ANALISIS BERITA *HOAX* DALAM BAHASA INDONESIA MENGGUNAKAN METODE *MULTILINGUAL BERT*

TUGAS AKHIR

Diajukan oleh:

IZZIA KHALKIA
NIM. 190705059
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

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

LEMBAR PERSETUJUAN

ANALISIS BERITA *HOAX* MENGGUNAKAN METODE *MULTILINGUAL BERT*

TUGAS AKHIR

Diajukan Kepada Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN) Ar-Raniry Banda Aceh
Sebagai Salah Satu Beban Studi Memperoleh Gelar Sarjana
pada Prodi Teknologi Informasi

Oleh:

IZZIA KHALKIA
NIM. 1907050059
Mahasiswa Fakultas Sains dan Teknologi
Program Studi Teknologi Informasi

Disetujui untuk Dimunaqasyahkan Oleh:

Pembimbing I,	Pembimbing II,
Hendri Ahmadian, M.I.M	**Mulkan Fadhli, M.T**
NIP. 198301042014031002	**NIP. 198811282020121006**

Mengetahui,
Ketua Program Studi Teknologi Informasi

Ima Dwitawati, MBA
NIP. 198210132014032002

LEMBAR PENGESAHAN

ANALISIS BERITA *HOAX* DALAM BAHASA INDONESIA MENGGUNAKAN METODE *MULTILINGUAL BERT*

TUGAS AKHIR

Telah Diuji Oleh Panitia Ujian Munaqasah Tugas Akhir
Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh dan Dinyatakan Lulus
Serta Diterima Sebagai Salah Satu Beban Studi Program Sarjana (S-1)
Dalam Prodi Teknologi Informasi

Pada Hari/Tanggal: Selasa, 15 Agustus 2023
28 Muharam 1444 H
di Darussalam, Banda Aceh

Panitia Ujian Munaqasyah Skripsi

Ketua,	Sekretaris,
Hendri Ahmadian, M.I.M	**Mulkan Fadhli, M.T**
NIP. 198301042014031002	**NIP. 198811282020121006**
Penguji I,	Penguji II,
Bustami, M.Sc	**Khairan AR, M.Kom**
NIP. 198604082014031001	**NIP.198607042014031001**

Mengetahui
Dekan Fakultas Sains dan Teknologi
UIN Ar-Raniry Banda Aceh

Dr. Ir. Muhammad Dirhamsyah, M.T.,IPU
NIP: 196210021988111001

LEMBAR PERNYATAAN KEASLIAN

Yang *BERT*anda tangan di bawah ini:

Nama	: Izzia Khalkia
NIM	: 1907005059
Program Studi	: Teknologi Informasi
Fakultas	: Sains dan Teknologi
Judul Tugas Akhir	: Analisis Berita *Hoax* dalam Bahasa Indonesia Menggunakan Metode *Multilingual BERT*

Dengan ini menyatakan bahwa dalam penulisan tugas akhir ini, saya:

1. Tidak menggunakan ide orang lain tanpa mampu mengembangkan dan mempertanggungjawabkan;
2. Tidak melakukan plagiasi terhadap naskah orang lain;
3. Tidak menggunakan karya orang lain tanpa menyebutkan sumber asli atau tanpa izin pemilik karya;
4. Tidak memanipulasi dan memalsukan data;
5. Mengerjakan sendiri karya ini dan mampu mempertanggungjawab atas karya ini;

Bila kemudian hari ini ada tuntutan dari pihak lain atas karya saya, dan telah melalui pembuktian yang dapat mempertanggungjawabkan dan ternyata memang ditemukan bukti bahwa saya telah melanggar pernyataan ini, maka saya siap dikenakan sanksi berdasarkan aturan yang berlaku di Fakultas Sains dan Teknologi UIN Ar-Raniry Banda Aceh.

Demikian pernyataan ini saya buat dengan sesungguhnya dan tanpa paksaan dari pihak manapun.

Banda Aceh,
Yang Menyatakan

(Izzia Khalkia)

ABSTRAK

Nama : Izzia Khalkia
NIM 190705059
Program Studi : Teknologi Informasi
Fakultas : Sains dan Teknologi (FST)

Judul :ANALISIS BERITA *HOAX* DALAM BAHASA INDONESIA MENGGUNAKAN METODE *MULTILINGUAL BERT*.

Tanggal Sidang : 15 Agustus 2023
Pembimbing I : Hendri Ahmadian, M.I.M
Pembimbing II : Mulkan Fadhli, M.T

Pengaruh teknologi informasi telah masuk ke berbagai bidang aktivitas manusia dan memberikan manfaat yang besar dalam berbagai bidang yaitu dalam bidang pendidikan, kesehatan, politik, ekonomi, teknologi komunikasi dan informasi. Penyebaran informasi dengan mudah merupakan hal yang *positive* namun tidak semua informasi yang disebarkan di internet berupa fakta, banyaknya berita beredar di internet adalah berita yang tidak benar atau disebut dengan *hoax*. Informasi *hoax* dapat merugikan banyak pihak yang tidak bersalah dengan meyakini pembaca tentang kejadian yang tidak benar. Cara tradisional untuk mengklasifikasikan berita asli atau palsu adalah dengan memeriksa berita dari sumber lain secara manual. Namun, itu membutuhkan usaha dan banyak waktu. Oleh karena itu upaya yang dapat dilakukan yaitu dengan mengklasifikasikan berita *hoax* menggunakan machine learning yaitu dengan metode *Multilingual BERT*, dimana *Multilingual BERT* atau multibahasa bekerja dengan sangat baik pada tugas transfer lintas bahasa, lebih unggul dari penyisipan kata non-*kontekstual statis*.

Penelitian ini menggunakan metode *Multilingual BERT* untuk *text classification*. *Dataset* penelitian ini berjumlah 1865 baris data. *Dataset* terdiri dari teks berita dan label berita *hoax*/non *hoax*. Metode *Multilingual BERT* pada penelitian ini menerapkan *Fine tuning Model* dan menyesuaikan *Hyperparameter* untuk mendapatkan hasil terbaik. Pada penelitian ini menggunakan bahasa pemograman *python* yang di aplikasikan di *Google collab*. Evaluasi terhadap hasil *text classification* dengan metode *Multilingual BERT* menggunakan *Confusion matrix*. Hasil evaluasi menggunakan metode *Confusion matrix* memilki nilai *accurasy* sebesar 96%, nilai *Precission* 75%, nilai *recall* 55% dan *F1-Score* 63%. Secara kesuluruhan hasil *text classification berita hoax* menggunakan metode *Multilingual BERT* dapat dikategorikan baik.

Kata Kunci : ***Text Classification, Multilingual BERT, Confusion matrix*****.**

ABSTRACT

Name	: Izzia Khalkia
Student ID	190705059
Study Program	: Information Technology
Faculty	: Sains and Technology (FST)
Title	: ANALYSIS OF HOAX NEWS IN INDONESIAN USING THE MULTILINGUAL BERT METHOD
Sassion Date	: 15 Agustus 2023
Supervisor I	: Hendri Ahmadian, M.I.M
Supervisor II	: Mulkan Fadhli, M.T

The influence of information technology has entered various fields of human activity and provides great benefits in various fields, namely in the fields of education, health, politics, economics, communication and information technology. Disseminating information easily is a positive thing, but not all information disseminated on the internet is fact, a lot of news circulating on the internet is news that is not true or is called a hoax. Hoax information can harm many innocent parties by believing readers about events that are not true. The traditional way to classify genuine or fake news is to manually check news from other sources. However, it requires effort and a lot of time. Therefore, efforts that can be made are to classify hoax news using machine learning, namely the Multilingual BERT method, where Multilingual BERT or multilingual works very well on cross-language transfer tasks, superior to static non-contextual word insertion.

This study uses the Multilingual BERT method for text classification. The dataset of this research is 1865 lines of data. The dataset consists of news text and hoax/non-hoax news labels. The Multilingual BERT method in this study applies the Fine tuning Model and adjusts the Hyperparameters to get the best results. This research uses the Python programming language which is applied in Google Collab. Evaluation of text classification results with the Multilingual BERT method using the Confusion matrix. The evaluation results using the Confusion Matrix method have an accuracy value of 96%, a precision value of 75%, a recall value of 55% and an F1-Score of 63%. Overall, the results of hoax news text classification using the Multilingual BERT method can be categorized as good.

Key Word's : ***Text Classification, Multilingual BERT, Confusion matrix***

KATA PENGANTAR

Segala puji dan syukur selalu kita panjatkan atas kehadiran Allah SWT yang atas segala rahmat dan hidayah-Nya kita masih dapat melihat Alam semesta yang indah ini. Tak lupa pula shalawat beriring salam selalu kita panjatkan untuk tuntunan suri tauladan Baginda Rasulullah Shallauhu‟alaihiwasalam dan beserta keluarga dan sahabat beliau yang senantiasa menjunjung tinggi nilai-nilai keislaman serta menggali ilmu yang tiada habisnya yang sampai saat ini masih bisa dinikmati oleh setiap manusia, sehingga penulis dapat menyelesaikan skripsi yang berjudul "**Analisis Berita *Hoax* dalam Bahasa Indoensia Menggunakan Metode *Multilingual BERT***".

Penyusunan tugas akhir ini diajukan sebagai salah satu syarat untuk menyelesaikan tugas akhir pada Program Studi Teknologi Informasi, Fakultas Sains dan Teknologi, UIN Ar-Raniry. Dalam penulisan proposal tugas akhir ini, penulis dengan segala kerendahan hati ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Kepada kedua orang tua yang penulis cintai karena Allah, Usman Daud dan Nurlaili yang senantiasa mendoakan, membimbing, mendidik, serta memberikan semangat dan dukungan kebaikan tanpa batas, semoga Allah membalas segala jasa-jasanya dengan kebaikan yaitu SurgaNya.
2. Segenap keluarga dan sahabat yang selalu menyemangati dan membantu penulis untuk menyelesaikan skripsi ini dari awal hingga akhir.
3. Bapak Dekan Fakultas Sains dan Teknologi Muhammad Dirhamsyah yang selalu mendukung dan memberi motivasi untuk kami.
4. Bapak Hendri Ahmadian, S.Si.,M.I.M sebagai pembimbing pertama dan Bapak Mulkan Fadhli, M.T sebagai pembimbing kedua, yang telah meluangkan waktunya dan mencurahkan pemikirannya dalam membimbing penulis untuk menyelesaikan skripsi ini.
5. Ketua Prodi Teknologi Informasi Ibu Ima Dwitawati, M.B.A. Sekretaris Prodi Teknologi Informasi Bapak Khairan AR.,M.Kom., serta staf Prodi yang telah ikut membantu proses pelaksanaan penelitian.

6. Kepada Staf Prodi Ibu Cut Ida Rahmadiana S,Si. yang telah membantu penulis dalam hal pengurusan administrasi dan surat-surat untuk keperluan penyelesaian tugas akhir.
7. Bapak dan Ibu dosen Program Studi Teknologi Informasi yang telah memberikan ilmu pengetahuan dalam bidang teknologi inofrmasi kepada penulis sehingga penulis mampu menyelesaikan tugas akhir karya ilmiah ini.
8. Sahabat dan teman-teman mahasiswa program studi Teknologi Informasi angkatan 2019 terkhusus nya kepada Nurhaliza, Alisa Sausan Nabila, Najla Haura, Cut Elfia Zahara dan Farla Nara Viana serta seluruh keluarga Teknologi Informasi yang telah memberikan dukungan dan semangat dalam penyelesaian tugas akhir ini.

Penulis menyadari sepenuhnya bahwa dalam penyusunan tugas akhir ini tidak cukup dikategorikan sempurna, untuk itu penulis dengan segala kerendahan hati menerima saran dan kritikan guna menyempurnakan penyusunan tugas akhir ini. Akhir kata, semoga tugas akhir ini dapat bermanfaat bagi penulis dan pembaca dan semoga dicatat sebagai sebuah amal kebaikan oleh Allah SWT.

Amin Ya Rabbal A‟lamin.

Banda Aceh, 04 April 2023

Penulis

Izzia Khalkia

DAFTAR ISI

LEMBAR PERSETUJUAN ..... **i**
LEMBAR PENGESAHAN ..... **ii**
LEMBAR PERNYATAAN KEASLIAN ..... **iii**
ABSTRAK ..... **iv**
KATA PENGANTAR ..... **vi**
DAFTAR ISI **1**
DAFTAR GAMBAR **3**
DAFTAR TABEL **4**
BAB I PENDAHULUAN **5**
1.1 Latar Belakang 5
1.2 Rumusan Masalah 7
1.3 Tujuan Penelitian 7
1.4 Batasan Penelitian 7
1.5 Manfaat Penelitian 7
BAB II TINJAUAN PUSTAKA **8**
2.1 Penelitian Terdahulu 8
2.2 Berita *Hoax* 11
2.6 *Deep learning* 13
2.7 *Transformers* 14
2.8 *BERT* 16
2.9 *Multilingual BERT* 18
2.10 *Pre-training* Model dan *Fine tunning* 19
2.11 *Tools* 20
2.11.1 Python 20
2.11.2 Google collaboratory 20
2.12 Model Evaluasi 21
2.12.1 Confusion matrix 21

BAB III METODOLOGI PENELITIAN 24
3.1 Jenis Penelitian 24
3.2 Tahapan Penelitian 24
3.3 Metode Pengumpulan Data 25
3.3.1 Observasi 25
3.3.2 Studi Literatur 25
3.4 *Pre-processing* Data 25
3.5 Pelabelan Data 26
3.6 Metode Simulasi 26
3.6.1 Menemukan Formulasi Permasalahan 26
3.6.2 Konsep Penelitian 26
3.7 Alur Implementasi Model 27
3.7.1 *Dataset* 27
3.7.2 Pembagian Data 28
3.7.3 Implementasi Model dan Arsitektur 29
3.8 Metode Analisis Hasil 32
3.8.1 Evaluasi 32
3.9 Alat Bantu Penelitian **32**
BAB IV HASIL PENELITIAN DAN PEMBAHASAN 33
4.1 Data *Preparation* 33
4.1.2 *Tokenization* 34
4.2 *Modelling* 36
4.2.1 *LoadModel* 37
4.2.2 *Fine Tuning & Train Model* 37
4.3 Hasil Analisis Terhadap Implementasi Model 39
4.4 Hasil Uji Coba menggunakan *Confusion matrix* 47
BAB V KESIMPULAN DAN SARAN 51
5.1 Kesimpulan 51
5.2 Saran 52
DAFTAR PUSTAKA 52
LAMPIRAN 55

DAFTAR GAMBAR

Gambar 2. 1 Proses NLP 12
Gambar 2. 2 Arsitektur *Transformers* (Mulyawan, 2023). 15
Gambar 2. 3 Representasi Input *BERT* (Devlin et al., 2019) 17
Gambar 2. 4 Tahapan *Pre-train* dan *Finetunning* 19
Gambar 3. 1 Tahapan penelitian 25
Gambar 3.2 Alur Implementasi Sistem 27
Gambar 3.3 Implementasi Model *BERT* 29
Gambar 4. 1 *DownLoadDataset* 34
Gambar 4. 2 Sampel *Dataset* Teks Berita *Hoax* 34
Gambar 4. 3 *BERT Tokenizer* from *Hugging face* 35
Gambar 4. 4 Proses Tokenisasi 35
Gambar 4. 5 *Padding* Data 36
Gambar 4. 6 *Load* model 37
Gambar 4. 7 *Code* untuk menggunakan *Hyperparameter* 37
Gambar 4. 8 *Train* Model 38
Gambar 4. 9 Diagram *Confusion matrix* Model dengan *Learning rate* 2e-5 dan 10 *epoch* 47
Gambar 4. 10 Hasil perhitungan *confusion matrix* 49

DAFTAR TABEL

Tabel 2. 1 Perbandingan Penelitian Sejenis 10
Tabel 2. 2 *Confusion matrix* 22
Tabel 3. 1 Pembagian jumlah data *training* dan testing 28
Tabel 3. 2 Algoritma implementasi model *BERT* 32
Table 4. 1 Hyperparameter *Multilingual BERT* 38
Table 4. 2 Perbandingan Grafik *Training loss* pada *epoch* 10 40
Table 4. 3 Hasil Perbandingan *Accurasy* Pada *Epoch* 10 43
Table 4. 4 Perbandingan Grafik *Training Loss* pada *epoch* 5 43
Tabel 4.5 Hasil perbandingan *Training loss* dengan *epoch* 5 46
Table 4. 5 Hasil perhitungan *Confusion matrix* 48

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengaruh teknologi informasi telah masuk ke berbagai bidang aktivitas manusia dan memberikan manfaat yang besar dalam berbagai bidang yaitu dalam bidang pendidikan, kesehatan, politik, ekonomi, teknologi komunikasi dan informasi. Beberapa alat teknologi yang dimanfaatkan untuk komunikasi dan informasi yaitu *smartphone*, *email*, televisi, internet dan media lainya.

Pengguna internet merupakan wadah yang tepat sebagai media komunikasi dan juga berperan sebagai media penyebaran informasi. Penyebaran informasi dengan mudah merupakan hal yang *positive* namun tidak semua informasi yang disebarkan di internet berupa fakta, banyaknya berita beredar di internet adalah berita yang tidak benar atau disebut dengan *hoax*.

Berita *hoax* merupakan informasi menyesatkan dan manipulatif dengan menyebarkan informasi yang salah namun dianggap benar. Motif umum untuk menyebarkan pesan semacam itu adalah untuk menyesatkan pembaca, merusak reputasi, atau mendapatkan keuntungan dari sensasi. Sementara berita nyata adalah konten nyata yang memberikan informasi nyata dari sumber yang sah seperti jurnalis tepercaya atau outlet berita terkemuka. Dengan meningkatnya jumlah pengguna internet dan keterjangkauan smartphone, sebagian besar masyarakat Indonesia memiliki akses ke media sosial. Informasi *hoax* dapat merugikan banyak pihak yang tidak bersaalah dengan meyakini pembaca tentang kejadian yang tidak benar. Cara tradisional untuk mengklasifikasikan berita asli atau palsu adalah dengan memeriksa berita dari sumber lain secara manual. Namun, itu membutuhkan usaha dan banyak waktu (Isa et al., 2022).

Setiap tahun semakin banyak berita *hoax* beredar di kalangan masyarakat, karena banyaknya berita *hoax* munculah tindakan-tindakan pencegahan berita *hoax* dan banyak media yang menyajikan layanan untuk melaporkan konten yang diduga mengandung unsur *hoax*. Saat ini, model transformator digunakan untuk sebagian besar tugas klasifikasi dalam *Natural Language Processing* (NLP).

Alasannya karena model transformer memiliki mekanisme perhatian, yang akan bekerja lebih baik daripada model pembelajaran mesin lainnya. Oleh karena itu upaya yang dapat dilakukan yaitu dengan mengklasifikasikan berita *hoax* menggunakan machine learning yaitu dengan metode *Multilingual BERT*, dimana *Multilingual BERT* atau multibahasa bekerja dengan sangat baik pada tugas transfer lintas bahasa, lebih unggul dari penyisipan kata non-kontekstual statis.

Penelitian klasifikasi berita *hoax* telah dilakukan sebelumnya oleh Sani Muhammad Isa, dkk pada tahun 2021 dengan menggunakan berita *hoax* yang berasal dari website *turnbackhoax.id*. Pada penelitian tersebut menggunakan metode *IndoBERT* untuk melakukan klasifikasi dengan data 1.028 berita bohong tentang COVID 19. Dalam penelitian tersebut diperoleh *accurasy* sebesar 94,66 %. Penelitian semacam ini telah dilakukan sebelumnya oleh Antonius Angga Kurniawan dan Metty Mustikasari pada tahun 2021 melakukan klasifikasi berita *hoax* dalam bahasa Indonesia, dalam penelitian nya metode CNN memiliki *accurasy* terbaik yaitu 0,88 daripada metoode LSTM 0,84.

Fokus penelitian ini yaitu melakukan pengklasifikasian terhadap berita *hoax* dalam bahasa Indonesia dengan menggunakan *dataset* dari github milik Andi Aqil Amanulhaq yaitu: https://github.com/a3x-crypt/indo-covid19-news.git . Klasifikasi teks berita *hoax* secara abstraksi yaitu area pemrosesan bahasa alami yang mengotomatiskan klasifikasi teks menjadi satu atau beberapa kategori yang sesuai berdasarkan kontennya dengan membangun model menggunakan data pelatihan. Dalam penelitian ini peneliti menerapkan model *Multilingual BERT* untuk mengklasifikasi teks berita *hoax*.

Bedasarkan uraian di atas, penulis berinisiatif untuk melakukan penelitian dengan judul "*Analisis Berita hoax dalam Bahasa Indonesia Menggunakan Metode Multilingual BERT*". Penulis berharap dengan adanya penelitian ini dapat mengetahui proses klasifikasi teks berita *hoax* serta mengetahui kinerja model *Multilingual BERT* dalam analisis berita *hoax* di Indonesia.

1.2 Rumusan Masalah

1. Bagaimana proses kalsifikasi berita *hoax* menggunakan model *Multilingual BERT* ?
2. Bagaimana tingkat *accurasy* dalam pengklasifikasian berita *hoax* dengan model *Multilingual BERT*?

1.3 Tujuan Penelitian

1. Mengetahui proses klasifikasi berita *hoax* menggunakan model *Multilingual BERT.*
2. Mengetahui tingkat *accurasy* pengklasifikasian berita *hoax* dengan model *Multilingual BERT.*

1.4 Batasan Penelitian

1. *Dataset* yang digunakan adalah *dataset* berita *hoax* yang diambil dari github yaitu https://github.com/a3x-crypt/indo-covid19-news.git yang berisi tentang berita *hoax* tentang covid 19 pada tahun 2019.
2. Metode yang digunakan pada penelitian ini adalah *Multilingual BERT*-(*Bidirectoral Encoder Representations from transformer*)
3. Bahasa pemograman yang digunakan pada penelitian ini adalah *Python* dengan menggunakan *Google Collab* sebagai media untuk menjalankan program.

1.5 Manfaat Penelitian

1. Dapat digunakan sebagai referensi baru untuk mengetahui performasi *Multilingual BERT* dalam melakukan pendeteksian berita *hoax*.
2. Dapat dimanfaatkan sebagai penilaian terhadap berita *hoax* di Indonesia.
3. Mengetahui tingkat *accurasy* dari implementasi *deep learning* menggunakan metode *Multilingual BERT* pada analisis klasifikasi teks berita *hoax*.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Terkait dengan penelitian yang penulis lakukan menggunakan model pembelajaran *deep learning*, dibutuhkan referensi atau penelitian terkait guna untuk terhindar dari duplikasi dan plagiarisme, sehingga penulis dapat mengembangkan sesuatu hal yang berbeda pada penelitian ini. Berikut ini adalah beberapa penelitian terkait yang berhubungan dengan penelitian penulis.

Penelitian mengenai "*Implementasi Deep learning Menggunakan Metode CNN dan LSTM Untuk Menentukan Berita Palsu dalam Bahasa Indonesia"*(Kurniawan & Mustikasari, 2021). Penelitian ini menggunakan metode CNN dan LSTM. Data yang diambil dari situs penyedian berita-berita *hoax* dan berita fakta yang sudah valid, yaitu *TurnbackHoax*.id. data yang digunakan berjumlah 1786 berita dengan jumlah berita fakta sebanyak 802 dan berita palsu sebanyak 984. Penelitian ini menghasilkan bahwa metode CNN dan LSTM berhasil diterapkan untuk menentukan berita fakta dan berita palsu dalam bahasa Indonesia dengan baik. CNN memiliki tingkat *accurasy test*, *precision* dan *recall* sebesar 0.88, sedangkan model LSTM memiliki tingkat *accurasy test*, *precision* sebesar 0,84 dan *recall* sebesar 0,83.

Penelitian mengenai "*Deteksi Berita Hoax Pada Website TurnbackHoax dengan Menggunakan Machine Learning* (Rahmawati, 2021). Penelitian ini menggunakan data dari situs website Turnback*Hoax* dengan jumlah berita yang di analisis sebanyak 4.551. Data di latih untuk dapat memprediksi klasifikasi dengan Machine Learning yang digunakan yaitu *Random Forest Clasifisier.* Model Terbaik dalam penelitian ini yaitu Support Vector Machine dengan *accurasy* sebesar 83% dan *recall* pada kelas *hoax* sebesar 99%. Tujuan dari pekerjaan ini adalah untuk menemukan solusi yang dapat digunakan oleh pengguna untuk mendeteksi dan menyaring situs yang berisi informasi palsu dan menyesatkan.

Penelitian mengenai "*IndoBERT untuk Deteksi Berita Palsu Indonesia"*(Isa et al., 2022). Penelitian ini menggunakan model Indo*BERT* dan *BERT* yang berfokus pada konteks dan perhatian kalimat masukan. *Dataset* yang dikumpulkan dari website *TurnbackHoax.id* yaitu sebanyak 3.465 berita palsu dan 766 berita nyata. Dari percobaan tersebut telah dibuktikan bahwa membangun deteksi berita palsu Indonesia menggunakan model Indo*BERT* mendapatkan *accurasy* sebesar 94,66 % pada *Precission*, daya ingat, dan skor F1.

Penelitian mengenai "*Implementasi Deteksi Judul Berita Clickbait Berbahasa Indonesia dengan pre-trained model Multilingual BERT Pada Aplikasi Berbasis Chrome Extension"*(Girinoto et al., 2022). Dalam penelitian ini menggunakan *dataset* yang diambil dari 12 website berita. *Headline* berita Indonesia dari sembilan kategori berita beranotasi dengan data dari 5297 baris data *BERT*anda non-*clickbait* dan 3316 baris data *BERT*anda clickbait. Kumpulan data ini memiliki perjanjian *interrater Fleis kappa* sebesar 0,42. Karena jumlah tag judul berita yang tidak seimbang, *dataset* di *subsampling* untuk menyeimbangkan jumlah tag clickbait dan tag non-clickbait. Setelah dilakukan subsampling, maka jumlah data yang digunakan untuk proses pelatihan adalah 6632 headline berita yang memiliki jumlah headline yang sama dan berimbang.

Penelitian mengenai "*Pendekatan Metode Transformer untuk Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia"* (Rendragraha et al., 2021). Model yang digunakan adalah model *BERT* dan model *Multilingual BERT Pre-training* yang berfungsi sebagai titik awal. Sistem menerima masukan berupa teks komentar, yang kemudian memberikan pengenal untuk mengklasifikasikan teks komentar tersebut sebagai ofensif, normal atau non-ofensif. Hasil model *Scratch*, yang dilatih pada *dataset* bahasa Indonesia, mencapai skor F1 rata-rata makro sebesar 50% dibandingkan dengan 54% untuk *Multilingual BERT*. *Dataset* yang diekstrak berukuran 95.876 KB dan bersumber dari Wikipedia, Twitter, Kompas dan Corpus *Frog Storytelling*.

Penelitian mengenai "Analisis Sentimen *Review* Film Berbahasa Inggris Dengan Pendekatan *Biderectional Encoder Representations From Transformers"*(Putri, 2020). Model yang digunakan adalah model *Multilingual BERT* yang dilakukan *fine-tuning* dengan beberapa *layer* untuk klasifikasi. Pada penelitian ini merekan melaakukan klasifikasi sentiment analisis terhadap review film dengan menggunakan *dataset Cornelledu* dari *Pobo*. Dari penelitian ini didapatkan hasil *accurasy* yang dihitung dengan menggunakan *confision matrix* sebesar 73%.

Tabel 2. 1 Perbandingan Penelitian Sejenis

Peneliti	Metode	Kasus	Jumlah *Dataset*	*Accurasy*
(Kurniawan & Mustikasari, 2021)	CNN dan LSTM	Menentukan Berita Palsu dalam Bahasa Indonesia	1.786	CNN, 0,88 LSTM, 0,84
(Rahmawati, 2021)	*Support Vektor Machine*	Deteksi Berita *Hoax* Pada *Website TurnbackHoax* dengan Menggunakan *Machine Learning*	4.551	83%
(Isa et al., 2022)	Indo*BERT*	Deteksi Berita Palsu di Indonesia	3.465	94,66%
(Rendragraha et al., 2021)	*BERT*	Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia	95,874 KB	50%
(Girinoto et al., 2022)	*Multilingual BERT*	Deteksi Berita *Clickbait* Berbahasa Indonesia	6632	92%
(Putri, 2020)	*Multilingual BERT*	Analisis Sentimen *Review Film* Berbahasa Inggris	2.000	73%

2.2 Berita *Hoax*

Berita palsu (*fake news*) adalah berita dibuat untuk menipu pembaca. Ada dua motivasi utama penyebaran berita bohong. Yang pertama adalah uang. Berita menjadi viral di media sosial dan dapat menghasilkan pendapatan iklan yang signifikan saat pengguna mengklik ke situs web asli. Hal inilah yang tampaknya menjadi motivasi utama sebagian besar pembuat berita bohong yang identitasnya terbongkar. Motivasi lain bersifat ideologis (Danu Nur Irwanto, 2021).

Informasi *Hoax* dapat merugikan banyak pihak yang tidak bersalah dengan meyakini pembaca tentang kejadian yang tidak benar. Bedasarkan Surat Al-Hujarat ayat 6 dapat dipahami bahwa ayat ini menunjukan dengan jelas perlunya memeriksa dengan teliti sebelum dipahami untuk mempercayai berita dan disebarkan kepada orang lain agar tidak menimbulkan musibah.

Allah SWT berfirman dalam Surat *Al-Hujarat* ayat 6:

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُوٓاْ إِن جَآءَكُمْ فَاسِقٌۢ بِنَبَإٖ فَتَبَيَّنُوٓاْ أَن تُصِيبُواْ قَوْمَۢا بِجَهَٰلَةٖ فَتُصْبِحُواْ عَلَىٰ مَا فَعَلْتُمْ نَٰدِمِينَ

Artinya :

“Hai orang-orang yang beriman, jika dating kepadamu orang fasik membawa suatu berita, maka periksalah dengan teliti, agar kamu tidak menimpakan suatu musibah kepada suatu kaum tanpa mengetahui keadaanya yang menyebabkan kamu menyesal atas perbuatanmu itu”(*Surat Al-Hujarat Ayat 6*, n.d.).

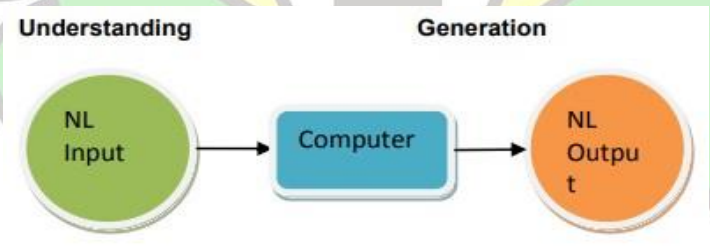
Karena banyaknya berita *hoax* yang beredar di kalangan masyarakat maka muncullah tindakan tindakan pencegahan berita *hoax* dengan melaporkan konten/berita yang di duga mengandung unsur *hoax*. Salah satu upaya yang dilakukan yaitu dengan mengklasifikasikan yang mana berita *hoax* dan mana yang fakta menggunakan *machine leaning* khususnya *BERT* (*Bidiretional Encoder Representation from Transformer*). Oleh karena itu, penggunaan machine learning untuk proses klasifikasi diharapkan memiliki keakuratan yang tinggi dalam hal *accurasy*. Sehingga kedepanya penyebaran berita *hoax* bisa berkurang dan masyarakat tidak dibuat bingung lagi dengan informasi atau berita yang beredar (Danu Nur Irwanto, 2021).

2.3 Klasifikasi Teks

Klasifikasi teks adalah area pemrosesan bahasa alami yang mengotomatiskan klasifikasi teks menjadi satu atau beberapa kategori yang sesuai berdasarkan kontennya dengan membangun model menggunakan data pelatihan. Klasifikasi teks diterapkan dalam berbagai konteks, mulai dari pengindeksan dokumen hingga kosa kata terpandu, pemfilteran dokumen, pembuatan metadata otomatis, dan beberapa aplikasi lain yang memerlukan organisasi dokumen. Ada beberapa strategi umum untuk menggunakan klasifikasi teks, yaitu *pre-processing* teks, ekstraksi fitur, pemodelan dengan teknik pembelajaran mesin yang sesuai, dan pelatihan dan pengujian classifier (Hayati, 2018).

2.4 *Natural Language Processing* (NLP)

Bahasa alami atau *Natural Language* adalah bahasa yang digunakan oleh orang-orang. Pemrosesan Bahasa Alami menggabungkan semua yang dibutuhkan komputer untuk memahami bahasa alami dan juga untuk menghasilkan bahasa alami. NLP adalah cabang dari kecerdasan buatan dan *linguistik* yang berfokus untuk membuat komputer memahami pernyataan atau kata-kata yang ditulis dalam bahasa manusia. Bahasa alami juga dikenal sebagai bahasa biasa, diucapkan atau ditulis oleh manusia untuk komunikasi umum (Martinez, 2010).



Gambar 2. 1 Proses NLP

Pada gambar diatas dapat dilihat bahwa ada dua proses yang terjadi yaitu:

1. *Natural Language Understanding* (NLU), bertugas untuk memahami bahwa didalam input adalah bahasa alami atau Natural Language.
2. *Natural Language Generation* (NLG), bertugas dalam sub generasi pemrosesan bahasa alami atau disebut juga dengan pembuatan teks (Martinez, 2010).

2.5 Artificial Inteligence

AI atau kecerdasan buatan adalah cabang ilmu komputer yang memandang upaya membangun komputer sebagai sesuatu yang dapat dilakukan manusia, bahkan lebih baik dari itu. AI merupakan mesin yang bertindak dan meniru kecerdasan layaknya manusia. Menurut Luger dan Wiliam, kecerdasan buatan adalah cabang ilmu computer yang berhubungan dengnan otomasi perilaku cerdas (1993). Kemudian menurut Haag dan Peter kecerdasan buatan adalah bidang studi yang berhubungan dengan penangkapan, pemodelan, da penyimpanan kecerdasan manusia ke dalam sebuah sistem teknologi informasi sehingga sistem tersebut dapat digunakan sebagai prosess pengambilan keputusan yang dilakukan oleh manusia (1996) (Dewi, 2020).

Kecerdasan buatan memilki banyak cakapan, supaya tidak salah dalam mengartikan kecerdasan buatan itu sendiri maka perlu diketahui pengertian dan cakupan dari kecerdasan buatan itu sendiri. Pengelompokan domain aplikasi kecerdasan buatan terbagi menjadi 3 aplikasi yaitu, Aplikasi Pengetahuan Kognitif, Aplikasi Robotika, Aplikasi antarmuka Alami. Aplikasi Pengetahuan Kognitif dibagi menjadi beberapa bagian yaitu: Sistem Pakar, Sistem Belajar, Logika Kabur, Algoritma Genetika, Jaringan Syaraf, dan Agen Cerdas. Aplikasi Robotika terdiri dari Persepsi Visual, Rangsangan, Ketangkasan, Daya Penggerak, dan Navigasi. Aplikasi Antarmuka Alami terdiri dari Bahasa Alami, Pengenalan Percakapan, Antarmuka Multisensor, dan Virtual Reality, kecerdasan buatan terdiri dari: Pengolahan Bahasa Alami, Visi Komputer, Pengenalan Percakapan, Robotika, Sistem Pakar, logika kabur, jaringan saraf, Algoritma genitika, Sistem AI hibrida, dan Agen Cerdas (Dewi, 2020).

2.6 Deep learning

Deep learning adalah salah satu cabang dari *machine learning* yang algoritmanya memiliki abstrasksi tingkat tinggi pada sekumpulan data. *Deep learning* disebut juga dengan *Representations Learning*. *Deep learning* juga dapat melakukan model komputasi yang memilki beberapa *layer* pengolahan untuk mempelajari representasi data. Teknologi *deep learning* telah digunakan di berbagai produk *BERT*eknologi tinggi seperti mobil self-driving. Selain itu, dia juga mendukung produk dan layanan yang kami gunakan setiap hari. Contohnya

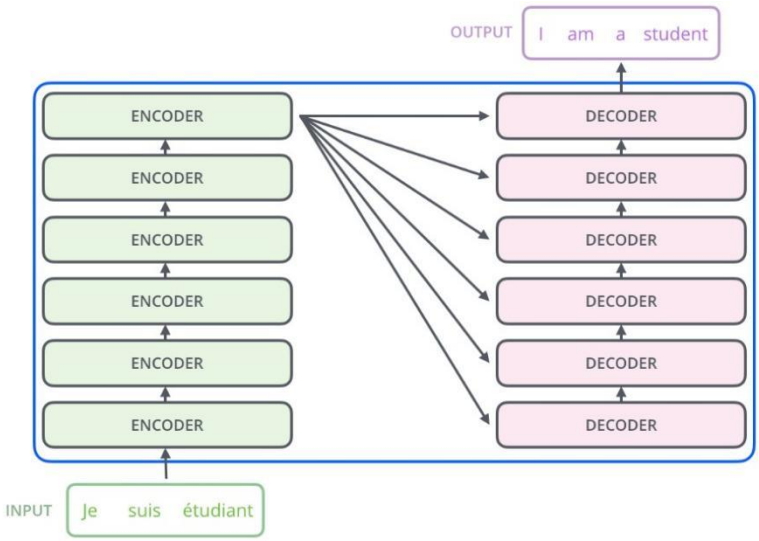
termasuk asisten digital, *Google* Terjemahan, dan perangkat yang diaktifkan suara (perangkat pintar yang dapat diaktifkan dengan suara) (Setiawan, 2021).

Deep learning adalah metode yang memanfaatkan *Artificial Neural Network* atau jaringan saraf tiruan, dimana *Artificial Neural Network* dibuat mirip dengan otak manusia. *Deep learning* pada komputer melakukan pembelajaran dengan cara mengklarifikasi langsung pada objek, seperti Gambar, suara, teks dan yang lainnya. Karena model ini dilatih dengan menggunakan *dataset* berlabel dengan jumlah yang banyak dan *Neural Network* dapat melakukan penyelesaian masalah secara akurat dan otomatis. Metode *Deep learning* sebagian besarnya menggunakan *Neural Network Architecture*, oleh karena itu *Deep learning* sering disebut dengan *Deep Neural Network*. Dari istilah *"deep"* dapat mengacu pada jumlah lapisan yang tersembunyi di *Neural Network* atau dengan kata lain *hidden layer*. Pada *Neural Network* tradisonal hanya mengandung 2-3 lapisan saja, namun berbeda hal dengan *Deep Neural Network* dapat memiliki 150 lapisan(Reza et al., 2020).

Deep learning dapat disebut juga dengan *Representation Leraning*. *Deep learning* melakukan model komputasi yang memiliki bebrapa *layer* pengolalahan agar mempelajari sebuah representasi data. Metode ini meningkatkan pengembangan pada berbagai bidang yaitu pengenalan suara (*voice recognition*), pengenalan *object* visual (*image recognition*), deteksi objek (*object detection*). *Deep learning* saat ini temasuk masuk kedalam terobosan berbagai bidang pada *artificial intelligence*.

2.7 Transformers

Transformers adalah model transmisi pertama yang sepenuhnya bergantung pada *Self-attention* untuk menghitung representasi input dan outputnya, sebuah mekanisme untuk mengetahui hubungan kontekstual antar kata. *Transformers* terutama digunakan dalam pemrosesaan bahasa alami dan *computer vision.* Model ini dirancang untuk memprosess tipe data *sequence* seperti bahasa alami untuk melakukan tugas seperti terjemah bahasa dan perigkasan teks. *Transformers* terdiri dari dua buah tumpukan(*stack*) (Rendragraha et al., 2021). Pada Gambar dibawah ini adalah arsitektur *transformers*.



Gambar 2. 2 Arsitektur *Transformers* (Mulyawan, 2023).

BERT menggunakaan arsitektur *neural network* yang disebut *transformers* yang teerlihat pada gambar 2.2 bahwa *BERT* hanya menerima *input* berupa vector dengan teknik *word embeddings*. Proses *embeddings* setiap token dalam urutan input direpresentasikan melallui proses ini. Karena arsitektur *transformers* tidak memilki koneksi berulang, maka posisi dari setiap token dalam urutan input harus secara eksplisit direpresentasikan dengan menambahkan vector *positional encoding* kedalam *input embeddings*. Urutan input besserta *positional encoding* nya kemudian dimasukkan kedalam mekanisme *multihead self attention*. Mekanisme ini memungkinkan model untuk focus pada bagian bagian yang berbeda dalam urutan input pada setiap lapisan dan menangkap ketergantungan jarak jauh antar token. Setelah mekanisme *self attention*, output dimasukkan ke dalam j*aringan feed forward*. Jaringan ini menerapkan tranformasi *non linear* untuk setiap posisi secara independen. Untuk meningkatkan pelatihaan model dan membantunya konvergen lebih cepat, *residual connection* dan lapisan *normalization* digunakan. *Residual connection* memungkinkan gradient untuk mengalir lebih mudah melalui jaringan, sedangkan lapisan *normalization* membantu untuk menstabilkan ditribusi nilai *output*.

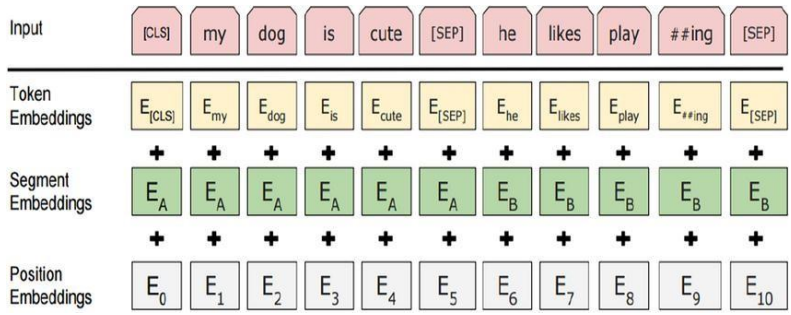
Arsitektur *transformers* memilki *stack encoder*. *Stack encoder BERT*anggung jawab untuk meng-eng*Code* unrutan input, sementara stack *decoder BERT*anggung jawab untuk mengghasilkan urutan output. Pada stack

decoder, mekanisme self attention dijadikan mask sehingga setiap posisi hanya dapat focus pada posisi selanjutnya termasuk posisi saat ini. Hal ini diperlukan untuk mencegah model curang dan menghasilkan output yang bergantung pada token masa mendatang. Selama proses decoding stack *decoder* juga memperhatikan output dari stack *encoder*. Hal ini memungkinkan model untuk menggunakan informasi dari urutan input untuk menghasilkan urutan output. Akhirnya, output dari stack *decoder* di proyeksikan menjadi vector probabilitas atas kosakata. Distribusi probabilitas ini digunaka untuk menghasilkan urutasn output token per token. (Rahmatullah, 2021).

2.8 BERT

Bidirectional Encoder Representations from Transformers (*BERT*) merupakan model representasi bahasa terlatih yang pertama kali di populerkan oleh *Google* pada 11 Oktober 2018. Model ini berbeda dengan model sebelumnya yang bersifat satu arah, model ini dibuat sebagai *pre-trained model* atau model yang sudah dilatih. Inovasi utama *BERT* adalah representasi kata dua arah (*bidirectional*) *BERT* adalah model representasi kata kontekstual terlatih berdasarkan Mask *Language Model* (Model Bahasa *BERT*openg) menggunakan transformator dua arah. Arsitektur model *BERT* adalah struktur dua arah transformer-*encoder-decoder* multi*layer*. *Transformers* mengikuti arsitektur keseluruhan ini dengan kesadaran diri dan pembuat enkode dan dekoder yang terhubung sepenuhnya yang ditumpuk dari titik ke titik (Tandijaya et al., 2021).

Kinerja kerangka kerja *BERT* memiliki dua langkah: pra-pelatihan dan penyempurnaan. Pra-pelatihan *BERT* tidak menggunakan metode kiri-ke-kanan atau kanan-ke-kiri tradisional, tetapi menggunakan *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP) pada data pra-pelatihan. MLM mengisi kesenjangan. Model menggunakan kata-kata konteks di sekitar token topeng untuk memprediksi kata mana yang akan diprediksi, sedangkan NSP adalah prediksi kalimat berikutnya yang diberikan pada kedua model. Setelah data pra-pelatihan, *BERT* melakukan penyempurnaan. Penyempurnaan atau lebih dikenal dengan *fine tuning* diinisialisasi dengan parameter yang telah dilatih sebelumnya, dan semua parameter penyempurnaan menggunakan data berlabel dari tugas hilir (Tandijaya et al., 2021).



Gambar 2. 3 Representasi Input *BERT* (Devlin et al., 2019)

Representasi input *BERT* di tampilkan pada gambar 2.3 berikut merupakan langkah langkah tokenisasi dalam *BERT*.

1. Tokenisasi : Membagi teks menjadi token token yang terdiri dari kata kata. *BERT* menggunakan tokenisasi Word pieces, yang berarti bebrapa token dapat dibagi lagi menjadi sub token.
2. Token *embeddings* : *BERT* menambahkan dua token khusus ke awal dan akhir setiap kalimat, yaitu [CLS] dan [SEP]. Token [CLS] digunakan untuk merepressentasikan kalimat secara keselurihan yang berada di awal kalimat, sedangkan token [SEP] di akhir kalimat digunakan untuk memisahkan kalimat dalam input yang bearada dari urutan input.
3. Konversi token menjadi ID : Setiap token dalam input kemudian dikonversi menjadi ID token yang sesuai menggunakan kamus yang telah di tetapkan. Selanjutnya, setiap ID token dikonversi menjadi vector dengan mengambil nilai *embeddings* dari*matrix embeddings* kata yang dilatih sebelumnya.
4. Segment *embeddings* : jika input terdiri dari dua kalimat, setiap token dalam input harus ditandai sebagai milik kalimat pertama atau kedua. Ini dilakukan dengan memberikan segmen ID 0 atau 1 ke setiap token, tergantung pada kalimat mana yang mengandung token tersebut.
5. Position *embeddings*: *BERT* menggunakan position *embeddings* untuk menambahkan informasi posisi absolut ke dalam representasi token. Ini dilakukan dengan menambahkan vector posisional yang telah ditentukan sebelumnya ke sitiap vector token.

Model *BERT* memiliki beberapa variasi berdasarkan ukuran *Hyperparameter* yang berbeda. Beberapa variasi model *BERT* yang umum dikenal berdasarkan ukuran *Hyperparameter* antara lain:

1. *BERT Base*: Ini adalah versi *BERT* yang paling umum dan terkenal. Model *BERT Base* memiliki sekitar 110 juta parameter. Biasanya, terdiri dari 12 lapisan (*layer*s) dari *encoder* dengan masing-masing lapisan berukuran 768 unit (*hidden size*).
2. *BERT Large*: Ini adalah versi yang lebih besar dari *BERT Base*. Model *BERT Large* memiliki sekitar 340 juta parameter. Biasanya, terdiri dari 24 lapisan dari *encoder* dengan masing-masing lapisan berukuran 1024 unit (*hidden size*).
3. *Multilingual BERT*: Versi ini adalah *BERT* yang dilatih pada beberapa bahasa. Model ini memiliki ukuran parameter yang mirip dengan *BERT-Base*, tetapi ia dilatih pada teks dari berbagai bahasa.

2.9 Multilingual BERT

Multilingual BERT adalah versi *BERT* tetapi dengan beberapa bahasa yang berbeda. Ini adalah salah satu model *BERT* terlatih. Namun, yang membedakan *BERT* multibahasa dengan model lainnya adalah model ini telah dilatih dalam 104 bahasa, termasuk bahasa Indonesia. Model ini sangat berguna untuk memecahkan masalah multibahasa.

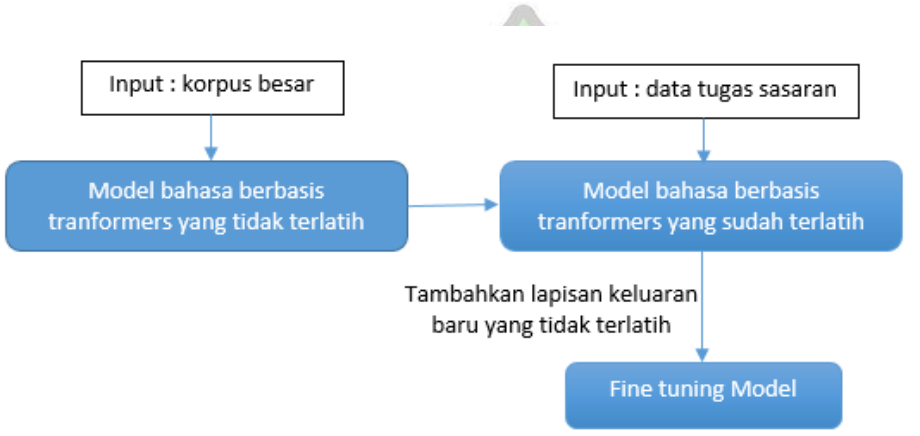
Ada dua model Multibahasa yang tersedia saat ini yaitu:

1. *BERT-Base*, *Multilingual Cased* : 104 bahasa, 12-lapisan, 768 tersembunyi, 12 head *layer*, parameter 110M.
2. *BERT-Base*, *Multilingual Uncased* : 102 bahasa, 12 lapisan, 768 tersembunyi, 12 head *layer*, parameter 110M (Devlin, 2019).

Pada penelitian ini, penulis memakai model *BERT Base, Multilingual uncased*, karena di lihat dari *dataset* yang penulis gunakan adalah bahasa Indonesia. Dan didalam bahasa Indonesia semua kata pada kalimat dimulai dengan huruf besar. Jadi model *cased* ini lebih relevan terhadap *dataset* yang penulis gunakan.

2.10 ***Pre-training* Model dan *Fine tunning***

Multilingual BERT adalah *pre-train model*, artinya model tersebut sudah dibuat dan dilatih. Namun, model tersebut tidak sepenuhnya akurat dan siap pakai untuk keperluan penelitian ini, sehingga dibutuhkan proses *finetune dan train model. Fine tuning* adalah proses memperbaharui semua parameter dalam model keperluan *task* yang baru. Sedangkan *train model* adalah proses belajar model dalam menentukan nilai *weights* dan *bias* yang terbaik dari label.



Gambar 2. 4 Tahapan *Pre-train* dan *Finetunning*

Model bahasa terlatih (*Pre-trained* Model) dikumpulkan dengan *dataset* yang besar. Model yang dihasilkan mampu memahami model bahasa yang diajarkan. Jika menggunakan model bahasa terlatih, dengan mudah menggunakan model terlatih data besar dan hanya melakukan proses *fine tunning* sesuai dengan kumpulan data yang diperlukan. Penyempurnaan dengan *BERT* dapat dilakukan hanya dengan menyesuaikan representasi input sebelum beralih ke model yang telah dilatih sebelumnya dan menambahkan lapisan output yang tidak terlatih yang telah dilatih ulang untuk tugas tertentu. *Finetunning* sangat mudah dilakukan karena maknisme *self-attention* pada *Transformers* membuat *BERT* bisa membuat model untuk berbagi tugas, baik pada kalimat tunggal (*Singgle sentence)* atau kalimat berpasangan, dengan menukar masukan da keluaran yang sesuai.

Pada tahap *finetunning*, *Hyperparameter* pada saat *pre-training* adalah sama, kecuali batch size,*learning rate*, dan jumlah *epoch*. Dropout probability selalu 0.1. Nilai *Hyperparameter* yang optimal untuk setiap task pada NLP selalu berbeda, namun model bekerja sangat baik pada saat *batch size* 16 dan 32,

learning rate (Adam) 5e-5, 3e-5, 2e-5, dan *epoch* sebanyak 2,3 dan 4 kali (Rahmatullah, 2021).

2.11 Tools

Dalam penelitian ini penulis menggunakan *tools* perangkat lunak berupa Bahasa pemograman *python dan Google Collab. Tools* tersebut yang akan penulis gunakan untuk melakukan proses klasifikasi teks menggunakan *Pre-trained* model *Multilingual BERT*.

2.11.1 Python

Python adalah bahasa pemrograman tingkat tinggi. *Python* adalah open source dan dirancang agar mudah dipahami dan diimplementasikan. Bahasa skrip *Python* dapat digunakan di *Mac, Windows*, *Linux*, dan sistem operasi lainnya. *Python* juga biasa digunakan dalam pengembangan web, pengembangan game, pemrograman numerik, aplikasi perkantoran, dan lainnya. *Python* yaitu memiliki manajemen memory otomatis dan dapat dipakai dalam berbagai jenis perangkat lunak. *Python* menyediakan dukungan untuk integrasi dengan pemograman yang lain. *Python* dikatakan sebagai bahasa pemograman yang memiliki sintak kode yang jelas.

Kekurangan pada *python* yaitu salah satunya programnya harus diproses terlebih dahulu sebelum dijalankan pada komputer. Namun juga terdapat banyak sekali kelebihan pada *pemograman* tingkat tinggi ini yaitu mudah dalam mempelajarinya, mudah ditulis, mudah dalam membaca, mudah dalam mencari kesalahan dan juga yang lain (Rena, 2019).

2.11.2 Google collaboratory

Google Collab adalah produk riset internal *Google*. Kolaborasi *Google* atau poran dari eBook Logika Maatematika untuk Analisis Algoritma oleh Dr. Putu Harry Gunawan, alat ini adalah kompiler dan editor online untuk bahasa pemrograman *Python*. *Python* digunakan dalam banyak cara, mulai dari web dan desktop hingga analisis data. Beberapa perusahaan telah mengadopsi bahasa *Python* karena kemudahan dan kesedderhanaannya. Salah satunya adaalah *Google*. *Google* saat ini menawarkan bahasa pemrograman *Python* melalui *Google Collab* atau *Google* Interactive Notebook. *Google Collab* juga menawarkan tiga jenis prosesor pembelajaran mesiin yang canggih: unit

pemrosesan pusat (CPU) , unit pemrosesan grafis (GPU), dan unit pemrosesan tensor (TPU). *Google collab* bisa dijadikan referensi bagi para data enthusiast untuk meningkatkan skill belajar *Python* mereka.

Tidak seperti produk *Google* standar lainnya seperti *Google Sheets, Google Drive, Google Docs,* dan lainnya, *Google collab* adalah produk berbasis *cloud*. Namun, kita dapat menggunakan *Google collab* secara gratis. *Google collab* dirancang khusus untuk developer atau peneliti yang kesulitan mengakses data dalam jumlah besar. Seperti yang Anda ketahui, *Google collab* adalah lingkungan pemrograman untuk bahasa pemrograman *Python* dalam format "notebook" (mirip dengan notebook *Jupyter*), atau dengan kata lain, seolah-olah *Google* meminjamkan komputer kepada kita secara gratis untuk membuat program atau data untuk memproses dari *Google*. Dalam hal ini, *Google collab* juga menawarkan lingkungan sumber yang sangat terbuka untuk mempelajari *Python* (Davita,2021).

2.12 Model Evaluasi

Model evaluasi yang digunakan oleh penulis yaitu *Confusion matrix*. Evaluasi merupakan kesimpulan akhir dari proses yang telah dilakukan atau output yang dikeluarkan dari proses panjang yang telah dilakukan, maka metode yang digunakan yaitu *Confusion matrix* untuk mengetahui keakuratan proses pengklasifikasian teks berita *hoax*.

Confusion matrix memiliki nilai atau persamaan untuk mendapatkan nilai keakuratan *system* tersebut. Maka proses dilakukan *Confusion matrix* sangat penting, sehingga *system* yang telah dibuat memiliki *system* yang baik untuk proses pengklasifikasian.

2.12.1 Confusion matrix

Confusion matrix biasanya digunakan untuk menghitung keakuratan pada sebuah objek deteksi. Adapun *Confusion matrix* ini terdapat 4 istilah sebagai hasil dari proses pendeteksian teks. Istilah tersebut yaitu *True positive (TP)* yaitu Jumlah data *positive* yang terklasifikasi benar, *False positive (FP)* yaitu Jumlah data *positive* yang terklasifikasi salah. *False negative (FN)* yaitu Jumlah data *negative* yang terklasifikasi salah. dan *True Nagative(TN)* yaitu Jumlah data *negative* yang terklasifikasi benar.

Keterangan :

- *True positive* (TP) yaitu dimana teks kalimat berita *hoax* telah berhasil oleh model *system*.
- *False positive* (FP) yaitu data bukan teks kalimat *hoax* namun diseteksi berita *hoax* oleh sistem.
- *False negative* (FN) yaitu data teks kalimat *hoax* namun diseteksi bukan teks *hoax* oleh *system*.
- *True negative* (TN) yaitu *system* tidak dapat mendeteksi teks kalimat *hoax*.

Kualitas klasifikasi dapat diperiksa menggunakan matriks konfusi (*Confusion matrix*). *Confusion matrix* membandingkan hasil klasifikasi dari pemrosesan sistem dengan hasil klasifikasi dari data sebenarnya dengan mengukur tingkat *accurasy*, *Precission* dan perolehan kembali. *Accurasy* adalah rasio prediksi yang benar terhadap total informasi yang diestimasi. Acuracy menghitung tingkat *accurasy* antara informasi yang diminta dengan jawaban atau hasil yang diberikan oleh sistem. Sementara itu, pengambilan digunakan untuk menghitung jumlah data yang diklasifikasikan dengan benar, data yang seharusnya termasuk dalam kategori tersebut(Rendragraha et al., 2021).

T

Confusion matrix	Prediksi	
	Negative	*Positive*
Negative	TN	FN
Positive	FP	TP

. 2 *Confusion matrix*

Menurut (Rahmatullah, 2021) *Confusion matrix* data ditentukan 3 nilai yaitu Accurasi, *Precission*, dan *recall*. Berikut Rumus untuk menghitung *Accurasy*, *Precission*, dan *recall* terhadap kelas *k* yaitu :

1. *Accuracy,* merupakan hasil perhitungan tingkat keakuratan deteksi objek terhadap objek *tajwid nun mati* secara keseluruhan. Adapun persamaan *accuracy* dapat dilihat dipersamaan 2.1.

- $Accurasy_k = \frac{TPk+TNk}{TPk+FPk+TNk+FNk} \times 100$ Persamaan (2.1)

2. *Precission,* merupakan persamaan mengenai jumlah prediksi yang benar dibandingkan dari keseluruhan hasil yang dapat di prediksi oleh *system*. Pada persamaan ini akan menghasilkan jumlah objek yang benar dari keseluruhan yang di deteksi oleh *system*. Adapun persamaan *precission* dapat dilihat dipersamaan 2.2

- $Precission_k = \frac{TPk}{TPk+FPk} \times 100$ Persamaan (2.2)

3. *Recall,* merupakan jumlah prediksi yang benar dibandingkan dengaan keseluruhan hasil pendenteksian tajwid sebenarnya. Persamaan ini akan menghasilkan jumlah objek deteksi dengan benar dari keseluruhan jumlah objek yang terdeteksi oleh *system.* Adapun persamaan *recall* dapat dilihat dipersamaan 2.3

- $Recall_k = \frac{TPk}{TPk+FNk} \times 100$ Persamaan (2.3)

4. *F1-Score*, selain *accurasy*, persisi, dan *recall*, pada tahap evaluasi juga dihitung *F1-Score* untuk mengetahui perfomasi model dengan cara mengkombinasikan nilai *Precission* dan *recall*. Nilai 1 menunjukkan bahwa model yang digunakan memilki performasi yang baik dalam melakukan prediksi, sedangkan nilai 0 menuunjukkan model memiliki performansi yang buruk. Pada binary classification, F1 Score dirumuskan sebagai berikut :

- $F1 - Score = \frac{2*Precission*recall}{Precission+recall} \times 100$ Persamaan (2.4)

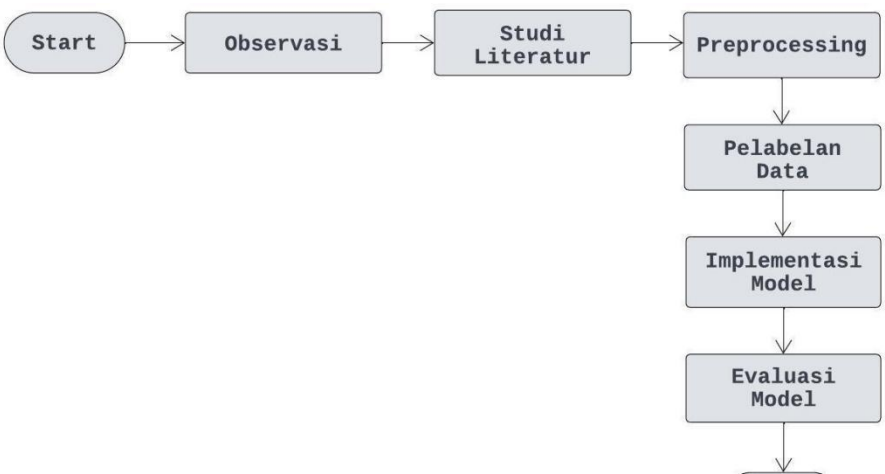
BAB III
METODOLOGI PENELITIAN

3.1 Jenis Penelitian

Penelitian ini merupakan jenis penelitian kuantitatif. Penelitian kuantitatif merupakan jenis penelitian yang didasarkan pada analisa dengan menggunakan metode numerik untuk memberikan jawaban atas pertanyaan penelitian. Tujuan penelitian kuantitatif yaitu untuk mengembangkan dan menggunakan model matematis atau teori-teori yang berkaitan dengan fenomena yang terjadi. Pemilihan jenis penelitian kuatitatif berdasarkan penyelesaian masalah dalam penelitian ini yang menggunakan pendekatan numerik dalam mengimplementasikan metode *Multilingual BERT* dan untuk mengklasifikasikan *Berita hoax/non hoax* dengan proses *text classification*.

3.2 Tahapan Penelitian

Pada penelitian ini, penulis membuat kerangka kerja yang berbentuk skema atau strategi untuk memudahkan penulis dalam melakukan penelitian sehingga skema tersebut penulis jadikan sebagai panduan dalam melakukan penelitian. Tahapan penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Tahapan penelitian

3.3 Metode Pengumpulan Data

3.3.1 Observasi

Proses observasi yang penulis lakukan adalah untuk mengidentifikasi masalah penelitian mengenai klasifikasi teks berita *hoax* yang diambil dari github yaitu: https://github.com/a3x-crypt/indo-covid19-news.git . Tahapan yang dilakukan adalah pencarian *dataset* berita *hoax* yang diambil dari github, kemudian men*downLoad*file dalam bentuk "csv". Teks berita *hoax* tersebut telah dikelompokkan kedalam dua kelas yaitu "*hoax*" dan "valid ". Data data tersebut nantinya akan di *pre-processing* terlebih dahulu untuk dijadikan data *training* dan data testing.

3.3.2 Studi Literatur

Studi pustaka merupakan salah satu metode yang penulis gunakan dalam proses pengumpulan data, yaitu dengan cara membaca, mengolah informasi, menulis catatan penting dari buku perpustakaan, meminjam bahan penunjang dan yang berkaitan dengan penelitian penulis. Referensi yang dikumpulkan oleh penulis adalah data dan informasi dalam berbagai website dan Jurnal terpercaya yang berkaitan dengan penelitian dan pengujian sistem. Penulis menggunakan referensi ini untuk menulis bab pengantar, landasan teori dan metode penelitian.

3.4 *Pre-processing* Data

Pada tahap ini, data yang dikumpulkan oleh penulis tidak lagi dilakukan pada tahap *pre-processing*, karena *dataset* yang digunakan pada penelitian ini sudah memiliki label sehingga dapat langsung digunakan pada model yang ingin dilatih. Namun, *pre-processing* diperlukan untuk data mentah yang akan diproses untuk memulihkan informasi di dalamnya karena banyak data kotor dan simbol yang

tidak perlu. *Pre-processing* data dilakukan dengan cara membuang data yang tidak perlu atau tidak sesuai agar data tersebut lebih mudah diolah, sehingga menghasilkan data yang sesuai dengan kebutuhan penelitian. Langkah-langkah yang diambil pada saat cleaning data seperti *lower casing*, penghapusan simbol, penghapusan slang, penghentian kata dan lain lain.

3.5 Pelabelan Data

Kemudian pada *dataset* tersebut sudah tercantum label pada setiap teks berita. Kelas kelas yang digunakan yaitu "0" untuk berita non *hoax* dan "1" untuk berita *hoax*. Pelabelan dilakukan dengan tujuan agar data teks berita dapat dimasukkan dalam proses *training* sehingga *system* dapat mengenali teks berita *hoax*.

3.6 Metode Simulasi

Penulis menggunakan metode simulasi dalam penelitian ini, Metode simulasi ini *BERT*ujuan untuk memberikan gambaran kinerja metode yang akan digunakan. Adapun tahapan pada metode simulasi penelitian ini teridiri dari dua tahapan utama yaitu perumusan konsep penelitian dan implementasi metode dan arsitektur. adapun tahap-tahap metode simulasi ini sebagai berikut:

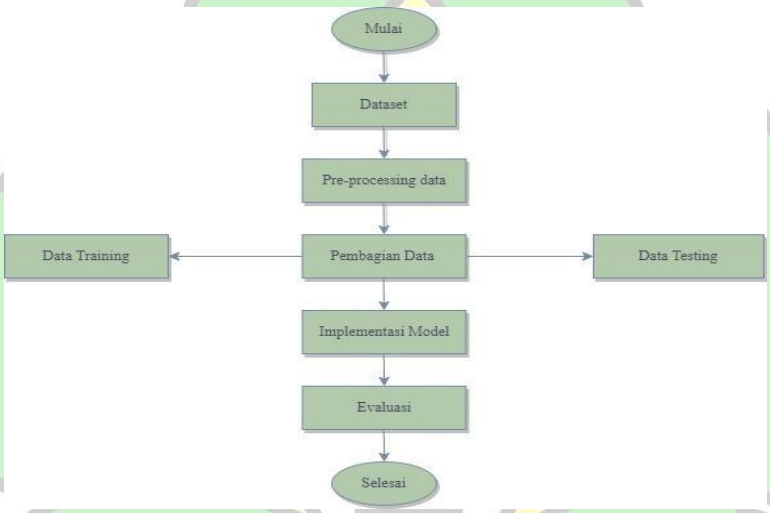
3.6.1 Menemukan Formulasi Permasalahan

Identifikasi masalah yang dilakukan oleh penulis bedaasarkan penelitian penelitian yang berhubungann dengan metode *Transformers* yang dilatih dengan *pre-train*ed model *Multilingual BERT* . Pada penelitian terdahulu belum pernah ada yang membahas pengujian sistem mengenai klasifikasi teks berita *hoax* dengan metode *Multilingual BERT*.

3.6.2 Konsep Penelitian

Penelitian ini menggunakan model *Multilingual BERT* untuk mengklasifikasi teks berita *hoax*. Pada proses pengujian sistem ini nantinya terdapat proses *training* pada teks, membuat model pengujian untuk mengenali yang mana berita *hoax* dan yang mana berita fakta untuk mendapatkan nilai *accurasy* tertinggi. Setelah itu hasil output yang didapat dari hasil pengujian *system* berupa teks yang telah di input dan dikenali oleh sistem.

3.7 Alur Implementasi Model



Gambar 3.2 Alur Implementasi Sistem

3.7.1 Dataset

Data yang digunakan pada penelitian ini merupakan data berita *hoax* mengenai kasus Covid 19 yang terjadi pada tahun 2019. Data ini diambil dari github milik public (https://github.com/a3x-crypt/indo-covid19-news.git). Kemudian data dari github tersebut *downLoad*terlebih dahulu dalam bentuk zip, selanjutnya diambil file didalamnya yang berbentuk "csv" agar dapat di input ke dalam sistem, sebagai alat analisis dan pustaka pemrosesan alami. Data yang sudah terkumpul sebanyak 1865 teks berita yang terdiri dari berita *hoax* dan *nonhoax*. Data berita yang akan di analisis merupakan teks berita berbahasa Indonesia. Proses pengolahan *dataset* ini menggunakan pemograman *python* dengab *library pandas.*

3.7.2 Pembagian Data

Sebelum tahap klasifikasi, data yang telah di labeli dibagi menjadi 3 yaitu data *train*, data test dan data validasi. Data *train* digunakan pengembangan model, data test digunakan untuk menguji dan melihat keakuratan model, sedangkan data validasi digunakan untuk memvalidasi kinerja model dan meminimalisir overfitting yang sering terjadi. *Scikit Learn* merupakan *library python* yang sering digunakan untuk split data. Data *training* merupakan *dataset* yang digunakan untuk menghasilkan model sistem dari metode *Multilingual BERT*. Proses sistem pada tahap ini adalah untuk mempersiapkan inputan data yang akan digunakan pada tahap pelatihan. Data *testing* adalah data yang digunakan untuk menghasilkan *value* evaluasi. Proses sistem pada tahap ini adalah untuk menginputkan data yang akan berbentuk matriks untuk menggambar hasil proses sistem.

Penulis menggunakan teknik *training* dan pengujian menentukan metode *Multilingual BERT*. Data teks berita dibagi menjadi data teks *train* dan data teks *test.* Untuk mendapatkan model yang ingin dibuat dibutuhkan suatu analisis dari *train*. Setelah data di *training* maka data *train* diuji coba ulang terlebih dahulu untuk melihat apakah *system* mampu mengklasifikasi teks berita. Data *test* berguna untuk mengetahui nilai keakuratan dari model yang telah di *training*.

Tabel 3. 1 Pembagian jumlah data *training* dan testing

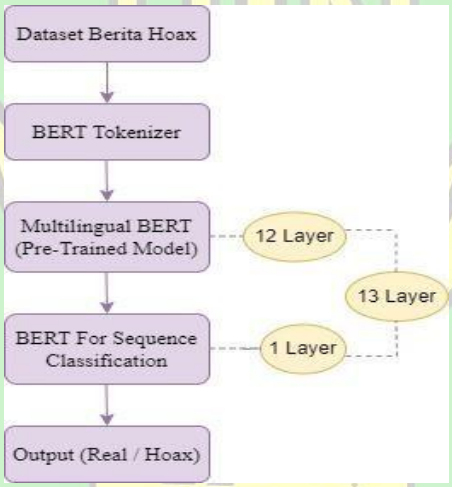
Label Data	Jumlah Data yang dipakai
Data *Training*	1426
Data Testing	187
Data Validation	252
Total *Dataset* Berita *Hoax*	**1865**

Dataset dibagi menjadi data *training* dan data *testing* dengan proporsi masing-masing, yaitu 80% data *training* dan 20% data *testing*dari keseluruhan data. Kemudian dari data *training* akan dibagi lagi menjadi data *training* itu sendiri dan data validasi dengan proporsi 70% dan 30% dari jumlah data *training*.

3.7.3 Implementasi Model dan Arsitektur

Implementasi dan model arsitektur merupakan tahapan untuk mentransformasikan rumusan konsep penelitian menjadi implementasi sistem yang dibangun oleh penulis. Setelah tahapan praproses data, *dataset* akan duubah menjadi input yang dapat diterima oleh *Multilingual BERT* yaitu dalam bentuk vector representasi kata menggunakan *Tokenizer*. Kemudian dilakukan proses fine-tuning dimana *pre-train* model *BERT* diadaptasi untuk melakukan klasifkasi berita *hoax*.

Untuk model klasifikasi, penulis menggunakan *BERT* For *Sequence classification*, yang merupakan *layer* klasifikasi di atas 12 *layer* M-*BERT*. Berikut ilustrasi dari model Arsitektur *Multilingual BERT*.

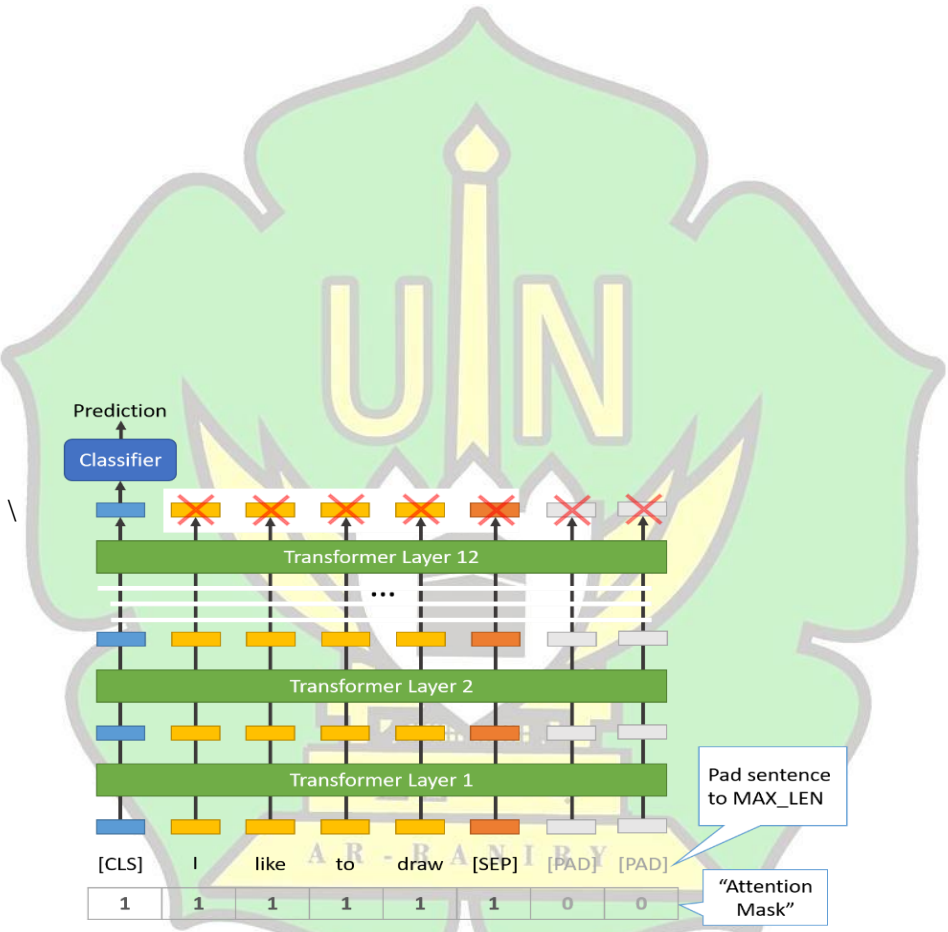


Gambar 3.3 Implementasi Model *BERT*

Berdasarkan Gambar 3.3 dapat disimpulkan bahwa arsitektur model M*BERT* terdiri dari 13 *layer*. 12 lapisan tersembunyi *mBERT* dan 1 lapisan pengklasifikasi di bagian *BERT For Sequence classification*. Setiap lapisan melakukan beberapa self attention pada kata yang disematkan dari lapisan sebelumnya.

Selanjutnya input yang telah disesuaikan kemudian diteruskan ke dalam jaringan *BERT* yaitu tumpukan 12 *layer Transformers Encoder* seperti yang tergambar pada gambar di atas *Encoder* disini *BERT*anggung jawab untuk mengambil input teks dan menghasilkan representasi vector melalui proses

embeddings. Tiap *layer encoder* memilki dua sub *layer* , yang pertama adalah *multi-head self-attention* mechanism, yang kedua *connected feed forward network*. Setelah melewati semua *encoder*, didapatkan vector output dari setiap token. Namun hanya vector output dari token [CLS]yang akan digunakan sebagai vector input untuk classifier. Dibawah ini adalah ilustrasi fnetuning *BERT* untuk klasifikasi.



Gambar 3. 4 Ilustrasi Fine-tuning *BERT* untuk Klasifikasi (McCormick, 2019)

Pada gambar diatas menunjukkan proses fine-tuning *Multilingual BERT* untuk klasifikasi dilakukan dengan menambahkan *layer* klasifikasi. *Library Transformers* memiliki kelas *BERTForSequenceClassification* yang didesain untuk tugas klasifikasi. Kelas *BERTForSequenceClassification* bekerja dengan cara memasukkan output dari pooler untuk menghitung logits. Nilai logits yang dihasilkan kemudian digunakan untuk mendapatkan nilai prediksi menggunakan softmax. Dalam konteks klasifikasi urutan, *BERT* dapat memahami

konteks global dari teks karena model ini dilatih secara *bi-directional* (memperhitungkan konteks sebelum dan sesudah kata). Hal ini memungkinkan *BERT* untuk mengatasi beberapa tantangan dalam tugas-tugas klasifikasi, seperti kata-kata yang memiliki makna bergantung pada konteks. (Rahmatullah, 2021). Dibawah ini beberapa source *Code* pada percobaan model *Multilingual BERT*.

1. *Dataset* berita *hoax*

```
import pandas as pd

df = pd.read_csv("/content/beritahoax.csv")
df.shape
.....
```

2. *BERT Tokenizer*

```
from transformers import BERTTokenizer

print("Loading BERT Tokenizer")
tokenizer = BERTTokenizer.from_pretrained('BERT-base-
Multilingual-uncased', do_lower_case=True)
.....
```

3. Implementasi model *BERT* (*pre-trai*ned)

```
model = BERTForSequenceClassification .from_pretrained(
    "BERT-base-Multilingual-uncased",


....
```

4. *BERT For Sequence classification*

```
model = BERTForSequenceClassification .from_pretrained(
    "BERT-base-Multilingual-uncased",
```

```
        num_labels = 2,
        output_attentions = False,
        output_hidden_states = False
)
model.cuda()
.....
```

5. Hasil Output atau *accurasy* yang didapatkan setelah proses *training* data

```
from sklearn.metrics import accuracy_score
acc = accuracy_score(flat_true_labels, flat_prediction)
print("ACC: %.3f" %acc)
.....
```

Tabel 3. 2 Algoritma implementasi model *BERT*

3.8 Metode Analisis Hasil

3.8.1 Evaluasi

Tahap akhir yang penulis lakukan yaitu analisa terhadap output berdasarkan asumsi yang telah dilakukan apakah sesuai dengan input. Lalu, tahap ini akan ditampilkan hasil output dengan input image. Evaluasi model *train loss* adalah mengevaluasi jumlah *loss* saat terjadinya proses *training* model dalam penelitian ini.

3.9 Alat Bantu Penelitian

Alat bantu pada penelitian ini adalah perangkat keras dan perangkat lunak komputer. Perangkat keras yang digunakan adalah satu unit Laptop *DELL intell* inside *core* i5 dengan spesifikasi sebagai berikut :

- Processor : Intel(R) Core(TM) i5-4310U CPU @2.00Hz (4CPUs), ~2,6GHz
- Operating *System* : Windows 10 Pro 64 bit (10.0, Build 19045)
- RAM 4 GB (2 GB available)

Perangkat lunak yang digunakan dalam penelitian ini adalah menggunakan beberapa *tools* seperti *Google* Collab untuk bahasa pemograman *python*, *tools* tersebut akan digunakan penulis dalam proses text klasifikasi berita *hoax* dengan metode *Multilingual BERT*.

Spesifikasi *google coolab* yang digunakan pada penelitian ini sebagai berikut:

- *Python* 3 type T4 *Google* Compute Engine backend (GPU)
- *System* RAM 1.3 / 12.7 GB

- Disk 24.4/ 78.2 GB

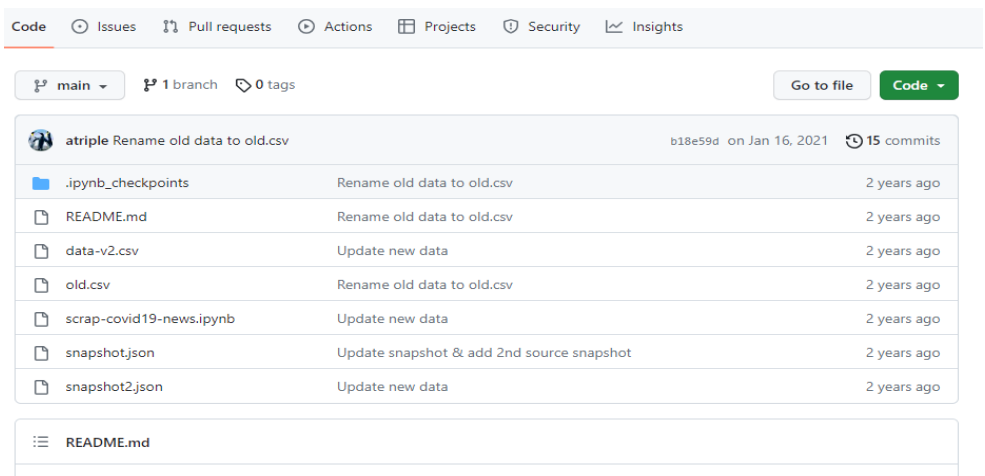
BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini, penulis akan menjelaskan proses klasifikasi teks berita *hoax* dan hasil dari pengklasifikasian berita *hoax* dengan model *Multilingual BERT* untuk mengetahui tingkat *accurasy*. Pada penelitian ini penulis akan menjelaskan hasil dari pengambilan data, hasil implementasi model serta menunjukkan hasil uji coba menggunakan metode *Multilingual BERT*. Lalu akan ditunjukkan pula evaluasi model menggunakan *confusion matrix*.

4.1 Data *Preparation*

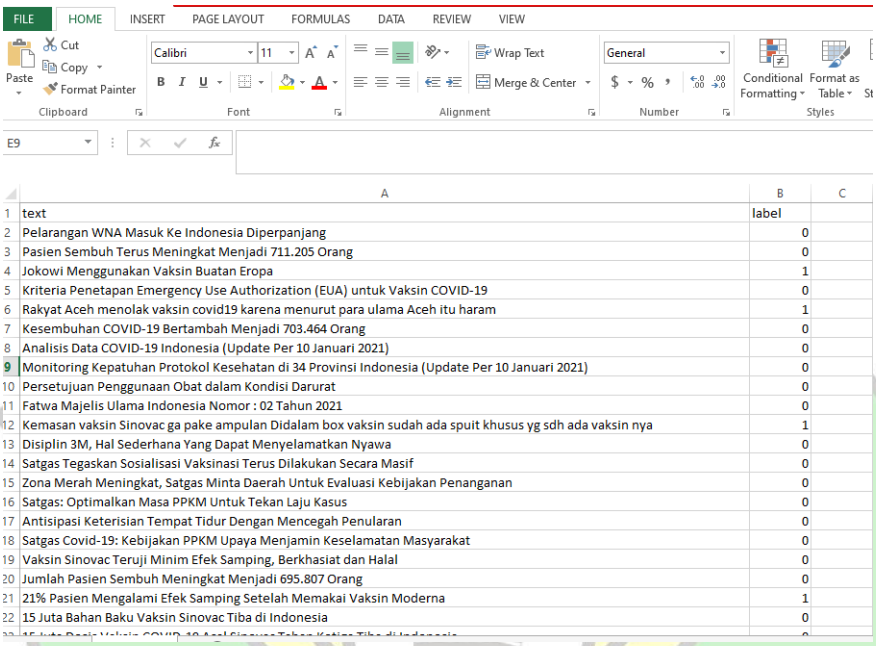
4.1.1 DownLoadDataset

Dataset yang dibutuhkan pada penelitian ini adalah kumpulan teks berbahasa Indonesia yang nantinya akan dipakai untuk melakukan *finetunning* dan *train* pada model *Multilingual BERT*. *Dataset* pada penelitian ini dapat di unduh di repository Github yaitu https://github.com/a3x-crypt/indo-covid19-news.git.



Gambar 4. 1 *DownLoadDataset*

Dataset ini memiliki tipe *file* csv (*comma-separated values). Dataset* yang diambil yaitu teks berita *hoax* dengan jumlah *dataset* sebanyak 1865 teks dan terdapat 2 label yaitu " *hoax*" dan "non *hoax*".



	A	B	C
1	text	label	
2	Pelarangan WNA Masuk Ke Indonesia Diperpanjang	0	
3	Pasien Sembuh Terus Meningkat Menjadi 711.205 Orang	0	
4	Jokowi Menggunakan Vaksin Buatan Eropa	1	
5	Kriteria Penetapan Emergency Use Authorization (EUA) untuk Vaksin COVID-19	0	
6	Rakyat Aceh menolak vaksin covid19 karena menurut para ulama Aceh itu haram	1	
7	Kesembuhan COVID-19 Bertambah Menjadi 703.464 Orang	0	
8	Analisis Data COVID-19 Indonesia (Update Per 10 Januari 2021)	0	
9	Monitoring Kepatuhan Protokol Kesehatan di 34 Provinsi Indonesia (Update Per 10 Januari 2021)	0	
10	Persetujuan Penggunaan Obat dalam Kondisi Darurat	0	
11	Fatwa Majelis Ulama Indonesia Nomor : 02 Tahun 2021	0	
12	Kemasan vaksin Sinovac ga pake ampulan Didalam box vaksin sudah ada spuit khusus yg sdh ada vaksin nya	1	
13	Disiplin 3M, Hal Sederhana Yang Dapat Menyelamatkan Nyawa	0	
14	Satgas Tegaskan Sosialisasi Vaksinasi Terus Dilakukan Secara Masif	0	
15	Zona Merah Meningkat, Satgas Minta Daerah Untuk Evaluasi Kebijakan Penanganan	0	
16	Satgas: Optimalkan Masa PPKM Untuk Tekan Laju Kasus	0	
17	Antisipasi Keterisian Tempat Tidur Dengan Mencegah Penularan	0	
18	Satgas Covid-19: Kebijakan PPKM Upaya Menjamin Keselamatan Masyarakat	0	
19	Vaksin Sinovac Teruji Minim Efek Samping, Berkhasiat dan Halal	0	
20	Jumlah Pasien Sembuh Meningkat Menjadi 695.807 Orang	0	
21	21% Pasien Mengalami Efek Samping Setelah Memakai Vaksin Moderna	1	
22	15 Juta Bahan Baku Vaksin Sinovac Tiba di Indonesia	0	

Gambar 4. 2 Sampel *Dataset* Teks Berita *Hoax*

Dapat dilihat pada gambar diatas terdapat label 0 dan label 1 dimana label 0 adalah teks berita non *hoax* dan label 1 adalah teks berita *hoax*. Jumlah data dengan label 0 yaitu 1800 sentence dan jumlah data dengan label 1 yaitu 65 sentence.

4.1.2 Tokenization

Langkah selanjutnya setelah mendapatkan *dataset* adalah melakukan tokenisasi. Proses tokenisasi adalah langkah penting dalam pemrosesan bahasa alami yang melibatkan pemisahan teks menjadi unit-unit yang lebih kecil, yang disebut token. Token bisa berupa kata, sub-kata (misalnya, suku kata), karakter, atau bahkan frase tergantung pada pendekatan tokenisasi yang digunakan.

Pada penelitian ini, *BERT Tokenizer* diambil dari repository model *Hugging face* dengan metode "*from_pretrained*". Kegunaan fungsi tersebut adalah untuk mengunduh *pree-trained tokenizer* milik *BERT-base Multilingual-*

uncased. Hal ini dilakukan karena *pre-trained model* seperti *BERT* hanya akan bekerja dengan baik jika diberi input token token yang sama seperti token token yang digunakan untuk tokenisasi *pre-trained model* tersebut. Akan tetapi, sebelum memulai proses tokenisasi dan proses proses lainya, pustaka pustaka (*library*) yang diperlukan untuk keperluan penelitian ini harus dimuat terlebih dahulu, seperti pustaka untuk mengatur *file path* (os, sys), membuat *neural network* (*torch*), mengunduh *pre-trained model* dan *tokenizer* (*transformers*), menghitung nilai *loss* dan *Loaddata* (*utils*), dan fungsi-fungsi tambahan untuk melatih model. Dan "*Uncased*" menunjukkan bahwa *tokenizer* tidak membedakan huruf kapital dan huruf kecil seperti yang ditunjukan pada gambar 4.3.

```
sentences = df.text.values

labels = df.label.values
from transformers import BERTTokenizer

print("Loading BERT Tokenizer")
tokenizer = BERTTokenizer.from_pretrained('BERT-base-Multilingual-
uncased', do_lower_case=True)
```

Gambar 4. 3 *BERT Tokenizer* from *Hugging face*

```
print("Original: ", sentences[560])

print("Tokenized: ", tokenizer.tokenize(sentences[560]))
print("Token IDS: ",
tokenizer.convert_tokens_to_ids(tokenizer.tokenize(sentences[560])
))
```

```
Original: Pelarangan WNA Masuk Ke Indonesia Diperpanjang
Tokenized:  ['pela', '##rangan', 'w', '##na', 'masuk', 'ke',
'indonesia', 'dip', '##er', '##pan', '##jang']
Token IDS:  [11977, 64747, 165, 10206, 33722, 11009, 11393, 80044,
10177, 14824, 51634]
```

Gambar 4. 4 Proses Tokenisasi

Gambar diatas merupakan proses tokenisasi dari sentence dengan nomor indeks 560. Kalimat nya adalah "Tips Keuangan: Dana Darurat Itu Penting", Teks telah dipisah menjadi unit-unit yang lebih kecil yang disebut token. Token bisa berupa kata-kata, karakter, atau bagian-bagian kata. kalimat tersebut memiliki 6 token atau kata kemudian dilakukan tokenisasi dengan membagi teks menjadi token token yang terdiri dari kata kata yang sesuai dengan *vocabulary BERT*.

Pada kata “darurat” dipisahakan menjadi sub kata „dar“, „##rurat“ karena kata pelarangan tidak ada dalam *vocabulary BERT* atau dictionary. Kemudian setiap token dalam input di konversi menjadi ID token yang sesuai menggunakan kamus token dimana angka-angka ini akan digunakan sebagai input oleh model pemrosesan bahasa untuk mengenal dan memproses teks yang telah ditetapkan. Selanjutnya pada tiap token dikodekkan sesuai dengan indeks *vocabulary*. Setelah ditambahkan token khusus yaitu “[CLS]” dan “[SEP]”, langkah selanjutnya adalah mengkodekan token-token tersebut menjadi indeks dalam *vocabulary*. Setiap token akan diwakili oleh indeks unik sesuai dengan posisinya dalam *vocabulary* model. *Vocabulary* biasanya berisi semua kata atau subkata yang mungkin muncul dalam data latih (Luo et al., 2022).

```
from tensorflow.keras.pre-processing.sequence import pad_sequences
MAX_LEN = 64
print("Padding/truncating all sentences to %d values" % MAX_LEN)
print('Padding token: "{:}", ID: {:}'.format(tokenizer.pad_token,
tokenizer.pad_token_id))

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

input_ids[0]
array([ 101, 11977, 64747, 165, 10206, 33722, 11009, 11393, 80044,
10177, 14824, 51634, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Gambar 4. 5 *Padding* Data

Setelah melakukan tokenisasi, pada gambar 4.5 menunjukkan bahwa *padding* data pada semua kalimat agar memiliki panjang yang sama sebesar “*MAX_LEN”* yang ditentukan, yaitu 64. Proses ini bertujuan untuk memastikan bahwa semua kalimat yang dimasukkan ke dalam model memiliki panjang yang sama dengan nilai *value* = 0 yang digunakan sebagai *padding* token ID.

4.2 Modelling

Dataset siap dipakai untuk tahap *modelling* setelah melalui proses *data preparation*. Dalam tahap modelling ini dijelaskan dari cara mempersiapkan model, dan melatih model.

4.2.1 *LoadModel*

Proses *Load*model dilakukan dengan menggunakan fungsi dari pustaka *transformers*, yaitu "from *transformers* import *BERTForSequenceClassification* , AdamW, *BERT*Config". *BERTForSequenceClassification* ini adalah kelas yang menggunakan arsitektur model *BERT* untuk tugas klasifikasi urutan (*sequence classification*), di mana pada penelitian ini mempunyai tugas untuk mengklasifikasikan input teks menjadi kategori *hoax / non hoax*. Berikut ini potongan *Code* untuk *Loadmodel.*

```
from transformers import BERTForSequenceClassification  , AdamW,
BERTConfig


model = BERTForSequenceClassification  .from_pretrained(
    "BERT-base-Multilingual-uncased",
    num_labels = 2,
    output_attentions = False,
    output_hidden_states = False)
```

Gambar 4. 6 *Load* model

4.2.2 *Fine Tuning & Train Model*

Model *pre-trained* seperti *BERT* tidak bisa langsung dipakai untuk mengklasifikasikan teks berita. Diperlukan proses *finetunning* dan *training* pada model supaya model mempunyai *accurasy* yang bagus sehingga bisa digunakan dan masuk ke tahap produksi. Berikut ini potongan *Code* tahapan *fine tuning* dan *traini model*

```
optimizer = AdamW(
    model.parameters(),
    lr = 2e-5,
    eps = 1e-8 )
```

Gambar 4. 7 *Code* untuk menggunakan *Hyperparameter*

Tahap pertama dalam *finetunning* yang di lakukan adalah memilih *optimizer* yaitu ADAM dan menentukan *learning rate* bernilai 0,00002. Dan juga *epsilon* yaitu 0,00000001. Setelah itu langkah selanjutnya adalah memberitahu model bahwa nantinya model akan di latih menggunakan GPU.

```
from transformers import get_linear_schedule_with_warmup
```

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

Gambar 4. 8 *Train* Model

Finetunning selanjutnya yang dilakukan adalah memilih jumlah *epoch*, dalam hal ini penulis memilih *epoch* sebanyak 10. Pada penelitian ini, penulis menggunakan model *BERT Base Multilingual Uncase* yang memiliki 12 *layer encoder*, 768 *hidden nodes*, dan 110 juta parameters. Berdasarkan *paper*, *hypermparameter* yang direkomendasikan untuk medapatkan performa yang optimal adalah menggunakan *batch size* 16 atau 32, *learning rate* sebesar 5e-5, 3e-5, 2e-5 dengan optimizer adam dengan *epoch* 5 dan 10. Tabel 4.1 menunjukkan *Hyperparameter* yang digunakan pada penelitian Tugas Akhir ini.

Table 4. 1 *Hyperparameter Multilingual BERT*

No.	***Hyperparameter***	**Ukuran**
1.	*Batch Size*	16 dan 32
2.	*Learning rate*	2e-5, 3e-5, 4e-5e-5
3.	*Epsilon*	1e-8
4.	*Epoch*	5 dan 10

Berikut adalah penjelasan dari parameter yang digunakan dalam penelitian ini:

1. Optimizer AdamW

   Pada penelitian ini saya menggunakan optimizer AdamW karena variasi dari algoritma optimasi Adam (Adaptive Moment Estimation) yang mengatasi masalah pembelajaran berlebihan pada model jaringan saraf. Dengan menggunakan optimizer AdamW dengan *learning rate* 2e-5 dan *epsilon* 1e-8, model *BERT* atau model Transformer tersebut siap untuk dilatih atau fine-tuned pada tugas tertentu seperti pemrosesan bahasa alami, klasifikasi teks,

atau tugas-tugas lain yang memanfaatkan representasi bahasa yang kuat dari model tersebut.

2. Batch Size

   Dari tabel di atas dapat diketahui bahwa *batch size* yang dipakai untuk *training* model *BERT* adalah 32. *Batch size* adalah ukuran jumlah sampel data yang diberikan ke model dalam satu iterasi (pembelajaran mini-batch) selama proses pelatihan (*training*) atau pengujian (inference) pada model machine learning, termasuk model bahasa seperti *BERT*.

3. *Learning rate*

   Kemudian pada penelitian ini saya memakai *learning rate* nya adalah 2e-5, 3e-5, 4e-5 dan 5e-5. "2e-5" mewakili 2 kali 10 pangkat -5, yang setara dengan 0,00002. Pemilihan *learning rate* yang optimal sangat tergantung pada arsitektur model, *dataset*, dan ukuran batch, dan seringkali memerlukan eksperimen untuk menemukan nilai yang paling sesuai.

4. *Epsilon*

   Epsilon digunakan untuk mencegah pembagian dengan nol saat menghitung gradient. Nilai *epsilon* ini biasanya sangat kecil untuk menghindari masalah numeric. Pada penelitian ini *epsilon* yang digunakan adalah 1e-8. *Epsilon* 1e-8 merupakan representasi dari bilangan desimal dalam notasi ilmiah (notasi *eksponensial*). Istilah ini biasanya digunakan dalam konteks komputasi numerik dan ilmu komputer terutama saat berbicara tentang *Precission* bilangan *floating-point*.

4.3 Hasil Analisis Terhadap Implementasi Model

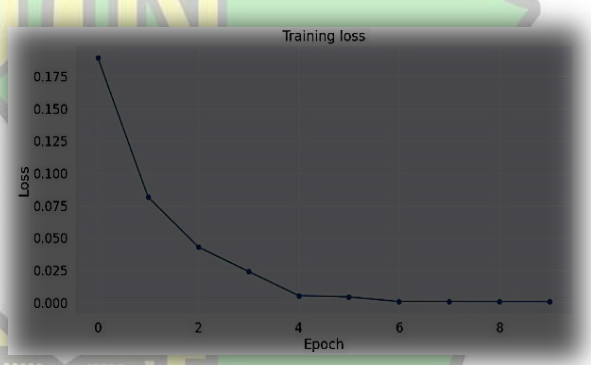
Bagian ini, penulis akan menjelaskan hasil dari beberapa percobaan yang dilakukan untuk membandingkan pengaruh nilai parameter *learning rate, batch size* dan *epoch* terhadap performa model *Multilingual BERT* dalam tugas klasifikasi teks berita *hoax*. Percobaan ini *BERT*ujuan untuk mengetahui kombinasi parameter terbaik yang dapat menghasilkan model dengan kinerja yang optimal.

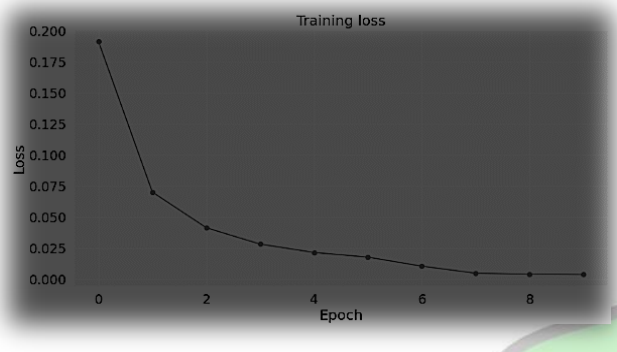
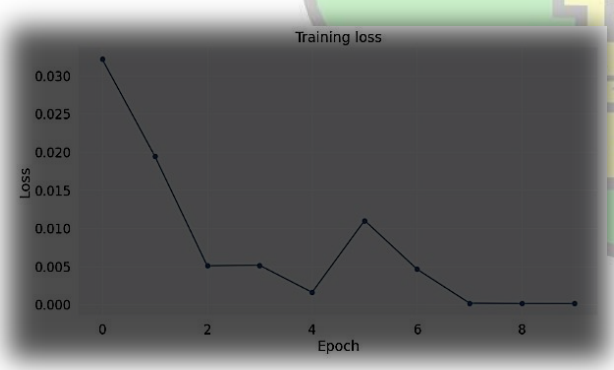
Berikut adalah penjelasan dari parameter yang digunakan untuk perbandingan pada penelitian ini.

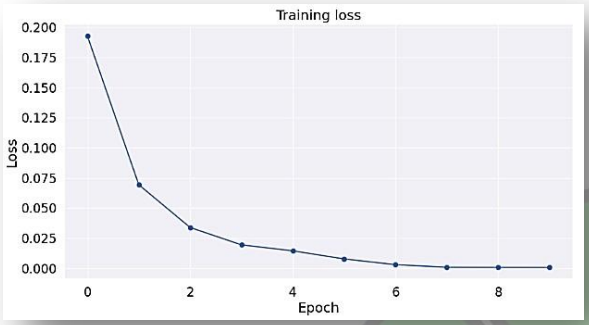
1. *Learning rate :* penulis akan membandingkan empat nilai *learning rate* berbeda yaitu : 2e-5, 3e-5, 4e-5 dan 5e-5
2. *Bach Size :* penulis akan membandingkan dua ukuran batch yang berbeda : 16, 32. Penggunaan *batch size* 16 dan 32 ditujukan untuk mendapatkan keseimbangan antara kecepatan pelatihan dan penggunaan memori.
3. *Epoch* : penulis akan membandingkan dua jumlah *epoch* yang berbeda : 5 dan 10.

Pada gambar dibawah ini menunjukkan grafik perbandingan *training loss* dari model *Multilingual BERT* menggunakan ***epoch* 10** dengan *learning rate* 2e-5, 3e-5, 4e-5, 5e-5 dengan *batch size* 32 dan 16.

Table 4. 2 Perbandingan Grafik *Training loss* pada *epoch* 10

1. Grafik *Training loss* dengan *learning rate* 2e-5 (*Batch size* 32)	Grafik *Training loss* denga learniing rate 2e-5 (*Batch size* 16)
	
Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,27 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafik turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 98%	Pada gambar diatas menunjukkan bahwa hasil dari *training loss* bearada pada kisaran 0,19 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafik turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 97%

2. Grafik *Training loss* denga learniing rate 3e-5 (*Batch size* 32)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,19 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 97%.	Grafik *Training loss* dengan *learning rate* 3e-5 (*Batch size* 16)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,30 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya tidak turun dengan stabil, pada *epoch* ke 2 dan 4 grafik naik berada pada kisaran nilai *loss* 0,01. Nilai *accurasy* yang didapatkan adalah 90%.
3. Grafik *Training loss* denga learniing rate 4e-5 (*Batch size* 32)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,31 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan pada *epoch* ke 4 grafik mengalami kenaikan yaitu	Grafik *Training loss* dengan *learning rate* 4e-5 (*Batch size* 16)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,26 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya turun dengan stabil. Nilai *accurasy* yang

berada pada kisaran nilai *loss* 0,11. Nilai *accurasy* yang didapatkan adalah 99%.	didapatkan adalah 90%.
4. Grafik *Training loss* denga *learning rate* 5e-5 (*Batch size* 32)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* memperoleh nilai *loss* sekitar 0,19 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan nilai *accurasy* terhadap data *training* semakin tinggi. Nilai *accurasy* yang didapatkan adalah 98%	Grafik *Training loss* denga *learning rate* 5e-5 (*Batch size* 16)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* bearada pada kisaran nilai *loss* sekitar 0,07 turun menjadi 0,00. dapat diketahui bahwa seiring berjalannya proses pelatihan nilai *accurasy* terhadap data *training* semakin tinggi. Nilai *accurasy* yang didapatkan adalah 91%

Dalam grafik *training loss* diatas dapat diketahui bahwa sumbu x biasanya mewakili iterasi atau *epoch* pelatihan, sementara sumbu y mewakili nilai *loss* pada iterasi atau *epoch* tertentu. Pada awal pelatihan, *loss* umumnya tinggi karena model belum mengetahui pola-pola yang ada dalam data dengan baik. Namun, seiring berjalannya waktu dan model mengenali pola-pola tersebut, *loss* secara *BERT*ahap akan menurun.

Bedasarkan dari tabel perbandingan *training loss* pada *epoch* 10 pada data pelatihan dapat disimpulkan bahwa *training loss* dengan hasil yang paling baik yaitu grafik *training losss* dengan *learning rate* 4e-5 dengan nilai *training loss* 0,30 turun menjadi 0.00 artinya semakin turun nilai *loss* nya maka semakin baik *accurasy* yang diperoleh. *Accurasy* yang di peroleh pada *learning rate* 4e-5

dengan *batch size* 32 adalah 99%. Berikut ini Tabel perbandingan *accurasy* dari hasil *training loss* dengan *Hyperparameter* yang berbeda.

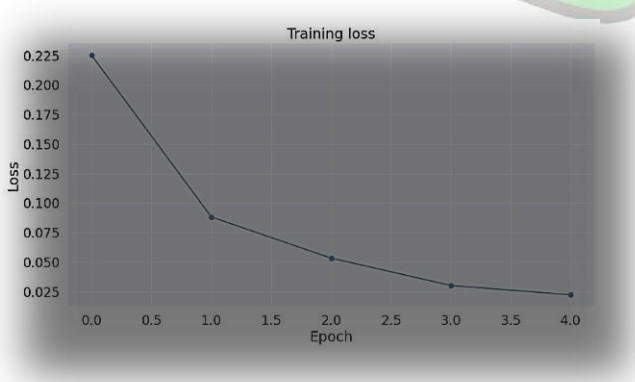
Table 4. 3 Hasil

No.	***Learning rate***	***Batch Size***	***Accurasy***	***Batch Size***	***Accurasy***
1	2e-5	32	98%	16	97%
2	3e-5	32	97%	16	90%
3	4e-5	32	99%	16	90%
4	5e-5	32	98%	16	91%

Perbandingan *Accurasy* Pada *Epoch* 10

Bedasarkan tabel perbandingan *accurasy* di atas dapat diketahui bahwa perubahan *batch size* dan *epoch* itu berpengaruh pada hasil *accurasy* yang didapatkan. Pada gambar dibawah ini menunjukkan grafik perbandingan *loss training* dari model menggunakan ***epoch* 5** dengan *learning rate* 2e-5, 3e-5, 4e-5, 5e-5 dengan *batch size* 32 dan 16.

Table 4. 4 Perbandingan Grafik *Training Loss* pada *epoch* 5

5. Grafik *Training loss* dengan *learning rate* 2e-5 (*Batch size* 32)	Grafik *Training loss* denga learniing rate 2e-5 (*Batch size* 16)
	
Pada gambar diatas menunjukkan bahwa hasil dari	Pada gambar diatas menunjukkan bahwa hasil

training loss berada pada kisaran 0,23 turun menjadi 0,02, dapat diketahui bahwa seiring berjalannya proses pelatihan grafik turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 97%	dari *training loss* bearada pada kisaran 0,25 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafik turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 97%
6. Grafik *Training loss* denga learniing rate 3e-5 (*Batch size* 32)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,03 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 98%.	Grafik *Training loss* dengan *learning rate* 3e-5 (*Batch size* 16)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,02 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya tidak turun dengan stabil, pada *epoch* ke 1 grafik naik berada pada kisaran nilai *loss* 0,03. Nilai *accurasy* yang didapatkan adalah 97%.
7. Grafik *Training loss* denga learniing rate 4e-5 (*Batch size* 32)	Grafik *Training loss* dengan *learning rate* 4e-5 (*Batch size* 16)

 Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,02 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan pada *epoch* ke 1 grafik mengalami kenaikan dengan nilai *loss* 0,11. Nilai *accurasy* yang didapatkan adalah 98%.	 Pada gambar diatas menunjukkan bahwa hasil dari *training loss* berada pada kisaran 0,05 turun menjadi 0,00, dapat diketahui bahwa seiring berjalannya proses pelatihan grafiknya turun dengan stabil. Nilai *accurasy* yang didapatkan adalah 97%.
8. Grafik *Training loss* denga *learning rate* 5e-5 (*Batch size* 32)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* memperoleh nilai *loss* sekitar 0,02 turun menjadi 0,01, dapat diketahui bahwa seiring berjalannya proses pelatihan nilai *accurasy* terhadap data *training* semakin tinggi. Nilai *accurasy* yang didapatkan adalah 98%	Grafik *Training loss* denga *learning rate* 5e-5 (*Batch size* 16)  Pada gambar diatas menunjukkan bahwa hasil dari *training loss* bearada pada kisaran nilai *loss* sekitar 2,00 turun menjadi 0,25. dapat diketahui bahwa seiring berjalannya proses pelatihan nilai *accurasy* terhadap data *training* semakin tinggi. Nilai *accurasy* yang didapatkan adalah 91%

Bedasarkan dari tabel perbandingan *training loss* pada *epoch* 5 pada data pelatihan dapat disimpulkan bahwa grafik *training loss* memilki perbedaan yang signifikan yang di uji dengan beberapa nilai parameter *learning rate , Batch size dan Epoch.* dengan hasil yang paling baik yaitu grafik *training losss* dengan *learning rate* 3e-5 dengan nilai *training loss* 0,03 turun menjadi 0.00 artinya semakin turun nilai *loss* nya maka semakin baik *accurasy* yang diperoleh. *Accurasy* yang di peroleh pada *learning rate* 3e-5 adalah 98%. Berikut ini Tabel perbandingan *accurasy* dari hasil *training loss* bedasarkan nilai *Hyperparameter*.

Tabel 4.5 Hasil perbandingan *Training loss* dengan *epoch* 5

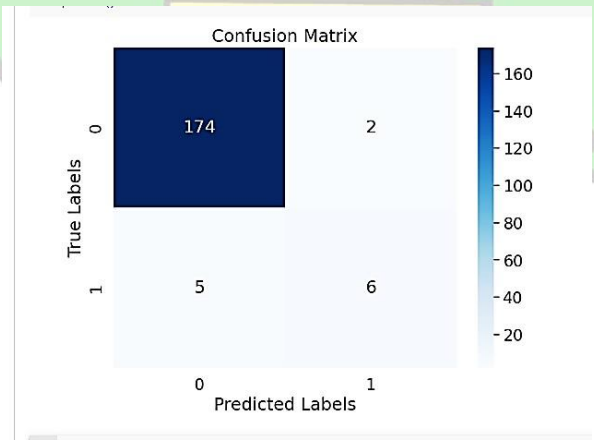
No.	***Learning rate***	**Batch Size**	***Accurasy***	**Batch Size**	***Accurasy***
1	2e-5	32	97%	16	97%
2	3e-5	32	98%	16	97%
3	4e-5	32	99%	16	97%
4	5e-5	32	98%	16	91%

Dari percobaan yang telah dilakukan dapat disimpulkan adalah nilai *accurasy* dari data pelatihan dapat di pengaruhi oleh nilai *loss* dan pola grafik naik turun. *Loss function* adalah metrik yang mengukur seberapa baik model dan cocok

dengan data pelatihan. Semakin rendah nilai *loss*, semakin baik model untuk menyesuaikan data. Namun grafik naik turun tidak selalu mengindikasikan performa yang buruk. Terkadang fluktuasi tersebut, bisa mencerminkan kompleksitas *dataset* atau pembelajaran yang lebih lambat. *Accurasy* adalah ukuran sejauh mana model berhasil mengklasifikasikan data dengan benar. Jika nilai *loss* rendah dan grafik *loss* menunjukkan tren penurunan, ini bisa mengindikasikan model yang semakin baik.

4.4 Hasil Uji Coba menggunakan *Confusion matrix*

Setelah dilakukan proses *pre-training* dan *Finetunning* model *Multilingual BERT,* maka selanjutnya dilakukan uji coba untuk mengetahui performa model terhadap data baru. Diagram *confusion matrix* dan performansi model pada hasil percobaan sebagai berikut.



Gambar 4. 9 Diagram *Confusion matrix* Model dengan *Learning rate* 2e-5 dan 10 *epoch*

Pada gambar diatas menunjukkkan bahwa ada 4 metrik dari percobaan *dataset* dari 187 *sentence*. Ada empat nilai penting yang didapatkan dari hasil perhitungan metrik-metrik klasifikasi bedasarkan *true* labels dan prediksi labels yaitu:

1. ***True positive* (TP)**: Ini adalah jumlah instance yang benar-benar termasuk dalam kelas *positive* dan juga diprediksi dengan benar oleh model sebagai kelas *positive*. Dalam kata lain, model berhasil mengidentifikasi instance *positive* dengan benar.
2. ***True negative* (TN)**: Ini adalah jumlah instance yang benar-benar termasuk dalam kelas *negative* dan juga diprediksi dengan benar oleh model sebagai kelas *negative*. Model berhasil mengidentifikasi instance *negative* dengan benar.
3. ***False positive* (FP)**: Ini adalah jumlah instance yang sebenarnya termasuk dalam kelas *negative*, tetapi diprediksi oleh model sebagai kelas *positive*. Kesalahan ini juga dikenal sebagai "Type I error" atau "*False* Alarm". Model salah memprediksi instance *negative* sebagai *positive*.
4. ***False negative* (FN)**: Ini adalah jumlah instance yang sebenarnya termasuk dalam kelas *positive*, tetapi diprediksi oleh model sebagai kelas *negative*. Kesalahan ini juga dikenal sebagai "Type II error" atau "Miss". Model gagal mengidentifikasi instance *positive*.

Dibawah ini adalah tabel hasil perhitungan *confusion matrix* bedasarkan warna terpekat dengan nilai TN yaitu 174, TP dengan nilai 6, FN dengan nilai 5, dan FP dengan nilai 2.

Table 4. 5 Hasil perhitungan *Confusion matrix*

No.	Jenis Perhitungan	Keterangan
	TN	174
1	TP	6
2	FN	5
3	FP	2
Total Data		**187**

Setelah dilakukan pengujian, maka diketahui bahwa 174 sampel benar di klasifikasikan sebagai *positive* (*True Negative*), 6 sampel benar di klasifikasikan sebagai *negative* (*True positive*), 5 sampel salah di klasifikasikan sebagai *positive* (*False Negative*) dan 2 sampel salah di klasifikasikan sebagai negative (*False positive*).

Bedasarkan diagram *confusion matrix* dapat dilihat bahwa model memilki performasi yang mirip dan cukup baik dalam mengklasifikasikan data. Rata rata yang *accurasy* yang didapatkan model dari tiap tiap percobaan sebesar 90%. Dari lima percobaan diketahui bahwa hanya dengan iterasi 10 *epoch* dan *learning rate* 2e-5 dengan *batch size* 32 sudah menunjukkan performasi yang cukup baik yaitu :

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


# Contoh confusion matrix
cm = [[ 174,  2],
      [ 5, 6]]
# Menampilkan hasil
print("Accuracy: {:.2f}".format(accuracy))
print("Precision: {:.2f}".format(precision))
print("Recall: {:.2f}".format(recall))
print("F1-Score: {:.2f}".format(f1))
Accuracy: 0.96
Precision: 0.75
Recall: 0.55
F1-Score: 0.63
```

Gambar 4. 10 Hasil perhitungan *confusion matrix*

Berdasakan hasil implementasi model terhadap data *testing*yaitu 187 sentence dengan perhitungan *confusion matrix* didapatkan hasil *accurasy* sebesar 96% dari model menggunakan batch size 32 dengan iterasi 10 *epoch*, Berikut perhitungan *Precission*, *recall* dan *F1-Score* pada *confusion matrix*.

Persamaan ***Accurasy*****k** = $Accuracy = \frac{Jumlah\ 0bjek\ Fang\ Terdeteksi\ Benar}{Jumlah\ Keseluruhan\ 0bjek\ Fang\ Terdeteksi}$ x 100

$$= \frac{174+6}{187} * 100$$

= 96 %

Persamaan *Precission*$_k$ = $Precision = \frac{TP}{(TP+FP)}$ x 100

$= \frac{6}{6+2} * 100$

= 75%

Persamaan *Recall*$_k$ = $Recall = \frac{TP}{(TP+FN)}$ x 100

$= \frac{6}{6+5} * 100$

= 55%

Persamaan *F1-Sore* = $F1\ Score = \frac{2\ x\ recall\ x\ precision}{recall+precision}$ x 100

$= \frac{2*75*55}{75+55} * 100$

= 63%

Tabel 4.6 Perhitungan *accurasy*, *Precission*,rekal dan *F1-Score*

No.	Jenis Pengukuran Model	Keterangan
1	*Accurasy*	96 %
2	*Precission*	75%
3	*Recall*	55%
4	*F1-Score*	63%

Pada tabel 4.6 Dengan empat nilai ini (TP, TN, FP, dan FN), kita dapat menghitung metrik evaluasi klasifikasi yang penting:

1. *Accurasy* :

   Dalam kasus ini, *accurasy* mencapai 96%, yang berarti sekitar 96% dari semua sampel telah diklasifikasikandengan benar oleh model, baik itu sebagai hasil *positive* maupun negative.

2. *Precission* :

Precission yang didapatkan sebesar 75% yang menunjukkan bahwa dari semua prediksi *positive* yang dibuat oleh model, hanya sekitar 75% yang benar benar relevan atau akurat.

3. *Recall* :

   Recall sebesar 55% mengindikasikan bahwa model hanya mampu mengidentifikasikan sekitar 55% dari semua kasus *positive* yang sebenarnya ada.

4. *F1-Score* :

   F1-Score sebesar 63% menunjukkan tingkat keseimbangan yang baik antara kemampuan model untuk menghasilkan hasil yang akurat (*Precission*) dan kemampuan model untuk mengidentifikasikan kasus *positive* yang ada (*recall*).

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pembahasan dan penelitian yang telah diuraikan pada bab sebelumnya, dapat disimpulkan bahwa:

- Proses dalam implementasi *deep learning* untuk klasifikasi teks berita *hoax* menggunakan metode *Multilingual BERT* berjalan dengan baik dan menghasilkan keberhasilan pengklasifikasian teks *berita hoax*.

- *Dataset* yang digunakan adalah sebanyak 1865 teks yang di ambil dari sebuah github milik *Andi Aqil Amanulhaq* yaitu: https://github.com/a3x-crypt/indo-covid19-news.git . Data tersebut sudah diberi label dengan 2 label yaitu 0 dan 1 yang berarti 0 untuk berita non *hoax* dan 1 untuk berita *hoax*. *Dataset* terbagi menjadi 3 yaitu data *training*, data *validation* dan data *testing*. Data *training* memiliki 1426 *teks*, data *validation* memiliki 252 teks dan data *testing*memiliki 187 *teks*. Arsitektur model *BERT* pada penelitian ini menghasilkan *accurasy* dari data *training* yaitu 97%, sedangkan dari data yang berlabel di ambil dari 187 data *testing*yang lebih spesifik dan di hitung menggunakan *confusion matrix* memperoleh *accurasy* 96%, *Precission* 75%, *recall* 55% dan *F1-Score* 63%.

5.2 Saran

Berdasarkan kesimpulan dari hasil dan pembahasan pada penelitian ini, maka penulis mengemukakan beberapa saran yang dapat digunakan untuk pengembangan lanjutan dari penelitian ini, sebagai berikut:

- Penulis mengharapkan saran perbaikan atau pengembangan oleh penulis seperti menambahkan jumlah *dataset* lebih dari 1865 teks, pada penelitian ini hanya menggunakan 1865 teks untuk *dataset* yang digunakan dalam proses klasifikasi teks berita *hoax*.
- Pada penelitian ini hanya memfokuskan pada satu jenis model saja yaitu *Multilingual BERT base uncased* saja, penulis berharap untuk penelitian selanjutnya bisa di coba dengan berbagai jenis model *BERT*, seperti *BERT large* dan lain lain.

DAFTAR PUSTAKA

Danu Nur Irwanto. (2021). *TUGAS AKHIR Identifikasi Berita Hoax Menggunakan Kombinasi Metode K-Nearest Neigbor (KNN) dan TF-IDF Berbasis Web Dengan Menggunakan Framework Codeigniter HALAMAN SAMPUL*. http://eprints.uwp.ac.id/id/eprint/3339/

Davita, A. W. (2021). *Belajar Python dengan Google collab*. https://www.dqlab.id/belajar-*python*-dengan-*google*-colab

Devlin, J. (2019). *Multilingual BERT*. https://github-com.translate.goog/*google*-research/*BERT*/blob/master/*Multilingual*.md?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=sc

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT*: *Pre-training* of deep bidirectional *transformers* for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*(Mlm), 4171–4186.

Dewi, A. O. P. (2020). Kecerdasan Buatan sebagai Konsep Baru pada Perpustakaan. *Anuva: Jurnal Kajian Budaya, Perpustakaan, Dan Informasi*, *4*(4), 453–460. https://doi.org/10.14710/anuva.4.4.453-460

Girinoto, G., Alwan, D. A., Gde K.T. D, G. A. N., Nabila, O. G., Arizal, A., & Priambodo, D. F. (2022). Implementasi Deteksi Judul Berita Clickbait Berbahasa Indonesia dengan *pre-train*ed model *Multilingual BERT* Pada Aplikasi Berbasis Chrome Extension. *Jurnal Ilmiah SINUS*, *20*(2), 25. https://doi.org/10.30646/sinus.v20i2.624

Hayati, R. (2018. (2018). Bab Ii Landasan Teori. *Journal of Chemical Information and Modeling*, 8–33.

Isa, S. M., Nico, G., & Permana, M. (2022). Indo*BERT* for Indonesian Fake News Detection. *ICIC Express Letters*, *16*(3), 289–297. https://doi.org/10.24507/icicel.16.03.289

Kurniawan, A. A., & Mustikasari, M. (2021). Implementasi *Deep learning* Menggunakan Metode CNN dan LSTM untuk Menentukan Berita Palsu dalam Bahasa Indonesia. *Jurnal Informatika Universitas Pamulang*, *5*(4), 544. https://doi.org/10.32493/informatika.v5i4.6760

Luo, Z., Xi, Y., Ma, J., Yang, Z., Mao, X., Fan, C., & Zhang, R. (2022). Dec*BERT*: Enhancing the Language Understanding of *BERT* with Causal Attention Masks. *Findings of the Association for Computational Linguistics: NAACL 2022 - Findings*, 1185–1197. https://doi.org/10.18653/v1/2022.findings-naacl.89

Martinez, A. R. (2010). Natural language processing. *Wiley Interdisciplinary Reviews: Computational Statistics*, *2*(3), 352–357. https://doi.org/10.1002/wics.76

McCormick, C. and N. R. (2019). *BERT Finetunning tutorial with pytorch*. https://mccormickml.com/2019/07/22/*BERT*-fine-tuning/

Mulyawan, R. (2023). *Transformers Neural Network*. https://rifqimulyawan.com/kamus/transformer-neural-network/.

Putri, C. A. (2020). Analisis Sentimen Review Film Berbahasa Inggris Dengan Pendekatan Bidirectional *Encoder* Representations from *Transformers*. *JATISI (Jurnal Teknik Informatika Dan Sistem Informasi)*, *6*(2), 181–193. https://doi.org/10.35957/jatisi.v6i2.206

Rahmatullah, B. (2021). *Sentiment Analysis Pelaksanaan Work From Home di Indonesia pada Masa Pandemi COVID-19 Menggunakan IndoBERT*.

Rahmawati, S. (2021). *DETEKSI BERITA HOAX PADA WEBSITE TURNBACKHOAX DENGAN MENGGUNAKAN MACHINE LEARNING UIN SYARIF HIDAYATULLAH JAKARTA 2021 M / 1442 M*.

Rena, P. N. (2019). *PeneraPan Metode Convolution Neural Network Pada Pendeteksian Gambar Notasi Balok*. Universitas Islam Negerii Syarif.

Rendragraha, A. D., Bijaksana, M. A., & Romadhony, A. (2021). Pendekatan Metode *Transformers* untuk Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia. *E-Proceeding of Engineering*, *8*(2), 3385–3395.

Reza, V., Snapp, P., Dalam, E., Di, I. M. A., Socialization, A., Cadger, O. F., To, M., Cadger, S., Programpadang, R., Hukum, F., Hatta, U. B. U. B., Sipil, F. T., Hatta, U. B. U. B., Danilo Gomes de Arruda, Bustamam, N., Suryani, S., Nasution, M. S., Prayitno, B., Rois, I., … Rezekiana, L. (2020). No 主観的健康感を中心とした在宅高齢者における健康関連指標に関する共分　　散構造分析Title. *Bussiness Law Binus*, *7*(2), 33–48. http://repository.radenintan.ac.id/11375/1/PERPUS PUSAT.pdf%0Ahttp://business-law.binus.ac.id/2015/10/08/pariwisata-

syariah/%0Ahttps://www.ptonline.com/articles/how-to-get-better-mfi-results%0Ahttps://journal.uir.ac.id/index.php/kiat/article/view/8839

Setiawan, R. (2021). *Mengenal Deep learning lebih jelas*. 9 Oktober 2021. https://www.dicoding.com/blog/mengenal-deep-learning/

Surat Al-hujarat ayat 6. (n.d.). https://tafsirweb.com/9776-surat-al-hujurat-ayat-6.html

Tandijaya, J. H., Liliana, & Sugiarto, I. (2021). Klasifikasi dalam Pembuatan Portal Berita Online dengan Menggunakan Metode *BERT*. *Jurnal Infra*, *9*(2), 320–325.

LAMPIRAN

Adapun lampiran lengkap hasil dari penelitian ini sebagai berikut :

Lampiran 1. Source *Code* Untuk Cek Resource pada *google* Collab

Code berikut merupakan program *python* untuk menginput *library* dan modul yang dibutuhkan untuk memeriksa ketersediaan GPU pada perangkat dan mengatur perangkat yang akan digunakan untuk komputasi oleh *library* PyTorch.

```
import torch

if torch.cuda.is_available():
```

```
  device = torch.device('cuda')

  print('there are %d GPU(s) available.' %
torch.cuda.device_count())

  print('we will use the GPU: ',
torch.cuda.get_device_name(0))

else:
  print("No GPU available, using the CPU instead")
  device = torch.device("cpu")
```

Lampiran 2. Source *Code* Untuk *LoadDataset*

Code berikut untuk membaca *dataset* berita *hoax* dari file CSV ke dalam sebuah DataFrame menggunakan *library* pandas, dan kemudian menampilkan lima baris pertama dari DataFrame tersebut.

```
import pandas as pd

# dataset berita hoax
dataset = '/content/beritahoax.csv'

# Membaca dataset ke dalam DataFrame
df = pd.read_csv("beritahoax.csv")

# Menampilkan dataset
print(df.head(5))
```

Code ini berfungsi untuk mengambil nilai dari dua kolom dalam DataFrame **df**, yaitu kolom **text** dan kolom **label**, dan menyimpannya ke dalam dua variabel terpisah: **sentences** dan **labels**.

```
sentences = df.text.values
labels = df.label.values
```

Code dibawaah ini untuk menggunakan *library* "*transformers*" dari *Hugging face* untuk mengunduh dan menginisialisasi *tokenizer* dari model *Multilingual BERT* (*BERT-base-Multilingual-uncased*) dalam bahasa yang tidak membedakan huruf besar dan kecil (do_lower_case=*True*).

```
from transformers import BERTTokenizer

print("Loading BERT Tokenizer")
tokenizer = BERTTokenizer.from_pretrained('BERT-base-
Multilingual-uncased', do_lower_case=True)
```

Code ini untuk mencetak hasil *tokenizer*

```
print("Original: ", sentences[0])

print("Tokenized: ", tokenizer.tokenize(sentences[0]))

print("Token IDS: ",
tokenizer.convert_tokens_to_ids(tokenizer.tokenize(sentences[0
])))
```

Code ini berfungsi untuk melakukan tokenisasi dan encoding teks dalam **sentences** menggunakan *tokenizer* yang telah diinisialisasi dengan model *Multilingual BERT*.

```
input_ids = []

for sent in sentences:
  enCoded_sent = tokenizer.enCode(
      sent,
      add_special_tokens = True
  )
  input_ids.append(enCoded_sent)

print("Original: ", sentences[0])
print("Token IDs: ", input_ids[0])
```

Code ini berfungsi untuk mengisi atau memotong panjang setiap baris dalam **input_ids** agar memiliki panjang yang sama, sehingga semua baris memiliki panjang yang seragam.

```
from tensorflow.keras.pre-processing.sequence import
pad_sequences


MAX_LEN = 64

print("Padding/truncating all sentences to %d values" %
MAX_LEN)
```

```
print('Padding token: "{:}", ID:
{:}'.format(tokenizer.pad_token, tokenizer.pad_token_id))

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

print("Done")
```

Code ini berfungsi untuk membuat attention mask dari **input_ids**. Attention mask adalah sebuah daftar (list) yang memiliki panjang yang sama dengan setiap baris dalam **input_ids**, dan mengindikasikan bagian mana dari baris yang merupakan token aktual dan bagian mana yang merupakan *padding*.

```
attention_mask = []

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

  attention_mask.append(att_mask)
```

Lampiran 3. *Source Code* untuk *Pembagian data training,testingdan validasi.*

Code ini berfungsi untuk membagi data menjadi empat set, yaitu data *training*, data validasi, data testing, dan attention mask yang sesuai untuk setiap set tersebut menggunakan *library* „sklearn.model_selection".

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_labels, test_labels =
train_test_split(input_ids,

        labels,

        random_state=2017,

        test_size=0.1)
train_mask, test_mask, _, _ = train_test_split(attention_mask,
                                               labels,
                                               random_state=20
17,
                                               test_size=0.1)

train_input, validation_input, train_labels, validation_labels
= train_test_split(train_input,

                   train_labels,
```

```
                    random_state=2018,

                    test_size=0.15)
train_mask, validation_mask, _, _ =
train_test_split(train_mask,
                                                    train_mas
k,
                                                    random_st
ate=2018,
                                                    test_size
=0.15)
```

Code ini berfungsi untuk mencetak bentuk (shape) dari setiap set data, yaitu data *training*, data validasi, dan data testing, serta bentuk dari attention mask yang sesuai dengan masing-masing set data. Dengan mencetak bentuk dari setiap set data, kita dapat melihat berapa banyak contoh (baris) dan berapa banyak fitur (kolom/token) dalam setiap set tersebut.

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

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

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

Lampiran 4 . *Source Code* untuk persiapan data model pre-traned *BERT*

Code ini berfungsi untuk menginisialisasi model *BERT* untuk tugas klasifikasi urutan (*sequence classification*) dengan menggunakan *library* "*transformers*" dari *Hugging face*.

```
from transformers import BERTForSequenceClassification  ,
AdamW, BERTConfig

model = BERTForSequenceClassification .from_pretrained(
    "BERT-base-Multilingual-uncased",
    num_labels = 3,
    output_attentions = False,
    output_hidden_states = False
)

model.cuda()
```

Code ini berfungsi untuk mencetak jumlah dan ukuran parameter-parameter (weights) dalam model *BERT* secara terperinci. Dengan melihat informasi ini, kita dapat memahami struktur model *BERT* dan mengeksplorasi ukuran setiap *layer* di dalamnya.

```
params = list(model.named_parameters())

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

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

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

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

Code ini berfungsi untuk menginisialisasi optimizer AdamW untuk melatih (*training*) model *BERT* dalam tugas klasifikasi urutan (*sequence classification*). Optimizer ini akan mengoptimalkan parameter-parameter dalam model *BERT* dengan memperbarui bobot-bobotnya berdasarkan gradien yang dihitung selama proses pelatihan.

```
optimizer = AdamW(
    model.parameters(),
    lr = 2e-5,
    eps = 1e-8
)
```

Code ini berfungsi untuk membuat scheduler laju pembelajaran (*learning rate*) dengan pemanasan (warm-up) linear untuk pelatihan model *BERT*.

```
from transformers import get_linear_schedule_with_warmup

epochs = 10

total_steps = len(train_dataloader) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps
= 0,
                                            num_training_step
s = total_steps)
```

Lampiran 5. *Source Code* untuk Fine tuning *BERT*

Code ini berfungsi untuk melatih (*training*) dan melakukan validasi pada model *BERT* dalam tugas klasifikasi urutan (*sequence classification*). Pelatihan dilakukan selama beberapa *epochs* (putaran) dengan menggunakan data *training*, dan setiap *epoch* akan diikuti oleh validasi pada data validasi.

```
import random

seed_val = 42

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

loss_values = []

for epoch_i in range(0, epochs):

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

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

  t0 = time.time()

  total_loss = 0
```

```
  model.train()

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

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

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

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

    model.zero_grad()

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

    loss = outputs[0]

    total_loss += loss.item()

    loss.backward()

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

    optimizer.step()

    scheduler.step()

  avg_train_loss = total_loss / len(train_dataloader)

  loss_values.append(avg_train_loss)

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

  # ==================================
  #              Validation
```

```
  # ========================================

  print("Running Validation...")

  t0 = time.time()

  model.eval()

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

  for batch in validation_dataloader:

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

    b_input_ids, b_input_mask, b_labels = batch

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

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

    tmp_eval_accuracy = flat_accuracy(logits, label_ids)

    eval_accuracy += tmp_eval_accuracy

    nb_eval_steps += 1

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

print("Training complete!")
```

Code ini berfungsi untuk membuat dan menampilkan grafik garis (line plot) dari nilai kerugian (*loss*) selama pelatihan model *BERT* pada setiap *epoch*. Grafik ini membantu dalam memvisualisasikan bagaimana nilai kerugian berubah seiring berjalannya waktu selama proses pelatihan

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
sns.set(style='darkgrid')
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (12,6)

plt.plot(loss_values, 'b-o')

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

plt.show()
```

Code ini berfungsi untuk melakukan prediksi label pada data uji (test data) menggunakan model *BERT* yang telah dilatih sebelumnya.

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

model.eval()

prediction, true_labels = [], []

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

  b_input_ids, b_input_mask, b_labels = batch

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

  logits = outputs[0]

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

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

print(" DONE.")
```

Code ini berfungsi untuk menghitung nilai koefisien korelasi Matthews (Matthews Correlation Coefficient, MCC) antara prediksi model dan label asli pada data uji. MCC adalah salah satu metrik evaluasi yang umum digunakan

dalam tugas klasifikasi biner untuk mengukur seberapa baik model klasifikasi dapat mengatasi kelas yang tidak seimbang.

```
from sklearn.metrics import matthews_corrcoef

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

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

mcc = matthews_corrcoef(flat_true_labels, flat_prediction)

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

Code ini berfungsi untuk menghitung dan mencetak nilai *accurasy* (accuracy) dari prediksi model pada data uji. *Accurasy* adalah salah satu metrik evaluasi yang paling umum digunakan dalam tugas klasifikasi untuk mengukur seberapa akurat model dapat memprediksi label yang benar.

```
from sklearn.metrics import accuracy_score

acc = accuracy_score(flat_true_labels, flat_prediction)

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

Lampiran 6. *Source Code* Untuk *Prediksi dan Evaluasi data*

Code ini berfungsi untuk mengonversi prediksi model dan label asli dari format tensor PyTorch menjadi array NumPy, kemudian melakukan dekode pada label ID ke label asli dengan menggunakan peta label (label_map), dan akhirnya mencetak label asli dan label yang diprediksi untuk setiap data uji.

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

# DeCode label IDs to original labels
label_map = {0: "Label_A", 0: "Label_B", 1:''} # Ubah sesuai
dengan label yang digunakan dalam dataset Anda

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

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

Code ini berfungsi untuk menghitung dan mencetak *confusion matrix* (matriks kebingungan) dari hasil prediksi model pada data validasi. *Confusion matrix* adalah tabel yang digunakan untuk menggambarkan performa model klasifikasi dengan membandingkan prediksi model dengan label asli dari data uji.

```
from sklearn.metrics import confusion_matrix

# Prediksi label pada data validasi
true_labels =
[0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0
,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0
,0,]
predicted_labels =
[0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0
,0]

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

# Tampilkan confusion matrix
print("Confusion matrix:")
print(cm)
```

Code ini berfungsi untuk menampilkan *confusion matrix* dalam bentuk heatmap menggunakan *library* **seaborn** dan **matplotlib.pyplot**. dan menampilkan *true* label dan prediksi label.

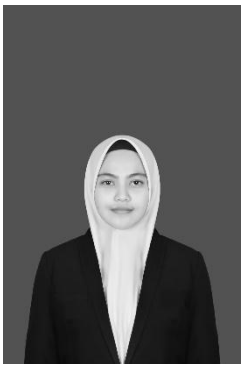
```
import seaborn as sns
import matplotlib.pyplot as plt

# Tampilkan confusion matrix dalam bentuk heatmap
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")

plt.title("Confusion matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")

plt.show()
```

Code ini berfungsi untuk menghitung dan mencetak beberapa metrik evaluasi klasifikasi, yaitu *accurasy* (accuracy), *Precission* (precision), *recall* (sensitivitas), dan *F1-Score*. Metrik-metrik ini memberikan informasi tentang performa model klasifikasi dengan lebih rinci daripada hanya menggunakan *confusion matrix* saja.

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

# Contoh confusion matrix
cm = [[ 174,  2],
      [ 5, 6]]

# Menghitung accurasy
accuracy = accuracy_score(true_labels, predicted_labels)

# Menghitung Precission
precision = precision_score(true_labels, predicted_labels)

# Menghitung recall
recall = recall_score(true_labels, predicted_labels)

# Menghitung F1-Score
f1 = f1_score(true_labels, predicted_labels)

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

DAFTAR RIWAYAT HIDUP



Izzia Khalkia, lahir di Beureunuen, Pidie pada tanggal 1 April 2001. Anak pertama dari lima bersaudara, dari pasangan Ibu Nurlaili dan Bapak Usman Daud. Penulis menyelesaikan pendidikan Sekolah Dasar di MIN 30 Beureunuen. Kemudian melanjutkan Pendidikan di jenjang Sekolah Menengah Pertama di MTsS Jeumala Amal Lueng Putu dan lulus tahun 2016. Penulis menempuh pedidikan jenjang Sekolah Menengah Atas di MAS Jeumala Amal Pidie Jaya dan lulus pada tahun 2019. Pada tahun yang sama penulis terdaftar sebagai mahasiswi pada program studi Teknologi Informasi, Fakultas Sains dan Teknologi, Universitas Islam Negeri Ar-Raniry, Banda Aceh melalui jalur seleksi PMB Lokal.