**SISTEM DETEKSI *HATE SPEECH* BERBAHASA INDONESIA MENGGUNAKAN METODE *INDONESIAN BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS* (INDOBERT)**

**SKRIPSI**

diajukan untuk menempuh ujian sarjana

pada Fakultas Matematika dan Ilmu Pengetahuan Alam  
Universitas Padjadjaran

RIZKY ANUGERAH

NPM 140810180049



UNIVERSITAS PADJADJARAN  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
PROGRAM STUDI TEKNIK INFORMATIKA  
SUMEDANG  
2021

**SKRIPSI**

**SISTEM DETEKSI *HATE SPEECH* BERBAHASA INDONESIA MENGGUNAKAN METODE *INDONESIAN BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS* (INDOBERT)**

**HATE SPEECH DETECTION SYSTEM IN INDONESIAN LANGUAGE USING INDONESIAN BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (INDOBERT) METHOD**

Telah dipersiapkan dan disusun oleh

RIZKY ANUGERAH

NPM 140810180049

Telah dipertahankan di depan Tim Penguji

pada tanggal …………….

Susunan Tim Penguji

1. Dr. Juli Rejito, M.Kom. Ketua Tim Penguji

NIP. 19680717 199303 1 003 ………………………

1. Dr. Intan Nurma Yulita, MT Pembimbing

NIP. 19850704 201504 2 003 ………………………

1. Aditya Pradana, S. T., M. Eng. Co-Pembimbing

NIP. 19841211 201504 1 002 ………………………

1. ……………………………… Penguji

NIP. ………………………... ………………………

1. ……………………………… Penguji

NIP. ………………………... ………………………

1. ……………………………… Penguji

NIP. ………………………... ………………………

# KATA PENGANTAR

*Assalamu’alaikum Warahmatullahi Wabarakatuh*

Dengan mengucap puji dan syukur penulis panjatkan ke hadirat Allah SWT yang telah memberikan berkah, rahmat, serta karunia-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “**SISTEM DETEKSI *HATE SPEECH* BERBAHASA INDONESIA MENGGUNAKAN METODE *INDONESIAN BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS* (INDOBERT)**” sebagai salah satu syarat menempuh sarjana pada Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Penulis menyadari selama proses penyusunan dan penulisan skripsi ini banyak hambatan serta rintangan yang dihadapi, namun pada akhirnya penulis dapat melaluinya berkat adanya bantuan, bimbingan, serta dukungan dari berbagai pihak. Untuk itu dalam kesempatan ini penulis mengucapkan terima kasih sebanyak-banyaknya kepada:

Keluarga penulis khususnya kepada orang tua penulis yang senantiasa memberikan bantuan baik berupa motivasi dan doa yang tiada hentinya kepada penulis selama penyusunan skripsi ini.

Dr. Intan Nurma Yulita, S.T., M.T., selaku pembimbing utama dan Aditya Pradana, S.T., M.Eng., selaku pembimbing pendamping yang telah meluangkan waktu dan pikirannya sehingga penulis dapat menyelesaikan skripsi ini.

Prof. Dr. Iman Rahayu, S.Si., M.Si., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dr. Setiawan Hadi, M.Sc.CS., selaku Kepala Departemen Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran.

Dr. Juli Rejito, M.Kom., selaku Ketua Program Studi S-1 Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Padjadjaran

Segenap dosen pengajar dan tata usaha Teknik Informatika Unpad yang telah mengajar, memberikan ilmu, pendidikan, dan pengetahuan kepada penulis selama masa perkuliahan.

Teman-teman Mahasiswa Teknik Informatika 2018 dan Keluarga Besar Himatif FMIPA Unpad atas kebersamaan dan pengalaman yang sangat berharga.

Peneliti tentunya menyadari bahwa skripsi ini masih terdapat banyak kekurangan. Oleh karena itu penulis berharap kepada semua pihak agar dapat menyampaikan kritik dan saran yang membangun untuk menambah kesempurnaan skripsi ini. Akhir kata, penulis juga berharap semoga skripsi ini dapat bermanfaat bagi pembaca.

*Wassalamu’alaikum Warahmatullahi Wabarakatuh*

Jatinangor, ............ 2022

Penulis

# ABSTRAK

Penggunaan internet saat ini

**Kata Kunci**: Analisis,

# ABSTRACT

*The use of the internet*

***Keywords:*** *Analysis*

# DAFTAR ISI

[KATA PENGANTAR iii](#_Toc102661673)

[ABSTRAK v](#_Toc102661674)

[ABSTRACT vi](#_Toc102661675)

[DAFTAR ISI vii](#_Toc102661676)

[DAFTAR TABEL xi](#_Toc102661677)

[DAFTAR GAMBAR xii](#_Toc102661678)

[DAFTAR LAMPIRAN xiii](#_Toc102661679)

[BAB I 1](#_Toc102661680)

[1.1 Latar Belakang 1](#_Toc102661681)

[1.2 Identifikasi Masalah 2](#_Toc102661682)

[1.3 Batasan Masalah 3](#_Toc102661683)

[1.4 Maksud dan Tujuan Penelitian 3](#_Toc102661684)

[1.5 Manfaat Penelitian 4](#_Toc102661685)

[1.6 Metodologi Penelitian 4](#_Toc102661686)

[1.7 Sistematika Penulisan 6](#_Toc102661687)

[BAB II 8](#_Toc102661688)

[2.1 Landasan Teori 8](#_Toc102661689)

[2.1.1 *Hate Speech* 8](#_Toc102661690)

[2.1.2 *Machine Learning* 9](#_Toc102661691)

[2.1.3 *Deep Learning* 9](#_Toc102661692)

[2.1.4 *Natural Language Processing* 10](#_Toc102661693)

[2.1.5 *Text Classification* 11](#_Toc102661694)

[2.1.6 *Text Preprocessing* 11](#_Toc102661695)

[2.1.7 *Bidirectional Encoder Representations from Transformers* (BERT) 14](#_Toc102661696)

[2.1.8 *Indonesian Bidirectional Encoder Representations from Transformers* (IndoBERT) 15](#_Toc102661697)

[2.1.9 *Confusion Matrix* 16](#_Toc102661698)

[2.1.10 Python 18](#_Toc102661699)

[2.2 Penelitian Sebelumnya 21](#_Toc102661700)

[BAB III 24](#_Toc102661701)

[3.1 Fase Persiapan dan Perencanaan Kebutuhan 24](#_Toc102661703)

[3.1.1 Objek Penelitian 24](#_Toc102661704)

[3.1.2 Perangkat Lunak 25](#_Toc102661705)

[3.1.3 Perangkat Keras 25](#_Toc102661706)

[3.1.4 Pengumpulan dan Persiapan Data 26](#_Toc102661707)

[3.2 Fase Perancangan Sistem 29](#_Toc102661708)

[3.2.1 Pengolahan Data 30](#_Toc102661709)

[3.2.2 Pembagian Data 33](#_Toc102661710)

[3.2.3 Klasifikasi dan Pemodelan Data 35](#_Toc102661711)

[3.3 Fase Evaluasi dan Pengujian Model 44](#_Toc102661712)

[3.3.1 Evaluasi Model 45](#_Toc102661713)

[BAB IV 49](#_Toc102661714)

[4.1 Hasil Pengumpulan dan Persiapan Data 49](#_Toc102661716)

[4.2 Hasil Pengolahan Data 50](#_Toc102661717)

[4.3 Hasil Pembagian Data 57](#_Toc102661718)

[4.4 Fase Evaluasi dan Pengujian Model 61](#_Toc102661719)

[4.4.1 a 63](#_Toc102661720)

[4.5 Fase Implementasi Sistem 63](#_Toc102661721)

[4.5.1 a 63](#_Toc102661722)

[BAB V 64](#_Toc102661723)

[5.1 Kesimpulan 64](#_Toc102661725)

[5.2 Saran 64](#_Toc102661726)

[DAFTAR PUSTAKA 65](#_Toc102661727)

[LAMPIRAN 71](#_Toc102661728)

# DAFTAR TABEL

[Tabel 2.1 *Confusion Matrix* 16](#_Toc102662075)

# DAFTAR GAMBAR

[Gambar 2.1 Hubungan antara *Artificial Intelligence*, *Machine Learning*, dan *Deep Learning* (Chollet, 2018) 10](#_Toc102662184)

[Gambar 2.2 Arsitektur BERT 15](#_Toc102662185)

# DAFTAR LAMPIRAN

# BAB I

**PENDAHULUAN**

## Latar Belakang

Saat ini penggunaan internet paling banyak digunakan untuk mengakses media sosial. Dengan adanya media sosial, setiap orang bisa saling berbagi informasi terhadap orang lain tanpa harus bertemu langsung satu dengan yang lainnya dan setiap orang juga memiliki kebebasan untuk mengemukakan pendapatnya di media sosial. Namun pada kenyataannya banyak juga masyarakat yang menyalahgunakan media sosial dengan memposting mengenai kalimat bernada ujaran kebencian atau *hate speech* seperti tuduhan, fitnah, berita *hoax*, ancaman, hinaan, maupun SARA, dimana postingan seperti ini tentunya sangat meresahkan bagi masyarakat.

Dengan merebaknya postingan pengguna media sosial di Indonesia yang bernada ujaran kebencian atau *hate speech*, membuat suatu sistem yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia merupakan langkah yang sangat tepat agar masyarakat tidak lagi sembarangan dalam memposting kalimat bernada ujaran kebencian atau *hate speech*. Alasan perlu dibuatnya sistem ini agar mempermudah masyarakat dalam menentukan kalimat mana yang termasuk *hate speech* atau *non-hate speech* secara otomatis karena dalam setiap waktu tentunya banyak pengguna media sosial di Indonesia yang memposting kalimat bernada *hate speech* sehingga masyarakat tidak perlu mengklasifikasikan setiap postingan yang ada satu per satu secara manual.

Untuk itu pada penelitian akan membuat sistem deteksi *hate speech* pada teks berbahasa Indonesia dengan metode IndoBERT yang digunakan untuk mengklasifikasikan teks mana yang bermakna *hate speech* dan *non-hate speech*. Sistem deteksi *hate speech* adalah sistem yang digunakan untuk mendeteksi apakah teks berbahasa Indonesia yang ingin diklasifikasikan merupakan *hate speech* atau *non-hate speech*. Sistem ini akan berfokus pada teks berbahasa Indonesia karena banyak postingan pengguna media sosial di Indonesia yang bernada *hate speech*. Alasan penggunaan IndoBERT karena terdapat model *pre-trained* berbahasa Indonesia yang disediakan jadi tidak perlu melakukan *train* dari awal dan tidak membutuhkan *data training* yang besar sehingga dapat mempersingkat waktu *train*. Berdasarkan hasil *benchmark* dengan metode lain yang dilakukan (Koto, Rahim, Lau, & Baldwin, 2020) didapatkan hasil bahwa penggunaan IndoBERT mendapatkan hasil yang lebih baik dibanding dengan metode lain seperti Bi-LSTM, mBERT, dan MalayBERT.

## Identifikasi Masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya, maka dapat dibuat identifikasi masalah sebagai berikut:

Bagaimana cara mengklasifikasikan teks berbahasa Indonesia yang bermakna *hate speech*?

Bagaimana cara mengimplementasikan metode IndoBERT dalam membuat suatu sistem yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia?

Bagaimana cara membuat sistem berbasis *website* yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia?

## Batasan Masalah

Berdasarkan identifikasi masalah yang telah dijelaskan sebelumnya dan untuk mendapatkan hasil sesuai yang diharapkan, maka dapat dibuat batasan masalah sebagai berikut:

1. Penelitian ini mencoba untuk membuat sistem berbasis *website* untuk mendeteksi kalimat bernada *hate speech* pada teks berbahasa Indonesia dengan metode yang digunakan yaitu IndoBERT.
2. Dataset yang digunakan didapat dari penelitian yang dilakukan oleh (Alfina, Mulia, Fanany, & Ekanata, 2017), (Putri, 2018), dan (Ibrohim & Budi, 2019).
3. Membagi label menjadi 4 jenis klasifikasi yaitu *weak* *hate speech*, *moderate* *hate speech*, *strong* *hate speech*, dan *non-hate speech*.

## Maksud dan Tujuan Penelitian

Maksud dari penelitian ini adalah melihat hasil dari sistem deteksi *hate speech* pada teks berbahasa Indonesia menggunakan metode IndoBERT.

Tujuan dari penelitian ini adalah:

1. Dapat mengklasifikasikan teks berbahasa Indonesia menjadi 4 jenis klasifikasi yaitu *weak* *hate speech*, *moderate* *hate speech*, *strong* *hate speech*, dan *non-hate speech*.
2. Untuk mengimplementasikan metode IndoBERT dalam membuat suatu sistem yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia.
3. Untuk membuat sistem berbasis *website* yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia.

## Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Menghasilkan sistem berbasis *website* yang dapat mendeteksi *hate speech* pada teks berbahasa Indonesia.
2. Mengurangi jumlah pengguna yang menyalahgunakan media sosial dengan memposting kalimat bernada ujaran kebencian atau *hate speech*.
3. Membantu masyarakat dan aparat hukum agar dapat lebih mudah untuk mengklasifikasikan postingan yang bermakna *hate speech* dan *non-hate speech* di media sosial.

## Metodologi Penelitian

Sistem yang akan dirancang pada penelitian ini merupakan sistem berbasis *website* yang dapat mendeteksi kalimat *hate speech* pada teks bahasa Indonesia. Alur perancangan sistem pada penelitian ini dapat dilihat pada diagram alur (*flowchart*) di bawah ini.



Gambar 1.1 Diagram Alur (*Flowchart*) Perancangan Sistem

Berdasarkan diagram alur (*flowchart*) perancangan sistem di atas, tahapan yang akan dilakukan pada penelitian ini adalah sebagai berikut:

1. Tahap pendahuluan, yaitu tahapan awal pada penelitian ini yang bertujuan untuk menentukan topik permasalahan, latar belakang, identifikasi masalah, batasan masalah, maksud, tujuan, dan manfaat masalah.
2. Tahap studi literatur, yaitu tahapan yang dilakukan penulis dengan mencari referensi dan informasi dari buku, jurnal, maupun artikel yang mendukung penelitian ini.
3. Tahap persiapan dan perencanaan kebutuhan, yaitu tahapan yang dilakukan penulis untuk mempersiapkan segala kebutuhan yang diperlukan dalam penelitian ini, seperti perangkat lunak, perangkat keras, menyiapkan dataset, dan lain-lain.
4. Tahap perancangan sistem, yaitu tahapan yang dilakukan penulis untuk merancang *flowchart* perancangan sistem serta mengolah data yang telah didapatkan pada tahapan sebelumnya, kemudian memisahkan data menjadi 3 bagian yang akan digunakan sebagai data *train*, *testing*, dan *validation*, lalu membuat model dengan *pre*-*trained* IndoBERT yang tersedia.
5. Tahap evaluasi dan pengujian model, yaitu tahapan yang dilakukan penulis untuk mengevaluasi hasil dari pemodelan yang sudah dibuat pada tahapan sebelumnya dengan menguji performa dari sistem menggunakan metrik pengukuran atau *performance metrics* yang ada yaitu *accuracy*, *precision*, *recall*, dan *F-measure* atau *F1-score*. Setelah itu, hasil evaluasi tersebut diuji dengan dibuatnya sistem berbasis *website* untuk menganalisis dan menguji hasil dari model yang sudah dibuat sebelumnya.
6. Tahap penulisan laporan, yaitu tahapan yang dilakukan penulis dengan menuliskan hasil dan mendokumentasikan proses penelitian agar tidak ada proses yang terlewatkan ke dalam laporan skripsi.

## Sistematika Penulisan

Untuk memberi gambaran yang jelas tentang penelitian ini, maka disusunlah sistematika penulisan yang berisi materi yang akan dibahas pada setiap bab. Sistematika dalam penulisan tugas akhir ini adalah sebagai berikut:

**BAB I PENDAHULUAN**

Pada bab ini dijelaskan tentang latar belakang dari topik penulisan skripsi, pokok permasalahan berupa identifikasi masalah, batasan masalah, maksud, tujuan, dan manfaat yang diharapkan dari penulisan skripsi, metodologi penelitian yang digunakan, serta sistematika penulisan.

**BAB II TINJAUAN PUSTAKA**

Pada bab ini dijelaskan seluruh tinjauan pustaka atau kajian literatur yang berhubungan dengan penelitian dan menunjang dalam proses pengerjaan skripsi. Tinjauan pustaka atau kajian literatur tersebut didapatkan dari studi pada penelitian sebelumnya, buku, artikel internet, dan jurnal terkait yang dapat mendukung pemecahan masalah dengan topik yang akan dibahas dalam penelitian ini..

**BAB III METODOLOGI PENELITIAN**

Pada bab ini dijelaskan tentang metode pengembangan aplikasi yang digunakan meliputi analisis kebutuhan sistem, perancangan aplikasi, diagram pemodelan sistem, model perancangan data dan rancangan antarmuka pengguna.

**BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini dijelaskan tentang implementasi aplikasi yang telah dibangun, tampilan aplikasi, dan pengujian aplikasi.

**BAB V KESIMPULAN DAN SARAN**

Bab ini merupakan penutup yang berisi kesimpulan dan saran dari penelitian yang sudah dilakukan.

# BAB II

**TINJAUAN PUSTAKA**

## Landasan Teori

Pada bab ini dibahas tentang seluruh landasan teori yang berhubungan dan menunjang dalam proses penelitian ini. Landasan teori tersebut berisi metode yang digunakan, penjelasan teoritis mengenai algoritma yang digunakan dalam proses pengembangan sistem, dan teori lainnya yang berguna untuk memahami permasalahan yang dibahas. Landasan teori yang ada didapatkan dari studi pada penelitian sebelumnya, buku, artikel internet, dan jurnal terkait yang dapat mendukung pemecahan masalah dengan topik yang akan dibahas dalam penelitian ini.

### *Hate Speech*

Ujaran kebencian atau *hate speech* adalah tindakan komunikasi yang dilakukan oleh suatu individu atau kelompok dalam bentuk provokasi, hasutan, ataupun hinaan kepada individu atau kelompok lain dalam hal berbagai aspek seperti ras, warna kulit, jenis kelamin, cacat, orientasi seksual, kewarganegaraan, agama dan lain-lain (Prawira, 2016).

Kejahatan ujaran kebencian dapat dilakukan melalui berbagai media, antara lain (Prawira, 2016) :

1. Dalam orasi kegiatan kampanye.
2. Spanduk atau *banner*.
3. Jejaring media sosial.
4. Penyampaian pendapat di muka umum (demonstrasi).
5. Ceramah keagamaan.
6. Media masa, cetak, ataupun elektronik.

### *Machine Learning*

*Machine Learning* adalah salah satu bidang ilmu di *Artificial Intelligence*. *Machine Learning* bertujuan untuk membuat mesin dilatih dengan banyak contoh atau dataset yang berhubungan dengan tugas yang dibutuhkan. Mesin mempelajari pola-pola yang diberikan berdasarkan dataset dan menghasilkan sebuah aturan sendiri. Sehingga ketika suatu data dimasukkan ke dalam mesin, mesin sudah dapat mengenali data tersebut (Fimoza, 2021).

*Machine Learning* merujuk pada sebuah metode yang membuat komputer memiliki kemampuan dalam mempelajari dan melakukan sebuah pekerjaan secara otomatis. Proses *Machine Learning* dilakukan melalui algoritma tertentu, sehingga pekerjaan yang diperintahkan kepada komputer dapat dilakukan secara otomatis (Hertzmann & Fleet, 2010).

### *Deep Learning*

*Deep Learning* adalah salah satu bidang *Machine Learning* yang memanfaatkan banyak *layer* pengolahan informasi nonlinier untuk melakukan ekstraksi fitur, pengenalan pola, dan klasifikasi (Deng & Yu, 2014)*. Deep Learning* bekerja untuk mempelajari sehingga tidak hanya dapat memprediksi tetapi juga merepresentasikan data dengan benar, sehingga cocok untuk melakukan prediksi (Goldberg, 2017).

Pendekatan *Deep Learning* memungkinkan komputer untuk mempelajari suatu model yang rumit dengan cara membangun suatu model yang sederhana (Goodfellow, Bengio, & Courvil, 2016). Berdasarkan hal tersebut, *Deep Learning* dapat dikatakan sebagai turunan dari *Machine Learning*, dimana metode ini terdiri dari banyak tingkatan proses informasi *non-linear* dan abstraksi untuk dapat melakukan *supervised*, *unsupervised learning* dan *representation*, klasifikasi, dan pengenalan pola (Deng & Yu, 2014).

Diagram, venn diagram

Description automatically generated

Gambar 2.1 Hubungan antara *Artificial Intelligence*, *Machine Learning*, dan *Deep Learning* (Chollet, 2018)

### *Natural Language Processing*

*Natural Language Processing* (NLP) merupakan salah satu cabang ilmu AI yang berfokus pada pengolahan bahasa natural. Bahasa natural adalah bahasa yang secara umum digunakan oleh manusia dalam berkomunikasi satu sama lain. Bahasa yang diterima oleh komputer butuh untuk diproses dan dipahami terlebih dahulu supaya maksud dari *user* bisa dipahami dengan baik oleh komputer (Jurafsky & Martin, 2008).

Ada berbagai contoh penerapan aplikasi dari NLP, contohnya yaitu *Chatbot* (aplikasi yang membuat *user* bisa seolah-olah melakukan komunikasi dengan komputer), *Stemming* atau *Lemmatization* (pemotongan kata dalam bahasa tertentu menjadi bentuk dasar pengenalan fungsi setiap kata dalam kalimat), *Summarization* (ringkasan dari bacaan), *Translation Tools* (menerjemahkan bahasa) dan aplikasi-aplikasi lain yang memungkinkan komputer mampu memahami instruksi bahasa yang diinputkan oleh *user* (Hidayat, Sujaini, & Dwinyoto, 2015).

### *Text Classification*

*Text Classification* atau klasifikasi teks merupakan salah satu bidang dari Pengolahan Bahasa Alami atau lebih dikenal sebagai *Natural Language Processing* (NLP) yang mengotomatisasikan pengklasifikasian teks ke satu atau lebih kategori yang tepat berdasarkan isinya dengan membangun model menggunakan data latih (Mowafy, Rezk, & El-Bakry, 2018).

Klasifikasi teks diterapkan dalam berbagai konteks, mulai dari pengindeksan dokumen berdasarkan kosa kata yang terkontrol, penyaringan dokumen, pembuatan *metadata* otomatis, dan berbagai aplikasi lainnya yang membutuhkan organisir dokumen (Sebastiani, 2001). Ada beberapa strategi umum dalam penggunaan klasifikasi teks, yaitu *text preprocessing*, *feature extraction*, *modeling* menggunakan teknik pembelajaran mesin yang sesuai, serta *training* dan *testing* pada *classifier* (Dalal & Zaveri, 2011).

### *Text Preprocessing*

*Text preprocessing* merupakan tahapan proses awal terhadap teks untuk mempersiapkan teks menjadi data yang akan diolah selanjutnya (Feldman & Sanger, 2007). *Text preprocessing* digunakan untuk mempersiapkan teks sebelum digunakan dalam pengujian atau pelatihan dengan tujuan untuk mengurangi *noise* yang ada pada data sehingga dapat meningkatkan kinerja *classifier* dan mempercepat proses klasifikasi (Sivakumar & Gunasundari, 2017).

Langkah-langkah *text* *preprocessing* pada penelitian yang dilakukan penulis berupa:

1. *Data Cleaning*

*Data Cleaning* atau pembersihan data adalah suatu teknik untuk menangani adanya data yang tidak lengkap (Han, Kamber, & Pei, 2011). Jika proses pembersihan data tidak dilakukan maka kualitas dari analisis data menjadi berkurang atau kurang akurat. Oleh sebab itu diperlukanlah pembersihan data tersebut.

Ada beberapa kriteria suatu data yang harus dibersihkan, yaitu (Han, Kamber, & Pei, 2011):

1. *Missing Value*, merupakan nilai atribut yang kosong pada objek data. Biasanya disebabkan karena kesalahan saat proses pengumpulan data. Beberapa cara penanganannya adalah dengan menghapus baris yang datanya tidak tersedia atau teknik menghitung nilai pengganti (imputasi).
2. *Noise*, merupakan kesalahan acak (*random error*) dalam sebuah variabel pada objek data. Cara penanganannya dengan cara penghalusan data (*smoothing*).
3. *Outlier*, merupakan sebuah karakteristik yang berbeda dari data lainnya pada suatu dataset. *Outlier* mempunyai informasi yang berguna untuk menganalisa kejadian-kejadian yang sering terjadi.
4. *Inconsistent*, merupakan penggunaan kode yang tidak konsisten. Hal ini disebabkan karena perbedaan sumber data yang didapatkan.
5. *Case Folding*

*Case folding* adalah tahapan proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil, serta menghilangkan karakter selain a-z (Mustaqhfiri, Abidin, & Kusumawati, 2011).

1. *Tokenizing*

*Tokenizing* adalah proses pemotongan string input berdasarkan tiap kata yang menyusunnya. Pemecahan kalimat menjadi kata-kata tunggal dilakukan dengan men-*scan* kalimat dengan pemisah (*delimiter*) *white space* (spasi, tab, dan *newline*) (Mustaqhfiri, Abidin, & Kusumawati, 2011).

1. *Normalization*

*Text Normalization* atau normalisasi teks adalah mengubah teks kalimat menjadi teks yang secara lengkap memperlihatkan cara pengucapannya. Normalisasi teks meliputi pengubahan singkatan, akronim, angka, tanggal, waktu, karakter-karakter khusus, dan simbol-simbol dengan bentuk huruf alphabet lengkap sehingga tidak terjadi ambiguitas berkenaan dengan cara pengucapan (Sproat, et al., 2001).

1. *Filtering*

*Filtering* adalah proses penghilangan *stopword*. *Stopword* adalah kata-kata yang sering kali muncul dalam dokumen namun artinya tidak deskriptif dan tidak memiliki keterkaitan dengan tema tertentu. Di dalam bahasa Indonesia, *stopword* dapat disebut sebagai kata tidak penting, misalnya “di”, ’’oleh”, “pada”, ’’sebuah”, ’’karena” dan lain sebagianya (Mustaqhfiri, Abidin, & Kusumawati, 2011).

1. *Stemming*

*Stemming* merupakan proses mencari akar (*root*) kata dari tiap token kata yaitu dengan pengembalian suatu kata berimbuhan ke bentuk dasarnya (*stem*) (Tala, 2003).

### *Bidirectional Encoder Representations from Transformers* (BERT)

*Bidirectional Encoder Representations from Transformers* atau BERT adalah algoritma *Deep Learning* yang dirancang untuk mengolah *Natural Language Processing* atau NLP yang dikembangkan oleh para peneliti di Google AI *Language* pada tahun 2018.

*Transformer* adalah sebuah mekanisme yang mempelajari hubungan kontekstual antara kata-kata dalam teks (Vaswani, et al., 2017). Dalam bentuk, *transformer* menyertakan dua mekanisme terpisah, yaitu *encoder* yang membaca input teks dan *decoder* yang menghasilkan prediksi untuk tugas tersebut. Karena tujuan BERT adalah untuk menghasilkan model bahasa, hanya diperlukan mekanisme *encoder* (Putra, Bijaksana, & Romadhony, 2021). Arsitektur pada metode BERT dapat dilihat pada Gambar 2.2.

Diagram

Description automatically generated

Gambar 2.2 Arsitektur BERT

Model BERT dapat digunakan dengan dua cara, yaitu *pre-training* dan *fine-tuning*. Untuk dapat melakukan *fine-tuning*, model tersebut harus sudah di-*pre-train* dengan menggunakan dataset dengan bahasa yang sama. Model-model yang ada biasanya menggunakan bahasa yang telah memiliki korpus yang besar dan siap digunakan (Wilie, et al., 2020).

### *Indonesian Bidirectional Encoder Representations from Transformers* (IndoBERT)

IndoBERT merupakan salah satu nonlingual model BERT untuk bahasa Indonesia. IndoBERT sendiri merupakan modifikasi dari BERT-Base yang sudah ada dengan mengikuti kofigurasi dari BERT-Base (*uncased*) yang memiliki 12 *hidden layers* masing masing memiliki 768d, 12 *attention heads*, dan *feed-forward hidden layers* 3,072d (Koto, Rahim, Lau, & Baldwin, 2020).

Jika di total IndoBERT di-*train* dengan lebih dari 220 juta kata. Terdiri dan 3 sumber utama, yaitu Wikipedia Indonesia (74 juta kata), artikel Kompas, Tempo, dan Liputan6 (total 55 juta), dan Web Corpus Indonesia (90 juta kata) (Koto, Rahim, Lau, & Baldwin, 2020).

### *Confusion Matrix*

*Confusion Matrix* merupakan sebuah metode untuk mengukur performa dan melakukan evaluasi terhadap model klasifikasi pada *Machine Learning*. *Confusion Matrix* berupa tabel yang menyatakan klasifikasi dan jumlah data uji benar dan data uji salah yang memuat informasi hasil perbandingan data prediksi dan data aktual (Terrada, Cherradi, Raihani, & Bouattane, 2019). Evaluasi dengan menggunakan metode *confusion matrix* menghasilkan nilai *accuracy*, *precision*, *recall*, *F1-score*, dan *support*. Model *Confusion Matrix* dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Confusion Matrix*

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Actual Class | |
| Positive (P) | Negative (N) |
| Predicted Class | Positive (P) | True Positive (TP) | False Positive (FP) |
| Negative (N) | False Negative (FN) | True Negative (TN) |

Keterangan:

1. *True Positive* (TP) merupakan banyaknya data yang kelas aktualnya adalah kelas positif dengan kelas prediksinya merupakan kelas positif.
2. *True Negative* (TN) merupakan banyaknya data yang kelas aktualnya adalah kelas negatif dengan kelas prediksinya merupakan kelas negatif.
3. *False Positive* (FP) merupakan banyaknya data yang kelas aktualnya adalah kelas negatif dengan kelas prediksinya merupakan kelas positif.
4. *False Negative* (FN) merupakan banyaknya data yang kelas aktualnya adalah kelas positif dengan kelas prediksinya merupakan kelas negatif.

*Confusion Matrix* dapat digunakan untuk menghitung berbagai *performance metrics* untuk mengukur kinerja model yang telah dibuat. Beberapa *performance metrics* yang biasa dan sering digunakan yaitu *accuracy*, *precission*, *recall*, dan *F1-score*.

Nilai *accuracy* merupakan metode pengujian berdasarkan tingkat kedekatan antara nilai prediksi dengan nilai aktual. Nilai *accuracy* digunakan untuk menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Dengan mengetahui jumlah data yang diklasifikasikan secara benar maka dapat diketahui akurasi hasil prediksi. Perhitungan nilai *accuracy* dapat dilihat pada Persamaan 2.1.

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

Namun nilai *accuracy* memiliki beberapa kelemahan seperti kurang informatif dan bias, sehingga perlu digunakannya metrik pengukuran lain untuk membantu dalam pengukuran performa model yang dibuat, yaitu nilai *precision*, *recall*, dan *F1-score*.

Nilai *precision* (*positive predictive value*) merupakan metode pengujian dengan melakukan perbandingan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi yang terambil oleh sistem, baik yang relevan maupun tidak. Ini berarti dari semua data prediksi positif yang dibuat, berapa banyak data yang benar-benar positif. Nilai *precision* digunakan untuk mengukur pola positif yang diprediksi dengan benar (*True Positive*) dari total pola prediksi dalam kelas positif. Perhitungan nilai *precision* dapat dilihat pada Persamaan 2.2.

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

Nilai *recall* merupakan metode pengujian yang membandingkan jumlah informasi relevan yang didapatkan sistem dengan jumlah seluruh informasi relevan yang ada dalam koleksi informasi, baik yang terambil atau tidak terambil oleh sistem. Ini berarti dari semua data aktual positif yang ada, berapa banyak dari data tersebut yang diprediksi dengan benar sebagai positif. Nilai *recall* digunakan untuk mengukur rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Perhitungan nilai *recall* dapat dilihat pada Persamaan 2.3.

|  |  |  |
| --- | --- | --- |
|  |  | (2.3) |

Nilai *F1-score* merupakan perbandingan rata-rata *precision* dan *recall* yang dibobotkan. Perhitungan nilai *F1-score* dapat dilihat pada Persamaan 2.4.

|  |  |  |
| --- | --- | --- |
|  |  | (2.4) |

### Python

Python adalah bahasa pemrograman interpretatif, berorientasi objek, dan semantik yang dinamis yang dapat digunakan di berbagai *platform* khususnya *platform* yang berfokus pada keterbacaan kode. Python memiliki sintaks yang sederhana, mudah dipelajari, dan lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks (Python Software Foundation, 2001).

*Interpreter* Python dan standar *library*-nya tersedia secara gratis untuk semua *platform* dan dapat secara bebas disebarkan. Saat ini kode Python dapat dijalankan di berbagai sistem operasi, beberapa contoh diantaranya adalah Windows, Linux/Unix, Mac OSX, dan FreeBSD. Python saat ini dikembangkan dan dikelola oleh tim relawan yang besar dan tersedia secara gratis dari Python Software Foundation.

Untuk lebih mudah dan cepat dalam mengembangkan suatu sistem atau aplikasi berbasis Python, maka *programmer* dapat menggunakan *library* dalam kode Python-nya. *Library* adalah sekumpulan kode yang memiliki fungsi-fungsi tertentu yang biasanya terkumpul dalam sebuah *namespace* atau *module* atau *package* yang dapat dipanggil atau di-*import* atau digunakan kembali ke program lain. Dengam adanya *library*, *programmer* tidak harus membuat kode dari awal untuk menjalankan suatu fungsi tertentu.

Dalam pengembangan sistem deteksi *hate speech* ini, penulis menggunakan beberapa *library* agar mempermudah proses pengembangan sistem ini *Library* Python yang digunakan dalam pengembangan sistem ini yaitu:

1. Pandas, merupakan salah satu *library* dalam Python yang digunakan untuk membuat tabel, mengubah dimensi data, mengecek data, dan lain-lain. Struktur data dasar pada Pandas dinamakan DataFrame, yang memudahkan untuk membaca sebuah *file* berformat .csv, .tsv, dan .txt.
2. Numpy, merupakan salah satu *library* dalam Python yang digunakan untuk memudahkan operasi komputasi, menghitung operasi matematika pada *array*, dan mengolah data numerik.
3. Matplotlib, merupakan salah satu *library* dalam Python yang digunakan untuk membuat grafik dalam berbagai jenis, seperti grafik garis, batang, lingkaran, histogram, dan lainnya.
4. Re atau Regular Expression (Regex), merupakan salah satu *library* dalam Python yang digunakan untuk mencari, mengganti pola tertentu, dan mengganti kemunculan sub-string tertentu dengan sub-string lain.
5. NLTK atau *Natural Language Tool Kit*, merupakan salah satu *library* dalam Python yang digunakan untuk membantu dalam bekerja dengan teks, memproses teks seperti melakukan *classification*, *tokenization*, *stemming*, *tagging*, *parsing*, dan *semantic reasoning*.
6. Sastrawi, merupakan salah satu *library* dalam Python yang digunakan untuk mengubah kata berimbuhan bahasa Indonesia menjadi bentuk dasarnya.
7. Imbalanced-Learn merupakan salah satu *library* dalam Python yang membantu dalam menyeimbangkan kumpulan data yang sangat condong atau bias terhadap beberapa kelas.
8. Random, merupakan salah satu *library* dalam Python yang digunakan untuk menghasilkan nilai acak atau *random*.
9. PyTorch, merupakan salah satu *library* dalam Python yang digunakan untuk komputasi *Machine Learning*, melakukan perhitungan tensor dengan akselerasi GPU, membuat grafik komputasi dinamis, dan menghitung gradien secara otomatis. Penggunaan *library* ini sama dengan *library* lain yang sejenis seperti Tensorflow dan Keras.
10. Tqdm, merupakan salah satu *library* dalam Python yang digunakan untuk menampilkan *progress bar* dengan perulangan yang sederhana.
11. Transformers, merupakan salah satu *library* dalam Python yang menyediakan ribuan model terlatih untuk melakukan tugas pada modalitas yang berbeda seperti klasifikasi teks, visi, audio, ekstraksi informasi, tanya jawab, *summarization*, translasi, *text generation*, dan lain-lain dalam berbagai bahasa. Transformers didukung oleh dua *library Deep Learning*, yaitu PyTorch dan TensorFlow.
12. Seaborn, merupakan salah satu *library* dalam Python yang digunakan untuk membuat grafik dan statistik dengan menggunakan Python. *Library* ini dibangun berdasarkan *library* matplotlib serta terintegrasi dengan struktur data pada *library* pandas.
13. Scikit-learn atau sklearn, merupakan salah satu *library* dalam Python yang digunakan untuk membantu melakukan *processing* data ataupun melakukan *training* data untuk kebutuhan *Machine Learning*.
14. Os, merupakan salah satu *library* dalam Python yang digunakan untuk berinteraksi dengan sistem operasi.
15. Pickle, merupakan salah satu *library* dalam Python yang digunakan untuk menyimpan dan membaca data ke dalam atau dari sebuah *file*.

## Penelitian Sebelumnya

*Deep Learning* adalah Penelitian yang dilakukan oleh penulis ini dibangun berdasarkan beberapa referensi dari penelitian yang sudah dilakukan sebelumnya. Referensi yang didapat dari penelitian sebelumnya sangat penting untuk dilakukan agar penelitian ini terhindar dari kesamaan atau duplikasi dari penelitian sebelumnya. Hal ini dilakukan sebagai bentuk kontribusi penelitian bagi penulis agar penelitian mengenai topik ini terus berkembang.

Pada penelitian yang dilakukan oleh (Ibrohim & Budi, 2019), peneliti mencoba untuk mengklasifikasi teks multi-label untuk kalimat *abusive* dan deteksi *hate speech* menggunakan pendekatan *Support Vector Machine* (SVM), *Naive Bayes* (NB), dan *Random Forest Decision Tree* (RFDT) *classifier* dan *Binary Relevance* (BR), *Label Power-set* (LP), dan *Classifier Chains* (CC) sebagai metode transformasi data. Hasil percobaan menunjukkan bahwa secara umum RFDT *classifier* menggunakan LP sebagai metode transformasi memberikan akurasi terbaik dengan waktu komputasi yang paling cepat dibanding dengan pendekatan lainnya.

Pada penelitian yang dilakukan oleh (Wilie, et al., 2020), peneliti mengenalkan IndoNLU, yaitu sebuah koleksi sumber untuk riset dalam topik *Natural Language Understanding* (NLU) untuk Bahasa Indonesia. Dari penelitian ini dihasilkanlah 4 model IndoBERT, yaitu IndoBERTBASE, IndoBERTLARGE, IndoBERT-liteBASE, dan IndoBERT-liteLARGE.

Pada penelitian yang dilakukan oleh (Koto, Rahim, Lau, & Baldwin, 2020), peneliti memaparkan tentang IndoBERT dan IndoLEM. IndoBERT adalah model BERT *pre-trained* berbahasa Indonesia yang dilatih dengan *Transformer*, sebuah algoritma yang diadaptasi dari algoritma CNN. Sedangkan IndoLEM adalah dataset NLP berbahasa Indonesia yang dapat digunakan untuk berbagai tugas seperti tugas *labelling*, analisis sentimen, *text summarization*, dan prediksi *tweet*.

Pada penelitian yang dilakukan oleh (Putra, Bijaksana, & Romadhony, 2021), peneliti mencoba untuk mengimplementasikan metode IndoBERT untuk merancang sistem deteksi penggunaan kalimat *abusive* pada teks bahasa Indonesia. Data terdiri dari tiga label, yaitu *Not Abusive*, *Abusive Not Offensive*, dan *Abusive And Offensive*. Hasil pengujian menunjukkan IndoBERT dapat lebih baik dalam mengklasifikasikan kalimat *abusive* pada teks bahasa Indonesia. Di model BERT-base, berhasil menghasilkan nilai *F1-score* untuk semua kelas sebesar 0.6842.

# BAB III

**ANALISIS DAN PERANCANGAN**



## Fase Persiapan dan Perencanaan Kebutuhan

Fase persiapan dan perencanaan kebutuhan dilakukan untuk mencari tahu apa saja yang diperlukan untuk merancang sistem ini. Hal yang dilakukan adalah dimulai dari menentukan objek penelitian, menganalisis kebutuhan perangkat lunak, menganalisis kebutuhan perangkat keras, dan menganalisis kebutuhan *library* Python yang digunakan dalam merancang sistem ini.

### Objek Penelitian

Objek penelitian adalah sasaran ilmiah untuk mendapatkan data dengan tujuan dan kegunaan tertentu tentang suatu hal objektif, valid, dan realiabel tentang suatu hal (variabel tertentu) (Sugiyono, 2012). Objek penelitian penting dalam penelitian karena merupakan tujuan yang dicapai untuk memperoleh solusi dari permasalahan yang terjadi. Dengan kata lain, objek penelitian adalah sesuatu yang menjadi fokus dalam penelitian. Objek penelitian akan dibahas dan dianalisis oleh penulis berdasarkan teori-teori yang sesuai dengan objek penelitian.

Objek penelitian ini adalah salah satu model nonlingual bahasa Indonesia yang dirancang untuk mengolah NLP berbasis *Deep Learning*, yaitu IndoBERT (*Indonesian Bidirectional Encoder Representations from Transformers*) untuk mengklasifikasikan teks atau kalimat yang bermakna *hate speech* dan *non-hate speech*. Model *pre-trained* ini digunakan untuk memprediksi data baru, yaitu data teks yang dimasukkan oleh pengguna pada sistem deteksi *hate speech* berbahasa Indonesia berbasis *website*.

### Perangkat Lunak

Kebutuhan perangkat lunak terdiri dari perangkat lunak yang digunakan dalam proses pengembangan sistem ini. Perangkat lunak yang digunakan yaitu:

1. Python, merupakan bahasa pemrograman yang digunakan untuk membangun sistem berbasis AI seperti *Data Science* dan *Machine Learning*.
2. Visual Studio Code, merupakan *source*-*code editor* yang dibuat olehMicrosoft untuk sistem operasi Windows, Linux, dan macOS yang akan membantu penulisan kode untuk membangun sistem sehingga lebih cepat dan mudah.
3. Google Colab, merupakan layanan perangkat lunak berbasis *cloud* yang pada dasarnya serupa dengan Jupyter Notebook yang dijalankan menggunakan peramban, seperti Mozilla Firefox dan Google Chrome.
4. Peramban (Google Chrome), digunakan untuk pengujian dan implementasi sistem.

### Perangkat Keras

Kebutuhan perangkat keras terdiri dari perangkat keras yang digunakan dalam proses pengerjaan penelitian ini. Spesifikasi perangkat keras yang digunakan yaitu:

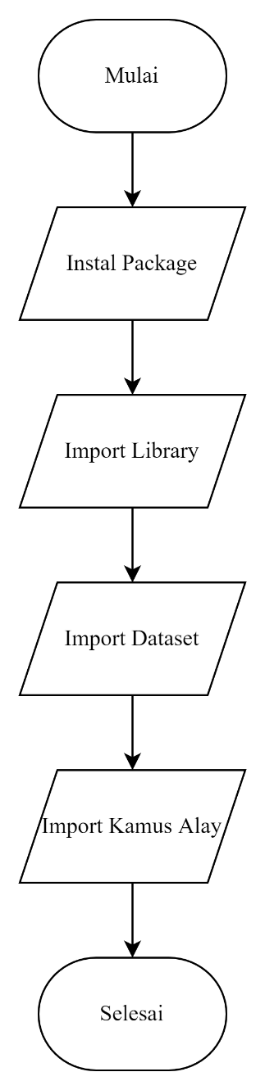
1. Sistem Operasi: Windows 10 Home
2. Prosesor: Intel Core i7-9750H
3. RAM: 16 GB DDR4
4. VGA: NVIDIA GeForce RTX 2060 6 GB

Sedangkan kebutuhan perangkat keras yang digunakan dalam proses pengembangan sistem ini menggunakan sumber daya komputasi yang disediakan oleh Google Cloud dengan spesifikasi yang digunakan yaitu:

1. Prosesor: Intel Xeon Processor @ 2.20 GHz
2. RAM: 13 GB
3. GPU: NVIDIA Tesla K80 12 GB GDDR5

### Pengumpulan dan Persiapan Data

Pengumpulan data adalah cara yang dilakukan peneliti untuk memperoleh data yang akan digunakan pada penelitian ini. Pengumpulan data memegang peranan yang sangat penting untuk mendapatkan informasi dari penelitian yang dilakukan dengan langkah-langkah yang dilakukan dapat dilihat pada *flowchart* di bawah ini.



Gambar 3.1 *Flowchart* Tahapan Persiapan Data

Langkah-langkah dalam pengumpulan data yang digunakan pada penelitian ini adalah sebagai berikut:

1. Studi Literatur

Pada bagian ini penulis melakukan pengumpulan data yang berhubungan dengan permasalahan penelitian yang bersumber dari studi pada penelitian sebelumnya, buku, artikel internet, dan jurnal terkait yang dapat mendukung pemecahan masalah dengan topik yang akan dibahas dalam penelitian ini.

1. Observasi

Pada bagian ini penulis melakukan pengamatan atau observasi secara langsung untuk mengumpulkan data yang dapat mendukung pemecahan masalah dengan topik yang akan dibahas dalam penelitian ini.

Data pada penelitian ini didapatkan dari dataset yang sudah dibangun dari penelitian (Alfina, Mulia, Fanany, & Ekanata, 2017), (Putri, 2018), dan (Ibrohim & Budi, 2019). Dataset didapatkan dengan cara melakukan *crawling* terhadap *tweet* menggunakan API Twitter yang diimplementasikan menggunakan *library* Tweepy. Kata kunci yang digunakan untuk melakukan *crawling* data adalah kata berbahasa Indonesia yang sering digunakan oleh masyarakat ketika memposting kalimat *hate speech* dan *abusive* di media sosial Twitter. Peneliti tersebut melakukan *crawling* terhadap data Twitter selama 6 bulan, dimulai dari tanggal 20 Maret 2018 sampai dengan tanggal 10 September 2018.

Selanjutnya setelah dataset didapat adalah melakukan persiapan data. Langkah pertama yang dilakukan adalah dengan melakukan konfigurasi Google Colab agar dapat membaca semua *file* yang ada di Google Drive. Langkah ini dilakukan agar semua *output file* yang dihasilkan setelah *code* dijalankan dapat langsung tersimpan di Google Drive.

Langkah selanjutnya yaitu menginstal *package* yang dibutuhkan agar nantinya *code* dapat dijalankan. Kelebihan yang dimiliki oleh Google Colab yaitu sudah tersedia banyak *package* sehingga tidak perlu melakukan instalasi *package* lagi, namun ada beberapa *package* yang belum tersedia sehingga perlu dilakukan instalasi pada *package* yang belum terinstal, seperti *package* NLTK, PySastrawi, Imbalance-Learn, dan Transformers.

Langkah selanjutnya yaitu mengimpor semua *library* Python yang dibutuhkan dalam pengembangan sistem deteksi *hate speech* ini. *Library* disini digunakan untuk membantu penulis agar dapat membangun sistem ini dengan lebih cepat dan efisien. *Library* Python yang digunakan pada pengembangan sistem ini yaitu:

1. Pandas, Numpy, Matplotlib, NLTK, Re, dan Sastrawi digunakan pada proses *preprocessing*.
2. Train Test Split digunakan pada proses *Hold-Out Cross Validation*.
3. TfidfVectorizer dan SMOTE digunakan pada proses *data balancing*.
4. Torch, Seaborn, Tqdm, dan Transformers digunakan pada proses pemodelan data.
5. Scikit-learn digunakan pada proses evaluasi model.

Langkah selanjutnya yaitu mengimpor dan membaca dataset yang telah didapat dengan menggunakan *library* pandas. Selanjutnya perbandingan jumlah data pada setiap label pada dataset dapat divisualisasikan dengan menampilkan diagram perbandingan jumlah data pada setiap label menggunakan *library* Matplotlib.

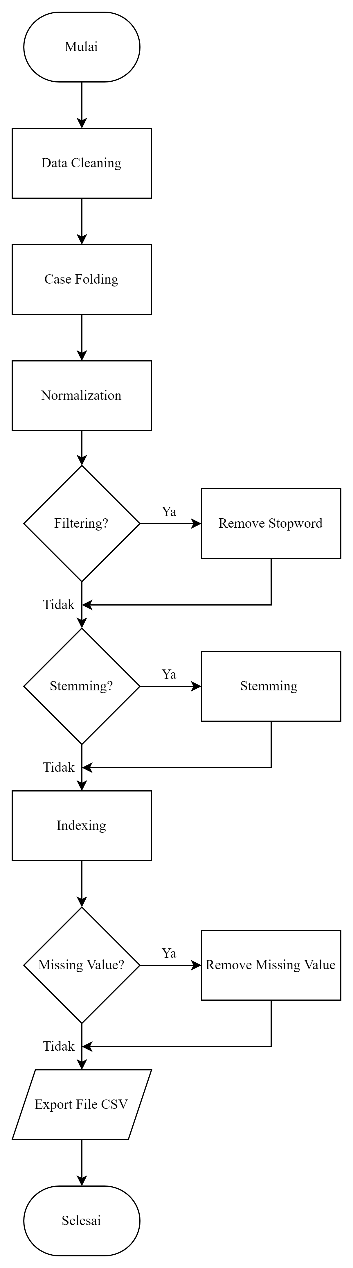
Langkah terakhir pada tahap persiapan data yaitu mengimpor kamus alay dari (Ibrohim & Budi, 2019) yang berisi kata-kata baku dalam bahasa Indonesia yang nantinya akan digunakan pada proses *normalization*.

## Fase Perancangan Sistem

Fase perancangan sistem berisi penjelasan alur pengembangan sistem yang dibuat beserta *flowchart*. Hal yang dilakukan adalah dimulai dari melakukan *preprocessing* data, *split up* dataset, dan pemodelan data.

### Pengolahan Data

Pengolahan data dilakukan agar data yang didapatkan sebelumnya dapat menjadi data yang lebih efisien, bermanfaat, dan memberikan hasil yang lebih baik untuk dapat diolah pada tahap selanjutnya. Pada tahap ini, hal yang dilakukan adalah dengan melakukan *preprocessing* data, pemberian label dan indeks, dan menghilangkan *missing value* pada dataset dengan langkah-langkah yang dilakukan dapat dilihat pada *flowchart* di bawah ini.



Gambar 3.2 *Flowchart* Tahapan Pengolahan Data

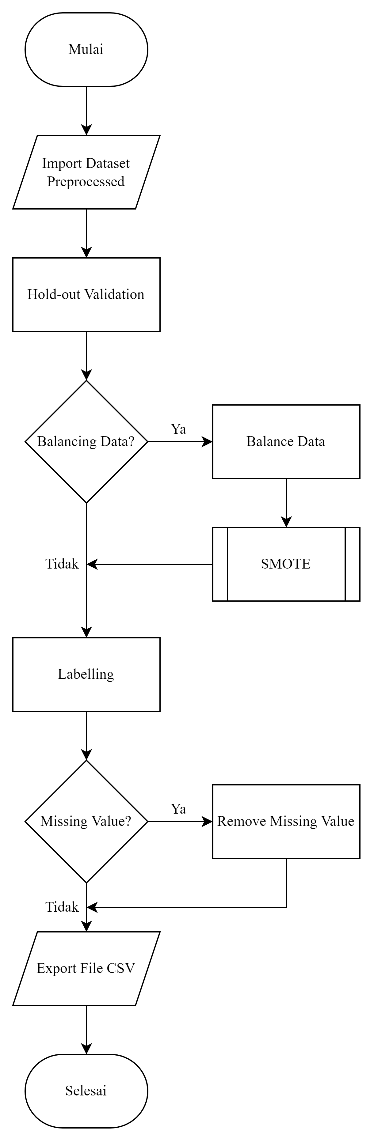
Pada penelitian ini, *preprocessing* data dilakukan untuk mengubah dataset yang awalnya tidak terstruktur menjadi terstruktur sehingga mempermudah data agar dapat lebih mudah untuk diproses. Langkah-langkah *preprocessing* data pada penelitian yang dilakukan penulis dimulai dari *data cleaning* yang merupakan tahap yang meliputi penghapusan URL atau *link*, *hashtag* (#), nama pengguna, *mention* (@), *retweet* (RT), tanda baca, baris baru, nomor, tanda baca, spasi kosong, emoji, dan transformasi beberapa sintaks HTML dengan menggunakan *regular expression*. Setelah itu, dilakukan proses *case folding* yang merupakan tahap untuk mengubah semua huruf menjadi huruf kecil (*lowercase*). Setelah itu, dilakukan proses *normalization* yang merupakan tahap untuk merubah kata dari yang awalnya tidak baku ke bentuk yang lebih baku menggunakan kamus kosakata baku yang diimpor dari (Ibrohim & Budi, 2019). Setelah itu, dilakukan proses *filtering* atau *stopword removal* yang merupakan tahap untuk menghilangkan *stopword* yang bertujuan untuk menghapus kata-kata yang sering bermunculan dan kata-kata yang tidak memiliki makna menggunakan *library* NLTK stopword dengan s*topword* yang digunakan didapat dari penelitian (Tala, 2003). Tahap terakhir pada *preprocessing* data yaitu dilakukan proses *stemming* yang merupakan tahap untuk mencari kata dasar dari tiap token kata yaitu dengan mengubah suatu kata berimbuhan atau menghilangkan imbuhan ke bentuk dasarnya dengan menggunakan *library* Sastrawi.

Tahap selanjutnya setelah dilakukannya *preprocessing* data yaitu pemberian label dan indeks. Untuk kelas *Non-Hate Speech* diberi label “Non\_HS” dengan indeks “0”, kelas *Weak Hate Speech* diberi label “HS\_Weak” dengan indeks “1”, kelas *Moderate Hate Speech* diberi label “HS\_ Moderate” dengan indeks “2”, dan kelas *Strong Hate Speech* diberi label “HS\_ Strong” dengan indeks “3”.

Tahap selanjutnya setelah pemberian label dan indeks adalah menghilangkan *missing value* pada dataset. *Missing Value* adalah hilangnya beberapa data yang telah diperoleh dari tahap *preprocessing* yang ditandai dengan adanya data dengan *value* *NaN* atau *Null*. Jika dibiarkan terdapat data dengan *value* yang hilang, maka dataset tidak akan dapat diproses pada tahap selanjutnya. Cara untuk mendeteksi apakah terdapat data *NaN* adalah dengan menggunakan fungsi *isnull()* pada *library* pandas, sedangkan cara penanganannya adalah dengan mengambil baris data yang memiliki *value* dengan menggunakan fungsi *notna()* pada *library* pandas.

### Pembagian Data

Data yang sudah diproses pada tahap sebelumnya akan dibagi menjadi 3 data, yaitu data *training*, *testing*, dan *validation* dengan langkah-langkah yang dilakukan dapat dilihat pada *flowchart* di bawah ini.



Gambar 3.3 *Flowchart* Tahapan Pembagian Data

Data *training* adalah data yang nantinya akan digunakan untuk melatih model yang sesuai, data *validation* adalah data yang nantinya akan digunakan untuk proses validasi model untuk mencegah terjadinya *overfitting*, sedangkan data *testing* adalah data yang nantinya akan digunakan untuk menguji dan mengetahui performa model yang didapatkan pada tahapan *training*. Tidak boleh adanya *overlap* antar subset data, karena akan merusak proses *training* model. Ukuran data *training* harus jauh lebih besar dari data *testing* dan *validation*, karena dengan data *training* yang terlalu kecil maka model tidak akan terlatih.

Proses pembagian data pada penelitian ini menggunakan metode *Hold-out Cross Validation*. *Hold-out Cross Validation* adalah salah satu metode yang digunakan untuk membagi data menjadi data *training*, *testing*, dan *validation*. Data keseluruhan yang digunakan berjumlah 13121 data dengan komposisi pembagian data menjadi 80% untuk *training*, 10% untuk *testing*, dan 10% untuk *validation*.

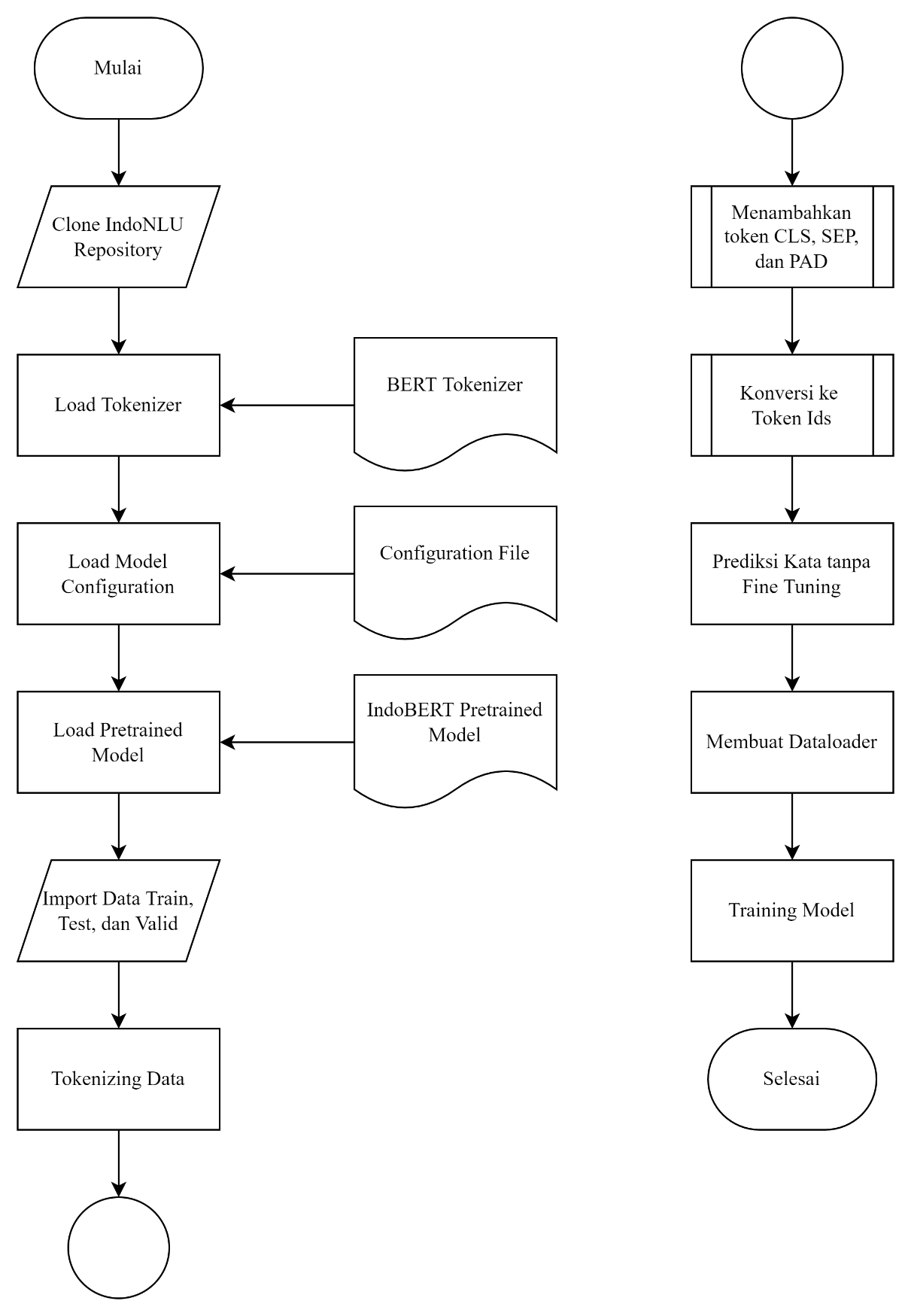
Pada tahap penelitian ini penulis melakukan 2 tahapan pengujian, yaitu bekerja dengan data yang tidak seimbang (*imbalance*) dan data yang seimbang (*balance*). Data *imbalance* merupakan keadaan dimana distribusi jumlah data di setiap kelas tidak seimbang atau tidak sama, sedangkan data *balance* merupakan keadaan dimana distribusi jumlah data di setiap kelas seimbang atau sama. Untuk menangani data *imbalance*, maka dilakukanlah tahap *data balancing* untuk menyeimbangkan jumlah data pada setiap label. Hal ini perlu dilakukan karena umumnya apabila kita melakukan proses klasifikasi pada data *imbalance* maka akan menghasilkan akurasi yang jelek dan cenderung memprediksi data ke kelas mayoritas. Salah satu tahap *data balancing* yang sering digunakan yaitu dengan metode *Synthetic Minority Over-sampling Technique* (SMOTE). SMOTE adalah salah satu metode yang digunakan untuk menangani data yang tidak seimbang dengan cara menambah jumlah data kelas minor agar sama atau setara dengan jumlah data kelas mayor, atau biasa disebut dengan *oversampling*.

Sebelum melakukan proses SMOTE, data pada kolom *tweet* diubah terlebih dahulu, dari yang awalnya berformat *string* diubah menjadi numerik. Hal tersebut dilakukan karena SMOTE hanya dapat dilakukan pada data numerik atau *float*. Data *string* dikonversi menjadi numerik menggunakan fungsi TfidfVectorizer. Setelah proses SMOTE dilakukan, data *tweet* yang berformat numerik tadi diubah kembali menjadi *string* dengan menggunakan fungsi inverse\_transform. Fungsi ini digunakan untuk mendapatkan kembali teks yang sebelumnya dikonversi menjadi numerik.

Setelah itu dilakukan kembali tahap pemberian label dan menghilangkan *missing value* yang mungkin ada pada saat setelah proses SMOTE. Setelah semua tahapan dilakukan, selanjutnya data disimpan dalam *file* *comma-separated values* (CSV) sehingga menghasilkan 3 data, yaitu data *training*, data *testing*, dan data *validation* yang akan digunakan pada tahap selanjutnya.

### Klasifikasi dan Pemodelan Data

Selanjutnya dilakukan proses klasifikasi teks bahasa Indonesia dan pemodelan data dengan menggunakan model *pre-trained* IndoBERTBASE yang didapat dari penelitian yang dilakukan oleh (Wilie, et al., 2020). Terdapat beberapa langkah yang dilakukan pada klasifikasi dan pemodelan data menggunakan metode IndoBERT dengan langkah-langkah yang dilakukan dapat dilihat pada *flowchart* di bawah ini.



Gambar 3.4 *Flowchart* Tahapan Pemodelan Data

Untuk melakukan klasifikasi dan pemodelan data, diperlukan konfigurasi dan model *pretrained* yang bisa didapatkan dari *repository* IndoNLU dengan cara melakukan *clone* *repository* IndoNLU terlebih dahulu. Setelah itu, muat semua sumber daya yang diperlukan, seperti BERT *Tokenizer*, konfigurasi model, model *pretrained*, dataset *training*, *testing*, dan *validation*. Agar model dapat digunakan untuk keperluan klasifikasi, diperlukan inisiasi terlebih dahulu dengan melakukan instalasi IndoBERT model *pre-trained* dengan *library* yang digunakan yaitu BertForSequenceClassification.

Selanjutnya pada tahapan klasifikasi, data yang didapat harus disesuaikan terlebih dahulu ke dalam bentuk representasi *input* yang dapat diterima dan dibaca oleh BERT. BERT tidak langsung menerima input berupa kalimat, tetapi kalimat tersebut akan melewati proses *embedding*. Proses *embedding* ini menggunakan tiga jenis *embedding*, yaitu *token embedding*, *segment embedding*, dan *positional embedding*.

Agar dapat dibaca oleh BERT, oleh karena itu diperlukan proses *tokenizing* pada data dengan menggunakan BERT *Tokenizer* yang berisi fungsi untuk melakukan *tokenizing*, yaitu proses untuk memisahkan setiap kata pada suatu kalimat menjadi token atau disebut *word tokenizing*. Untuk melakukan proses *tokenizing*, *tokenizer* akan mengecek apakah setiap kata pada kalimat yang dimasukkan ada pada kamus atau tidak. Jika ada, maka proses pemisahan kalimat menjadi kata per kata akan dilakukan. Jika tidak ada, *tokenizer* akan memisahkan kata menjadi sub kata yang kemungkinan kemunculannya paling besar pada kamus. Jika *tokenizer* tidak menemukan sub kata pada kamus, kata tersebut akan dipisah menjadi karakter per karakter. Cara terbaik untuk mengatasi hal tersebut adalah dengan memisahkan kata yang tidak ada pada kamus menjadi sub kata dengan simbol “##”. Hal ini dilakukan karena BERT menggunakan kamus yang spesifik yang dimana tergantung dengan model apa yang dipakai.

Disini BERT *Tokenizer* akan menambahkan beberapa token khusus, seperti token [CLS], [SEP], [PAD], [UNK], dan [MASK]. Token [CLS] adalah token yang digunakan di awal kalimat, sedangkan token [SEP] adalah token yang digunakan untuk memisahkan satu kalimat dengan kalimat selanjutnya. Kalimat yang sudah diberi token khusus ini akan menjadi *token embeddings*. Hal ini dilakukan sebagai penanda bahwa *hidden layer* dari arsitektur *Transformers* milik BERT akan melakukan pembelajaran klasifikasi teks, dimana token ini akan terus mengalami pembaharuan bobot dari setiap lapisan *Transformers*, sehingga hasil pembobotan akhir dari token ini yang akan menjadi *output* dari model *trained* (Devlin, Chang, Lee, & Toutanova, 2019). Lalu kalimat-kalimat tersebut disesuaikan dengan panjang maksimum yang telah ditentukan dengan memberi *padding* dengan token [PAD] yang penempatannya berada setelah token [SEP].

Setelah token [CLS], [SEP], dan [PAD] ditambahkan, BERT akan mengubah setiap token kata menjadi ID yang unik sesuai dengan kamus dan bilangan unik tersebut disimpan sebagai *token ids*. Token ID ini didapatkan ketika model di-*training* yang akan membuat setiap kata, sub kata, dan karakter pada kamus memiliki ID-nya masing-masing. Proses pengubahan kata menjadi ID ini dilakukan karena model *pre-trained* BERT hanya dapat mengerti ID dari token.

Untuk membedakan kalimat pertama dengan kalimat selanjutnya, dilakukanlah tahap *segment embeddings* dengan memberi angka 1 atau huruf A pada kalimat pertama, memberi angka 2 atau huruf B pada kalimat kedua, dan seterusnya. Tokenizer dapat mengetahui kalimat pertama dan kalimat selanjutnya dengan melihat token [SEP] yang berfungsi untuk memisahkan antara dua kalimat. Agar BERT mengetahui posisi dari setiap kata, dilakukanlah tahap *positional embeddings* untuk setiap token untuk menunjukkan posisi setiap kata pada kalimat. Hal ini dilakukan agar BERT tidak memperlakukan kata yang sama yang muncul lebih dari satu kali pada suatu kalimat dengan makna yang sama. Semua proses tersebut akan mengembalikan hasil *input ids* yang berisi token ID yang di-*input*-kan, dan *attention mask* yang berisi urutan *array* yang berisikan nilai 0 atau 1, yang dimana nilai 0 berarti token tersebut merupakan token dari hasil *padding*, sedangkan nilai 1 berarti token tersebut merupakan token dari teks asli. Hal ini bertujuan agar *self-attention layer* dari BERT tidak menghiraukan token *padding* sehingga tidak akan mempengaruhi hasil dari model yang di-*training*.

Setelah teks melalui tahap pemrosesan di atas, maka teks dapat langsung ditokenisasi dengan model *pre-trained* yang sudah ditentukan, yang dimana pada penelitian ini model yang ditentukan yaitu model *pre-trained* IndoBERTBASE. *Output* dari proses *tokenizing* ini adalah *array* *tensor* yang meliputi hasil *embedding* dari setiap kata berdasarkan pengenalan kata dari model *pre-trained* yang dimiliki.

Sebelum melakukan proses *training* model, dilakukan pembuatan DataLoader yang berguna agar membantu menghemat penggunaan memori, mempercepat proses pengambilan data, meningkatkan kecepatan *training* model, serta meminimalisir terjadinya *data loss*. DataLoader merupakan fungsi pada *library* PyTorch yang berperan sebagai *iterator* yang akan menggabungkan dataset dan *sample* pada proses iterasi. Karena dataset pada penelitian ini dibagi menjadi tiga, maka terdapat 3 DataLoader yang digunakan yaitu DataLoader *training*, DataLoader *validation*, dan DataLoader *testing*.

Setelah pembuatan DataLoader, tahapan yang dilakukan selanjutnya adalah mencoba untuk memprediksi contoh teks tanpa melakukan *training* model dan hanya mengandalkan model *pre-trained* dari IndoBERT. Hasil yang didapat akan dibandingkan dengan hasil prediksi teks dengan *trained* model yang didapatkan pada proses *training* model yang akan dilakukan pada tahap selanjutnya.

Selanjutnya adalah proses *training* model dengan melakukan *fine-tuning*. *Fine-tuning* adalah proses memuat model BERT yang telah dilatih untuk kemudian model tersebut dikonfigurasi untuk membuatnya dapat melakukan tugas serupa lainnya. Tujuan dilakukan *fine-tuning* adalah agar hasil *trained* model yang dihasilkan pada proses *training* model dapat menghasilkan *output* prediksi yang optimal. Disini akan dilakukan proses *fine-tuning* menggunakan varian model *pre-trained* IndoBERTBASE P1 (*Phase* 1 - *uncased*) dengan memanfaatkan *library* Transformers dari HuggingFace.

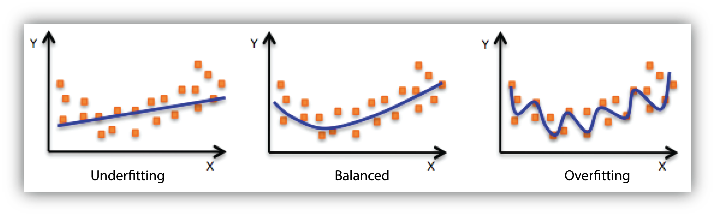
Agar hasil yang didapatkan pada proses *training* mendapatkan hasil yang optimal, diperlukan penentuan nilai *hyperparameter* yang tepat. Perlu menjadi catatan bahwa *parameter* dan *hyperparameter* merupakan dua variabel yang berbeda. *Parameter* adalah variabel dari suatu model yang nilainya akan diperbarui selama proses *training*, sedangkan *hyperparameter* adalah variabel yang nilainya digunakan untuk mengontrol atau menentukan bagaimana suatu model di-*training*.

Untuk menentukan nilai *hyperparameter* yang tepat, perlu mempertimbangkan juga antara kemampuan perangkat yang digunakan, waktu yang harus diluangkan untuk melakukan proses *training*, dan keoptimalan hasil yang didapatkan. Maka dari itu, pada penelitian ini akan dilakukan pencarian nilai *hyperparameter* terbaik.

Perlu diingat kembali bahwa nilai *hyperparameter* ini tidak diubah selama proses *training*. Maka dari itu hal yang dilakukan selanjutnya adalah melakukan beberapa percobaan untuk mendapatkan nilai *hyperparameter* yang tepat, salah satunya yaitu dengan melakukan *hyperparameter tuning*. *Hyperparameter tuning* adalah proses untuk menemukan nilai *hyperparameter* yang dapat menghasilkan model dengan performa terbaik. Menurut penelitian dari BERT yang dilakukan (Devlin, Chang, Lee, & Toutanova, 2019), ada tiga rekomendasi *hyperparameter* yang dapat disesuaikan untuk mengoptimalkan performa pada saat *fine-tuning*, di antaranya yaitu *batch size*, *learning rate*, dan *epoch*. *Batch size* adalah jumlah sampel yang disebarkan ke dalam *neural network*. Semakin besar nilai *batch size*, maka semakin lama juga waktu yang dibutuhkan untuk menyelesaikan satu *batch* yang artinya proses *training* yang dilakukan akan memakan memori yang semakin banyak. *Learning rate* adalah nilai yang akan menentukan seberapa banyak *weight* pada *neural network* yang akan diubah. Semakin besar nilai *learning rate*, maka ketelitian jaringan akan semakin berkurang yang artinya proses *training* yang dilakukan akan semakin cepat. *Epoch* adalah nilai yang akan menentukan berapa kali *neural network* melihat seluruh dataset. Semakin besar nilai *epoch*, maka semakin baik tingkat akurasi yang dihasilkan pada proses *training* yang artinya proses *training* yang dilakukan akan semakin lama. Perlu diingat juga bahwa jika nilai *epoch* diset terlalu besar maka akan menyebabkan terjadinya *overfitting*, begitu pula sebaliknya jika nilai *epoch* diset terlalu kecil maka akan menyebabkan terjadinya *underfitting*.

Sampai tahap ini, maka proses pemodelan data sudah bisa dilakukan dengan dimasukannya dataset yang telah dipersiapkan, dibersihkan, dan diberi label ke dalam model. Langkah pertama yang dilakukan adalah memuat *pre-trained* model IndoBERT yang akan dilatih beserta konfigurasi *hyperparameter* dari model tersebut. Berdasarkan konfigurasi *hyperparameter* tersebut, akan ditentukan berapa banyak iterasi dan kombinasi yang akan dijalankan untuk melatih model. Dalam menjalankan proses *training* tidak bisa dipungkiri bahwa akan ditemukan beberapa kesalahan atau *error* yang akan mempengaruhi hasil *training* model. Maka dari itu diperlukan algoritma atau metode optimalisasi yang dapat digunakan untuk meminimalkan *error* pada proses *training*, algoritma atau metode optimalisasi ini yang disebut sebagai *Gradient Descent*. Biasanya algoritma optimalisasi menggunakan *optimizer* untuk menentukan nilai-nilai parameter. Salah satu *optimizer* terbaik dan sering dipilih pada proses *training* adalah *Adaptive Moment Estimation* atau Adam. Adam dipilih sebagai *optimizer* yang digunakan pada proses *training* data dengan konfigurasi sesuai dengan pengaturan bawaan dari model BERT. *Hyperparameter* yang dikonfigurasi sesuai dengan hasil dari percobaan pencarian *hyperparameter* paling optimal pada tahap sebelumnya.

Selanjutnya dilakukan pemantauan terhadap nilai *accuracy* dan *loss* *performance metrics* dari data *training* dan data *validation* pada setiap *epoch* selama proses *training*. Jika konfigurasi *hyperparameter* yang ditentukan sebelumnya sudah optimal, maka akan meningkatkan nilai *accuracy* dan menurunkan nilai *loss* *performance metrics* untuk data *train* maupun data *validation*. Metode yang paling sering digunakan untuk memantau nilai *accuracy* dan *loss* *performance metrics* adalah menggunakan kurva pembelajaran atau *Learning Curve*. *Learning Curve* merupakan kurva yang dianggap sebagai alat yang efektif untuk memantau performa model yang akan menangani tugas baru ke depannya, dimana *Learning Curve* menyajikan representasi matematis dari proses pembelajaran yang terjadi selama iterasi tugas sebelumnya (Anzanello & Fogliatto, 2011). *Learning Curve* terdiri dari dua komponen perhitungan, yaitu *Training* *Learning Curve* dan *Validation* *Learning Curve*. *Training* *Learning Curve* diukur dari data *training* untuk memberikan gambaran seberapa baik model dalam melakukan *learning*. Sedangkan *Validation* *Learning Curve* diukur dari data *validation* yang terpisah dari data *training*. Di sini model akan di-*test* seberapa baik model dalam memprediksi suatu data baru berdasarkan hasil dari *learning* yang telah dilakukan. Berdasarkan hasil *fine-tuning* tersebut, maka model dapat dipersiapkan dengan lebih optimal sebelum masuk ke dalam fase evaluasi dan pengujian model. *Learning Curve* pada proses *fine-tuning* digambarkan pada Gambar 3.6.

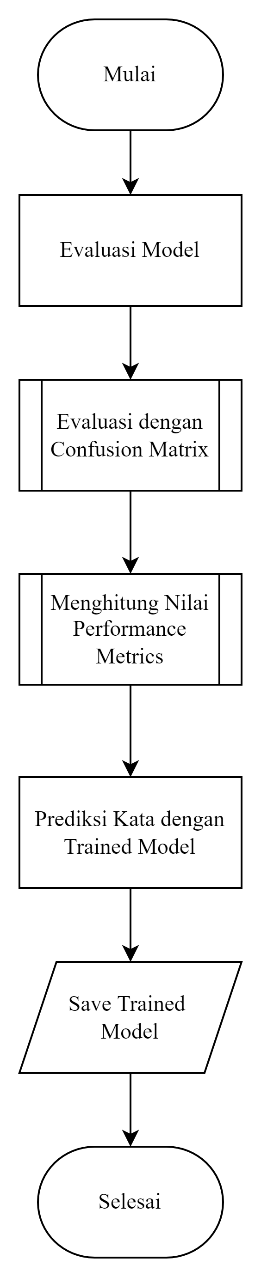


Gambar 3.5 *Learning Curve* pada F*ine-Tuning* (Amazon Web Services, 2022)

*Underfitting* dan *overfitting* merupakan hasil dari performa *Machine Learning* yang buruk dalam proses *fine-tuning*. *Underfitting* adalah suatu keadaan yang terjadi dimana model pelatihan data yang dibuat tidak mewakilkan keseluruhan data yang akan digunakan, sehingga menghasilkan performa yang buruk dalam pelatihan data dan tidak dapat memprediksi data dengan baik. Sedangkan *overfitting* adalah suatu keadaan yang terjadi dimana model pelatihan data yang dibuat terlalu fokus dan memiliki prediksi yang terlalu baik pada data *training* tertentu, sehingga tidak bisa melakukan prediksi dengan tepat jika diberikan dataset lain yang serupa dan prediksinya buruk pada data *testing*. *Underfitting* dan *overfitting* dapat diatasi dengan cara mengatur *hyperparameter* yang tepat saat proses *hyperparameter tuning* sehingga didapatkan model yang *balance*.

## Fase Evaluasi dan Pengujian Model

Fase evaluasi dan pengujian model merupakan tahap yang dilakukan untuk mengetahui apakah hasil yang didapat dari proses pemodelan data sudah dapat memprediksi dan mengklasifikasikan kata dengan benar atau belum. Hal yang dilakukan adalah dimulai dari melakukan evaluasi data dengan *confusion matrix*, menghitung nilai *performance metrics* dari data *testing*, menguji *trained* model yang sudah di-*fine-tuning* dengan memprediksi kata, dan menyimpan *trained* model untuk nantinya digunakan pada fase selanjutnya.



Gambar 3.6 *Flowchart* Tahapan Evaluasi dan Pengujian Model

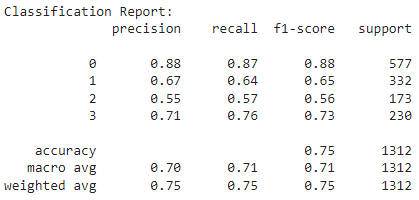
### Evaluasi Model

Setelah proses *training* model dilakukan, *trained* model yang dihasilkan perlu di-*test* atau dievaluasi untuk mengetahui apakah *trained* model tersebut dapat mengklasifikasikan kata dengan hasil yang akurat atau tidak. Tahap *testing* ini akan dilakukan pada dataset pengujian atau *testing* menggunakan dataset dengan rasio 10% dari keseluruhan jumlah data dengan jumlah 1312 data, dengan *performance metrics* yang dihitung yaitu *accuracy*, *precision*, *recall*, dan *Fl-score*. Langkah evaluasi model yang dilakukan adalah dengan menggunakan *confusion matrix* yang digunakan untuk menghitung *performance metrics*, *confusion matrix* juga dapat divisualisasikan dengan fungsi heatmap pada *library* seaborn. Sedangkan persamaan untuk menghitung keempat *performance metrics* yang dihitung tersebut dapat dilihat pada persamaan (2.1) sampai dengan (2.4).



Gambar 3.7 Contoh *Confusion Matrix*

Seluruh hasil *performance metrics* yang didapat dari proses perhitungan dengan *confusion matrix* dapat ditampilkan dengan fungsi *classification report* pada *library* sklearn. *Classification report* menampilkan hasil *performance metrics* dari seluruh kelas yang di-*train* beserta rata-rata dari masing-masing *performance metrics,* yaitu *accuracy* atau *micro average*, *macro average*, dan *weighted average*. Secara umum, jika data yang digunakan merupakan data *imbalance*, nilai *average* yang digunakan yaitu *macro average* karena *Machine Learning* akan memperlakukan semua kelas secara setara atau sama. Sedangkan jika data yang digunakan merupakan data *balance*, nilai *average* yang digunakan yaitu *micro average* atau *accuracy*.



Gambar 3.8 *Performance Metrics* dari Classification Report

Gambar 3.9 merupakan contoh hasil *performance metrics* yang didapat dari proses perhitungan dengan *confusion matrix* yang ditampilkan dengan fungsi *classification report*. Nilai *performance metrics*, *micro average*, *macro average*, dan *weighted average* berada di angka kisaran 0 sampai 1 untuk satuan desimal atau 0% sampai 100% untuk satuan persen, sedangkan nilai *support* merupakan jumlah data yang diuji. Jika *performance metrics* memiliki nilai yang lebih rendah dari 60%, diharuskan untuk melakukan *hyperparameter tuning* dan *fine-tuning* kembali karena model tersebut tidak akan menghasilkan prediksi yang tepat. Jika *performance metrics* memiliki nilai diantara 60% dan 70%, model yang telah di-*train* merupakan model yang tidak baik dan disarankan untuk melakukan *hyperparameter tuning* dan *fine-tuning* kembali agar *performance metrics* dapat mencapai nilai minimal 70%. Jika *performance metrics* memiliki nilai diantara 70% dan 80%, model yang telah di-*train* merupakan model yang baik dan apabila ingin mendapat model yang lebih baik lagi, disarankan untuk melakukan *hyperparameter tuning* dan *fine-tuning* kembali agar *performance metrics* dapat mencapai nilai minimal 80%. Jika *performance metrics* memiliki nilai diantara 80% dan 90%, model yang telah di-*train* merupakan model yang sangat baik yang berarti model dapat melakukan prediksi dengan sangat tepat. Jika *performance metrics* memiliki nilai diantara 90% dan 100%, kemungkinan hasil dapat mengakibatkan terjadinya *overfitting*.

Setelah hasil *performance metrics* didapatkan, selanjutnya adalah membandingkan *performance metrics* yang didapatkan pada percobaan pertama dengan percobaan-percobaan lainnya dimana percobaan yang lainnya dilakukan dengan cara melakukan *fine-tuning* ulang dan merubah *hyperparameter* yang sudah diatur pada fase sebelumnya sampai semua kombinasi *hyperparameter* yang ada di-*train* pada model untuk mendapatkan *trained* model yang paling optimal. Berdasarkan hasil evaluasi yang telah dibandingkan, maka *trained* model dengan hasil yang terbaik akan dipilih untuk disimpan dan digunakan pada fase implementasi sistem.

### Pengujian Model

Sebelum model disimpan, *trained* model yang sudah didapat sebelumnya akan diuji terlebih dahulu dengan cara memprediksi contoh teks yang sama seperti yang dilakukan pada fase sebelumnya dan hasil pengujian ini akan dibandingkan dengan hasil prediksi teks dengan *pretrained* model dari IndoBERT. Yang akan dibandingkan pada tahap pengujian ini adalah prediksi teks yang akan menampilkan hasil kelas atau label klasifikasi, dan nilai probabilitas dari hasil klasifikasi atau nilai *softmax*-nya.

Fungsi *softmax* akan menghitung nilai probabilitas dari setiap kelas label dan *softmax* juga akan membantu sistem untuk menentukan hasil dari kelas label ketika sistem ini dijalankan untuk memprediksi teks atau kalimat berbahasa Indonesia. *Softmax* memiliki rentang nilai dengan nilai 0 sampai 1. Jika fungsi *softmax* digunakan untuk mengklasifikasikan multi kelas, *softmax* akan mengembalikan nilai probabilitas dari masing-masing kelas. Hasil label yang akan ditampilkan sistem adalah nilai *softmax* dengan nilai probabilitas yang paling tinggi.

Setelah dilakukannya pengujian pada model, tahap terakhir pada fase ini adalah menyimpan model yang telah didapat dengan menggunakan fungsi save\_pretrained dari *library* Transformers. *Output* yang akan dihasilkan pada tahap ini adalah *trained* model, berserta *file-file* pendukung lainnya seperti BERT *Tokenizer*, konfigurasi model BERT, dan dataset yang sudah di tokenisasi.

## Fase Implementasi Sistem

Fase implementasi d.

# BAB IV

**HASIL DAN PEMBAHASAN**



## Hasil Pengumpulan dan Persiapan Data

Data pada penelitian ini didapatkan dari dataset yang sudah dibangun dari penelitian (Alfina, Mulia, Fanany, & Ekanata, 2017), (Putri, 2018), dan (Ibrohim & Budi, 2019). Dataset didapatkan dengan cara melakukan *crawling* terhadap *tweet* menggunakan API Twitter yang diimplementasikan menggunakan *library* Tweepy. Cuplikan isi data pada dataset dapat dilihat pada tabel 4.1.

Tabel 4.1 Cuplikan Dataset *Hate Speech* Multi Label

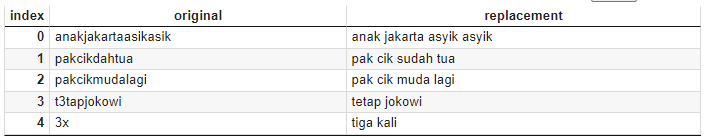
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tweet** | **HS** | **HS\_Weak** | **HS\_Moderate** | **HS\_Strong** |
| - disaat semua cowok berusaha melacak perhatian gue. loe lantas ... | 1 | 1 | 0 | 0 |
| RT USER: USER siapa yang telat ngasih tau elu?edan sarap gue ... | 1 | 0 | 0 | 1 |
| 41. Kadang aku berfikir, kenapa aku tetap percaya pada Tuhan padahal ... | 0 | 0 | 0 | 0 |
| USER USER AKU ITU AKU\n\nKU TAU MATAMU ... | 0 | 0 | 0 | 0 |
| USER USER Kaum cebong kapir udah keliatan dongoknya ... | 1 | 0 | 1 | 0 |

Total data yang didapatkan sebanyak 13169 data yang terdiri dari 2 label, yaitu *Hate Speech* sebanyak 7309 data dan *Non-Hate Speech* sebanyak 5860 data. Pada data *Hate Speech* dibagi menjadi 3 kelas, yaitu *Weak* *Hate Speech* sebanyak 3383 data, *Moderate* *Hate Speech* sebanyak 1705 data, dan *Strong* *Hate Speech* sebanyak 2221 data. Perbandingan jumlah data pada setiap label dapat dilihat pada gambar 4.1.

Gambar 4.1 Histogram Perbandingan Jumlah Data Pada Setiap Label

Cuplikan kamus alay yang nantinya akan digunakan pada proses *normalization* dapat dilihat pada tabel 4.2.

Tabel 4.2 Kamus Alay (Ibrohim & Budi, 2019)



## Hasil Pengolahan Data

Pada fase pengolahan data dilakukanlah proses *preprocessing* data yang terdiri dari beberapa proses yaitu *data cleaning*, *case folding*, *normalization*, *filtering*, dan *stemming* dengan hasil dari langkah-langkah yang dilakukan dijelaskan sebagai berikut.

1. *Data Cleaning*

*Data Cleaning* dilakukan dengan menghapus beberapa karakter yang tidak dibutuhkan untuk proses klasifikasi seperti URL atau *link*, *hashtag* (#), nama pengguna, *mention* (@), *retweet* (RT), tanda baca, baris baru, nomor, tanda baca, spasi kosong, emoji, dan sintaks HTML menggunakan *regular expression.* Penerapan *data cleaning* pada dataset dapat dilihat pada tabel 4.3.

Tabel 4.3 Penerapan *Data Cleaning* pada Dataset

Text, table

Description automatically generated

Berikut adalah *script* atau kode program *data cleaning*:

|  |
| --- |
| nltk.download("stopwords")  nltk.download("punkt")  stop\_words = nltk.corpus.stopwords.words("indonesian")  def data\_cleaning(text):  emoticon\_byte\_regex = r"\s\*(?:\\x[A-Fa-f0-9]{2})+"  url\_regex = "((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+)||(http\S+))"  text = re.sub(emoticon\_byte\_regex, "", text) # remove emoticon bytes  text = re.sub(url\_regex, "", text) # remove every url  text = re.sub(r"<[^>]\*>", "", text) # remove html tags  text = re.sub(r"@[A-Za-z0-9]+", "", text) # remove twitter usernames  text = re.sub(r"\\n", " ", text) # remove every new line '\n'  text = re.sub("@[\w\-]+", "", text) # remove mentions  text = re.sub("RT", "", text) # remove every retweet symbol  text = re.sub("USER", "", text) # remove every user  text = re.sub(" URL", " ", text) # remove word URL  text = re.sub(" url", " ", text) # remove word url  text = re.sub("\\+", " ", text) # remove backslash  text = re.sub("\s+", " ", text) # remove special regular expression character  text = re.sub("[^0-9a-zA-Z]", " ", text) # remove punctuation  text = re.sub("[^a-zA-Z]", " ", text) # remove numbers  text = re.sub(" +", " ", text) # remove extra spaces  word\_tokens = word\_tokenize(text)  filtered\_sentence = []  for word\_token in word\_tokens:  if word\_token not in stop\_words:  filtered\_sentence.append(word\_token)  # join words  text = " ".join(filtered\_sentence)  return text  df["text\_cleaned"] = df["Tweet"].apply(data\_cleaning)  df.text\_cleaned = df.text\_cleaned.str.strip()  df[["Tweet", "text\_cleaned"]].head() |

1. *Case Folding*

*Case Folding* dilakukan dengan mengubah semua huruf menjadi huruf kecil. Penerapan *case folding* pada dataset dapat dilihat pada tabel 4.4.

Tabel 4.4 Penerapan *Case Folding* pada Dataset

Table

Description automatically generated

Berikut adalah *script* atau kode program *case folding*:

|  |
| --- |
| # lowercase letters  def case\_folding(text):  text = text.lower()  return text  df["text\_folded"] = df["text\_cleaned"].apply(case\_folding)  df[["text\_cleaned", "text\_folded"]].head() |

1. *Normalization*

*Normalization* dilakukan dengan merubah kata dari yang awalnya tidak baku menjadi baku menggunakan kamus kosakata baku yang diimpor dari (Ibrohim & Budi, 2019). Penerapan *normalization* pada dataset dapat dilihat pada tabel 4.5.

Tabel 4.5 Penerapan *Normalization* pada Dataset

Table

Description automatically generated

Berikut adalah *script* atau kode program *normalization*:

|  |
| --- |
| alay\_dict\_map = dict(zip(alay\_dict["original"], alay\_dict["replacement"]))  def normalize\_alay\_dict(text):  return " ".join(  [  alay\_dict\_map[word] if word in alay\_dict\_map else word  for word in text.split(" ")  ]  )  df["text\_normalized"] = df["text\_folded"].apply(normalize\_alay\_dict)  df[["text\_folded", "text\_normalized"]].head() |

1. *Filtering (Stopword Removal)*

*Filtering* dilakukan dengan menghilangkan *stopword* yang bertujuan untuk menghapus kata-kata yang tidak memiliki makna dengan s*topword* yang digunakan didapat dari (Tala, 2003). Penerapan *filtering* pada dataset dapat dilihat pada tabel 4.6.

Tabel 4.6 Penerapan *Filtering* pada Dataset

Table

Description automatically generated

Berikut adalah *script* atau kode program *filtering*:

|  |
| --- |
| # stopword from Tala, F. Z. (2003)  id\_stopword\_dict = pd.read\_csv(  "https://raw.githubusercontent.com/stopwords-iso/stopwords-id/master/stopwords-id.txt",  header=None,  )  id\_stopword\_dict = id\_stopword\_dict.rename(columns={0: "stopword"})  def remove\_stopword(text):  text = " ".join(  [  "" if word in id\_stopword\_dict.stopword.values else word  for word in text.split(" ")  ]  )  text = re.sub(" +", " ", text) # remove extra spaces  text = text.strip()  return text  df["text\_filtered"] = df["text\_normalized"].apply(remove\_stopword)  df[["text\_normalized", "text\_filtered"]].head() |

1. *Stemming*

*Stemming* dilakukan dengan menghilangkan kata berimbuhan ke bentuk dasarnya. Penerapan *stemming* pada dataset dapat dilihat pada tabel 4.7.

Tabel 4.7 Penerapan *Stemming* pada Dataset

Graphical user interface, table

Description automatically generated

Berikut adalah *script* atau kode program *stemming*:

|  |
| --- |
| factory = StemmerFactory()  stemmer = factory.create\_stemmer()  def stemming(text):  return stemmer.stem(text)  df["text"] = df["text\_filtered"].apply(stemming)  df[["text\_filtered", "text"]].head() |

Selanjutnya setelah dilakukannya *preprocessing* data yaitu pemberian indeks, dengan kelas *Non-Hate Speech* diberi indeks “0”, kelas *Weak Hate Speech* diberi indeks “1”, kelas *Moderate Hate Speech* diberi indeks “2”, dan kelas *Strong Hate Speech* diberi indeks “3”. Dataset yang telah diberi indeks dapat dilihat pada tabel 4.8.

Tabel 4.8 Dataset *Preprocessed* yang Telah Diberi Indeks

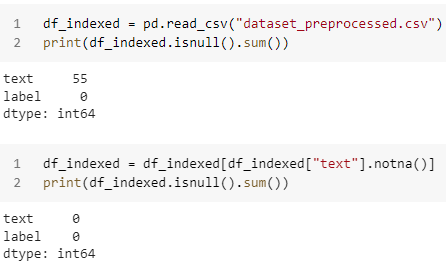
Graphical user interface, text, application, email

Description automatically generated

Berikut adalah *script* atau kode program *indexing*:

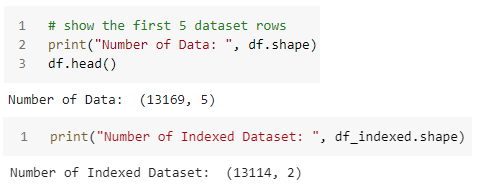
|  |
| --- |
| def label\_classification\_weak(hs):  label = ""  if int(hs) == 1:  label = 1  else:  label = 0  return label  def label\_classification\_moderate(hs):  label = ""  if int(hs) == 1:  label = 2  else:  label = 0  return label  def label\_classification\_strong(hs):  label = ""  if int(hs) == 1:  label = 3  else:  label = 0  return label  df["label\_weak"] = df["HS\_Weak"].apply(label\_classification\_weak)  df["label\_moderate"] = df["HS\_Moderate"].apply(label\_classification\_moderate)  df["label\_strong"] = df["HS\_Strong"].apply(label\_classification\_strong)  df["label"] = df[["label\_weak", "label\_moderate", "label\_strong"]].max(1)  df = df[["text", "label"]]  df.to\_csv("dataset\_preprocessed.csv", index=False)  df.head() |

Langkah terakhir adalah menghilangkan *missing value* pada dataset yang ditandai dengan adanya data dengan *value* *NaN* atau *Null*. Jumlah *missing value* yang ada pada dataset beserta *script* atau kode programnya dapat dilihat pada gambar 4.2.



Gambar 4.2 Perbandingan Jumlah *Missing Value* Sebelum dan Setelah Diproses

Jumlah data sebelum dan setelah dihilangkannya *missing value* beserta *script* atau kode programnya dapat dilihat pada gambar 4.3.



Gambar 4.3 Perbandingan Jumlah Data Sebelum dan Setelah Menghilangkan *Missing Value*

## Hasil Pembagian Data

Selanjutnya adalah proses pembagian data atau *split-up dataset* menggunakan metode *Hold-out Cross Validation* dengan data keseluruhan yang digunakan berjumlah 13114 data dengan komposisi pembagian data menjadi 80% untuk *training*, 10% untuk *validation*, dan 10% untuk *testing*.

Berikut adalah *script* atau kode program *Hold-out Cross Validation*:

|  |
| --- |
| # define dataset  X = df["text"]  y = df["label"]  # split into 80:10:10 ration  X\_train, X\_rem, y\_train, y\_rem = train\_test\_split(X, y, train\_size=0.8, random\_state=random\_state)  X\_valid, X\_test, y\_valid, y\_test = train\_test\_split(  X\_rem, y\_rem, test\_size=0.5, random\_state=random\_state  )  # describe info about train, valid, and test set  print("Number of Train Dataset: ", y\_train.shape)  print(y\_train.value\_counts())  print("\nNumber of Valid Dataset: ", y\_valid.shape)  print(y\_valid.value\_counts())  print("\nNumber of Test Dataset: ", y\_test.shape)  print(y\_test.value\_counts())  df\_train = pd.concat([X\_train, y\_train], axis=1)  df\_valid = pd.concat([X\_valid, y\_valid], axis=1)  df\_test = pd.concat([X\_test, y\_test], axis=1)  df\_train.to\_csv("dataset\_training.csv", index=False)  df\_valid.to\_csv("dataset\_validation.csv", index=False)  df\_test.to\_csv("dataset\_testing.csv", index=False) |

Pada tahap pembagian data ini penulis melakukan 2 tahapan pengujian, yaitu bekerja dengan data yang tidak seimbang (*imbalance*) dan data yang seimbang (*balance*). Hal ini dilakukan untuk mengetahui hasil mana yang lebih baik diantara bekerja dengan data *imbalance* atau *balance*. Tahap *data balancing* yang digunakan yaitu menggunakan metode SMOTE. Perbandingan jumlah data *imbalance* dengan data *balance* dapat dilihat pada tabel 4.9.

Tabel 4. Perbandingan Jumlah Data *Imbalance* dan Data *Balance*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Data | Jenis Data | Label | | | | |
| **0** | **1** | **2** | **3** | **Total** |
| Training | Imbalance Data | 4667 | 2380 | 1384 | 1765 | 10496 |
| Balance Data | 4667 | 4667 | 4667 | 4667 | 18668 |
| Validation | Imbalance Data | 593 | 347 | 146 | 226 | 1312 |
| Balance Data | 593 | 593 | 593 | 593 | 2372 |
| Testing | Imbalance Data | 562 | 348 | 174 | 229 | 1313 |
| Balance Data | 562 | 562 | 562 | 562 | 2248 |

Berikut adalah *script* atau kode program metode SMOTE:

|  |
| --- |
| # convert strings into numericals using TfidfVectorizer  vec\_train = TfidfVectorizer()  vec\_test = TfidfVectorizer()  vec\_valid = TfidfVectorizer()  df\_train = pd.read\_csv('dataset\_training.csv')  X\_train = vec\_train.fit\_transform(df\_train['text'])  df\_test = pd.read\_csv('dataset\_testing.csv')  X\_test = vec\_test.fit\_transform(df\_test['text'])  df\_valid = pd.read\_csv('dataset\_validation.csv')  X\_valid = vec\_valid.fit\_transform(df\_valid['text'])  sm = SMOTE(random\_state = 42)  X\_train\_res, y\_train\_res = sm.fit\_resample(X\_train, y\_train.ravel())  X\_test\_res, y\_test\_res = sm.fit\_resample(X\_test, y\_test.ravel())  X\_valid\_res, y\_valid\_res = sm.fit\_resample(X\_valid, y\_valid.ravel())  # convert back into strings using inverse\_transform  X\_train\_res = vec\_train.inverse\_transform(X\_train\_res)  X\_test\_res = vec\_test.inverse\_transform(X\_test\_res)  X\_valid\_res = vec\_valid.inverse\_transform(X\_valid\_res)  pd.DataFrame({'text': X\_train\_res}).to\_csv("X\_train.csv", index=False)  pd.DataFrame({'label': y\_train\_res}).to\_csv("y\_train.csv", index=False)  pd.DataFrame({'text': X\_test\_res}).to\_csv("X\_test.csv", index=False)  pd.DataFrame({'label': y\_test\_res}).to\_csv("y\_test.csv", index=False)  pd.DataFrame({'text': X\_valid\_res}).to\_csv("X\_valid.csv", index=False)  pd.DataFrame({'label': y\_valid\_res}).to\_csv("y\_valid.csv", index=False)  print("Before OverSampling, counts of label '0': {}".format(sum(y\_train == 0)))  print("Before OverSampling, counts of label '1': {}".format(sum(y\_train == 1)))  print("Before OverSampling, counts of label '2': {}".format(sum(y\_train == 2)))  print("Before OverSampling, counts of label '3': {} \n".format(sum(y\_train == 3)))  print('After OverSampling, the shape of train\_y: {} \n'.format(y\_train\_res.shape))  print("After OverSampling, counts of label '0': {}".format(sum(y\_train\_res == 0)))  print("After OverSampling, counts of label '1': {}".format(sum(y\_train\_res == 1)))  print("After OverSampling, counts of label '2': {}".format(sum(y\_train\_res == 2)))  print("After OverSampling, counts of label '3': {}".format(sum(y\_train\_res == 3)))  print("Before OverSampling, counts of label '0': {}".format(sum(y\_valid == 0)))  print("Before OverSampling, counts of label '1': {}".format(sum(y\_valid == 1)))  print("Before OverSampling, counts of label '2': {}".format(sum(y\_valid == 2)))  print("Before OverSampling, counts of label '3': {} \n".format(sum(y\_valid == 3)))  print('After OverSampling, the shape of valid\_y: {} \n'.format(y\_valid\_res.shape))  print("After OverSampling, counts of label '0': {}".format(sum(y\_valid\_res == 0)))  print("After OverSampling, counts of label '1': {}".format(sum(y\_valid\_res == 1)))  print("After OverSampling, counts of label '2': {}".format(sum(y\_valid\_res == 2)))  print("After OverSampling, counts of label '3': {}".format(sum(y\_valid\_res == 3)))  print("Before OverSampling, counts of label '0': {}".format(sum(y\_test == 0)))  print("Before OverSampling, counts of label '1': {}".format(sum(y\_test == 1)))  print("Before OverSampling, counts of label '2': {}".format(sum(y\_test == 2)))  print("Before OverSampling, counts of label '3': {} \n".format(sum(y\_test == 3)))  print('After OverSampling, the shape of test\_y: {} \n'.format(y\_test\_res.shape))  print("After OverSampling, counts of label '0': {}".format(sum(y\_test\_res == 0)))  print("After OverSampling, counts of label '1': {}".format(sum(y\_test\_res == 1)))  print("After OverSampling, counts of label '2': {}".format(sum(y\_test\_res == 2)))  print("After OverSampling, counts of label '3': {}".format(sum(y\_test\_res == 3)))  def data\_cleaning(text):  text = re.sub("['']", "", text)  text = text.replace('[', '').replace(']', '')  return text  df\_train = pd.read\_csv('X\_train.csv')  df\_test = pd.read\_csv('X\_test.csv')  df\_valid = pd.read\_csv('X\_valid.csv')  df\_train["text"] = df\_train["text"].apply(data\_cleaning)  df\_train.text = df\_train.text.str.strip()  df\_test["text"] = df\_test["text"].apply(data\_cleaning)  df\_test.text = df\_test.text.str.strip()  df\_valid["text"] = df\_valid["text"].apply(data\_cleaning)  df\_valid.text = df\_valid.text.str.strip()  df\_train.to\_csv("X\_train.csv", index=False)  df\_test.to\_csv("X\_test.csv", index=False)  df\_valid.to\_csv("X\_valid.csv", index=False)  # save data training  df\_X\_train = pd.read\_csv('X\_train.csv')  df\_y\_train = pd.read\_csv('y\_train.csv')  df\_train = pd.concat([df\_X\_train, df\_y\_train], axis=1)  # save data validation  df\_X\_valid = pd.read\_csv('X\_valid.csv')  df\_y\_valid = pd.read\_csv('y\_valid.csv')  df\_valid = pd.concat([df\_X\_valid, df\_y\_valid], axis=1)  # save data testing  df\_X\_test = pd.read\_csv('X\_test.csv')  df\_y\_test = pd.read\_csv('y\_test.csv')  df\_test = pd.concat([df\_X\_test, df\_y\_test], axis=1)  df\_train.to\_csv("dataset\_training.csv", index=False)  df\_valid.to\_csv("dataset\_validation.csv", index=False)  df\_test.to\_csv("dataset\_testing.csv", index=False) |

## Hasil Evaluasi dan Pengujian Model

Seperti yang sebelumnya sudah dijelaskan bahwa pada penelitian ini dataset yang ada bersifat *imbalance*, maka dari itu pada tahap evaluasi dan pengujian model ini penulis melakukan tahapan pengujian dengan melakukan *fine-tuning* pada 2 dataset yang berbeda, yaitu dengan dataset yang *imbalance* dan *balance*. Setelah itu penulis membandingkan hasil *confusion matrix* dan *performance metrics* dari kedua dataset tersebut untuk mengetahui hasil mana yang lebih baik dalam mengklasifikasikan teks berbahasa Indonesia.

Selanjutnya adalah tahap evaluasi dan pengujian model dengan melakukan *fine-tuning*, terlebih dahulu penulis melakukan tahap *hyperparameter tuning* dengan beberapa percobaan *hyperparameter tuning* yang akan diuji sebagai berikut:

1. *Batch Size*: 16 dan 32
2. *Learning Rate*: 5e-5, 3e-5, dan 2e-5
3. *Epoch*: 2, 3, dan 4

Berdasarkan *hyperparameter* yang akan diuji, pengujian akan dilakukan sebanyak 18 kali untuk satu dataset. Karena terdapat 2 dataset yang akan diuji, maka total pengujian akan dilakukan sebanyak 36 kali.

Berikut adalah *script* atau kode program *fine-tuning*:

|  |
| --- |
| batch\_size=16 # batch size  lr=2e-5 # learning rate  epochs=2 # epochs  attention\_probs\_dropout\_prob=0.1 # dropout  hidden\_dropout\_prob=0.1 # dropout  max\_seq\_len=512 # maximum length  num\_labels=4 # number of labels  num\_workers=2 # number of workers  eps=1e-8 # epsilon  weight\_decay=1e-2 #weight decay  device='cuda' # set device to cuda GPU  # load tokenizer  print('Loading BERT tokenizer...')  tokenizer = BertTokenizer.from\_pretrained('indobenchmark/indobert-base-p1')  # load config  print('Loading BERT config...')  config = BertConfig.from\_pretrained('indobenchmark/indobert-base-p1', attention\_probs\_dropout\_prob=attention\_probs\_dropout\_prob, hidden\_dropout\_prob=hidden\_dropout\_prob, num\_labels=num\_labels)  config.num\_labels = MultiLabelHateSpeechClassificationDataset.NUM\_LABELS  # instantiate model  print('Instantiate BERT model...')  model = BertForSequenceClassification.from\_pretrained('indobenchmark/indobert-base-p1', config=config)  training\_dataset\_path = 'dataset\_training.csv'  testing\_dataset\_path = 'dataset\_testing.csv'  validation\_dataset\_path = 'dataset\_validation.csv'  training\_dataset = MultiLabelHateSpeechClassificationDataset(training\_dataset\_path, tokenizer, lowercase=True)  testing\_dataset = MultiLabelHateSpeechClassificationDataset(testing\_dataset\_path, tokenizer, lowercase=True)  validation\_dataset = MultiLabelHateSpeechClassificationDataset(validation\_dataset\_path, tokenizer, lowercase=True)  training\_loader = HateSpeechClassificationDataLoader(dataset=training\_dataset, max\_seq\_len=max\_seq\_len, batch\_size=batch\_size, num\_workers=num\_workers, shuffle=True)  testing\_loader = HateSpeechClassificationDataLoader(dataset=testing\_dataset, max\_seq\_len=max\_seq\_len, batch\_size=batch\_size, num\_workers=num\_workers, shuffle=False)  validation\_loader = HateSpeechClassificationDataLoader(dataset=validation\_dataset, max\_seq\_len=max\_seq\_len, batch\_size=batch\_size, num\_workers=num\_workers, shuffle=False)  # labeling index  w2i, i2w = MultiLabelHateSpeechClassificationDataset.LABEL2INDEX, MultiLabelHateSpeechClassificationDataset.INDEX2LABEL  print(w2i)  print(i2w)  optimizer = AdamW(model.parameters(), lr=lr, eps=eps, weight\_decay=weight\_decay)  # tell pytorch to run this model on the GPU  model = model.cuda()  train\_acc\_lists = []  train\_pre\_lists = []  train\_rec\_lists = []  train\_f1\_lists = []  eval\_acc\_lists = []  eval\_pre\_lists = []  eval\_rec\_lists = []  eval\_f1\_lists = []  train\_loss\_lists = []  eval\_loss\_lists = []    # training  n\_epochs = epochs  for epoch in range(n\_epochs):  model.train()  torch.set\_grad\_enabled(True)    total\_train\_loss = 0  list\_hyp, list\_label = [], []  train\_pbar = tqdm(training\_loader, leave=True, total=len(training\_loader))  for i, batch\_data in enumerate(train\_pbar):  # forward model  loss, batch\_hyp, batch\_label = forward\_sequence\_classification(model, batch\_data[:-1], i2w=i2w, device=device)  # update model  optimizer.zero\_grad()  loss.backward()  optimizer.step()  tr\_loss = loss.item()  total\_train\_loss = total\_train\_loss + tr\_loss  # calculate metrics  list\_hyp += batch\_hyp  list\_label += batch\_label  train\_pbar.set\_description("(Epoch {}) TRAIN LOSS:{:.4f} LR:{:.8f}".format((epoch+1),  total\_train\_loss/(i+1), get\_lr(optimizer)))  # calculate train metric  metrics = multi\_label\_hate\_speech\_classification\_metrics\_fn(list\_hyp, list\_label)  print("(Epoch {}) TRAIN LOSS:{:.4f} {} LR:{:.8f}".format((epoch+1),  total\_train\_loss/(i+1), metrics\_to\_string(metrics), get\_lr(optimizer)))    train\_acc\_lists.append(metrics['ACC'])  train\_pre\_lists.append(metrics['PRE'])  train\_rec\_lists.append(metrics['REC'])  train\_f1\_lists.append(metrics['F1'])  current\_train\_loss = round(total\_train\_loss/(i+1), 4)  train\_loss\_lists.append(current\_train\_loss)  # evaluate on validation  model.eval()  torch.set\_grad\_enabled(False)    total\_loss, total\_correct, total\_labels = 0, 0, 0  list\_hyp, list\_label = [], []  pbar = tqdm(validation\_loader, leave=True, total=len(validation\_loader))  for i, batch\_data in enumerate(pbar):  batch\_seq = batch\_data[-1]  loss, batch\_hyp, batch\_label = forward\_sequence\_classification(model, batch\_data[:-1], i2w=i2w, device=device)    # calculate total loss  valid\_loss = loss.item()  total\_loss = total\_loss + valid\_loss  # calculate evaluation metrics  list\_hyp += batch\_hyp  list\_label += batch\_label  metrics = multi\_label\_hate\_speech\_classification\_metrics\_fn(list\_hyp, list\_label)  pbar.set\_description("VALID LOSS:{:.4f} {}".format(total\_loss/(i+1), metrics\_to\_string(metrics)))    metrics = multi\_label\_hate\_speech\_classification\_metrics\_fn(list\_hyp, list\_label)  print("(Epoch {}) VALID LOSS:{:.4f} {}".format((epoch+1),  total\_loss/(i+1), metrics\_to\_string(metrics)))    eval\_acc\_lists.append(metrics['ACC'])  eval\_pre\_lists.append(metrics['PRE'])  eval\_rec\_lists.append(metrics['REC'])  eval\_f1\_lists.append(metrics['F1'])  current\_eval\_loss = round(total\_loss/(i+1), 4)  eval\_loss\_lists.append(current\_eval\_loss) |

Setelah *fine-tuning* dilakukan, selanjutnya adalah mengevaluasi *trained* model yang didapat dari tahap sebelumnya dengan menggunakan *confusion matrix* dan menghitung nilai *performance metrics* yaitu *accuracy*, *precision*, *recall*, dan *F1-score*.

Berikut adalah *script* atau kode program *classification report* dan *confusion matrix*:

|  |
| --- |
| # evaluate on test  model.eval()  torch.set\_grad\_enabled(False)  total\_loss, total\_correct, total\_labels = 0, 0, 0  list\_hyp, list\_label = [], []  print('Loading Testing DataLoader...')  pbar = tqdm(testing\_loader, leave=True, total=len(testing\_loader))  for i, batch\_data in enumerate(pbar):  \_, batch\_hyp, \_ = forward\_sequence\_classification(model, batch\_data[:-1], i2w=i2w, device=device)  list\_hyp += batch\_hyp  df = pd.DataFrame({'label':list\_hyp}).reset\_index()  df.to\_csv('prediction.txt', index=False)  print(df)  prediction = pd.read\_csv('prediction.txt')  prediction\_list = []  for i in prediction['label'] :  if i == 'HS\_Weak':  prediction\_list.append(1)  elif i == 'HS\_Moderate':  prediction\_list.append(2)  elif i == 'HS\_Strong':  prediction\_list.append(3)  else :  prediction\_list.append(0)  data\_test = pd.read\_csv('dataset\_testing.csv')  def label\_classification(hs):  label = ''    if str(hs) == 'HS\_Weak':  label = 1  elif str(hs) == 'HS\_Moderate':  label = 2  elif str(hs) == 'HS\_Strong':  label = 3  else:  label = 0  return label  data\_test['label'] = data\_test['label'].apply(label\_classification)  data\_test = data\_test[['text', 'label']]  data\_test.head()  print(f"Accuracy: {round(accuracy\_score(data\_test['label'], prediction\_list), 2)}")  print(f"Precision: {round(precision\_score(data\_test['label'], prediction\_list, average='macro'), 2)}")  print(f"Recall: {round(recall\_score(data\_test['label'], prediction\_list, average='macro'), 2)}")  # micro avg of f1-score  f1\_score\_micro = f1\_score(data\_test['label'], prediction\_list, average='micro') \* 100  print("F1-score (Micro Avg) is {}%." .format(round(f1\_score\_micro),1))  # macro avg of f1-score  f1\_score\_macro = f1\_score(data\_test['label'], prediction\_list, average='macro') \* 100  print("F1-score (Macro Avg) is {}%." .format(round(f1\_score\_macro),1))  # weighted avg of f1-score  f1\_score\_weighted = f1\_score(data\_test['label'], prediction\_list, average='weighted') \* 100  print("F1-score (Weighted Avg) is {}%." .format(round(f1\_score\_weighted),1))  print("Classification Report:")  print(classification\_report(data\_test['label'], prediction\_list))  print("\nConfusion Matrix:")  f, ax = plt.subplots(figsize=(10,5))  sns.heatmap(confusion\_matrix(data\_test['label'], prediction\_list), annot=True, fmt=".0f", ax=ax)  plt.xlabel("True Label")  plt.ylabel("Predicted Label")  plt.show() |

Kemudian *trained* model dengan hasil yang terbaik akan dipilih untuk disimpan dan nantinya akan dimuat pada tahap implementasi sistem. Output yang dihasilkan pada tahap ini dapat dilihat pada gambar berikut.

Gambar 4. *Output File* pada Tahap *Save Model*

Berikut adalah *script* atau kode program *save model*:

|  |
| --- |
| output\_dir = '/content/drive/MyDrive/Skripsi/Code'  # output\_dir = '/content/drive/MyDrive/Skripsi/Code/multi-label-hate-speech-detection/models'  # create output directory  if not os.path.exists(output\_dir):  os.makedirs(output\_dir)  print("Saving model to %s" % output\_dir)  # save a trained model, configuration and tokenizer using 'save\_pretrained()'.  model\_to\_save = model.module if hasattr(model, 'module') else model  model\_to\_save.save\_pretrained(output\_dir)  tokenizer.save\_pretrained(output\_dir)  # save  # with open('/content/drive/MyDrive/Skripsi/Code/multi-label-hate-speech-detection/models/tokenizer.pkl', 'wb') as handle:  with open('/content/drive/MyDrive/Skripsi/Code/tokenizer.pkl', 'wb') as handle:  pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST\_PROTOCOL)  !ls -l --block-size=K /content/drive/MyDrive/Skripsi/Code  # !ls -l --block-size=K /content/drive/MyDrive/Skripsi/Code/multi-label-hate-speech-detection/models |

fefef

## Hasil Implementasi Sistem

Agar tujuan pada penelitian ini tercapai, maka dibuatlah sistem berbasis website untuk menganalisis hasil dari model yang sudah dibuat sebelumnya. Skenario pengujiannya dibagi menjadi dua, yang pertama yaitu pengguna nantinya bisa memasukkan teks atau kalimat yang ingin dicek, lalu output yang dihasilkan adalah klasifikasi label apakah teks atau kalimat yang dimasukkan termasuk hate speech atau non-hate speech.

Selanjutnya yang kedua yaitu klasifikasi hate speech dengan memasukkan kata kunci yang ingin dicari, lalu output yang dihasilkan yaitu hasil klasifikasi yang didapatkan dengan melakukan crawling pada media sosial Twitter dan output-nya dapat dilihat dalam format CSV.

### a

# BAB V

**KESIMPULAN DAN SARAN**



## Kesimpulan

Berdasarkan berbagai hal yang telah disampaikan pada bab sebelumnya serta perancangan dan implementasi yang sudah dilakukan penulis, maka dapat diambil simpulan sebagai berikut:

Agar pengguna

## Saran

Berdasarkan penelitian yang dilakukan, penulis memberikan saran sebagai berikut:

Aplikasi ini dapat diterapkan.

# DAFTAR PUSTAKA

Alfina, I., Mulia, R., Fanany, M. I., & Ekanata, Y. (2017). Hate Speech Detection in the Indonesian Language: A Dataset and Preliminary Study. *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 233-237. doi:10.1109/icacsis.2017.8355039

Amazon Web Services. (2022). *Model Fit: Underfitting vs. Overfitting*. Dipetik April 22, 2022, dari AWS: https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

Anzanello, M. J., & Fogliatto, F. S. (2011, September). Learning curve models and applications: literature review and research directions. *International Journal of Industrial Ergonomics, 41*(5), 573-583. doi:10.1016/j.ergon.2011.05.001

Chollet, F. (2018). *Deep Learning with Python.* Shelter Island: Manning Publications Co.

Dalal, M. K., & Zaveri, M. (2011, August). Automatic Text Classification: A Technical Review. *International Journal of Computer Applications, 28*(2). doi:10.5120/3358-4633

Deng, L., & Yu, D. (2014, June). Deep Learning: Methods and Applications. *Foundations and Trends in Signal Processing, 7*, 197-387. doi:https://doi.org/10.1561/2000000039

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* *I*, hal. 4171-4186. Minneapolis, Minnesota: Association for Computational Linguistics. doi:10.48550/arXiv.1810.04805

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data.* Cambridge Univ. Press.

Fimoza, D. (2021). Analisis Sentimen Terhadap Film Indonesia Dengan Pendekatan Bert .

Gestama, P. (2020, February 1). *PENGERTIAN SDLC adalah: Fungsi, Metode dan Tahapan SDLC*. Dipetik February 21, 2022, dari Salamadian: https://salamadian.com/sdlc-system-development-life-cycle/

Goldberg, Y. (2017, April). Neural Network Methods for Natural Language Processing. (G. Hirst, Penyunt.) *Synthesis Lectures on Human Language Technologies, 10*(1), 1-309. doi:https://doi.org/10.2200/S00762ED1V01Y201703HLT037

Goodfellow, I., Bengio, Y., & Courvil, A. (2016). *Deep Learning.* Massachusetts: MIT Press.

Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques* (3rd ed.). Morgan Kaufman.

Hertzmann, A., & Fleet, D. (2010). *Machine Learning and Data Mining Lecture Notes.* Toronto: Computer Science Department, University of Toronto.

Hidayat, A., Sujaini, H., & Dwinyoto, R. (2015). Aplikasi Penerjemah Dua Arah Bahasa Indonesia – Bahasa Melayu Sambas Berbasis Web dengan Menggunakan Decoder Moses. *Jurnal Sistem dan Teknologi Informasi (JustIN), 3*(3).

Hossin, M., & Sulaiman, M. N. (2015, March). A Review on Evaluation Metrics for Data Classification Evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2). doi:10.5121/ijdkp.2015.5201

Ibrohim, M. O., & Budi, I. (2019, August). Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter. *Proceedings of the Third Workshop on Abusive Language Online*, 46-57. doi:10.18653/v1/W19-3506

Jurafsky, D., & Martin, J. H. (2008). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd ed.). New Jersey: Prentice Hall.

Koto, F., Rahim, A., Lau, J. H., & Baldwin, T. (2020, November 2). IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP. *In Proceedings of the 28th COLING*. Diambil kembali dari https://arxiv.org/pdf/2011.00677.pdf

Mowafy, M., Rezk, A., & El-Bakry, H. M. (2018, January). Building Unstructured Crime Data Prediction Model (Practical Approach). *INTERNATIONAL JOURNAL OF COMPUTER APPLICATION*. doi:10.26808/rs.ca.i8v4.01

Mustaqhfiri, M., Abidin, Z., & Kusumawati, R. (2011). Peringkasan Teks Otomatis Berita Berbahasa Indonesia Menggunakan Metode Maximum Marginal Relevance. *Jumal Ilmu Komputer dan Teknologi Informasi, 4*(4). doi:10.18860/mat.v0i0.1578

Prawira, Y. (2016). Upaya Kepolisian Dalam Menanggulangi Kejahatan Ujaran Kebencian (Hate Speech) Berdasarkan Surat Edaran Kapolri NO.SE/06/X/2015. *Skripsi*.

Putra, H. K., Bijaksana, M. A., & Romadhony, A. (2021, April). Deteksi Penggunaan Kalimat Abusive Pada Teks Bahasa Indonesia Menggunakan Metode IndoBERT. *e-Proceeding of Engineering, 8*(2), 3028-3038.

Putri, T. T. (2018). Analisis dan deteksi hate speech pada sosial twitter berbahasa indonesia. *Master's thesis*, Master's thesis, Faculty of Computer Science, Universitas Indonesia.

Pypi.org. (2018). *Pandas.* Diambil kembali dari Python Package Index: https://pypi.org/project/pandas/

Python Software Foundation. (2001). *What is Python? Executive Summary.* Retrieved March 14, 2022, from Python: https://www.python.org/doc/essays/blurb/

Sebastiani, F. (2001). Machine Learning in Automated Text Categorization. *ACM Computing Surveys, 34*, 1-47. doi:10.1145/505282.505283

Sivakumar, A., & Gunasundari, R. (2017). A Survey on Data Preprocessing Techniques for Bioinformatics and Web Usage Mining. *International Journal of Pure and Applied Mathematics, 117*(20), 785-794.

Sommerville, I. (2011). *Software Engineering (9th Edition).* New York: Addison-Wesley.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., & Richards, C. (2001, July). Normalization of Non-Standard Words. *Computer Speech & Language, 15*(3), 287-333. doi:10.1006/csla.2001.0169

Sugiyono. (2012). *Memahami Penelitian Kualitatif.* Bandung: ALFABETA.

Tala, F. Z. (2003). *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.* Amsterdam.

Terrada, O., Cherradi, B., Raihani, A., & Bouattane, O. (2019). Classification and Prediction of atherosclerosis diseases using machine learning algorithms. *2019 5th International Conference on Optimization and Applications (ICOA)* (hal. 1-5). Kenitra, Morocco: IEEE. doi:10.1109/ICOA.2019.8727688

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017, December 6). Attention Is All You Need. *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, 6000-6010. Diambil kembali dari https://arxiv.org/abs/1706.03762

Wilie, B., Vincentio, K., Winata, G. I., Cahyawijaya, S., Li, X., Lim, Z. Y., . . . Purwarianti, A. (2020, December). IndoNLU: Benchmark and Resources for Evaluating Indonesian Natural Language Understanding. *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 843-857. Diambil kembali dari https://aclanthology.org/2020.aacl-main.85

# LAMPIRAN

**RIWAYAT HIDUP**

**DATA PRIBADI**

Nama : Rizky Anugerah

NPM : 140810180049

Tempat, Tanggal Lahir : Sumedang, 20 November 2000

Jenis Kelamin : Laki-Laki

Agama : Islam

Status : Belum Menikah

Alamat : Jalan Anggrek 2 No. 11 Perumnas Kuningan

Nomor HP : 081224043885

E-mail : [rizky18011@mail.unpad.ac.id](mailto:rizky18011@mail.unpad.ac.id)

**RIWAYAT PENDIDIKAN**

2005-2006 : TK Bhayangkari

2006-2012 : SD Negeri 1 Ciporang

2012-2015 : SMP Negeri 2 Kuningan

2015-2018 : SMA Negeri 2 Kuningan

2018-2021 : Teknik Informatika Universitas Padjadjaran

**PENGALAMAN ORGANISASI**

|  |  |  |
| --- | --- | --- |
| **Instansi** | **Jabatan** | **Tahun** |
| BE Himatif FMIPA Unpad | Staff Media dan Informasi | 2020 |

**PENGALAMAN KEPANITIAAN**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instansi** | **Nama Kepanitiaan** | **Jabatan** | **Tahun** |
| Himatif FMIPA Unpad | CBS | Staf Publikasi dan Dokumentasi | 2019 |
| BEM Kema FMIPA Unpad | Mipa Bersatu | Staf Akomodasi | 2019 |
| BEM Kema FMIPA Unpad | OSEAN | Staf Publikasi dan Dokumentasi | 2019 |
| Himatif FMIPA Unpad | Informatics Sport, Art, and Games | Staf Publikasi dan Dokumentasi | 2019 |
| Himatif FMIPA Unpad | Musyawarah Besar Himatif FMIPA Unpad | Staf Dana dan Konsumsi | 2018 |