2 / 15.0 GB
- Disk 27.4 / 78.2 GB

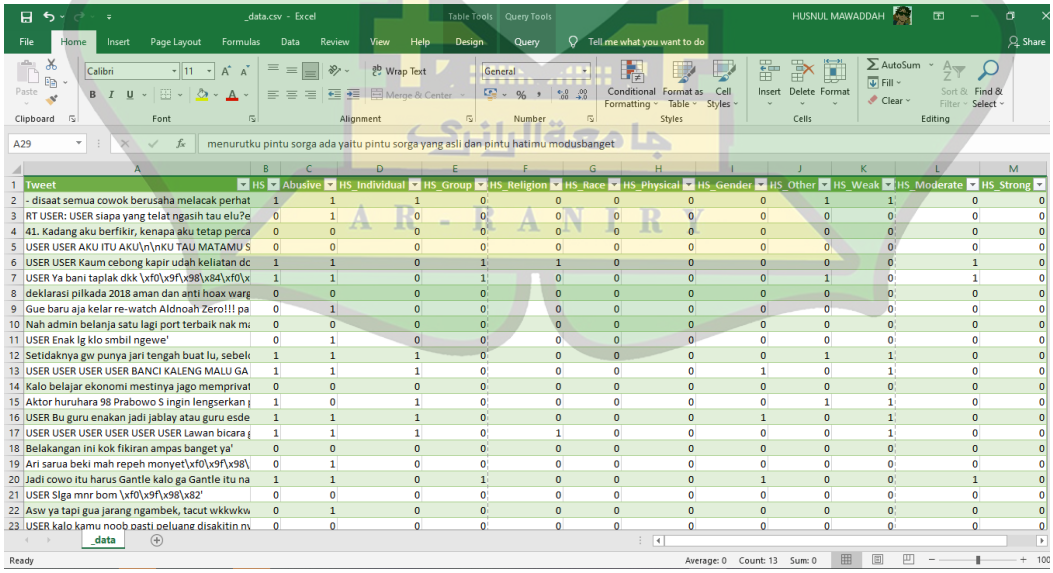
BAB IV

HASIL DAN PEMBAHASAN

Bagian ini akan djelaskan mengenai data, metode, serta hasil yang didapatkan berdasarkan uji coba yang telah dilakukan peneliti setelah melakukan percobaan dengan 2 model yang berbeda, yaitu *SVM* Fitur Manual dan *BERT*.

4.1 Pengumpulan Dataset

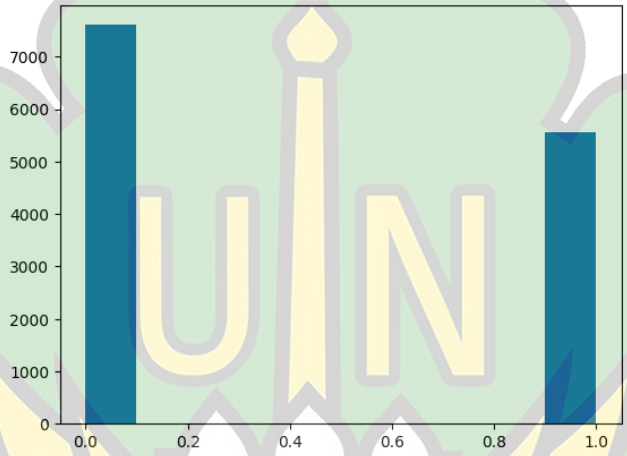
Dataset yang digunakan berasal dari sumber *GitHub* dengan tautan: https://*GitHub*.com/okkyibrohim/id-multi-label-hate-speech-and-abusive-language-detection.git. Dataset yang diambil yaitu *Multi Label Hate speech and Abusive Language Detection in Indonesian Twitter* dengan jumlah sampel data sebanyak 13,169 *Post (Tweet)*. Dataset ini memiliki 13 atribut yang dapat dilihat pada gambar 4.1 berikut.



	Tweet	HS	Abusive	HS_Individual	HS_Group	HS_Religion	HS_Race	HS_Physical	HS_Gender	HS_Other	HS_Weak	HS_Moderate	HS_Strong
2	- disaat semua cowok berusaha melacak perhat	1	1	1	0	0	0	0	0	1	1	0	0
3	RT USER: USER siapa yang telat ngasih tau elu?e	0	1	0	0	0	0	0	0	0	0	0	0
4	41. Kadang aku berfikir, kenapa aku tetap perca	0	0	0	0	0	0	0	0	0	0	0	0
5	USER USER AKU ITU AKU\n\nKU TAU MATAMU S	0	0	0	0	0	0	0	0	0	0	0	0
6	USER USER Kaum cebong kapir udah keliatan dc	1	1	0	1	1	0	0	0	0	0	1	0
7	USER Ya bani taplak dkk \xf0\x9f\x98\x84\xf0\x	1	1	0	1	0	0	0	0	1	0	1	0
8	deklarasi pilkada 2018 aman dan anti hoax warg	0	0	0	0	0	0	0	0	0	0	0	0
9	Gue baru aja kelar re-watch Aldnoah Zero!!! pa	0	1	0	0	0	0	0	0	0	0	0	0
10	Nah admin belanja satu lagi port terbaik nak ma	0	0	0	0	0	0	0	0	0	0	0	0
11	USER Enak lg klo smbil ngewe'	0	1	0	0	0	0	0	0	0	0	0	0
12	Setidaknya gw punya jari tengah buat lu, sebel	1	1	1	0	0	0	0	0	1	1	0	0
13	USER USER USER USER BANCI KALENG MALU GA	1	1	1	0	0	0	0	1	0	1	0	0
14	Kalo belajar ekonomi mestinya jago memprivat	0	0	0	0	0	0	0	0	0	0	0	0
15	Aktor huruhara 98 Prabowo S ingin lengserkan	1	0	1	0	0	0	0	0	1	1	0	0
16	USER Bu guru enakan jadi jablay atau guru esde	1	1	1	0	0	0	0	1	0	1	0	0
17	USER USER USER USER USER USER Lawan bicara	1	1	1	0	1	0	0	0	0	1	0	0
18	Belakangan ini kok fikiran ampas banget ya'	0	0	0	0	0	0	0	0	0	0	0	0
19	Ari sarua beki mah repeh monyet\xf0\x9f\x98\	0	1	0	0	0	0	0	0	0	0	0	0
20	Jadi cowo itu harus Gantle kalo ga Gantle itu na	1	1	0	1	0	0	0	1	0	0	1	0
21	USER Slga mnr bom \xf0\x9f\x98\x82'	0	0	0	0	0	0	0	0	0	0	0	0
22	Asw ya tapi gua jarang ngambek, tacut wkkwkw	0	1	0	0	0	0	0	0	0	0	0	0
23	USER kalo kamu noob pasti peluang disakitin ny	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 4. 1 Contoh Dataset

Plot *HS* dilakukan dengan histogram, yang mana terdapat 7000 lebih *Post (Tweet)* dengan nilai 0 (bukan *hate speech*), dan 5000 lebih *Post (Tweet)* dengan nilai 1 (*hate speech*). Histogram plot *HS* yang dihasilkan ditunjukkan pada gambar 4.2 berikut.



Gambar 4. 2 Plot *HS*

Pada Gambar 4. 2 Plot *HS* tersebut memperlihatkan hasil klasifikasi *Post (Tweet)* berdasarkan kategori-kategori tertentu, yakni "Bukan *Hate speech*" dan "*Hate speech*". Dari total data yang dianalisis, 7608 *Post (Tweet)* tergolong sebagai bukan *hate speech*, sementara 5561 *Post (Tweet)* lainnya diklasifikasikan sebagai *hate speech*. Berikut hasil telah disusun dalam format tabel untuk memudahkan pemahaman dan penggunaan sebagai referensi, dapat dilihat pada Tabel 4.1 di bawah ini.

Tabel 4. 1 Jumlah *Post (Tweet)* Berdasarkan Label

No	Label	*Post (Tweet)*	Jumlah
1	0	Bukan *Hate speech*	7608
2	1	*Hate speech*	5561

4.2 *Pre-processing* Data

4.2.1 *Pre-processing SVM*

Pre-processing untuk klasifikasi *hate speech* pada *X (Twitter)* menggunakan metode *Support Vector Machine (SVM)* dan *BERT* memiliki pendekatan yang berbeda. *SVM* memanfaatkan tokenisasi, penghapusan *stopwords*, *stemming*, dan *vectorization*. Proses ini telah masuk ke dalam tahap awal *pre-processing* data dengan *source code* ditulis seperti pada gambar 4.3. Penerapan setiap fungsi pada data *Post (Tweet)* yang telah dijadikan satu sebagai *pre-processing* kemudian di *apply*.

```
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def lowercase(text):
  return text.lower()

def remove_unnecessary_char(text):
  text = re.sub('\n',' ',text)
  text = re.sub('rt',' ',text)
  text = re.sub('user',' ',text)
  text = re.sub('((@[^\s]+)|(#[^\s]+))',' ',text)
  text = re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))',' ',text)
  text = re.sub(' +',' ',text)
  return text

def remove_nonalphanumeric(text):
  text = re.sub('[^0-9a-zA-Z]+',' ', text)
  return text

alay_dict_map = dict(zip(alay_dict['original'], alay_dict['replacement']))
def normalize_alay(text):
  return ' '.join([alay_dict_map[word] if word in alay_dict_map else word for word in text.split(' ')])

def remove_abusivewords(text):
  text = ' '.join(['' if word in id_abusivewords_dict.abusivewords.values else word for word in text.split(' ')])
  text = re.sub(' +', ' ', text)
  text = text.strip()
  return text

def stemming(text):
  return stemmer.stem(text)
```

Gambar 4. 3 Tahap *Pre-processing* dengan *SVM*

Selanjutnya pada gambar 4.4 menunjukkan hasil dari proses *pre-processing* pada teks. *Pre-processing* adalah langkah awal yang penting dalam pemrosesan teks yang membantu membersihkan dan mempersiapkan teks mentah sebelum analisis lebih lanjut.

```
dataset['Tweet'] = dataset['Tweet'].apply(preprocess)
df = dataset[['Tweet', 'HS']]
df.head()
```

1 to 5 of 5 entries Filter

index	Tweet	HS
0	di saat semua cowok usaha lacak perhati gue kamu lantas remeh perhati yang gue kasih khusus ke kamu basic kamu cowok	1
1	siapa yang telat beri tau kamu gue gaul dengan cigax jifla cal sama siapa itu licew juga	0
2	41 kadang aku pikir kenapa aku tetap percaya pada tuhan padahal aku selalu jatuh kali kali kadang aku rasa tuhan itu tinggal aku sendiri ketika orang tua rencana pisah ketika kakak lebih pilih jadi kristen ketika aku anak ter	0
3	aku itu aku dan ku tau mata tapi lihat dari mana itu aku	0
4	kaum sudah lihat dari awal tambah lagi haha	1

Gambar 4. 4 Hasil dari *Apply Pre-processing*

4.2.2 Labelling Data

Untuk pemberian label, data yang kita import pada penelitian ini telah terlabeli dengan baik. Seperti yang terlampir dengan angka 0 dan 1, di mana 0 bukan *hate speech* dan 1 adalah *Post (Tweet)* yang berisi *hate speech*. Pada penelitian ini kita hanya mengambil dua *column* yaitu *Post (Tweet)* dan *HS*, final yang dihasilkan ditunjukkan pada gambar 4.5.

1 to 5 of 5 entries Filter

index	Tweet	HS
0	di saat semua cowok usaha lacak perhati gue kamu lantas remeh perhati yang gue kasih khusus ke kamu basic kamu cowok	1
1	siapa yang telat beri tau kamu gue gaul dengan cigax jifla cal sama siapa itu licew juga	0
2	41 kadang aku pikir kenapa aku tetap percaya pada tuhan padahal aku selalu jatuh kali kali kadang aku rasa tuhan itu tinggal aku sendiri ketika orang tua rencana pisah ketika kakak lebih pilih jadi kristen ketika aku anak ter	0
3	aku itu aku dan ku tau mata tapi lihat dari mana itu aku	0
4	kaum sudah lihat dari awal tambah lagi haha	1

Show 25 per page

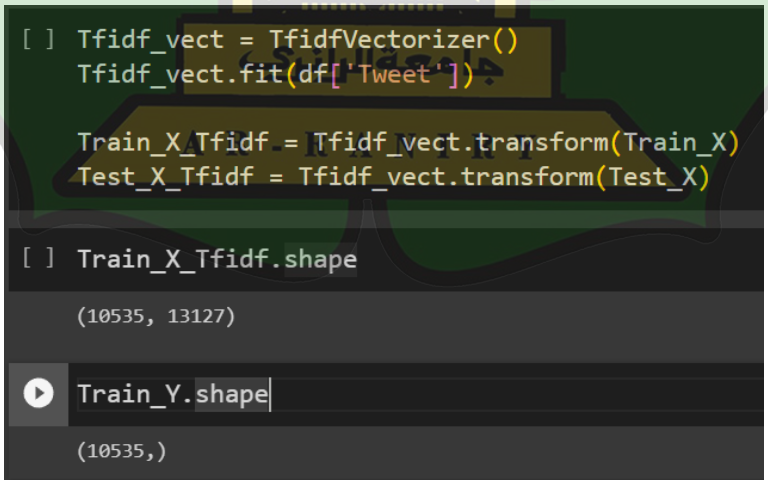
Gambar 4. 5 Data *Column* Dataset

4.2.3 *Split* Data *Testing* dan *Training*

Pada tahapan *pre-processing*, peneliti melakukan split data dengan module *Scikit-learn* train_test_split, yaitu module yang dapat membagi dataset menjadi data *train* dan *test* dengan pembagian 80% untuk data *training* dan 20% untuk *testing*, artinya *Training data* sebanyak 10535 dan *Testing data* sebanyak 2634, serta melakukan nilai frekuensi dengan *TF-IDF*.

4.2.4 *Term Frequency Inverse Document Frequency* (*TF-IDF*)

Tahap ini melakukan konversi teks menjadi representasi numerik menggunakan metode *TF-IDF* (*Term Frequency-Inverse Document Frequency*). Ini memungkinkan teks-teks tersebut diubah menjadi angka-angka, merepresentasikan seberapa penting kata-kata tersebut dalam setiap dokumen, yang dapat digunakan sebagai fitur untuk melatih model atau melakukan prediksi di tahap selanjutnya. *Source code* beserta output dapat dilihat pada gambar 4.6, dan penjelasan singkat untuk setiap langkah-langkah pada gambar tersebut dapat dilihat pada tabel 4.2.



Gambar 4. 6 Algoritma *TF-IDF* pada *SVM*

Tabel 4. 2 Langkah-langkah Konversi Teks ke Representasi TF-IDF

No	Langkah	Deskripsi
1	Inisialisasi	Objek ‘TfidfVectorizer()’ digunakan untuk mengubah teks menjadi representasi vektor TF-IDF.
2	Pembelajaran Kosakata	'Tfidf_vect.fit(df[‘*Tweet*’])’ mempelajari kata-kata dalam kolom ‘*Tweet*’ dari dataframe ‘df’.
3	Konversi Data Latih	‘Train_X_Tfidf = Tfidf_vect.transform(Train_X)’ mengonversi teks dari data latih ke representasi vektor.
4	Konversi Data Uji	‘Test_X_Tfidf = Tfidf_vect.transform(Test_X)’ mengonversi teks dari data uji ke representasi vektor.
5	Representasi Numerik	Menghasilkan angka yang merepresentasikan pentingnya kata-kata dalam setiap dokumen untuk pelatihan/prediksi.

4.2.5 *Pre-processing BERT*

1. *Case Folding* dan *Tokenization*

Tahapan ini dilakukan *lowercase*, normalisasi *lowercase* tidak diperlukan karena model *BERT* sudah dilatih dengan teks yang sudah dinormalisasi. Peneliti memanfaatkan *BERT Fine Tuning*, menggunakan *pre-trained* model *BERT* untuk menyelesaikan tugas klasifikasi teks yang spesifik pada dataset baru. Model *BERT* diinisialisasi dengan parameter *pre-trainednya* dan disesuaikan dengan dataset yang berlabel, memanfaatkan *transformers* dari *Hugging Face* pada *platform* Google Colab dengan GPU Tesla T4. Model yang digunakan, yaitu *BERT For Sequence Classification*, menambahkan layer untuk klasifikasi kalimat *multilabel* dan *multiclass*. Proses tokenisasi menggunakan *tokenizer WordPiece* dengan *“BERT-base-multilingual-uncased”* untuk mengubah teks menjadi token-token, memisahkan setiap kata menjadi token tersendiri. Penggunaan special token

seperti [CLS] untuk tugas klasifikasi teks juga diaplikasikan dalam model *BERT*. Berikut pada tabel 4.2 contoh *tokenized* dan token ids pada salah satu kalimat original dari dataset.

Tabel 4. 3 Contoh *Tokenized* Data

Original	***Tokenized***	**Token IDS**
Moga ini ada tindak lanjutnya!; ; Pelaporan terhadap ?penista agama? Ade Armando akan dilakukan langsung oleh Ketua Umum FPI KH. Shobri Lubis ke Bareskrim Mabes Polri, Gambir, Jakarta Pusat pada Selasa (10/4) pukul 13.00 WIB.; #Perhaps USER;	‘moga’, ‘ini’, ‘ada’, ‘tindak’, ‘lanjutnya’, ‘!’, ‘;’, ‘;’, ‘pelaporan’, ‘terhadap’, ‘?’, ‘penis’, ‘##ta’, ‘agama’, ‘?’, ‘ade’, ‘arm’, ‘##ando’, ‘akan’, ‘dilakukan’, ‘langsung’, ‘oleh’, ‘ketua’, ‘umum’, ‘fpi’, ‘kh’, ‘.’, ‘sho’, ‘##br’, ‘##i’, ‘lubis’, ‘ke’, ‘bar’, ‘##eskrim’, ‘mabes’, ‘polri’, ‘,’, ‘gambir’, ‘,’, ‘jakarta’, ‘pusat’, ‘pada’, ‘selasa’, ‘(‘, ‘10’, ‘/’, ‘4’, ‘)’, ‘pukul’, ‘13’, ‘.’, ‘00’, ‘wib’, ‘.’, ‘;’, ‘#’, ‘perh’, ‘##aps’, ‘user’, ‘;’	16983, 92, 176, 5053, 11577, 30457, 30473, 30473, 11032, 618, 30477, 3356, 155, 1300, 30477, 11571, 6741, 8792, 150, 477, 728, 213, 2120, 752, 17550, 454, 30470, 9596, 5547, 30356, 20616, 43, 248, 25263, 14887, 5487, 30468, 23545, 30468, 678, 1417, 126, 3943, 30464, 740, 30471, 401, 30465, 1413, 1328, 30470, 4230, 2230, 30470, 30473, 30459, 1304, 15938, 6273, 30473

2. Split Data Testing dan Training

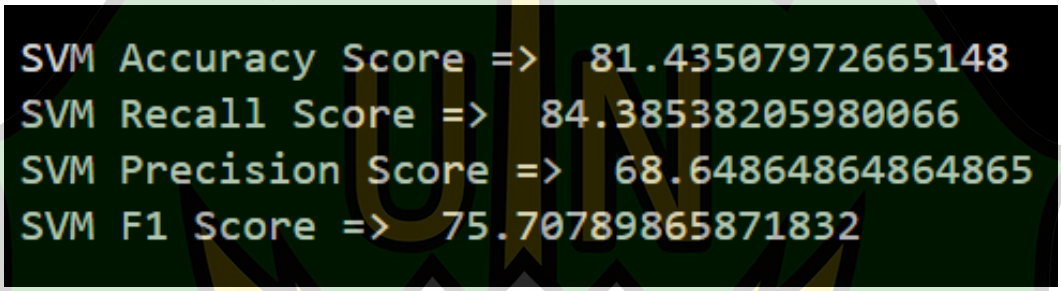
Dalam uji coba *BERT* ini, 80% data untuk pelatihan dan pengujian 20%, yang berarti *Training data* sebanyak 10535 dan *Testing data* sebanyak 2634.

4.3 Analisis Hasil dari Implementasi

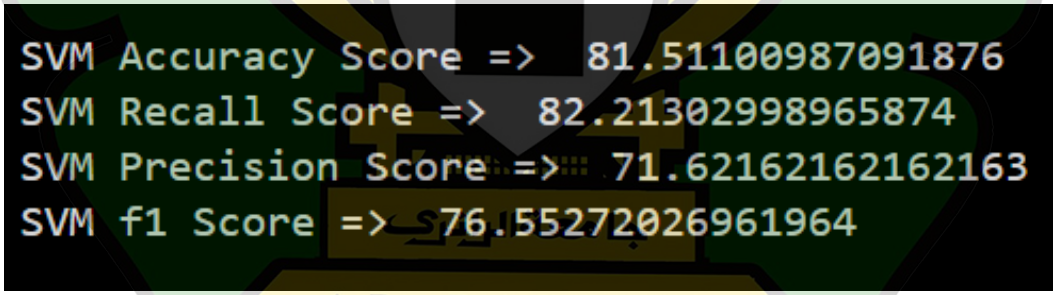
4.3.1 Klasifikasi dengan *SVM*

Berikutnya setelah melakukan ekstraksi fitur pada klasifikasi dengan *SVM*, model *Support Vector Machine (SVM)* diinisialisasi menggunakan fungsi SVC() yang

berasal dari perpustakaan *Scikit-learn*. Berbagai nilai evaluasi seperti akurasi, *Recall*, presisi, dan skor *F1* dihitung menggunakan metrik seperti '*Accuracy_Score*', '*Recall_Score*', '*Precision_Score*', dan '*F1_Score*' dari *Scikit-learn*. Nilai-nilai ini memberikan gambaran seberapa baik model *SVM* dalam memprediksi data yang belum pernah dilihat sebelumnya, hasil yang diperoleh dapat dilihat pada gambar 4.7 berikut.



Gambar 4. 7 Hasil Akurasi Menggunakan Metrik



Gambar 4. 8 Hasil Akurasi dengan *Best Tuning*

Dapat dilihat dalam evaluasi awal pada gambar 4.7 model *SVM* memperoleh akurasi sekitar 81.43%, mampu memprediksi dengan tepat dari total data. Kemampuan model mengenali kasus positif yakni *recall* mencapai 84.38%, sementara presisi hanya

sekitar 68.64%, artinya dari prediksi yang positif, sekitar 68.64% adalah kasus yang tepat. Nilai *F1 Score* sebagai perpaduan presisi dan *Recall* mencapai 75.70%.

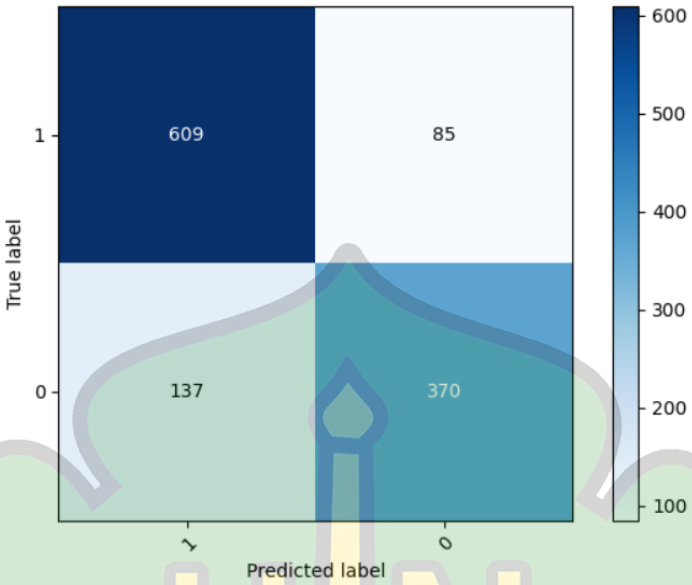
Kemudian gambar 4.8 adalah hasil setelah penulis mencoba menyesuaikan setting *hyperparameter* untuk model *Support Vector Machine (SVM)* menggunakan metode 'GridSearchCV' dari pustaka 'sklearn.model_selection'. *Hyperparameter* adalah pengaturan yang memengaruhi bagaimana model *SVM* belajar dan membuat prediksi. Hasil *Tuning hyperparameter* menunjukkan pengaturan terbaiknya adalah:

- Nilai C terbaik adalah 10.
- Menggunakan kernel 'rbf'.
- Skor terbaik yang diperoleh sebesar 0.80, menandakan tingkat akurasi model yang optimal dengan konfigurasi tersebut.

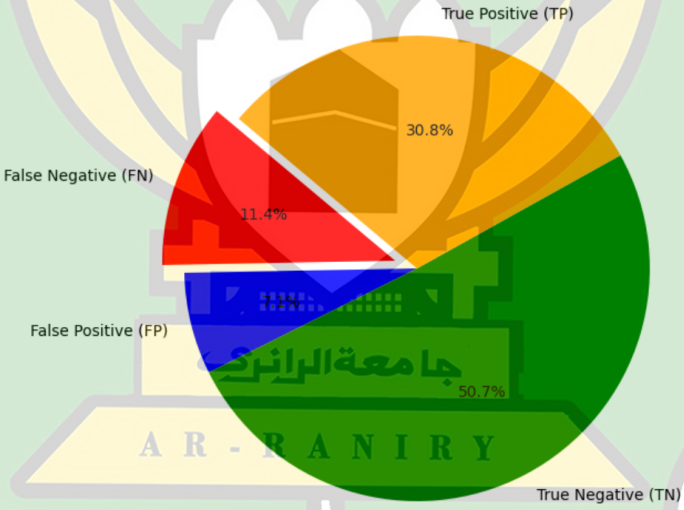
Tabel 4. 4 Hasil Akurasi *SVM*

No	***SVM***	***Accuracy***	***Precision***	***Recall***	***F1 Score***
1	Tanpa *Hyperparameter*	81.43	68.64	84.38	75.70
2	Setelah *Tuning*	81.51	71.62	82.21	76.55

Setelah penyetelan *hyperparameter*, dapat dilihat pada tabel 4.4 terjadi peningkatan sedikit pada akurasi menjadi 81.51%, namun ada penurunan pada *Recall* menjadi 82.21%. Walau begitu, terjadi peningkatan pada presisi menjadi 71.62%, menunjukkan perbaikan dalam identifikasi kasus positif dari prediksi model. *F1 Score* juga meningkat menjadi 76.55%. Perubahan ini menandakan bahwa penyetelan parameter sedikit mempengaruhi kinerja model, walaupun dampaknya tidak signifikan secara besar.



Gambar 4. 9 *Confusion Matrix SVM*



Gambar 4. 10 *Pie chart* Hasil Klasifikasi *SVM*

Confusion Matrix SVM menunjukkan evaluasi performa model melalui empat kriteria utama. Penjelasan hasilnya dapat dilihat pada Gambar 4.10. *Pie chart* klasifikasi *SVM* memberikan visualisasi yang menjelaskan informasi dari *Confusion*

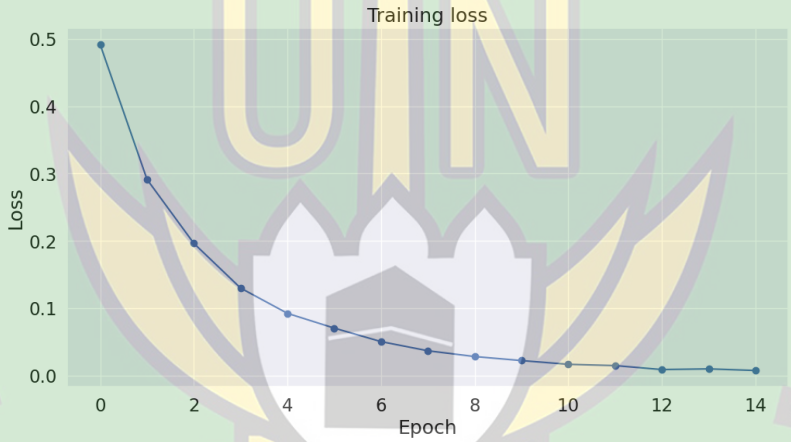
Matrix tersebut secara lebih intuitif. Berikut penjelasan *Pie chart* Klasifikasi *SVM* untuk data *testing* sebanyak 1200 *post:*

- *False Negative* (FN): Sekitar 11.4% dari total berarti 137 *post*, menggambarkan kasus yang keliru diidentifikasi sebagai non-*hate speech* padahal sebenarnya merupakan *hate speech*.
- *False Positive* (FP): Sekitar 7.1%, 85 *post* menunjukkan kesalahan dalam mengklasifikasikan konten non-*hate speech* sebagai *hate speech*.
- *True Negative* (TN): Mencapai 50.7%, 609 *post* menggambarkan kemampuan model mengenali konten non-*hate speech* secara akurat.
- *True Positive* (TP): Sekitar 30.8%, 370 *post* menunjukkan kemampuan model dalam mengklasifikasikan sebagian besar konten yang sebenarnya *hate speech* dengan benar.

4.3.2 Klasifikasi dengan *BERT*

Untuk klasifikasi dengan *BERT*, pada percobaan pertama kita akan melatih model menggunakan 10 *Epoch*, batch size 32, dan *learning rate* 2e-5. Dalam pelatihan model ini, rata-rata waktu yang dibutuhkan untuk menyelesaikan satu *Epoch* (satu putaran data melalui model) adalah sekitar 6 menit 51 detik. Grafik *training loss* untuk uji coba pertama dapat dilihat pada gambar 4.11 berikut.

Gambar 4. 11 Parameter *Training loss* Uji Coba Pertama



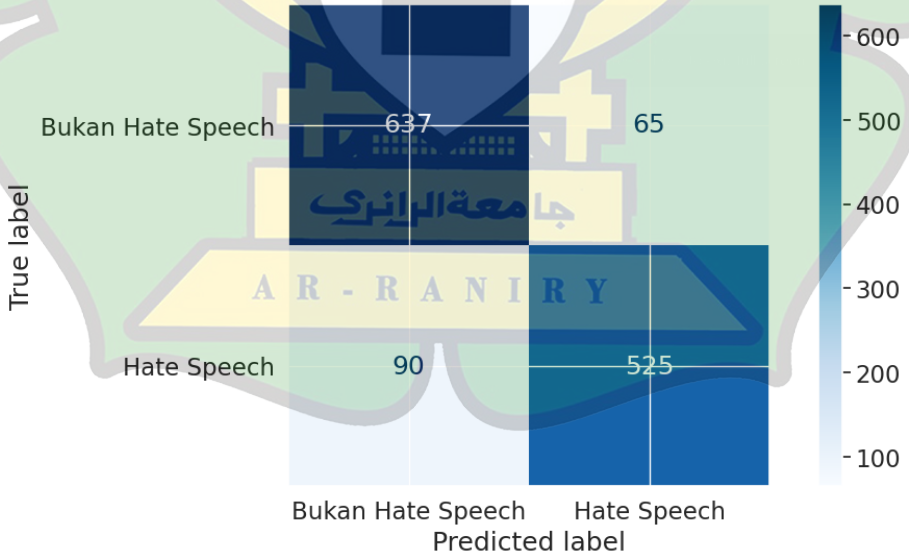
Gambar 4. 12 Parameter *Training loss* Uji Coba Kedua

Selanjut nya, pada gambar 4.12 penulis mencoba uji coba kedua, dimana penulis menambah jumlah *Epoch HS* menjadi 15 untuk memberikan lebih banyak iterasi pada proses pelatihan. Proses pelatihan model klasifikasi dengan 15 *Epoch* menunjukkan adanya penurunan berkelanjutan dalam *training loss* pada setiap iterasi. Berikut hasil akurasi kedua percobaan yang dirangkum kedalam bentuk tabel agar mempermudah dalam melihat hasil setiap akurasi.

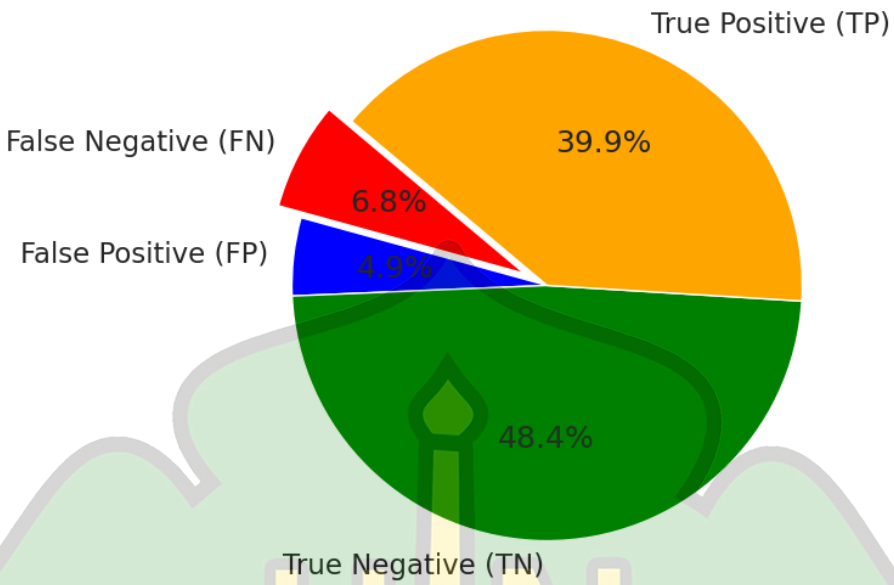
Tabel 4. 5 Hasil Akurasi *BERT*

No	*Epoch*	*Accuracy*	*Precision*	*Recall*	*F-1 Score*
1	10	0.98	0.96	1.00	0.98
2	15	0.97	0.94	1.00	0.97

Pada tabel 4.5 dapat dilihat dari tabel yang diberikan, hasil akurasi *BERT* terlihat bahwa pada *Epoch* 10, akurasi model meningkat menjadi 98%, sedangkan pada *Epoch* 15, akurasi model turun sedikit menjadi 97%. Hal ini menunjukkan bahwa pada *Epoch* 10, model berhasil mencapai performa yang lebih tinggi dengan akurasi yang lebih tinggi juga. Karena hasil terbaik didapatkan pada *Epoch* 10, maka fokus perbandingan akan difokuskan pada akurasi dari *Epoch* tersebut untuk dibandingkan dengan performa model *SVM*. Berikut *Confusion Matrix BERT* yang diperoleh.



Gambar 4. 13 *Confusion Matrix BERT*



Gambar 4. 14 *Pie chart* Hasil Klasifikasi *BERT*

Dari hasil *Confusion Matrix BERT* pada gambar 4.13, dapat dilihat *pie chart* gambar 4.14 untuk hasil klasifikasi *BERT* dengan data *testing* setara *SVM* yaitu 1200 *post* menghasilkan:

- *True Positive* (TP) mencakup sekitar 39.9%: 525 *post* konten *hate speech* yang terdeteksi dengan benar oleh model.
- *True Negative* (TN) mencakup sekitar 48.4%: 637 *post* konten non-*hate speech* yang terdeteksi dengan benar oleh model.
- *False Positive* (FP) mencakup sekitar 4.9%: 65 *post* konten non-*hate speech* yang keliru terdeteksi sebagai *hate speech*.
- *False Negative* (FN) mencakup sekitar 6.8%: 90 konten *hate speech* yang keliru terdeteksi sebagai non-*hate speech*.

4.4 Perbandingan Hasil dan Performa antara *SVM* dan *BERT*

Kedua model yaitu *SVM* dan *BERT* telah melalui beberapa percobaan guna memperoleh nilai optimal sebelum dibandingkan untuk menentukan kinerja terbaik di antara keduanya. Berikut tabel hasil perbandingan performa dari kedua model yang di uji coba dalam penelitian ini.

Tabel 4. 6 Hasil *Accuracy*, *Precision*, *Recall* dan F-1 dari Kedua Model

No	Metode	*Accuracy*	*Precision*	*Recall*	*F1-Score*
1	*SVM*	81%	71%	82%	76%
2	*BERT*	98%	96%	100%	98%

Tabel 4.6 di atas menunjukkan hasil perbandingan dua metode klasifikasi, yaitu *Support Vector Machine (SVM*) dan *Bidirectional Encoder Representations from Transformers (BERT).* Berikut adalah penjelasan tabel yang lebih terperinci:

- *Accuracy*: *SVM* menghasilkan 81% dan *BERT* 98% untuk akurasi. Perbandingan: *BERT* memiliki akurasi yang jauh lebih tinggi dibandingkan *SVM*.
- *Precision*: *SVM* memperoleh 71% dan *BERT* sebanyak 96% untuk nilai presisi. Perbandingan: *BERT* memiliki nilai *Precision* yang lebih tinggi, menunjukkan kecenderungan untuk mengidentifikasi dengan lebih tepat positif yang sebenarnya.
- *Recall*: Untuk *SVM* memiliki *recall* 82% dan *BERT* berhasil memperoleh 100%. Perbandingan: *BERT* memiliki *Recall* yang lebih tinggi, menandakan

kemampuannya dalam menemukan lebih banyak instance positif secara keseluruhan.

- *F-1 Score*: *SVM* memperoleh 76% dan *BERT* 98% untuk nilai *F-1 Score*. Perbandingan: *BERT* memiliki nilai *F-1 Score* yang jauh lebih tinggi, menunjukkan keseimbangan yang lebih baik antara *Precision* dan *Recall* dibandingkan dengan *SVM*.

Perbandingan *using time* pada kedua model adalah pada proses klasifikasi menggunakan metode *SVM*, waktu yang dibutuhkan untuk menerapkan (*apply*) hasil dari tahap *preprocessing* yang mungkin cukup lama, yakni sekitar 1005.092 detik atau sekitar 16,75 menit, disebabkan oleh faktor yang mungkin adalah kompleksitas dari fitur-fitur yang dihasilkan dari *preprocessing* dan ukuran dari data yang diolah. Sedangkan pada proses klasifikasi menggunakan *BERT*, waktu yang paling lama terjadi saat pelatihan model, dengan rata-rata 6 menit 51 detik per *epoch* atau sekitar 1,14 jam untuk 10 *epoch*. Durasi waktu yang signifikan ini terjadi karena proses latihan model *BERT* melibatkan berulangnya pengolahan data melalui model dengan tingkat kekompleksan yang tinggi. Setiap *epoch* membutuhkan waktu yang cukup lama karena *BERT* memproses data secara mendalam dan melakukan penyesuaian berulang untuk memperbaiki model terhadap data yang digunakan. Hal ini menyebabkan penggunaan waktu yang signifikan saat melatih model *BERT* pula.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan penulis, dapat disimpulkan bahwa pada penelitian ini:

1. Dengan membandingkan *SVM* dan *BERT* dalam klasifikasi *hate speech*, terlihat perbedaan signifikan dalam kinerja keduanya. *BERT*, menggunakan pemrosesan bahasa alami *end-to-end*, menunjukkan keunggulan dalam hal akurasi, presisi, *recall*, dan *F1-Score* dibandingkan dengan metode fitur manual *SVM*. *BERT* mencapai akurasi 98%, sementara *SVM* hanya 81%, menandakan tingkat ketepatan yang jauh lebih tinggi dalam mengidentifikasi konten *hate speech.* Selain itu, nilai presisi, *recall*, dan *F1-Score BERT* yang masing-masing mencapai 96%, 100%, dan 98% menunjukkan kemampuan *BERT* dalam mengenali dan membedakan *hate speech* secara lebih akurat dibandingkan dengan SVM yang memiliki nilai yang jauh lebih rendah, yaitu 71%, 82%, dan 76%.
2. Terdapat perbedaan yang signifikan dalam kemampuan metode *SVM* dan *BERT* dalam mengklasifikasikan *hate speech* di *platform* media sosial *X (Twitter)*. *BERT* memiliki performa terbaik dalam melakukan klasifikasi pada dataset *Multi Label Hate speech and Abusive Language Detection in Indonesian X (Twitter)* dengan jumlah sampel data sebanyak 13,169 *Post (Tweet)*. Meskipun *SVM* mampu melakukan klasifikasi dengan tingkat keakuratan sebesar 81%, hasil dari *BERT* menunjukkan tingkat keunggulan yang jauh lebih tinggi dengan

akurasi mencapai 98%. Oleh karena itu, algoritma *BERT* cukup baik dalam melakukan klasifikasi teks, memberikan keunggulan yang signifikan dalam menangani kompleksitas bahasa dan konteks yang ada dalam media sosial *X (Twitter)*.

5.2 Saran

Berdasarkan kesimpulan dari hasil penelitian ini, penulis mengusulkan saran untuk penelitian selanjutnya ialah sebagai berikut:

1. Melakukan eksplorasi lebih lanjut terkait penggunaan *BERT* dalam konteks yang lebih luas di *platform* media sosial. Memperluas analisis terhadap jenis konten yang lebih variatif dan mungkin menggali bagaimana *BERT* berperilaku dalam mengenali *hate speech* di *platform* lain selain *X (Twitter)*.
2. Mengoptimalkan penggunaan *SVM* dan *BERT* secara komputasional agar lebih efisien. Menggunakan perangkat dengan spesifikasi yang cukup sehingga dalam proses lebih cepat dalam menghitung akurasinya.
3. Penelitian lebih lanjut diperlukan untuk mengevaluasi *SVM* dengan fitur yang lebih kompleks atau teknik ekstraksi fitur yang lebih maju sebagai upaya untuk meningkatkan kinerja *SVM* dalam mengenali dan membedakan ujaran kebencian. Selain itu, alternatif lain yang dapat dipertimbangkan adalah mengganti metode lain yang akan dibandingkan dengan *BERT*, dengan harapan menemukan pendekatan yang lebih efisien dalam menangani masalah *hate speech* di *platform* media sosial.

DAFTAR PUSTAKA

[1] A. D. Riyanto, "Hootsuite (We are Social): Indonesian Digital Report 2023," 12 Juni 2023. [Online]. Available: https://andi.link/hootsuite-we-are-social-indonesian-digital-report- 2023/.

[2] Kominfo, "Indonesia Peringkat Lima Pengguna *X (Twitter)*," Kominfo, [Online]. Available: https://www.kominfo.go.id/content/detail/2366/indonesia-peringkat-lima-pengguna-*X (Twitter)*/0/sorotan_media. [Accessed 08 Agustus 2023].

[3] H. Kurniawan, "5 Negara Pengguna *X (Twitter)* Terbesar, Indonesia Nomor Berapa?," SindoNews.com, 30 Mei 2023. [Online]. Available: https://tekno.sindonews.com/read/1112317/207/5-negara-pengguna-*X (Twitter)*-terbesar-indonesia-nomor-berapa 1685408720#:~:text=Dilansir%20dari%20*X (Twitter)*%20The%20World,30%2F5%2F2023).&text=Disusul%20kemudian%20dengan%20Jepang%20sebanyak,bawahnya%20sebanyak%2027. [Accessed 08 Agustus 2023].

[4] R. Setiawan, "Mengenal Deep Learning Lebih Jelas," Dicoding Indonesia, 9 10 2021.[Online].Available:https://www.dicoding.com/blog/mengenal-deeplearning/ . [Accessed 18 08 2023].

[5] J. S. A. N. Kikye Martiwi Sukiakhy, "Penggunaan Data Pada *X (Twitter)* Dalam Klasifikasi Dan Visualisasi Cyberbullying Dengan Algoritma *SVM* (*Support Vector Machine*)," Cyberspace: Jurnal Pendidikan Teknologi informasi, vol. 7, no. Volume 7, Nomor 1, Maret 2023, hal. 1 - 14, pp. 1 - 14, 2023.

[6] A. S. A. F. A. B. Karimah Mutisari Hana, "Multi-label Classification of Indonesian *Hate Speech* on *X (Twitter)* Using *Support Vector Machine*s," research gate, Bandung, 2021.

[7] D. H. F. Alan Tusa Bagus W, "Klasifikasi Emosi Pada Teks dengan Menggunakan Metode Deep Learning," Syntax Literate : Jurnal Ilmiah Indonesia, vol. 6, no. Issue No. 1, November 2021, pp. 1-8, 2021.

[8] A. A. &. K. M. Hind Saleh, "Detection of *Hate speech* using *BERT* and *Hate speech* Word Embedding with Deep Model," Applied Artificial Intelligence: An International Journal, vol. 37, no. 2023, VOL. 37, NO. 1, e2166719 (405 pages), pp. 1-24, 2023.

[9] M. N. I. A. W. A. Lestari#3, "(Mengungkap Praktik Prostitusi Online pada Media Sosial *X (Twitter)*)," Jurnal Ilmu Siber, vol. 2, no. Vol. 2, No. 3, 08 2023, p. 3, 2023.

[10] Y. S. Alam Rizki Fitriansyah, "Analisis Sentimen Terhadap Pembangunan Kereta Cepat Jakarta - Bandung Pada Media Sosial *X (Twitter)* Menggunakan Metode *SVM* dan GloVe Word Embedding," e-Proceeding of Engineering, vol. 10, no. Vol.10, No.2 April 2023 | Page 1713, p. 1714, 2023.

[11] P. Vicente, "Sampling *X (Twitter)* users for social science research: evidence from a systematic review of the literature," Quality & Quantity, 2023.

[12] M. Nisa Anggraini Batubara, "Ujaran Kebencian pada Berita-Berita Covid-19 di Instagram," LINGUA, vol. 20, no. LINGUA, Vol. 20, No. 1, Maret 2023, pp. 92-106, 2023.

[13] F. R. U. P. N. S. Rija Muhamad Yazid, "Deteksi Ujaran Kebencian dengan Metode Klasifikasi Naïve Bayes dan Metode N-Gram pada Dataset Multi-Label *X (Twitter)* Berbahasa Indonesia," Informatics And Digital Expert (INDEX), vol. 4, no. Informatics And Digital Expert (INDEX) - VOL.4 NO. 2 (2022) 46-52, pp. 46-52, 2022.

[14] P. S. S. M. S. G. Luh Putu Ary Sri Tjahyanti, "Peran Artificial Intelligence (*AI*) Untuk Mendukung Pembelajaran Di Masa Pandemi Covid-19," Komputer dan Teknologi Sains (KOMTEKS), vol. 1, no. Vol. 1, No. 1, Oktober 2022, hlm. 15-21 , pp. 15-21 , 2022.

[15] Sulistiana, "Optimasi *Support Vector Machine* (*SVM*) Menggunakan Grid Search Dan Unigram Guna Meningkatkan Akurasi Review Pada Situs Ecommerce," Semarang, 12 Mei 202, Semarang, 2020.

[16] N. A. Salsabila, "Analisis Sentimen Pada Media Sosial *X (Twitter)* Terhadap Tokoh Gus Dur Menggunakan Metode Naïve Bayes Dan *Support Vector Machine* (*SVM*)," no. 25 10 2022, 25 10 2022.

[17] A. A. Karim, "Perbandingan Metode Random Forest, K-Nearest Neighbor, dan *SVM* Dalam Prediksi Akurasi Pertandingan Liga Italia," Prosiding Seminar Nasional Teknologi Dan Sains, Vol. 2, No. Prosiding Seminar Nasional Teknologi Dan Sains Tahun 2023, Vol. 2, 2023.

[18] M. H. A. Hardy, "Implementasi Model Hybrid CNN-*SVM* Untuk Deteksi Leukocoria," Jakarta, 28 Desember 2022, Jakarta, 2022.

[19] E. R. M. D. S. Ganjar Gingin Tahyudin, "Klasifikasi Gender Berdasarkan Citra Wajah Menggunakan Vision Transformer," e-Proceeding of Engineering , vol. 10, no. e- Proceeding of Engineering : Vol.10, No.2 April 2023 , p. 1808, 2023.

[20] C. S. B. D. Adine Nayla, "Deteksi *Hate speech* Pada *X (Twitter)* Menggunakan Algoritma *BERT*," e-Proceeding of Engineering, vol. 10, no. e-Proceeding of Engineering : Vol.10, No.1 Februari 2023, p. Page 256, 2023.

[21] D. B. D. S. d. Rumah, "Mengenal Google Colab, Cara Jalankan Program Python," DQLab, 20 03 2023. [Online]. Available: https://dqlab.id/mengenal-google-colab-cara-jalankan-program-python. [Accessed 18 08 2023].

[22] M. R. E. F. G. Alfredo Torence, "Penerapan Data Mining Menggunakan Algoritma K- Means Clustering Dalam Pengelompokkan Data Penerima Vaksinasi Covid-19," SISTEM INFORMASI TGD, vol. 2, no. Volume 2, Nomor 3, Mei 2023, Hal 482-488, pp. 482-488, 2023.

[23] Wikipedia, "Wikipedia," [Online]. Available: https://www.wikipedia.org/ . [Accessed 24 April 2022].

[24] S. A. M. S. Sanjana Sharma, "Degree based Classification of Harmful *Speech* using *X (Twitter)* Data," p. 106–112, August 2018.

[25] Y. S. P. W. D. E. Kevin Antariksa, "Klasifikasi Ujaran Kebencian pada Cuitan dalam Bahasa Indonesia," Jurnal Buana Informatika, vol. 10, no. 4 Jurnal Buana Informatika, Volume 10, Nomor 2, Oktober 2019: 164-171, pp. 164-171, 2019.

LAMPIRAN

Berikut adalah *source code* untuk kedua metode yakni *SVM* dan *BERT*.

❍ Klasifikasi menggunakan Fitur Manual *SVM*

****IMPORT FILES****

```
from  google.colab import files
files.upload()

!unzip X (Twitter)
```

****INSTALL PACKAGES****

```
!pip install Sastrawi
!pip install torch
!pip install transformers
!pip install Scipy
!pip install Scikit-learn
!pip install nvidia-smi
!pip install nusacrowd

!nvidia-smi
```

****IMPORT LIBRARY****

```
#packages to load
import os
import re
import csv
import torch
import torch.nn as nn
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import shutil
import sys
import seaborn as sns


#data preprocessing

from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer

#NLP tools
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
from sklearn.model_selection import train_test_split
from plotly import graph_objs as go
```

****LOAD DATASET YANG AKAN DIUJI****

```
dataset = pd.read_csv('re_dataset.csv',encoding='latin-1')

alay_dict = pd.read_csv('new_kamusalay.csv',encoding='latin-1',
header=None)
alay_dict = alay_dict.rename(columns={0:'original',1:'replacement'})

id_abusivewords_dict = pd.read_csv('abusive.csv', header=None)
id_abusivewords_dict = id_abusivewords_dict.rename(columns={0:
'abusivewords'})
```

****LOAD DATASET****

```
dataset.head(5)
dataset.describe()
dataset.isna().sum()
alay_dict.head(5)
id_abusivewords_dict.head()
```

****PLOT HS****

```
plt.hist(dataset.HS)
plt.show

dataset.HS.value_counts()
```

****PRE-PROCESSING****

```
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def lowercase(text):
  return text.lower()

def remove_unnecessary_char(text):
  text = re.sub('\n',' ',text)
  text = re.sub('rt',' ',text)
  text = re.sub('user',' ',text)
  text = re.sub('((@[^\s]+)|(#[^\s]+))',' ',text)
  text =
re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))',' 
',text)
  text = re.sub(' +',' ',text)
  return text

def remove_nonalphanumeric(text):
  text = re.sub('[^0-9a-zA-Z]+',' ', text)
```

```
  return text

alay_dict_map = dict(zip(alay_dict['original'],
alay_dict['replacement']))
def normalize_alay(text):
  return ' '.join([alay_dict_map[word] if word in alay_dict_map else
word for word in text.split(' ')])

def remove_abusivewords(text):
  text = ' '.join(['' if word in
id_abusivewords_dict.abusivewords.values else word for word in
text.split(' ')])
  text = re.sub(' +', ' ', text)
  text = text.strip()
  return text

def stemming(text):
  return stemmer.stem(text)
```

****PREPROCESSING FUNCTION****

```
def preprocess(text):
  text = lowercase(text)
  text = remove_unnecessary_char(text)
  text = remove_nonalphanumeric(text)
  text = normalize_alay(text)
  text = remove_abusivewords(text)
  text = stemming(text)
  return text
```

****APPLY****

```
def classify(hs):
    retval = ""
    if int(hs) == 1:
        retval = 'negative'
    else:
        retval = 'positive'
    return retval

dataset['Tweet'] = dataset['Tweet'].apply(preprocess)
dataset['hs_class'] = dataset['HS'].apply(classify)
dataset[['Tweet', 'hs_class']].sample(8)

df = dataset[['Tweet', 'HS']]
df.head()
```

****TRAIN AND TEST****

```
Train_X,Test_X,Train_Y,Test_Y=
model_selection.train_test_split(df['Tweet'],df['HS'],test_size=0.2)

Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
```

```
Test_Y = Encoder.fit_transform(Test_Y)
```

****TF-IDF** **FEATURE****

```
Tfidf_vect = TfidfVectorizer()
Tfidf_vect.fit(df['Tweet'])

Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)

Train_X_Tfidf.shape

Train_Y.shape
```

****BUILD, TRAIN, EVALUATE MODEL****

```
SVM = SVC()
SVM.fit(Train_X_Tfidf,Train_Y)

predictions_SVM = SVM.predict(Test_X_Tfidf)
print("SVM Accuracy Score => ",accuracy_score(predictions_SVM,
Test_Y)*100)
print("SVM Recall Score => ",recall_score(predictions_SVM,
Test_Y)*100)
print("SVM Precision Score => ",precision_score(predictions_SVM,
Test_Y)*100)
print("SVM F1 Score => ",f1_score(predictions_SVM, Test_Y)*100)

from sklearn.metrics import confusion_matrix
confusion_matrix(Test_Y, predictions_SVM)

from sklearn.metrics import classification_report
print(classification_report(Test_Y, predictions_SVM))
```

****HYPERPARAMETER** **TUNING****

```
hyperparameters = {'kernel':('linear', 'rbf'), 'C':[1,10]}
svm = SVC()
svm_tuned = GridSearchCV(svm, hyperparameters, cv=5)

svm_tuned.fit(Train_X_Tfidf,Train_Y)

print('Best C:', svm_tuned.best_estimator_.C)
print('Best Kernel:', svm_tuned.best_estimator_.kernel)
print('Best Score:', svm_tuned.best_score_)
```

****BEST TUNING APPLY****

```
svm = SVC(C=10, kernel='rbf', degree=3, gamma='scale')
svm.fit(Train_X_Tfidf,Train_Y)

predict_test = svm.predict(Test_X_Tfidf)
print("SVM Accuracy Score => ",accuracy_score(predict_test,
Test_Y)*100)
```

```
print("SVM Recall Score => ",recall_score(predict_test, Test_Y)*100)
print("SVM Precision Score => ",precision_score(predict_test,
Test_Y)*100)
print("SVM f1 Score => ",f1_score(predict_test, Test_Y)*100)

from sklearn.metrics import classification_report

# Prediksi untuk data testing
predict_test = svm.predict(Test_X_Tfidf)
print("Classification Report for Testing Data:")
print(classification_report(Test_Y, predict_test))

# Prediksi untuk data training
predict_train = svm.predict(Train_X_Tfidf)
print("Classification Report for Training Data:")
print(classification_report(Train_Y, predict_train))

train = dataset[['hs_class', 'Tweet']]
base_train = train
train.to_csv('train_preprocessed.csv', index = False)
train.sample(5)
```

****TRY TO GROUP HS****

```
temp=
train.groupby('hs_class').count()[Tweet'].reset_index().sort_values(
by='Tweet',ascending=False)
temp.style.background_gradient(cmap='Greens')

train = dataset[['hs_class', 'Tweet']]
train_pos = train[train['hs_class']=='positive']
train_neg = train[train['hs_class']=='negative']
train_neg, removed = train_test_split(train_neg, test_size=(1.9/7),
shuffle=True)
train = pd.concat([train_pos, train_neg])
temp =
train.groupby('hs_class').count()['=Tweet'].reset_index().sort_value
s(by='Tweet',ascending=False)
temp.style.background_gradient(cmap='Greens')
```

****CONFUSION MATRIX****

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    if not title:
        title = 'Normalized confusion matrix' if normalize else
'Confusion matrix, without normalization'
```

```
    cm = confusion_matrix(y_true, y_pred)
    classes = classes[unique_labels(y_true, y_pred)]

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print('Normalized confusion matrix')
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
rotation_mode="anchor")

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    plt.show()  # Ensure the plot is displayed
    return ax

np.set_printoptions(precision=2)


from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels
def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
  if not title:
    if normalize:
      title = 'Normalized confusion matrix'
    else:
      title = 'Confusion matrix, without normalization'

  cm = confusion_matrix(y_true, y_pred)
  classes = classes[unique_labels(y_true, y_pred)]
  if normalize:
      cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
      print('Normalized confusion matrix')
  else:
```

```
        print('Confusion matrix, without normalization')

    print(cm)

    fig, ax = plt.subplots()
    im = ax.imshow(cm,interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
      for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], fmt),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

np.set_printoptions(precision=2)

class_names = Test_Y
plot_confusion_matrix(Test_Y, predict_test, classes=class_names,
                      title= 'Confusion matrix, without
normalization')
plt.show()

import matplotlib.pyplot as plt

# Mengambil confusion matrix yang telah dihasilkan sebelumnya
conf_matrix = np.array([[1352, 172],
                        [315, 795]])

# Menghitung FN, FP, TN, TP dari confusion matrix
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
TP = conf_matrix[1, 1]

# Data untuk pie chart
sizes = [FN, FP, TN, TP]
labels = ['False Negative (FN)', 'False Positive (FP)', 'True
Negative (TN)', 'True Positive (TP)']
colors = ['red', 'blue', 'green', 'orange']
explode = (0.1, 0, 0, 0) # Untuk menyorot bagian tertentu jika
diinginkan

# Membuat pie chart
```

```
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Memastikan lingkaran terlihat sebagai lingkaran
plt.title('')
plt.show()

import matplotlib.pyplot as plt

# Mengambil confusion matrix yang telah dihasilkan sebelumnya
conf_matrix = np.array([[1352, 172],
                        [315, 795]])

# Menghitung FN, FP, TN, TP dari confusion matrix
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
TP = conf_matrix[1, 1]

# Membuat bar plot horizontal
categories = ['False Negative (FN)', 'False Positive (FP)', 'True
Negative (TN)', 'True Positive (TP)']
values = [FN, FP, TN, TP]

plt.figure(figsize=(8, 6))
plt.barh(categories, values, color=['red', 'blue', 'green',
'orange'])
plt.xlabel('Values')
plt.ylabel('Categories')
plt.title('Confusion Matrix Values')
plt.show()



def classify(Tweet):
  pred = SVM.predict(Tfidf_vect.transform([Tweet]))
  if pred == 1:
    return "Hate Speech"
  return "Bukan Hate Speech"



classify('gua benci banget liat warna tas lu anjir jelek!')

classify('kita bagus nya ngapain ya hari ini')

classify('gendut, makin berat ya lu haha')
```

❍ **Klasifikasi menggunakan *BERT* (*epoch* 10)**

****1) CHECK RESOURCE****

```
!nvidia-smi


!pip install transformers
!pip install wget
```

****2) DOWNLOAD & LOAD DATASET****

```
import wget
import os
import pandas as pd


print("Downloading dataset...")
url = 'https://github.com/okkyibrohim/id-multi-label-hate-speech-
and-abusive-language-detection.git'


!git clone https://github.com/okkyibrohim/id-multi-label-hate-
speech-and-abusive-language-detection.git

%cd id-multi-label-hate-speech-and-abusive-language-detection

dataset = pd.read_csv('re_dataset.csv', encoding='latin-1')

alay_dict = pd.read_csv('new_kamusalay.csv', encoding='latin-1',
header=None)
alay_dict = alay_dict.rename(columns={0: 'original', 1:
'replacement'})

id_abusivewords_dict = pd.read_csv('abusive.csv', header=None)
id_abusivewords_dict = id_abusivewords_dict.rename(columns={0:
'abusivewords'})

df = dataset[['Tweet', 'HS']]
df.head()

import seaborn as sns
import matplotlib.pyplot as plt

plt.hist(dataset.HS)
plt.show

sentences = df.Tweet.values
labels = df.HS.values
```

****3) LOAD BERT TOKENIZER****

```
from transformers import BertTokenizer, AutoModel

print("Loading BERT Tokenizer")
tokenizer = BertTokenizer.from_pretrained("bert-base-multilingual-
uncased", do_lower_case=True)


df.sample(8)

print("Original: ", sentences[133])
print("Tokenized: ", tokenizer.tokenize(sentences[133]))
print("Token IDS: ",
tokenizer.convert_tokens_to_ids(tokenizer.tokenize(sentences[133])))

input_ids = []

for sent in sentences:
  encoded_sent = tokenizer.encode(
      sent,
      add_special_tokens = True
  )
  input_ids.append(encoded_sent)

print("Original: ", sentences[133])
print("Token IDs: ", input_ids[133])

print("Max sentence length: ", max([len(sen) for sen in input_ids]))

from keras.preprocessing.sequence import pad_sequences

MAX_LEN = 256


print("Padding/truncating all sentences to %d values" % MAX_LEN)
print('Padding token: "{:}", ID: {:}'.format(tokenizer.pad_token,
tokenizer.pad_token_id))

input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype='long',
value=0, truncating='post', padding='post')

print("Done")

input_ids[133]

attention_mask = []

for sent in input_ids:
  att_mask = [int(token_id > 0) for token_id in sent]

  attention_mask.append(att_mask)
```

4) PERSIAPKAN DATA

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_labels, test_labels =
train_test_split(input_ids,labels,random_state=2017,

test_size=0.1)train_mask, test_mask, _, _ =
train_test_split(attention_mask,labels,random_state=2017,test_size=0
.1)

train_input, validation_input, train_labels, validation_labels =
train_test_split(train_input,train_labels,random_state=2018,test_siz
e=0.15)

train_mask, validation_mask, _, _ =
train_test_split(train_mask,train_mask,random_state=2018,test_size=0
.15)


import numpy as np
print("== Train ==")
print("Input: ", train_input.shape)
print("Label: ", train_labels.shape)
print("Mask: ", np.array(train_mask).shape)

print("\n== Validation ==")
print("Input: ", validation_input.shape)
print("Label: ", validation_labels.shape)
print("Mask: ", np.array(validation_mask).shape)

print("\n== Test ==")
print("Input: ", test_input.shape)
print("Label: ", test_labels.shape)
print("Mask: ", np.array(test_mask).shape)


import torch
train_input = torch.tensor(train_input)
train_labels = torch.tensor(train_labels)
train_mask = torch.tensor(train_mask)

validation_input = torch.tensor(validation_input)
validation_labels = torch.tensor(validation_labels)
validation_mask = torch.tensor(validation_mask)

test_input = torch.tensor(test_input)
test_labels = torch.tensor(test_labels)
test_mask = torch.tensor(test_mask)

from torch.utils.data import TensorDataset, DataLoader,
RandomSampler, SequentialSampler

batch_size = 32

train_data = TensorDataset(train_input, train_mask, train_labels)
```

```
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler,
batch_size=batch_size)

validation_data = TensorDataset(validation_input, validation_mask,
validation_labels)
validation_sampler = SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data,
sampler=validation_sampler, batch_size=batch_size)

test_data = TensorDataset(test_input, test_mask, test_labels)
test_sampler = SequentialSampler(test_data)
test_dataloader = DataLoader(test_data, sampler=test_sampler,
batch_size=batch_size)
```

****5) PERSIAPKAN MODEL PRE-TRAINED BERT****

```
from transformers import BertForSequenceClassification, AdamW,
BertConfig

model = BertForSequenceClassification.from_pretrained(
    "bert-base-multilingual-uncased",
    num_labels = 2,
    output_attentions = False,
    output_hidden_states = False
)

model.cuda()

params = list(model.named_parameters())

print("The BERT model has {:} different named
parameters.".format(len(params)))

print("==== Embedding Layer ====")
for p in params[0:5]:
  print("{:<60} {:>12}".format(p[0], str(tuple(p[1].size()))))

print("==== First Transformers ====")
for p in params[5:21]:
  print("{:<60} {:>12}".format(p[0], str(tuple(p[1].size()))))

print("==== Output Layer ====")
for p in params[-4:]:
  print("{:<60} {:>12}".format(p[0], str(tuple(p[1].size()))))

from transformers import BertForSequenceClassification
import torch
from torch.optim import AdamW


optimizer = AdamW(
    model.parameters(),
    lr = 2e-5,
```

```
    eps = 1e-8
)


from transformers import get_linear_schedule_with_warmup
epochs = 10
total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps =
total_steps)



import numpy as np

def flat_accuracy(preds, labels):
  pred_flat = np.argmax(preds, axis=1).flatten()
  labels_flat = labels.flatten()
  return np.sum(pred_flat == labels_flat) / len(labels_flat)

import time
import datetime

def format_time(elapsed):
  elapsed_rounded = int(round(elapsed))
  return str(datetime.timedelta(seconds=elapsed_rounded))
```

****6) TRAINING BERT****

```
import random
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
seed_val = 42

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

loss_values = []

for epoch_i in range(0, epochs):

  # ========================================
  #               Training
  # ========================================

  print("======= Epoch {:} / {:} =======".format(epoch_i+1, epochs))
  print("Training...")

  t0 = time.time()

  total_loss = 0
```

```
  model.train()

  # For each batch of training data
  for step, batch in enumerate(train_dataloader):

    # Progress update every 40 batches
    if step % 40 == 0 and not step == 0:
      elapsed = format_time(time.time() - t0)

      print("Batch {:>5,} of {:>5,}.     Elapsed: {:}".format(step,
len(train_dataloader), elapsed))

    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)

    model.zero_grad()

    outputs = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask,
                    labels=b_labels)

    loss = outputs[0]

    total_loss += loss.item()

    loss.backward()

    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)

    optimizer.step()

    scheduler.step()

  avg_train_loss = total_loss / len(train_dataloader)

  loss_values.append(avg_train_loss)

  print("   Average training loss: {0:.2f}".format(avg_train_loss))
  print("   Training epoch took: {:}".format(format_time(time.time()
- t0)))

  # ========================================
  #               Validation
  # ========================================

  print("Running Validation...")

  t0 = time.time()

  model.eval()

  eval_loss, eval_accuracy = 0, 0
  nb_eval_steps, nb_eval_examples = 0, 0
```

```
  for batch in validation_dataloader:

    batch = tuple(t.to(device) for t in batch)

    b_input_ids, b_input_mask, b_labels = batch

    with torch.no_grad():
      outputs = model(b_input_ids,
                      token_type_ids=None,
                      attention_mask=b_input_mask)

    logits = outputs[0]
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()

    tmp_eval_accuracy = flat_accuracy(logits, label_ids)

    eval_accuracy += tmp_eval_accuracy

    nb_eval_steps += 1

  print("   Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
  print("   Validation took: {:}".format(format_time(time.time() -
t0)))

print("Training Complete!")


import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='darkgrid')
sns.set(font_scale=1.5)

plt.rcParams["figure.figsize"] = (12,6)
plt.plot(loss_values, 'b-o')

plt.title("Training loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")

plt.show()
```

****7) PREDICT DAN EVALUATE****

```
print("Predicting labels for {:,} test
sentences".format(len(test_input)))

model.eval()

prediction, true_labels = [], []

for batch in test_dataloader:
  batch = tuple(t.to(device) for t in batch)
```

```
  b_input_ids, b_input_mask, b_labels = batch

  with torch.no_grad():
    outputs = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask)

  logits = outputs[0]

  logits = logits.detach().cpu().numpy()
  label_ids = b_labels.to('cpu').numpy()

  prediction.append(logits)
  true_labels.append(label_ids)

print(" DONE.")

from sklearn.metrics import matthews_corrcoef

flat_prediction = [item for sublist in prediction for item in
sublist]
flat_prediction = np.argmax(flat_prediction, axis=1).flatten()

flat_true_labels = [item for sublist in true_labels for item in
sublist]

mcc = matthews_corrcoef(flat_true_labels, flat_prediction)

print("MCC: %.3f" %mcc)

from sklearn.metrics import accuracy_score

acc = accuracy_score(flat_true_labels, flat_prediction)

print("ACC: %.3f" %acc)


# Convert tensor to numpy arrays
flat_true_labels = np.array(flat_true_labels)
flat_prediction = np.array(flat_prediction)

# Decode label IDs to original labels
label_map = {0: "Bukan Hate Speech", 1: "Hate Speech"}  # Ubah
sesuai dengan label yang digunakan dalam dataset

true_labels = [label_map[label] for label in flat_true_labels]
predicted_labels = [label_map[label] for label in flat_prediction]

# Print original labels and predicted labels side by side
for i in range(len(flat_true_labels)):
    print("Data ke-%d - True: %s, Predicted: %s" % (i+1,
flat_true_labels[i], flat_prediction[i]))


from sklearn.metrics import confusion_matrix
```

```
# Hitung confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)
print(cm)


from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score

# Contoh confusion matrix
cm = [[637, 65],
 [90, 525]]


# Calculate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=["Bukan Hate Speech", "Hate Speech"])
disp.plot(cmap='Blues', values_format='d')
plt.show()


from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

# Mengambil confusion matrix yang telah dihasilkan sebelumnya
conf_matrix = np.array([[637, 65],
                        [90, 525]])

# Menghitung FN, FP, TN, TP dari confusion matrix
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
TP = conf_matrix[1, 1]

# Data untuk pie chart
sizes = [FN, FP, TN, TP]
labels = ['False Negative (FN)', 'False Positive (FP)', 'True
Negative (TN)', 'True Positive (TP)']
colors = ['red', 'blue', 'green', 'orange']
explode = (0.1, 0, 0, 0)  # Untuk menyorot bagian tertentu jika
diinginkan

# Membuat pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Memastikan lingkaran terlihat sebagai lingkaran
plt.title('')
plt.show()
```

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# Nilai dari confusion matrix
cm = [[637, 65],
 [90, 525]]

# Konversi confusion matrix ke true_labels dan predicted_labels
true_labels = [0] * cm[0][0] + [1] * cm[1][1] + [0] * cm[1][0] + [1]
* cm[0][1]
predicted_labels = [0] * cm[0][0] + [1] * cm[1][1] + [0] * cm[0][1]
+ [1] * cm[1][0]

# Menghitung metrik evaluasi
accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)
f1 = f1_score(true_labels, predicted_labels)

# Menampilkan hasil
print("Accuracy: {:.2f}".format(accuracy))
print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1-Score: {:.2f}".format(f1))


def classify_with_BERT(Tweet):
    # Tentukan panjang maksimum teks yang diperbolehkan
    max_length = 256  # Sesuaikan sesuai dengan kebutuhan aplikasi

    # Tokenize the Post (Tweet) dengan panjang maksimum yang telah
ditentukan
    inputs = tokenizer(Tweet, padding=True, truncation=True,
max_length=max_length, return_tensors="pt")

    # Pindahkan tensor input ke perangkat yang sama dengan model
(GPU atau CPU)
    inputs.to(model.device)

    # Forward pass through the model
    with torch.no_grad():
        outputs = model(**inputs)

    # Proses output model sesuai kebutuhan
    predicted_class = torch.argmax(outputs.logits, dim=1).item()

    print("Predicted class:", predicted_class)
    print("Output logits:", outputs.logits)

    if predicted_class == 1:
        return "Hate Speech"
    return "Bukan Hate Speech"

Tweet = "wah, bagus ya warna langit nya hari ini gimana ya? Ayuk
main hari ini"
```

```
result = classify_with_BERT(Tweet)
print(result)


Tweet = "Anjir, lu kata tiap karakter mesti punya sisi buruk nya?
Ini Mc beneran gila, emang boleh separah itu?"
result = classify_with_BERT(Tweet)
print(result)
```

❍ **Klasifikasi menggunakan *BERT* (*epoch* 15)**

Dalam uji coba kedua, ada penyesuaian pada pengaturan *epoch* yang digunakan dalam proses pengujian. Meskipun kode sumbernya sama dengan uji coba pertama, pada uji coba kedua ini, penggunaan *epoch* diperpanjang menjadi 15. *Epoch* adalah istilah yang menandakan berapa kali seluruh dataset dilatihkan pada algoritma pembelajaran mesin. Jadi, perubahan ini mengacu pada pengulangan pembelajaran data yang dilakukan oleh algoritma. Meskipun semua kode lainnya tetap sama dengan uji coba sebelumnya yang menggunakan 10 *epoch*, pengaturan *epoch* diperpanjang menjadi 15 kali pengulangan pada uji coba kedua ini. Berikut *source code* untuk *epoch* 15:

****5) PERSIAPKAN MODEL PRE-TRAINED BERT****

```
from transformers import get_linear_schedule_with_warmup
epochs = 15
total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0,
                                            num_training_steps =
total_steps)


import numpy as np

def flat_accuracy(preds, labels):
  pred_flat = np.argmax(preds, axis=1).flatten()
  labels_flat = labels.flatten()
  return np.sum(pred_flat == labels_flat) / len(labels_flat)

import time
import datetime
```

```
def format_time(elapsed):
  elapsed_rounded = int(round(elapsed))
  return str(datetime.timedelta(seconds=elapsed_rounded))
```

****6) TRAINING BERT****

```
import random
device = torch.device("cuda" if torch.cuda.is_available() else
"cpu")
seed_val = 42

random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

loss_values = []

for epoch_i in range(0, epochs):

  # ========================================
  #               Training
  # ========================================

  print("======= Epoch {:} / {:} =======".format(epoch_i+1, epochs))
  print("Training...")

  t0 = time.time()

  total_loss = 0

  model.train()

  # For each batch of training data
  for step, batch in enumerate(train_dataloader):

    # Progress update every 40 batches
    if step % 40 == 0 and not step == 0:
      elapsed = format_time(time.time() - t0)

      print("Batch {:>5,} of {:>5,}.    Elapsed: {:}".format(step,
len(train_dataloader), elapsed))

    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)

    model.zero_grad()

    outputs = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask,
                    labels=b_labels)

    loss = outputs[0]
```

```
    total_loss += loss.item()
    loss.backward()
    torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
    optimizer.step()
    scheduler.step()
  avg_train_loss = total_loss / len(train_dataloader)

  loss_values.append(avg_train_loss)

  print("   Average training loss: {0:.2f}".format(avg_train_loss))
  print("   Training epoch took: {:}".format(format_time(time.time()
- t0)))

  # ==================================
  #           Validation
  # ==================================

  print("Running Validation...")
  t0 = time.time()
  model.eval()
  eval_loss, eval_accuracy = 0, 0
  nb_eval_steps, nb_eval_examples = 0, 0

  for batch in validation_dataloader:
    batch = tuple(t.to(device) for t in batch)
    b_input_ids, b_input_mask, b_labels = batch
    with torch.no_grad():
      outputs = model(b_input_ids,
                      token_type_ids=None,
                      attention_mask=b_input_mask)
    logits = outputs[0]
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    tmp_eval_accuracy = flat_accuracy(logits, label_ids)
    eval_accuracy += tmp_eval_accuracy
    nb_eval_steps += 1

  print("   Accuracy: {0:.2f}".format(eval_accuracy/nb_eval_steps))
  print("   Validation took: {:}".format(format_time(time.time() -
t0)))

print("Training Complete!")


import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style='darkgrid')
sns.set(font_scale=1.5)

plt.rcParams["figure.figsize"] = (12,6)
plt.plot(loss_values, 'b-o')

plt.title("Training loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
```

```
plt.show()
```

****7) PREDICT DAN EVALUATE****

```
print("Predicting labels for {:,} test
sentences".format(len(test_input)))

model.eval()

prediction, true_labels = [], []

for batch in test_dataloader:
  batch = tuple(t.to(device) for t in batch)

  b_input_ids, b_input_mask, b_labels = batch

  with torch.no_grad():
    outputs = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask)

  logits = outputs[0]

  logits = logits.detach().cpu().numpy()
  label_ids = b_labels.to('cpu').numpy()

  prediction.append(logits)
  true_labels.append(label_ids)

print(" DONE.")

from sklearn.metrics import matthews_corrcoef

flat_prediction = [item for sublist in prediction for item in
sublist]
flat_prediction = np.argmax(flat_prediction, axis=1).flatten()

flat_true_labels = [item for sublist in true_labels for item in
sublist]

mcc = matthews_corrcoef(flat_true_labels, flat_prediction)

print("MCC: %.3f" %mcc)

from sklearn.metrics import accuracy_score

acc = accuracy_score(flat_true_labels, flat_prediction)

print("ACC: %.3f" %acc)


# Convert tensor to numpy arrays
flat_true_labels = np.array(flat_true_labels)
```

```
flat_prediction = np.array(flat_prediction)

# Decode label IDs to original labels
label_map = {0: "Bukan Hate Speech", 1: "Hate Speech"}  # Ubah
sesuai dengan label yang digunakan dalam dataset

true_labels = [label_map[label] for label in flat_true_labels]
predicted_labels = [label_map[label] for label in flat_prediction]

# Print original labels and predicted labels side by side
for i in range(len(flat_true_labels)):
    print("Data ke-%d - True: %s, Predicted: %s" % (i+1,
flat_true_labels[i], flat_prediction[i]))


from sklearn.metrics import confusion_matrix

# Hitung confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)
print(cm)


from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score

# Contoh confusion matrix
cm = [[637, 65],
 [90, 525]]


# Calculate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=["Bukan Hate Speech", "Hate Speech"])
disp.plot(cmap='Blues', values_format='d')
plt.show()

from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

# Mengambil confusion matrix yang telah dihasilkan sebelumnya
conf_matrix = np.array([[637, 65],
                        [90, 525]])

# Menghitung FN, FP, TN, TP dari confusion matrix
TN = conf_matrix[0, 0]
FP = conf_matrix[0, 1]
FN = conf_matrix[1, 0]
TP = conf_matrix[1, 1]

# Data untuk pie chart
```

```
sizes = [FN, FP, TN, TP]
labels = ['False Negative (FN)', 'False Positive (FP)', 'True
Negative (TN)', 'True Positive (TP)']
colors = ['red', 'blue', 'green', 'orange']
explode = (0.1, 0, 0, 0)  # Untuk menyorot bagian tertentu jika
diinginkan

# Membuat pie chart
plt.figure(figsize=(8, 6))
plt.pie(sizes, labels=labels, colors=colors, explode=explode,
autopct='%1.1f%%', startangle=140)
plt.axis('equal')  # Memastikan lingkaran terlihat sebagai lingkaran
plt.title('')
plt.show()


from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

# Nilai dari confusion matrix
cm = [[637, 65],
 [90, 525]]

# Konversi confusion matrix ke true_labels dan predicted_labels
true_labels = [0] * cm[0][0] + [1] * cm[1][1] + [0] * cm[1][0] + [1]
* cm[0][1]
predicted_labels = [0] * cm[0][0] + [1] * cm[1][1] + [0] * cm[0][1]
+ [1] * cm[1][0]

# Menghitung metrik evaluasi
accuracy = accuracy_score(true_labels, predicted_labels)
precision = precision_score(true_labels, predicted_labels)
recall = recall_score(true_labels, predicted_labels)
f1 = f1_score(true_labels, predicted_labels)

# Menampilkan hasil
print("Accuracy: {:.2f}".format(accuracy))
print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1-Score: {:.2f}".format(f1))
```

RIWAYAT HIDUP PENULIS

1. Nama : Husnul Mawaddah
2. NIM : 190212035
3. Tempat / Tgl. Lahir : Darussalam / 08 Desember 1999
4. Jenis Kelamin : Perempuan
5. Alamat : Desa Simpang Peut, Kec. Kuala Kab. Nagan Raya
6. Status : Mahasiswa
7. E-Mail Institusi : 190212035@student.ar-raniry.ac.id
8. Nama Orang Tua
   a. Ayah : Syarifuddin
   b. Ibu : Nurmaila Wardati
   c. Pekerjaan Ayah : Wiraswasta
   d. Pekerjaan Ibu : Ibu Rumah Tangga
9. Alamat Orang Tua : Desa Simpang Peut, Kec. Kuala Kab. Nagan Raya
10. Pendidikan
    a. SD : MIN Rukoh
    b. SMP : SMP Negeri 2 Kuala
    c. SMA : MAS Darul Ihsan
    d. Perguruan Tinggi : Universitas Islam Negeri Ar-Raniry

Banda Aceh, 08 Desember 2023

Penulis,

Husnul Mawaddah

190212035