ANALISIS SENTIMEN BERBASIS ASPEK PEMAIN PUBGM BERDASARKAN REVIEW *APP STORE* DENGAN *BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER* (BERT)

SKRIPSI

Hanif Misbah Ananda

191401105

PROGRAM STUDI S-1 ILMU KOMPUTER

FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI

UNIVERSITAS SUMATERA UTARA

MEDAN

2023

ANALISIS SENTIMEN BERBASIS ASPEK PEMAIN PUBGM BERDASARKAN REVIEW *APP STORE* DENGAN *BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER* (BERT)

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana Komputer

Hanif Misbah Ananda
191401105

PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2023

iii

PERSETUJUAN

Judul	: ANALISIS SENTIMEN BERBASIS ASPEK PEMAIN PUBGM BERDASARKAN REVIEW APP STORE DENGAN BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER (BERT)
Kategori	: SKRIPSI
Nama	: HANIF MISBAH ANANDA
Nomor Induk Mahasiswa	191401105
Program Studi	: SARJANA(S1) ILMU KOMPUTER
Fakultas	: ILMU KOMPUTER DAN TEKNOLOGI INFORMASI UNIVERSITAS SUMATERA UTARA

Telah diuji dan dinyatakan lulus di Medan , 27 Oktober 2023

Dosen Pembimbing I

Handrizal S.Si., M.Comp.Sc

NIP. 197706132017061001

Dosen Pembimbing II

Fuzy Yustika Manik, S.Kom.,M.Kom

NIP. 198710152019032010

Diketahui/disetujui oleh
Ketua Program Studi S1
Ilmu Komputer,

Dr. Amalia ST., M.T.
NIP. 197812212014042001

PERNYATAAN

ANALISIS SENTIMEN BERBASIS ASPEK PEMAIN PUBGM BERDASARKAN REVIEW *APP STORE* DENGAN *BIDIRECTIONAL ENCODER REPRESENTATION FROM TRANSFORMER* (BERT)

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya

Medan, 5 Oktober 2023

Hanif Misbah Ananda
191401105

UCAPAN TERIMA KASIH

Dengan penuh bersyukur, penulis ingin mengungkapkan terima kasih kepada Allah SWT atas rahmat, karunia, dan kemudahan yang diberikan, yang memungkinkan penyelesaian laporan hasil penelitian ini sebagai salah satu persyaratan untuk meraih gelar Sarjana Ilmu Komputer di Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.

Selama proses penulisan laporan ini, penulis merasa sangat beruntung karena mendapatkan banyak bimbingan, arahan, dan bantuan yang berarti dari segala pihak. Maka dari itu, dengan tulus hati, penulis ingin menyampaikan ucapan terima kasih yang mendalam kepada:

1. Prof. Dr. Muryanto Amin S.Sos., M.Si. sebagai Rektor Universitas Sumatera Utara
2. Dr. Maya Silvi Lydia B.Sc., M.Sc. sebagai Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
3. Dr. Amalia S.T., M.T. sebagai Ketua Program Studi S- 1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Bapak Handrizal S.Si., M.Comp.Sc sebagai dosen pembimbing I yang telah membantu dan memberikan banyak masukan, kritik dan saran terhadap penulis sehingga penulis dapat menyelesaikan skripsi ini.
5. Ibu Fuzy Yustika Manik S.Kom., M.Kom sebagai dosen pembimbing II yang telah banyak memberikan ilmu dan masukan terhadap skripsi yang dikerjakan oleh penulis.
6. Seluruh tenaga pengajar dan pegawai di Fakultas Ilmu Komputer dan Teknologi Informasi USU yang telah membantu penulis dalam penulisan skripsi.

7. Teristimewa, rasa terimakasih yang tulus tak terhingga disampaikan kepada kedua orang tua tercinta, Papa Alfi Noviansyah dan Mama Nani Adriani , yang selalu menyayangi, mendoakan, dan mendukung penulis dalam tiap segi kehidupan yang penulis lalui.
8. Rasa terima kasih kepada saudara dan saudari saya tercinta Miftah Kamil Ananda dan Khalishah Laila Ananda.
9. Teman-teman yang telah membantu Febriando Manik dan M Amirul Ilmi.
10. Teman-teman sepermainan Simiy, Apiisu, Cimmu, Tanara, dan Boba yang sering bermain bersama penulis.
11. Semua orang yang telah memberikan banyak bantuan, baik secara langsung maupun tidak langsung, tanpa sebutan nama satu per satu.

Penulis dengan tulus meminta maaf atas segala potensi kesalahan yang mungkin terjadi selama proses penelitian dan penyusunan laporan ini. Saya berharap bahwa hasil penelitian ini akan menjadi kontribusi yang berarti dalam pengembangan ilmu yang memahami sentimen pengguna dalam konteks digital.

Medan, 5 Oktober 2023

Penulis

ABSTRAK

Players Unknown's Battlegrounds Mobile (PUBGM) merupakan salah satu game mobile yang sangat populer di kalangan gamer, di mana pemain berpartisipasi dalam pertempuran online untuk bertahan hidup hingga hanya satu pemain yang tersisa. Seiring dengan popularitasnya, banyak pemain PUBGM berbagi pengalaman mereka melalui ulasan di *App Store*. Oleh karena itu, analisis ulasan pemain di *App Store* sangat penting untuk memahami pandangan mereka terhadap game ini dan untuk meningkatkan pengalaman bermain. Penelitian ini menggunakan metode analisis sentimen dengan menggunakan model *Bidirectional Encoder Representations from Transformers* (BERT) berbasis aspek *User Experience* . Hasil penelitian ini mengindikasikan bahwa peningkatan jumlah *epoch* tidak selalu menghasilkan akurasi yang lebih tinggi. Akurasi analisis sentimen juga dipengaruhi oleh kualitas dan jumlah dataset yang digunakan. dan mencapai akurasi sebesar 84%, 82%, dan 83% dalam tiga percobaan dengan pengaturan *hyperparameters* yang berbeda. Pengujian terhadap jumlah 7 *epoch* menunjukkan bahwa *epoch* 3 memberikan hasil yang baik, dan oleh karena itu digunakan untuk menganalisis sentimen.

Kata Kunci: *BERT, Analisis Sentimen, PUBGM (Players Unknown's Battlegrounds Mobile), User Experience, App Store.*

ABSTRACT

Players Unknown's Battlegrounds Mobile (PUBGM) is one of the highly popular mobile games among gamers, where players engage in online battles to survive until only one player remains. With its popularity, many PUBGM players share their experiences through reviews on the App Store. Therefore, analyzing player reviews on the App Store is crucial to understand their perspectives on the game and enhance the gaming experience. This research employs sentiment analysis using the Bidirectional Encoder Representations from Transformers (BERT) model based on User Experience aspects. The findings of this study indicate that increasing the number of epochs does not always lead to higher accuracy. Sentiment analysis accuracy is also influenced by the quality and quantity of the dataset used, achieving accuracies of 84%, 82%, and 83% in three experiments with different hyperparameter settings. Testing with 7 epochs reveals that epoch 3 yields good results and is thus employed for sentiment analysis.

Keywords: BERT, Sentiment Analysis, PUBGM (Players Unknown's Battlegrounds Mobile), User Experience, App Store.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

1.1 Latar Belakang

PUBGM atau *Players Unknown's Battlegrounds Mobile* adalah salah satu game mobile yang populer di kalangan gamer. Dalam game ini, pemain harus bertahan hidup dalam pertempuran online dengan pemain lainnya hingga hanya satu orang yang bertahan hidup. Karena popularitasnya yang tinggi, banyak pemain PUBGM yang sering membicarakan pengalaman mereka bermain game pada *App Store.*

Dalam industri permainan seluler, PlayerUnknown's Battlegrounds Mobile (PUBGM) telah menjadi fenomena global dengan jutaan pemain yang aktif bermain di seluruh dunia. Seiring dengan popularitasnya yang terus meningkat, munculnya ulasan pengguna menjadi faktor penting dalam memahami pandangan dan pengalaman pemain terhadap permainan.

App Store adalah toko aplikasi digital yang dikelola oleh Apple *Inc.* Ini adalah platform distribusi resmi untuk aplikasi dan permainan yang dirancang khusus untuk perangkat Apple, seperti iPhone, iPad, dan Mac. Di Apple *App Store,* pengguna Apple dapat mencari, mengunduh, dan menginstal berbagai aplikasi yang tersedia, baik yang gratis maupun berbayar. Penting untuk dicatat bahwa Apple memiliki persyaratan ketat untuk aplikasi yang ingin masuk ke *App Store* mereka. Setiap aplikasi harus melewati proses pengujian dan persetujuan yang ketat untuk memastikan bahwa mereka aman, berfungsi dengan baik, dan mematuhi pedoman yang telah ditetapkan oleh Apple.

Salah satu platform yang penting untuk mengumpulkan ulasan pengguna adalah *App Store*, di mana pengguna dapat memberikan penilaian dan komentar terkait aplikasi yang sudah di unduh termasuk PUBGM . Ulasan ini menjadi sumber berharga untuk menganalisis sentimen dan opini pemain terhadap berbagai aspek permainan, seperti kualitas grafis, gameplay, dan pengalaman bermain secara keseluruhan.

Analisis sentimen adalah metode yang efektif untuk menggali pendapat dan sikap pengguna terhadap suatu entitas. Dalam hal ini, analisis sentimen berbasis aspek pemain PUBGM menggunakan metode *Bidirectional Encoder*

Representations from Transformers (BERT) digunakan untuk memahami sentimen yang terkandung dalam ulasan pemain PUBGM di *App Store*.

BERT adalah model pemrosesan bahasa alami (NLP) yang telah terbukti efektif dalam memahami konteks kata-kata dalam sebuah kalimat. BERT memungkinkan pemahaman yang lebih baik tentang sentimen yang terkandung dalam kalimat, termasuk nuansa dan hubungan antara kata-kata.

Dengan menerapkan BERT pada ulasan pemain PUBGM di *App Store*, analisis sentimen berbasis aspek pemain PUBGM dapat dilakukan dengan lebih akurat. Hasil analisis ini dapat memberikan wawasan berharga kepada pengembang permainan dan pemangku kepentingan lainnya tentang bagaimana pemain merespons berbagai aspek permainan, baik secara positif maupun negatif. Dengan demikian, analisis sentimen berbasis aspek pemain PUBGM berdasarkan ulasan di *App Store* dengan menggunakan metode BERT memiliki potensi untuk memberikan pemahaman yang mendalam tentang persepsi pemain terhadap permainan ini. Informasi ini dapat digunakan untuk memperbaiki dan meningkatkan PUBGM, memperbaiki pengalaman bermain, dan meningkatkan kepuasan pengguna secara keseluruhan.

Berdasarkan penelitian sebelumnya yang dilakukan oleh Wahit desta P, Fandli Supandi, Yuni Ambar S tentang "Analisis Sentimen Kepuasan Pemain PUBG Berdasarkan Komentar di Media Sosial" pada penelitian tersebut hasil evaluasi yang didapat menggunakan metode klasifikasi *Naïve Bayes* adalah nilai akurasi sebesar 41,7% pada test pertama menggunakan 1000 dataset, pada test kedua mendapat nilai akurasi sebesar 43,429% menggunakan 1500 dataset meningkat sebesar 2% dari percobaan pertama (Desta et al., 2023).

1.2 Rumusan Masalah

Ulasan para pemain PUBGM di *App Store* perlu dianalisis agar dapat mengetahui pandangan pemain terhadap game ini, Analisis sentimen juga diperlukan agar meningkatkan kualiatas dan pengalaman bermain. Berdasarkan hal tersebut maka diperlukan suatu sistem yang mampu menganalisis dan menspesifikasikan penilaian oleh pemain pada komentar *App Store* dengan menggunakan *BERT*.

1.3 Batasan Masalah

Dalam penelitian ini, peneliti memiliki batasan terhadap penyelesaian masalah.

Adapun batasan masalah dalam pelaksanaan penelitian ini adalah:

1. Model algoritma BERT yang digunakan adalah model BERT$_{BASE}$.
2. Data yang diklasifikasikan adalah komentar yang diambil dari proses *scraping* data dari web *App Store* sebanyak 3715 data
3. Ulasan pemain pada *App Store* hanya bahasa Indonesia dan Bahasa inggris.
4. Kelas untuk klasifikasi sentimen yaitu positif, negatif, dan netral.
5. Kelas untuk klasifikasi aspek berdasarkan aspek *user experience* seperti *Gameplay*, Grafis, Koneksi Jaringan, dan Performa.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk menganalisis sentimen yang dirasakan oleh pemain atau penggemar PUBGM dalam konteks pengalaman pengguna, dengan memanfaatkan BERT (*Bidirectional Encoder Representations from Transformers*) sebagai alat utama analisisnya.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini adalah:

1. Memperoleh informasi mengenai aspek-aspek dari *user experience* yang mempengaruhi sentimen pemain atau penggemar PUBGM, yang dapat digunakan untuk meningkatkan kualitas game dan membuat lingkungan yang nyaman dan aman untuk bermain.
2. Memperoleh akurasi yang dilakukan oleh algoritma BERT dalam bidang analisis sentimen berbasis aspek.
3. Sebagai rujukan penelitian kedepannya mengenai analisis sentimen berbasis aspek.

1.6 Metodologi Penelitian

Metodologi penelitian yang dilakukan dalam penelitian ini adalah:

1. Studi Pustaka

   Tahap awal penelitian melibatkan pengumpulan referensi dan sumber ilmiah seperti jurnal, artikel, buku, dan sumber informasi lainnya terkait

dengan topik penelitian, seperti analisis sentimen berbasis aspek atau algoritma BERT. Tujuan utamanya adalah untuk mendapatkan data dan informasi yang diperlukan untuk mendukung penelitian ini. Dengan referensi ini, penelitian dapat membangun dasar teoritis yang kuat dan merancang metodologi yang tepat .

2. Identifikasi Masalah

   Pada tahap ini, akan dilakukan pengidentifikasian dan analisis masalah dari pengumpulan berbagai sumber literatur sehingga memperoleh pemahaman untuk menemukan metode yang paling tepat yang kemudian akan diterapkan dalam pemecahan masalah terhadap penelitian ini.

3. Analisis dan Perancangan Sistem

   Pada tahap ini, dilakukan penyelesaian terhadap permasalahan yang telah dianalisis melalui studi literatur. Proses ini mencakup perancangan desain arsitektur umum, pengumpulan data, serta pengembangan antarmuka sistem.

4. Implementasi Sistem

   Tahap ini dilakukan pembuatan sistem yang menggunakan pemrograman Bahasa *Python*.

5. Pengujian

   Pada tahap ini, saya melakukan pengujian sistem dan percobaan sesuai dengan kebutuhan yang telah saya tetapkan sebelumnya. Tujuan utama adalah memastikan bahwa program yang telah saya buat beroperasi sesuai dengan harapan yang saya tetapkan.

6. Analisis Hasil

   Pada tahap ini, dilakukan analisis terhadap hasil penelitian, yang mencakup evaluasi mendalam terhadap temuan-temuan yang telah ditemukan dalam penelitian ini.

7. Dokumentasi

   Pada tahap ini, peneliti akan mengawali proses dokumentasi serta penulisan laporan mengenai hasil penelitian terkait aplikasi yang telah berhasil dikembangkan, sesuai dengan format yang telah ditetapkan untuk skripsi ini.

1.7 Sistematika Penyusunan

Sistematika penulisan dari skripsi ini terdiri dari lima bagian utama untuk diketahui yaitu sebagai berikut:

BAB 1: Pendahuluan

Bab ini merupakan tahap awal dari penelitian ini, yang akan memberikan gambaran lengkap tentang konteks penelitian. Dalam bab ini, akan dijelaskan latar belakang penelitian, permasalahan yang menjadi fokus, tujuan penelitian, rumusan masalah, manfaat penelitian, batasan-batasan yang relevan, dan tata cara penyusunan laporan. Bab ini akan membantu pembaca untuk memahami esensi penelitian serta alasan mengapa penelitian ini penting dilakukan.

BAB 2: Landasan Teori

Bab landasan teori adalah tempat di mana kita menggali dan merangkum literatur terkait yang relevan dengan topik penelitian. Dalam bab ini, peneliti melakukan pencarian dan analisis terhadap berbagai sumber informasi seperti jurnal ilmiah, buku, makalah, atau artikel yang berkaitan dengan topik penelitian. Tujuan dari bab ini adalah untuk memperoleh pemahaman yang mendalam tentang topik penelitian, mengidentifikasi kekosongan pengetahuan, dan membenarkan pentingnya penelitian. Bab ini akan menjelaskan berbagai teori yang relevan dengan isu yang menjadi fokus penelitian, termasuk Aplikasi PUBG *Mobile*, *Natural Language Processing*, Analisis Sentimen, *Machine Learning*, *Neural Network*, *Deep Learning*, serta BERT (*Bidirectional Encoder Representations from Transformers*).

BAB 3: Analisis dan Perancangan

Bab ini mencakup penilaian sentimen terhadap Aplikasi PUBG Mobile yang terdapat di *AppStore* dengan menggunakan model Bahasa BERT. Tujuan dari analisis ini adalah untuk mengkategorikan sentimen menjadi tiga kelompok utama: positif, negatif, dan netral. Selain itu, bab ini juga melibatkan analisis aspek *User Experience*. Bab ini juga mencakup analisis sistem yang mencakup identifikasi masalah serta analisis kebutuhan dalam merancang arsitektur umum dan perancangan sistem.

BAB 4: Implementasi dan Pengujian

Bab Implementasi dan Pengujian adalah bagian penting dalam sebuah penelitian yang fokus pada penerapan dan evaluasi praktis dari sistem atau solusi yang telah dirancang. Dalam bab ini, peneliti menjelaskan langkah-langkah yang diambil untuk menerapkan solusi yang telah dirancang ke dalam lingkungan nyata atau simulasi.

BAB 5: Kesimpulan dan Saran

Bab Kesimpulan dan Saran merangkum temuan penelitian, menjawab pertanyaan penelitian, dan mengemukakan kesimpulan yang didapat dari hasil penelitian.

BAB 2
LANDASAN TEORI

2.1 PUBG Mobile

PUBG Mobile, singkatan dari "PlayerUnknown's Battlegrounds Mobile," adalah salah satu permainan video populer yang telah mengambil dunia game dengan badai. Permainan ini menghadirkan pengalaman pertempuran royale yang intens, di mana pemain dari seluruh dunia bersaing untuk menjadi yang terakhir bertahan dalam pertempuran. PUBG Mobile dikembangkan oleh PUBG Corporation dan Tencent Games, dan menjadi salah satu judul yang paling banyak diunduh di platform permainan seluler. Permainan ini menawarkan grafis yang memukau, taktik permainan yang mendalam, dan mode permainan yang beragam, semuanya menghadirkan tantangan yang menarik bagi pemainnya. Bagi para penggemar game mobile, PUBG Mobile telah menjadi salah satu pilihan utama untuk menguji keterampilan bertahan dan bermain strategis dalam pertempuran royale yang mendebarkan. Dengan komunitas yang besar dan turnamen e-sport yang sering diadakan, PUBG Mobile terus menjadi salah satu game seluler yang paling diminati di seluruh dunia. Mari kita lihat lebih lanjut tentang bagaimana analisis sentimen dapat diterapkan pada pengalaman pemain PUBG Mobile.

PUBG Mobile adalah permainan battle royale yang populer yang dapat dimainkan di perangkat seluler. Dalam permainan ini, pemain akan dijatuhkan ke pulau terpencil dengan 99 pemain lainnya. Mereka harus mencari senjata, perlengkapan, dan bertahan hidup sambil memerangi pemain lain. Peta permainan akan terus menyusut, memaksa pemain untuk saling berhadapan dalam wilayah yang semakin sempit. Tujuan utama adalah menjadi pemain terakhir yang bertahan hidup. Permainan ini menawarkan grafis yang mengesankan, taktik yang mendalam, dan gameplay yang seru.

2.2 Natural Language Processing (NLP)

Natural Language Processing adalah suatu bidang dari ilmu komputer dan kecerdasan buatan yang berfokus pada interaksi antara manusia dengan bahasa

alami komputer. Tujuannya adalah untuk membuat komputer dapat memahami, menganalisis, dan memproduksi teks atau ucapan dalam bahasa manusia (Jurafsky dan Martin, 2019. Secara rinci, NLP melibatkan serangkaian tugas dan teknik yang meliputi:

1. Tokenisasi: Memisahkan teks menjadi unit-unit yang lebih kecil, seperti kata atau frasa.
2. Pemisahan Kata: Memisahkan kata-kata dalam kalimat agar dapat dipahami secara individu.
3. Parsing: Menganalisis struktur kalimat untuk mengidentifikasi hubungan gramatikal antara kata-kata.
4. Pengenalan Entitas Nama: Mengidentifikasi dan mengklasifikasikan entitas bernama seperti orang, tempat, atau organisasi dalam teks.
5. Pemberian Kategori: Mengklasifikasikan teks ke dalam kategori tertentu, seperti topik atau sentimen.
6. Analisis Sentimen: Menentukan sikap atau pendapat dari teks, apakah itu positif, negatif, atau netral.
7. Pemahaman Bahasa Alami: Memahami arti dari teks yang lebih kompleks, termasuk pemahaman semantik, sintaksis, dan pragmatik.
8. Penerjemahan Mesin: Menerjemahkan teks dari satu bahasa ke bahasa lain menggunakan model statistik atau berbasis aturan.
9. Ringkasan Teks: Menghasilkan ringkasan yang singkat dari teks yang lebih panjang.
10. Generasi Bahasa Alami: Menghasilkan teks yang baru berdasarkan input atau kondisi tertentu.
11. Analisis Wacana: Menganalisis hubungan dan struktur teks dalam konteks wacana yang lebih luas.

Untuk mencapai tujuan-tujuan tersebut, NLP menggunakan berbagai pendekatan dan teknik seperti pemrosesan statistik, pembelajaran mesin, pemrosesan bahasa alami berbasis aturan, jaringan saraf, dan model bahasa. Selain itu, NLP juga bergantung pada sumber daya seperti kamus, korpus teks, dan alat-alat linguistik seperti lema, stopwords, dan stemmer.

Penerapan NLP sangat luas dan dapat ditemukan di berbagai bidang, termasuk mesin pencari, chatbot, analisis sentimen media sosial, sistem otomatisasi penjawab pertanyaan, deteksi spam, pemrosesan bahasa alami pada asisten virtual, dan banyak lagi. Dengan kemajuan teknologi dan peningkatan pemahaman tentang bahasa manusia, NLP terus berkembang dan memiliki peran penting dalam pengolahan informasi dan interaksi manusia dengan komputer.
tersebut.

2.3 Analisis Sentimen

Analisis Analisis sentimen adalah studi komputasi dari opini-opini, sentimen, serta emosi yang diekspresikan dalam teks (Liu, 2012) Penggunaan analisis sentimen telah diterapkan dalam mencari informasi dan merespon permintaan dari konsumen, sementara bagi pemilik bisnis, alat ini berguna dalam pengambilan keputusan operasional seperti strategi branding dan langkah-langkah pencegahan atau perbaikan. Terdapat beberapa ciri khas dari analisis sentimen, meliputi:

1. Analisis Subjektivitas: Selain menilai sentimen, analisis ini juga mencoba untuk mengidentifikasi sejauh mana teks bersifat subjektif atau objektif. Subjektivitas dapat mengandung opini atau pandangan pribadi.
2. Kombinasi dengan Teknologi Lain: Analisis sentimen sering digabungkan dengan teknologi lain seperti Natural Language Processing (NLP) untuk meningkatkan akurasi dan pemahaman konteks.

Analisis sentimen dapat diimplementasikan ke dalam 3 level (Liu, 2012), yaitu, terdapat beberapa level yang digunakan untuk mengklasifikasikan teks berdasarkan sentimen yang terkandung di dalamnya. Level pertama adalah level dokumen, di mana seluruh dokumen pendapat dikelompokkan menjadi sentimen positif atau negatif secara keseluruhan. Level kedua adalah level kalimat, di mana setiap kalimat dalam teks diidentifikasi sebagai opini positif, negatif, atau netral. Namun, jika pada level dokumen dan kalimat analisis tidak dapat menemukan preferensi yang sesungguhnya, maka digunakan level entitas dan aspek. Pada level ini, analisis akan lebih mendetail dengan mengidentifikasi dan mengklasifikasikan aspek-aspek tertentu dari entitas yang menjadi fokus opini, sehingga memberikan analisis yang lebih halus dan lebih terperinci.

2.4 Machine Learning

Machine learning merupakan cabang ilmu dalam bidang Kecerdasan Buatan. Tujuan dari *machine learning* adalah untuk melatih mesin dengan contoh atau *dataset* yang berkaitan dengan tugas yang harus dilakukan. Mesin akan mempelajari pola-pola dari *dataset* tersebut dan menghasilkan aturan sendiri. Dengan demikian, mesin dapat mengenali data yang dimasukkan ke dalamnya dengan baik. Secara rinci, Machine Learning melibatkan beberapa konsep dan komponen utama, termasuk:

1. Data Training adalah tahap di mana model Machine Learning dijalani melalui proses pembelajaran menggunakan data yang telah diberi label. Data training ini berperan sebagai "materi ajar" bagi model, memungkinkannya untuk memahami pola-pola dan menggabungkan pengetahuan dari data tersebut ke dalam pemahaman yang lebih umum.
2. Algoritma: Algoritma Machine Learning adalah metode matematis dan statistik yang digunakan untuk menggali informasi dari data. Ini termasuk metode seperti regresi, pohon keputusan, naive bayes, support vector machines, dan neural networks.
3. Fitur dan Vektor: Data input pada Machine Learning direpresentasikan sebagai fitur atau atribut numerik yang relevan. Fitur ini membantu model dalam mempelajari pola dari data. Data input ini sering diwakili dalam bentuk vektor.
4. Proses Pelatihan (Training Process): Proses di mana model Machine Learning disesuaikan dengan data training. Ini melibatkan memperbarui parameter model untuk mengoptimalkan prediksi model sesuai dengan data training.
5. Evaluasi dan Validasi: Setelah model dilatih, model tersebut dievaluasi dan divalidasi menggunakan data yang terpisah, yang disebut sebagai data validasi atau data pengujian. Evaluasi ini membantu mengevaluasi sejauh mana model dapat melakukan prediksi dengan tepat. dan mampu menggeneralisasi pengetahuan dari data yang belum dilihat sebelumnya.

6 Prediksi dan Inferensi: Setelah model dilatih dan divalidasi, model tersebut dapat digunakan untuk melakukan prediksi atau inferensi pada data baru yang belum pernah dilihat sebelumnya.

Dalam dunia Machine Learning, terdapat berbagai jenis dan pendekatan yang berbeda, di antaranya adalah Supervised Learning (pembelajaran terbimbing), Unsupervised Learning (pembelajaran tanpa pengawasan), dan Reinforcement Learning (pembelajaran dengan penguatan). Setiap jenis Machine Learning ini memiliki tujuan dan teknik yang berbeda, yang digunakan tergantung pada situasi dan permasalahan yang ingin dipecahkan.

Penerapan Machine Learning sangat luas dan dapat ditemukan di berbagai bidang, termasuk pengenalan suara dan gambar, analisis data, rekomendasi produk, deteksi penipuan, pengenalan pola, pemrosesan bahasa alami, dan banyak lagi. Dengan kemajuan teknologi dan ketersediaan data yang melimpah, Machine Learning terus berkembang dan menjadi salah satu bidang yang paling penting dalam ilmu komputer.

2.5 Aspect-Based Sentiment Analysis

Analisis sentimen dapat diklasifikasikan menjadi sentimen positif, negatif, dan netral terhadap seluruh dokumen. Adapun perluasan dari bidang analisis sentimen adalah analisis sentimen berbasis aspek (ABSA) yang dilakukan terhadap aspek-aspek tertentu yang terdapat dalam dokumen. ABSA adalah tugas yang lebih kompleks daripada tradisional Analisis sentimen tingkat teks. Ini berfokus pada mengidentifikasi atribut atau aspek entitas yang disebutkan dalam teks, bersama dengan sentimen yang diungkapkan terhadap setiap aspek (Hoang et al., 2019). Analisis sentimen berbasis aspek dilakukan karena analisis sentimen pada tingkat dokumen hanya mengidentifikasi sentimen secara keseluruhan dalam sebuah kalimat, tetapi tidak semua bagian dalam kalimat memiliki opini positif, dan tidak semua bagian memiliki opini negatif. Oleh karena itu, penting untuk memperhatikan aspek-aspek yang terdapat dalam kalimat agar analisis kalimat tersebut lebih menyeluruh. Dengan melakukan analisis sentimen berbasis aspek, maka sentimen negatif, positif, dan netral dapat ditentukan pada setiap aspek dalam kalimat. Tujuan dari Analisis Sentimen Berbasis Aspek (ABSA) adalah

untuk membuat klasifikasi pada setiap kalimat, memisahkan antara sentimen positif, negatif, dan netral pada setiap aspek yang terdapat dalam ulasan, review atau opini.

Sebagai contoh, mari kita ambil kasus ulasan pengguna aplikasi GOJEK. Dalam ulasan tersebut, ada dua aspek yang dibahas, yaitu "layanan" dan "biaya". Dengan ABSA, kita dapat memisahkan sentimen positif terkait dengan layanan yang dianggap sangat bagus, sementara juga menyoroti sentimen negatif yang berkaitan dengan biaya yang dianggap cukup tinggi. Hal ini memungkinkan kita untuk memiliki pemahaman yang lebih mendalam tentang bagaimana orang merasakan berbagai aspek yang berbeda dari suatu entitas.

Jadi, ABSA memberikan pendekatan yang lebih komprehensif dalam menganalisis sentimen, yang tidak hanya mengklasifikasikan sentimen keseluruhan, tetapi juga memecahnya menjadi sentimen positif, negatif, dan netral untuk setiap aspek yang relevan dalam teks.

2.6 User Experience

User experience merupakan hasil evolusi dari bidang-bidang sebelumnya yang memfokuskan pada pemahaman tentang bagaimana pengguna merasakan saat menggunakan sebuah sistem atau aplikasi. Konsep *user experience* bertujuan untuk menciptakan kepuasan khusus bagi pengguna saat berinteraksi dengan sistem tersebut. *User experience* menekankan pada aspek-aspek seperti pengalaman, persepsi, dan nilai-nilai yang terkait dengan interaksi antara manusia dan produk. Inilah sebabnya mengapa *user experience* dianggap sebagai konsep yang sangat luas. Selain itu, *user experience* tidak hanya terbatas pada satu jenis produk atau layanan, tetapi dapat diterapkan pada berbagai hal, seperti perangkat lunak, situs web, aplikasi seluler, perangkat keras, dan banyak lagi.

User experience, sering disingkat sebagai UX, merupakan hasil evolusi dari berbagai bidang yang sebelumnya memfokuskan pada pemahaman bagaimana pengguna merasakan dan berinteraksi dengan sebuah sistem atau aplikasi. Konsep user experience bertujuan untuk menciptakan pengalaman yang memuaskan bagi pengguna saat mereka berinteraksi dengan suatu produk atau layanan. Hal ini mencakup berbagai aspek, seperti bagaimana pengguna merasakan,

mempersepsikan, dan menilai pengalaman mereka saat berhubungan dengan produk atau sistem tertentu. *User experience* adalah konsep yang sangat luas, mencakup berbagai elemen dan aspek yang menciptakan pengalaman pengguna yang positif. Selain itu, UX tidak hanya berlaku untuk satu jenis produk atau layanan saja, melainkan dapat diterapkan pada berbagai hal, termasuk perangkat lunak, situs web, aplikasi seluler, perangkat keras, dan banyak lagi. Kualitas UX dapat sangat memengaruhi bagaimana pengguna berinteraksi dengan produk atau layanan tersebut, dan sering kali menjadi faktor penting dalam menentukan kesuksesan dan penerimaan pengguna terhadap produk atau sistem tersebut.

2.7 Text Mining

Text mining merujuk pada proses pengambilan informasi dan pola tersembunyi dari teks yang sebelumnya tidak diketahui, dan berpotensi berharga secara otomatis atau semi-otomatis dari luar biasa data tekstual yang tidak terstruktur, seperti teks bahasa alami atau Natural language (Hassani et al., 2020). *Text mining* adalah suatu proses yang bertujuan untuk mengubah teks yang tidak terstruktur dalam dokumen dan database menjadi data yang terstruktur. (Tan et al., 2000). *Text mining* adalah proses analisis teks berbahasa alami dalam jumlah besar untuk mengenali pola leksikal dan mengekstrak informasi bermanfaat. Teknik ini sering digunakan untuk kategorisasi, ekstraksi entitas, dan analisis sentimen, dengan tujuan mengidentifikasi wawasan dan tren dalam data terstruktur berukuran besar, *Text Mining* melibatkan langkah-langkah berikut:

1. Preprocessing: Langkah awal dalam *Text Mining* adalah *preprocessing,* di mana teks dipersiapkan untuk analisis lebih lanjut. Ini melibatkan tahap seperti tokenisasi, pembersihan data (menghapus karakter yang tidak relevan atau mengganggu seperti tanda baca atau karakter khusus), pemisahan kata (memisahkan kata-kata dalam kalimat), dan normalisasi (mengubah kata-kata menjadi bentuk dasar seperti stemming atau lemmatization).
2. Representasi Teks: Setelah *preprocessing*, teks diubah menjadi representasi yang lebih terstruktur untuk analisis. Salah satu representasi yang umum digunakan adalah vektor kata (word vectors) atau matriks

dokumen-kata (*document-term matrix*), di mana setiap kata atau dokumen mewakili nilai numerik. Representasi ini memungkinkan komputer untuk memanipulasi dan menghitung kesamaan antara kata-kata atau dokumen.

3. Analisis Sentimen: Salah satu tugas umum dalam *Text Mining* adalah analisis sentimen, di mana pendapat, sikap, atau emosi dalam teks diklasifikasikan menjadi positif, negatif, atau netral. Ini melibatkan teknik seperti klasifikasi teks, penggunaan kamus kata-kata positif/negatif, atau bahkan pendekatan berbasis pembelajaran mesin untuk mengenali sentimen dalam teks.
4. Tema dan Topik: Text Mining juga dapat digunakan untuk mengidentifikasi tema dan topik yang dominan dalam koleksi teks. Ini melibatkan teknik seperti analisis frekuensi kata-kata (*term frequency analysis*), pemodelan tema (*topic modeling*), atau clustering untuk mengelompokkan teks berdasarkan kesamaan tema.
5. Ekstraksi Informasi: *Text Mining* juga mencakup ekstraksi informasi, di mana informasi khusus atau entitas dari teks diekstrak dan diidentifikasi. Contohnya termasuk pengenalan entitas bernama seperti orang, tempat, atau organisasi, pengenalan tanggal atau angka, atau pengenalan hubungan antara entitas dalam teks.
6. Klasifikasi dan Prediksi: Metode *Text Mining* juga dapat digunakan untuk mengklasifikasikan atau memprediksi teks berdasarkan atribut atau label tertentu. Misalnya, klasifikasi dokumen berita ke dalam kategori berita politik atau olahraga, atau prediksi ulasan produk sebagai positif atau negatif.
7. Ringkasan Teks: *Text Mining* juga dapat digunakan untuk menghasilkan ringkasan teks yang singkat dan informatif dari dokumen yang lebih panjang. Ini melibatkan teknik seperti ekstraksi frasa penting atau pemodelan bahasa untuk menghasilkan ringkasan yang ringkas namun menggambarkan inti dari teks.

Penerapan *Text Mining* sangat luas dan dapat ditemukan dalam berbagai bidang, seperti analisis media sosial, pengolahan bahasa alami, pencarian informasi, analisis opinii, pengelompokan teks, pemrosesan dokumen, dan banyak

lagi. Dengan adanya kemajuan teknologi komputasi dan ketersediaan data teks yang melimpah, *Text Mining* menjadi alat yang sangat berguna untuk menggali wawasan dari teks dan mendukung pengambilan keputusan yang lebih baik.

2.8 Deep Learning

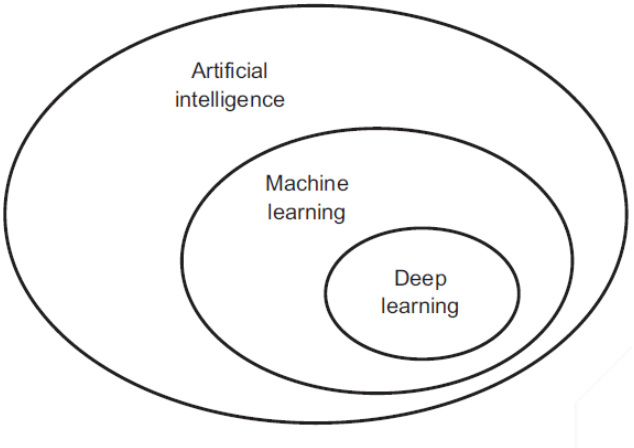
Deep Learning adalah cabang dari *machine learning* yang merupakan bagian dari Kecerdasan Buatan. *Deep Learning* merupakan sebuah metode pembelajaran mesin yang menggunakan jaringan neural yang sangat dalam, terdiri dari banyak lapisan (*layer*), untuk mempelajari representasi yang semakin kompleks dari data yang diberikan (Goodfellow, 2016). Pada *Deep Learning*, jaringan saraf yang terdiri dari multiple layers (layer-layer yang mendalam) digunakan untuk memproses data. Setiap layer di dalam jaringan saraf terdiri dari banyak unit komputasi yang disebut *neuron* atau node. Setiap neuron menerima input dari neuron di layer sebelumnya, melakukan komputasi, dan mengirimkan output ke neuron di layer berikutnya. Proses ini dilakukan secara berulang-ulang melalui layer-layer yang mendalam hingga mencapai layer output, di mana hasil prediksi atau pengklasifikasian diperoleh.

Deep Learning memungkinkan pembelajaran yang otomatis dan hierarkis dari fitur-fitur yang lebih kompleks dan abstrak dalam data. Dengan memperluas jumlah layer dan unit di dalam jaringan saraf, deep learning dapat mengekstraksi representasi yang lebih dalam dan kompleks dari data yang berdampak pada kinerja yang lebih baik dalam tugas-tugas seperti pengenalan pola, pengenalan wajah, pengenalan suara, pemrosesan bahasa alami, dan lain-lain.

Beberapa jenis arsitektur deep learning yang umum digunakan antara lain *Convolutional Neural Networks* (CNN) yang efektif dalam pengolahan gambar dan pengenalan pola spasial, *Recurrent Neural Networks* (RNN) yang cocok untuk data berurutan seperti teks dan suara, dan *Generative Adversarial Networks* (GAN) yang digunakan dalam pembuatan dan rekonstruksi data baru.

Deep Learning menjadi sangat populer dan sukses karena kemampuannya untuk mempelajari representasi yang kompleks dan abstrak dari data secara *end-to-end,* tanpa perlu ekstraksi fitur manual. Keberhasilannya dalam berbagai bidang

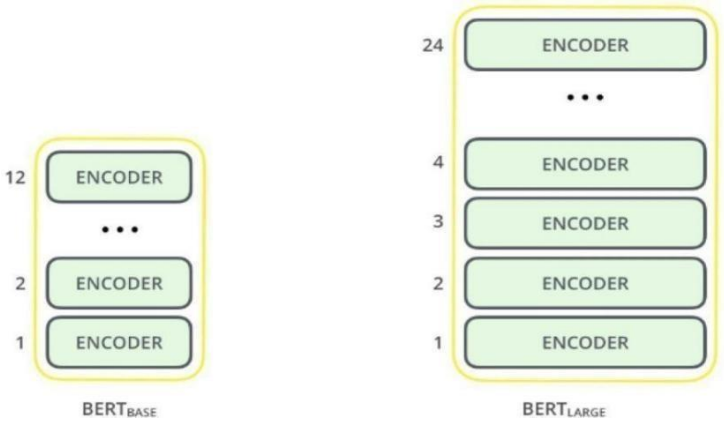
seperti pengenalan wajah, terjemahan mesin, kendaraan otonom, dan banyak lagi, telah membawa kemajuan yang signifikan dalam bidang kecerdasan buatan. Pada Gambar 2.1 dibawah akan ditunjukkan hubungan antara kecerdasan buatan, machine learning dan deep learning.



Gambar 2.1 Keterkaitan antara *Machine Learning*, *Deep Learning* , dan Kecerdasan Buatan (Chollet, 2018)

2.9 BERT (Bidirectional Encoder Representations from Transformers)

BERT (*Bidirectional Encoder Representations from Transformers*) pertama kali diperkenalkan oleh para peneliti Google AI pada tahun 2018 yang merupakan salah satu algoritma deep learning yang dipakai dalam *Natural Language Processing* (NLP). BERT adalah model bahasa mendalam yang berbasis Transformer. Model ini menggunakan struktur Transformer yang mendalam untuk belajar representasi bahasa yang kaya tanpa pengawasan. Dengan arsitektur Transformer ini, BERT dapat mengembangkan pemahaman yang lebih baik tentang konteks dan keterhubungan antara kata-kata dalam sebuah kalimat. BERT memiliki dua varian, yaitu BERT-base dan BERT-large. (Devlin et al., 2018). Berikut merupakan perbedaan BERT dijelaskan pada gambar 2.2.



Gambar 2.2 Ukuran $BERT_{BASE}$ dan $BERT_{LARGE}$

BERT memiliki mekanisme *transformers* yang mempunyai kemampuan untuk menangkap hubungan konteks dari kata-kata dalam suatu teks (Vaswani et al., 2017). *Transformers* mampu memproses dan mengonversi konsep yang dipahaminya melalui suatu mekanisme bernama *self-attention mechanism*. *Self-attention mechanism* adalah suatu metode yang digunakan oleh transformers untuk mengubah hubungan antara kata-kata yang terkait menjadi representasi yang dapat diproses oleh mekanisme tersebut. Terdapat dua mekanisme dalam *transformers* yaitu:

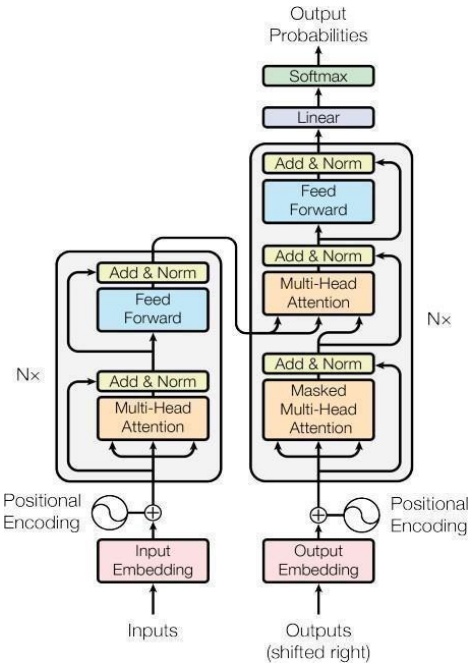
1. *Encoder*
   Encoder Dalam algoritma BERT (*Bidirectional Encoder Representations from Transformers*) peran *encoder* sangat penting. *Encoder* pada BERT bertanggung jawab untuk menghasilkan representasi kontekstual dari teks *input*. Dalam konteks BERT, *encoder* mengambil teks *input*, seperti kalimat atau dokumen, dan melakukan transformasi pada kata-kata di dalamnya. *Encoder* BERT menggunakan arsitektur *Transformer*, yang terdiri dari beberapa lapisan tersembunyi. Setiap lapisan tersebut melakukan pemrosesan paralel yang memungkinkan pemahaman konteks dua arah (*bidirectional*) dalam teks. *Encoder* BERT mengekstrak fitur-fitur penting dari kata-kata dalam teks dan mengkodekannya menjadi representasi yang kaya secara informasi. Representasi ini mempertimbangkan konteks kata sebelumnya dan konteks kata berikutnya dalam kalimat atau dokumen. Dengan demikian, *Encoder* BERT memiliki kemampuan untuk menangkap hubungan antar kata dan memahami

konteks dengan lebih baik. Representasi kontekstual yang dihasilkan oleh *encoder* BERT kemudian dapat digunakan untuk berbagai tugas pemrosesan bahasa alami, seperti analisis sentimen, pemahaman bahasa, dan tugas-tugas lainnya. Dalam tugas analisis sentimen sebagai contoh, *encoder* BERT mengubah teks input menjadi representasi yang mencerminkan sentimen atau perasaan yang terkandung dalam kalimat tersebut.

2. *Decoder*

   Dalam arsitektur BERT (*Bidirectional Encoder Representations from Transformers*), tidak ada bagian khusus yang disebut "*decoder*." BERT adalah model yang terdiri dari lapisan-lapisan transformer yang bekerja dalam arah dua arah (bidirectional), yang berarti ia dapat memproses dan memahami konteks kata dari kedua arah sebelum dan sesudah kata tertentu.

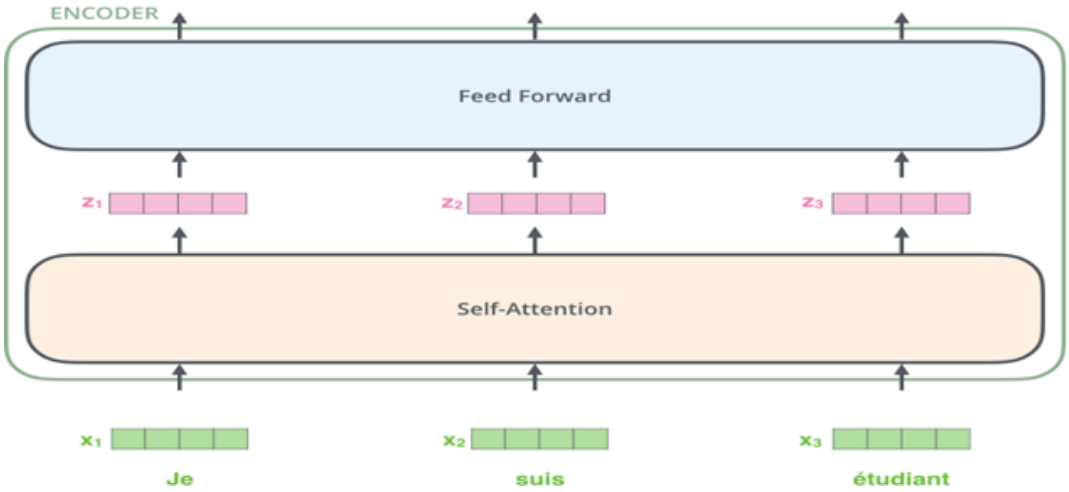
   Namun, untuk beberapa tugas pemrosesan bahasa alami yang membutuhkan langkah-langkah generatif atau pemodelan bahasa secara berurutan, seperti machine translation atau language generation, arsitektur transformer biasanya membutuhkan komponen "*decoder*". *Decoder* bertanggung jawab untuk menghasilkan teks secara berurutan berdasarkan representasi kontekstual yang diberikan oleh *encoder*. Decoder di dalamnya terdiri dari tumpukan yang terdiri dari enam lapisan teridentifikasi. Setiap lapisan ini memiliki struktur yang serupa dengan yang ada pada *encoder*, yaitu terdiri dari dua bagian. Tiap lapisan memiliki dua sub-lapisan yang serupa dengan *encoder*, namun dengan penambahan lapisan perhatian di antara kedua sub-lapisan ini. Lapisan perhatian tambahan ini membantu node saat ini untuk memperoleh konten kunci yang membutuhkan perhatian (Vaswani et al., 2017) melalui penerapan perhatian multi-head pada hasil keluaran *encoder*. Seperti halnya pada *encoder*, lapisan self-attention di dalam decoder memungkinkan setiap posisi di decoder untuk berhubungan dengan semua posisi sebelumnya dan posisi saat ini. Gambar 2.3 akan menjelaskan arsitektur *transformers* dan alur kerja *encoder* dan *decoder*.



Gambar 2.3 Transformers *Encoder* (kiri) Decoder (kanan)

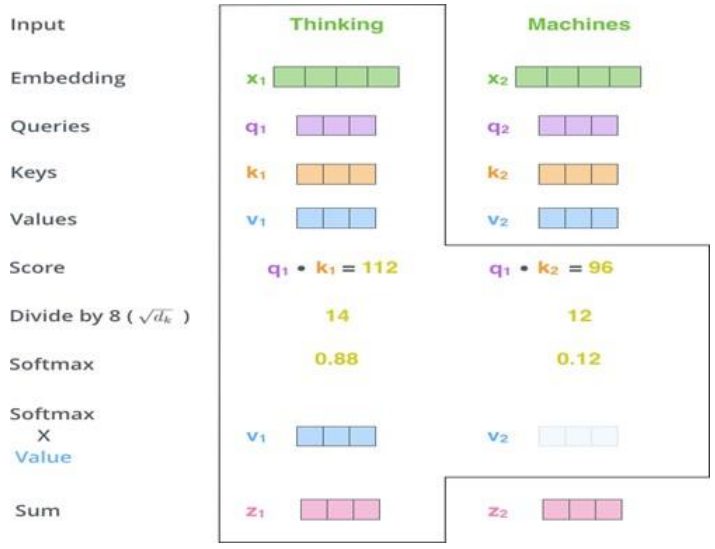
Dalam arsitektur BERT base, langkah-langkah yang terjadi di *encoder* dan decoder tidak terjadi secara terpisah seperti dalam arsitektur Transformer standar. Sebagai gantinya, BERT hanya memiliki *encoder* dan tidak melibatkan decoder seperti yang digunakan dalam tugas generatif seperti mesin terjemahan. Berikut adalah langkah-langkah yang terjadi di *encoder* BERT base:

1. *Input Embedding*:
    - Teks *input*, seperti kalimat atau dokumen, diubah menjadi representasi vektor dengan menggunakan lapisan *embedding*.
    - Setiap kata dalam teks diberikan vektor yang menyimpan informasi kata tersebut.
2. Self-Attention:
    - *Self-attention layer* digunakan untuk menekankan kata-kata yang relevan dalam teks pada gambar 2.3 akan ditunjukkan proses dari *self-attention*.
    - Representasi vektor dari setiap kata diproses melalui serangkaian perhitungan attention, di mana bobot perhatian dihitung antara semua pasangan kata dalam teks.
    - Dengan self-attention, BERT dapat memahami konteks dan hubungan antara kata-kata dalam teks.



Gambar 2.4 Sistemasi pada Self-attention Layer (Alammar, 2018)

3. Feed-Forward Neural Network:
    - Representasi vektor dari *self-attention layer* diproses melalui beberapa lapisan linier dan fungsi aktivasi.
    - *Feed-forward Neural Network* membantu dalam memperoleh representasi yang lebih kompleks dan abstrak dari teks.
4. Stacked *Encoder*:
    - Langkah-langkah di atas, yaitu *self-attention* dan feed-forward neural network, diulang dalam beberapa blok transformasi (*transformer blocks*) yang saling terhubung.
    - Setiap blok transformasi memiliki *self-attention layer dan feed-forward neural* networknya sendiri.
    - Dengan memiliki beberapa blok transformasi, BERT *base* dapat mempelajari representasi yang semakin dalam dan kompleks dari teks.
    - Seluruh langkah-langkah di atas terjadi di *encoder* BERT base untuk memproses teks *input* dan menghasilkan representasi vektor yang kaya dari teks tersebut. Representasi ini dapat digunakan dalam berbagai tugas pemrosesan bahasa alami, seperti analisis sentimen, klasifikasi teks, dan pemahaman *bahasa* pada gambar 2.4 akan ditunjukkan prosses dari *encoder*.



Gambar 2.5 Sistemasi pada *Encoder* (Alammar, 2018)

Perbedaan antara BERT base dan BERT large terletak pada ukuran dan kompleksitas modelnya. Berikut adalah perbedaan utama antara keduanya:

1. Ukuran Model
   - BERT base: Model BERT base memiliki ukuran yang relatif lebih kecil dibandingkan dengan BERT large. Biasanya, BERT base memiliki sekitar 110 juta parameter.
   - BERT large: Model BERT large memiliki ukuran yang lebih besar dibandingkan dengan BERT base. Biasanya, BERT large memiliki sekitar 340 juta parameter.
2. Jumlah Layer
   - BERT base: Model BERT base terdiri dari 12 layer transformasi (transformer blocks) dalam *encoder*.
   - BERT large: Model BERT large terdiri dari 24 layer transformasi (transformer blocks) dalam *encoder*.
3. Kompleksitas dan Representasi
   - BERT base: Meskipun BERT base lebih kecil, model ini sudah cukup mampu dalam melakukan tugas-tugas pemrosesan bahasa alami seperti analisis sentimen, klasifikasi teks, dan pemahaman bahasa.
   - BERT large: Dengan ukuran dan jumlah layer yang lebih besar, BERT large memiliki kapasitas yang lebih tinggi dalam memperoleh pemahaman yang lebih mendalam tentang teks dan kemampuan

representasi yang lebih kuat. Model ini biasanya lebih baik dalam tugas-tugas pemrosesan bahasa yang kompleks.

4. Waktu dan Sumber Daya
    - Karena ukuran dan kompleksitas yang lebih tinggi, BERT large membutuhkan lebih banyak waktu dan sumber daya komputasi untuk melatih dan menjalankan model dibandingkan dengan BERT base.
    - BERT base lebih efisien dalam hal waktu dan sumber daya, sehingga lebih mudah diakses dan digunakan.

Pemilihan antara BERT base dan BERT large tergantung pada kebutuhan tugas pemrosesan bahasa yang spesifik. Jika tugas tersebut relatif sederhana, BERT base sudah dapat memberikan hasil yang memadai. Namun, jika tugas tersebut lebih kompleks dan membutuhkan pemahaman yang lebih mendalam, BERT large dapat menjadi pilihan yang lebih baik.

Representasi *input* pada BERT melibatkan tiga jenis *embedding layers*, yaitu token *embedding*, segment *embedding*, dan *positional embedding*. Berikut adalah penjelasan mengenai ketiga *embedding layers* tersebut:

1. Token *Embedding*:
    - Token *embedding* mengubah setiap token dalam kalimat menjadi representasi vektor dengan dimensi tertentu.
    - Setiap token diubah menjadi vektor dengan menggunakan tabel *embedding* yang telah dilatih sebelumnya. Tabel *embedding* ini berisi vektor numerik untuk setiap token dalam kamus yang digunakan oleh BERT.
    - Token *embedding* memungkinkan BERT untuk memahami arti dan representasi masing-masing kata dalam kalimat.
2. Segment *Embedding*:
    - Segment *embedding* digunakan untuk membedakan dan memisahkan dua kalimat dalam satu pasangan kalimat.
    - Setiap kalimat dalam pasangan kalimat diberi label segment *embedding* yang berbeda. Misalnya, kalimat pertama dapat diberi label "0" dan kalimat kedua diberi

label "1".

- Segment *embedding* membantu BERT untuk memahami perbedaan dan hubungan antara dua kalimat dalam konteks tertentu.

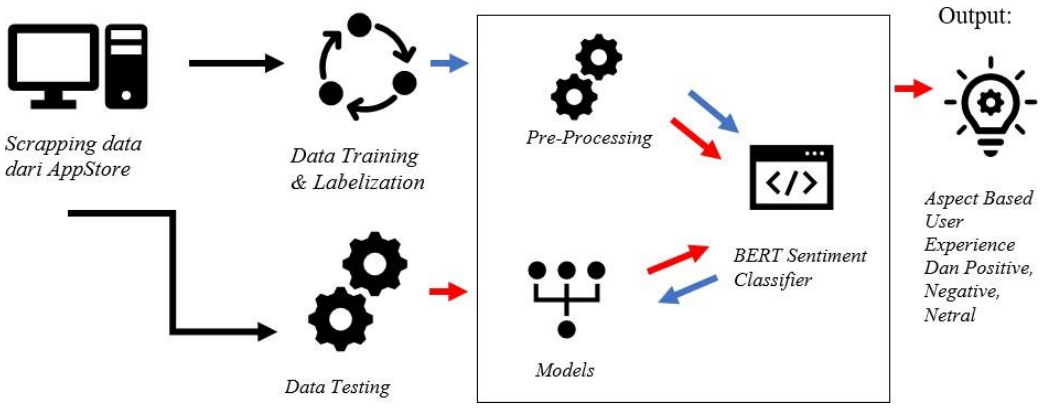
3. *Positional Embedding*:
    - *Positional embedding* digunakan untuk menyertakan informasi urutan kata dalam kalimat.
    - Setiap token diberi vektor *positional embedding* yang mencerminkan posisi relatif token dalam kalimat.
    - *Positional embedding* memungkinkan BERT untuk memahami urutan kata dan hubungan spasial antara kata-kata dalam kalimat.

BAB 3
ANALISIS DAN PERANCANGAN

Bab ini akan membahas perancangan dan analisis yang dipakai dalam sentimen analisis menggunakan algoritma BERT yang berbasis aspek.

3.1 Arsitektur Umum

Dalam analisis sentimen terhadap komentar PUBGM di *App Store*, telah diajukan metode yang menggunakan pendekatan Bidirectional *Encoder* Representations from Transformers (BERT). Metode ini melibatkan beberapa langkah seperti yang dijelaskan pada Gambar 3.1.



Gambar 3.1 Arsitektur Umum

Arsitektur umum sistem analisis sentimen dengan menggunakan algoritma BERT melibatkan beberapa langkah penting. Pertama, teks input yang berisi komentar atau ulasan diberikan kepada model BERT yang di scrapping dari *AppStore*. Selanjutnya data yang telah di labelling akan di split untuk data training dan data testing. Langkah berikutnya adalah melakukan tokenisasi pada teks input, di mana teks dipecah menjadi token-token yang lebih kecil untuk diproses oleh model. Setelah itu, token-token tersebut diberi tanda khusus seperti token [CLS] sebagai token awalan dan token [SEP] sebagai penanda akhir teks atau pemisah antara dua teks jika ada. Kemudian, token-token tersebut dilewatkan melalui layer embedding, di mana setiap token diubah menjadi representasi vektor dengan menggunakan token embedding, segment embedding, dan positional embedding. Selanjutnya, representasi vektor tersebut melewati beberapa lapisan transformer

dalam model BERT untuk memperoleh pemahaman kontekstual yang lebih dalam. Setelah melewati lapisan transformer, keluaran akhir dari token [CLS] diambil dan diberikan ke lapisan klasifikasi tambahan untuk memprediksi sentimen dari teks input, seperti positif, negatif, atau netral. Hasil prediksi sentimen dapat digunakan untuk analisis lebih lanjut atau pemantauan terhadap komentar atau ulasan yang diberikan. Arsitektur ini memanfaatkan kekuatan representasi kontekstual dari model BERT untuk memahami teks input dengan lebih baik dan menghasilkan hasil analisis sentimen yang lebih akurat.

3.2 Scraping

Pada penelitian ini menggunakan data yang diperoleh dari web *AppStore* yang diambil menggunakan library app_store_scraper. Agar dapat mengscrap aplikasi yang di inginkan maka input id aplikasi yang sudah didapat pada web *AppStore* pada kasus ini id aplikasi nya ialah 1330123889, Lalu untuk mendapat kan data yang diinginkan maka menggunakan review_description agar mendapatkan data set yang sesuai keinginan. Dataset yang sudah melalui proses tersebut kemudian disimpan dalam bentuk *excel worksheet* (.xlsx)

3.3 Labelisasi Dataset

Labelisasi dataset merupakan langkah penting dalam analisis sentimen karena membantu dalam melatih model atau algoritma untuk mengenali dan mengklasifikasikan sentimen pada data baru. Dengan menggunakan dataset yang telah dilabeli, model pembelajaran mesin dapat belajar pola dan konteks yang berkaitan dengan sentimen tertentu, sehingga dapat digunakan untuk mengklasifikasikan sentimen pada teks yang belum diketahui labelnya. Dalam konteks analisis sentimen, sentimen dapat diklasifikasikan menjadi beberapa kategori umum, seperti positif, negatif, atau netral.

Proses labelisasi dataset dapat dilakukan secara manual oleh seorang penilai manusia yang membaca dan memberikan label pada setiap data, atau menggunakan metode otomatis dengan bantuan algoritma atau model analisis sentimen yang telah dilatih sebelumnya. Contoh dataset yang sudah dilabelisasi sebagaimana terlihat pada Tabel 3.1 dengan nilai sentiment 1 sebagai positif , nilai sentiment 0 sebagai netral dan nilai sentiment -1 sebagai negatif.

Tabel 3.1 Contoh Dataset

Komentar	Sentimen
Gokil banget update kali ini, bener2 gila... Baru update, langsung main & dapet WWCD! Arigato PUBGM yang udah terus berkembang... Saran saya aja nih ya, kalau nentuin match yang setara aja tier nya... Saya baru crown 1 aja musuhnya qonqueror 5,3, dan seterusnya... Itu aja saran saya, jaya terus KRAFTON & PUBG MOBILE!!!ðŸ˜€ðŸ¤—	1
Banyakin bugnya tod anjg ini lagi habis update susah kali mau login anjg	-1
kenapa punyaku datanya keriset yaa setelah update. kocak ni game emng	-1
Sejak di pegang level infinite pubg sudah hancur. Level invinite memang terlalu bodoh. Banyak bug di tdm dan rank. Kalo memang anda tidak bisa menjaga game ini, hapus saja lah bodoh, sadar lu itu bodoh level infinite	-1
Update mulu anj bikin hape nge frem gw kan cuman mau main biasa ðŸ™,hape kentang kek gw mana bisa anj kalo update Â² game mulu sebulan sekali lu?update?	-1
Saya main game ini dari awal liris sampai sekarang, makin sini game ini makin berat. Update game makin gede, HP dengan RAM 8GB aja sering ngeframe apalagi turun di event. Tolonglah dari pada banyak update skin mending update gimana caranya biar game ini bisa ringan, biar lebih banyak yg bisa mainin. Terima kasih	-1
Gamenya bagus terima kasih telah mengembalikan mode metro royale tapi tidak bisa unggah ke awan saya main di jam 18.53 di negara saya	0
Tidak bisa tambah bind dengan isntan jadi ga enak tolong perbaiki biar bisa bind instan tanpa menunggu	-1

3.4 Pre-processing Dataset

Pada penelitan ini Pre-processing merupakan tahap awal dalam pengolahan data yang melibatkan serangkaian teknik untuk membersihkan dan mempersiapkan data mentah sebelum diproses lebih lanjut, seperti penghapusan data duplikat, penghapusan nilai yang hilang, dan transformasi data (Ahmed dan Kamal, 2020). Tahapan - tahapan yang terdapat pada *pre-processing* adalah *case folding*, *tokenizing*, *filtering*, dan *stemming*. Tahap *pre-processing* sangat penting karena dapat mempengaruhi kualitas analisis data dan hasil akhirnya. Dengan membersihkan dan mempersiapkan data dengan baik, analisis selanjutnya dapat berjalan lebih lancar dan menghasilkan wawasan yang lebih akurat.

3.4.1 Case Folding

Proses *Case folding* yaitu dengan konversi karakter teks menjadi huruf kapital atau tidak kapital untuk memudahkan pemrosesan teks dalam pengolahan bahasa alami (Ali dan Idris, 2018). Hanya karakter dari huruf 'a' hingga 'z' yang akan diterima, sementara karakter selain huruf akan dihapus dan dianggap sebagai pemisah (*delimiter*).

3.4.2 Data Cleaning

Proses *data cleaning* merupakan tahap penting dalam analisis data karena data yang kotor atau tidak valid dapat menghasilkan hasil yang salah atau mengarah pada kesimpulan yang salah. Contoh kata atau karakter yang dapat membuat data kotor seperti karakter yang berulang dua atau lebih, link, *username* (*@username*), *hashtag* (#), angka, simbol-simbol, spasi berlebih, tanda baca. Dengan membersihkan data sebelum menganalisisnya, kita dapat meningkatkan kualitas dan keandalan hasil analisis serta memastikan kepercayaan pada temuan dan keputusan yang diambil dari data tersebut.

3.4.3 Tokenizing

Tokenizing adalah langkah memecah teks menjadi kata-kata individual yang disebut token, sehingga mempermudah pemrosesan teks dalam pengolahan bahasa alami dan analisis data. Data atau dokumen awal biasanya berupa *string*. Dalam tahap ini, string input akan dipecah menjadi kata-kata berdasarkan spasi. Penulis menggunakan fungsi *word_tokenize* yang sudah disediakan *library* NLTK.

3.4.4 Stopwords Removal

Penghapusan *stopwords* adalah langkah untuk mengeliminasi kata-kata umum yang kurang memiliki makna khusus dan tidak memberikan kontribusi penting dalam analisis teks. Kata-kata ini sering muncul dalam teks tanpa memberikan informasi yang berarti tentang konten teks yang sedang dianalisis. Contoh *stopwords* dalam bahasa Indonesia termasuk "yang", "dan", "di", "dari", "adalah", dan sebagainya.

3.4.5 Stemming

Stemming adalah proses mengubah kata-kata dalam teks menjadi kata dasar atau kata dasar dalam bahasa yang sama untuk meningkatkan efektivitas dan efisiensi dalam analisis data (Setiawan, Sutawijaya dan Purwanto, 2018). Jadi, *stemming* adalah teknik yang universal dalam pemrosesan teks, tetapi perlu memperhatikan aturan bahasa yang berbeda untuk menjalankannya dengan efektif, terutama ketika berurusan dengan bahasa yang berbeda seperti bahasa Inggris dan bahasa Indonesia.

3.4.6 Normalisasi

Normalisasi dalam pemrosesan teks menggunakan Python adalah proses mengubah teks menjadi bentuk standar atau normal agar lebih mudah untuk diproses. Tujuan normalisasi adalah menghilangkan variasi dan keseragaman teks, sehingga kata-kata yang sebenarnya memiliki makna yang sama dapat diperlakukan secara serupa. Jika ada kata-kata baku di dalam komentar seperti “d”, “n”, “trims” dll akan diubah menjadi kata baku yaitu “di”, “dan”, dan “terima kasih”. Proses normalisasi dapat membantu dalam mempersiapkan data teks sebelum melakukan analisis lebih lanjut seperti klasifikasi, pemodelan bahasa, atau analisis sentimen. Dengan mengaplikasikan langkah-langkah normalisasi, kita dapat mendapatkan teks yang lebih terstruktur dan siap digunakan dalam proses pemrosesan dan analisis yang lebih lanjut.

3.5 Implementasi BERT

Dalam penelitian ini, penulis mengambil keuntungan dari dua model yang

populer dalam bidang pemrosesan bahasa alami, yaitu BERTBASE dan *bert-multilingual-base-cased*. Keunggulan dari kedua model ini adalah kemampuannya dalam mendukung sebanyak 104 bahasa, termasuk Bahasa Indonesia. Dengan memanfaatkan model-model tersebut, peneliti dapat melakukan analisis dan pemrosesan teks dengan lebih luas dan lebih efisien.

Model BERTBASE dan *bert-multilingual-base-cased* dapat diakses dengan mudah melalui *library Transformers* yang disediakan oleh *HuggingFace*. *Library* ini menawarkan akses ke ribuan model pre-trained, yang telah dilatih dengan data besar sebelumnya, sehingga dapat dengan mudah digunakan untuk berbagai tugas pemrosesan bahasa alami. Dari klasifikasi teks hingga ekstraksi informasi, dari sistem tanya jawab hingga ringkasan teks, dan bahkan translasi, library ini menyediakan solusi untuk berbagai tantangan dalam pemrosesan bahasa alami, dan semua ini dapat diaplikasikan dalam lebih dari 100 bahasa berbeda.

Hal yang menarik adalah bahwa *library Transformers* didukung oleh dua *library deep learning* terkenal, yaitu *PyTorch* dan *TensorFlow*. Dengan memanfaatkan kekuatan dari kedua library ini, pengguna dapat melakukan pemrosesan bahasa alami dengan tingkat keakuratan dan performa yang tinggi. Kedua library ini menyediakan dukungan untuk mengoperasikan model BERTBASE dan *bert-multilingual-base-cased*, sehingga memudahkan para peneliti dan praktisi dalam menggali potensi penuh dari model-model tersebut.

3.5.1 Penjelasan BERT

Wordpiece adalah metode pemisahan kata yang digunakan dalam model Bahasa-Berdasar Transformer (BERT) dan beberapa model NLP (Natural Language Processing) lainnya. Ini adalah pendekatan yang digunakan untuk mengatasi permasalahan pengkayaan kosakata dan penggabungan kata dalam pengolahan teks. Dalam metode *Wordpiece*, kata-kata dalam kalimat dibagi menjadi potongan-potongan yang lebih kecil yang disebut "*subword units*" atau "*subwords*". *Subwords* ini dapat berupa kata-kata utuh atau potongan kata. Pendekatan ini memungkinkan model untuk mempelajari representasi yang lebih kaya dari teks yang mengandung kosakata yang tidak umum atau terfragmentasi. Misalnya, kata "unhappiness" dapat dibagi menjadi dua subword yaitu "un" dan "happiness". Demikian pula, kata "unhappy" dapat dibagi menjadi "un" dan

"happy". Dalam model BERT, setiap subword diberikan token khusus yang disebut "*subword* token" dan semua *subword* digunakan sebagai masukan untuk model. Dengan menggunakan metode *Wordpiece*, model seperti BERT dapat memahami kata-kata yang tidak terlihat dalam data latih dan dapat mengatasi variasi bentuk kata yang umum dalam bahasa. Ini membantu meningkatkan kinerja model dalam tugas-tugas NLP seperti pemahaman bahasa alami, penerjemahan mesin, dan pengenalan entitas berdasarkan konteks. Langkah-langkah membuat *vocabulary* dalam BERT antara lain:

1. *Preprocessing*: Lakukan *preprocessing* pada teks, seperti menghapus tanda baca, mengubah huruf menjadi lowercase, menghapus karakter khusus, dan melakukan normalisasi teks.
2. *Tokenization*: Pecah teks menjadi token-token, misalnya dengan memisahkan berdasarkan spasi atau menggunakan metode tokenisasi yang lebih kompleks seperti menggunakan library seperti NLTK atau spaCy.
3. Menghitung frekuensi: Hitung frekuensi kemunculan setiap token dalam teks. Ini dapat dilakukan dengan mengiterasi melalui teks dan menghitung jumlah kemunculan setiap token.
4. *Filtering*: Lakukan *filtering* pada token-token yang tidak relevan atau tidak signifikan. Ini dapat meliputi penghapusan *stopwords* (kata-kata umum yang tidak memberikan banyak informasi), tanda baca, karakter khusus, atau kata-kata yang muncul sangat jarang.
5. Pembentukan *vocabulary*: Dalam tahap ini, gabungkan semua token yang telah melalui tahap-tahap sebelumnya untuk membentuk *vocabulary.*
6. *Encoding*: Setiap token dalam *vocabulary* diberikan sebuah kode numerik, baik berupa indeks atau representasi vektor. Ini memungkinkan komputer untuk memproses kata-kata dalam bentuk numerik.

Penggunaan metode *Wordpiece* dalam model seperti BERT membawa

dampak signifikan dalam pemahaman teks yang lebih mendalam dan kemampuan mengatasi variasi bentuk kata yang umum dalam bahasa. Hal ini, pada gilirannya, berkontribusi pada peningkatan kinerja model dalam berbagai tugas NLP, termasuk pemahaman bahasa alami, penerjemahan mesin, dan pengenalan entitas berdasarkan konteks. Tahapan-tahapan dalam pembuatan vocabulary untuk model seperti BERT sangat penting dalam memahami dan mengelola teks dengan baik.

Proses persiapan kalimat menjadi representasi *input* dalam BERT dilakukan oleh tokenizer. Berikut langkah-langkah untuk mengubah kalimat menjadi representasi *input* dalam algoritma BERT:

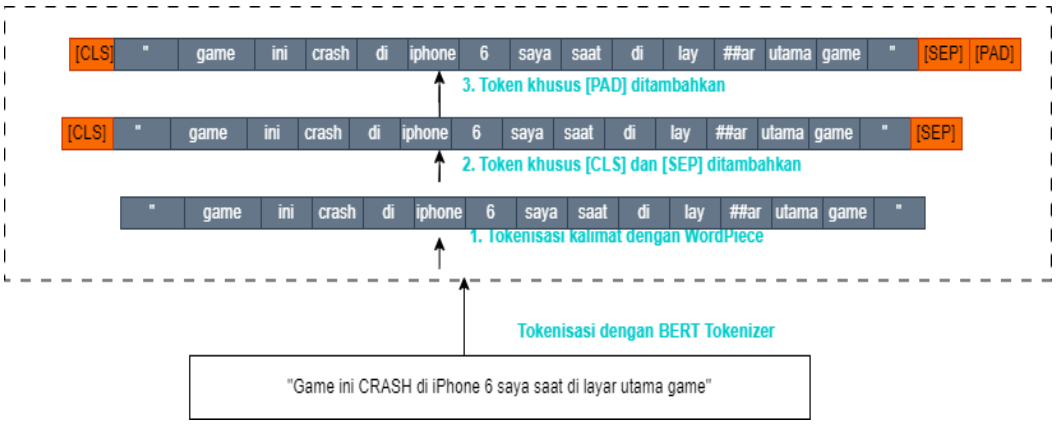
1. Tokenisasi *WordPiece*: Teks awal dibagi menjadi unit yang lebih kecil, yang disebut *WordPiece*. *WordPiece* adalah teknik tokenisasi di manakata-kata dalam teks dapat dipecah menjadi sub-kata yang lebih kecil berdasarkan frekuensi kemunculannya dalam corpus pelatihan. Misalnya, kata "playing" dapat dipecah menjadi "play" dan "##ing". Ini membantu dalam mengatasi kata-kata yang tidak dikenal atau kata-kata yang jarang muncul dalam corpus. Selanjutnya akan dilakukan penambahan token Khusus. Token-token khusus ditambahkan sebagai bagian dari tokenisasi. Token [CLS] ditambahkan di awal teks sebagai token kelas, yang digunakan untuk merepresentasikan kalimat secara keseluruhan. Token [SEP] ditambahkan sebagai pemisah antara dua kalimat atau bagian teks yang berbeda. Jika ada kalimat lebih dari satu, token [SEP] juga digunakan sebagai pemisah antara kalimat-kalimat tersebut. Lalu setelah tahap ini selesai akan dilakukan *padding* dan truncation dimana panjang teks dibatasi menjadi ukuran tertentu sesuai dengan batas yang ditentukan. Jika teks terlalu panjang, maka akan dilakukan truncation atau pemotongan. Jika teks terlalu pendek, maka akan dilakukan *padding* dengan menambahkan token khusus [PAD] agar memiliki panjang yang sama dengan teks lainnya. Pada Gambar 3.2 merupakan tahap awal sebuah kalimat diubah menjadi sebuah token-token kata. Kalimat yang diambil sebagai contoh adalah “Game ini CRASH di iPhone 6 saat di layer utama game”.

Gambar 3.2 Pengerjaan Tokenisasi dengan *WordPiece*

2. Setiap kalimat dalam proses tokenisasi diberi token-token khusus, yaitu [CLS] pada awal kalimat dan [SEP] pada akhir kalimat. Token [CLS] digunakan sebagai indikator awal sebuah kalimat dan juga sebagai indikator sentimen saat dilakukan klasifikasi sentimen. Sementara itu, token [SEP] digunakan untuk memisahkan satu kalimat dari kalimat berikutnya. Setelah diberi token-token khusus ini, kalimat-kalimat tersebut menjadi token *embeddings*. Tahap *token embeddings* ini ditunjukkan dalam Gambar 3.3.

Gambar 3.3 Pengerjaan Token *Embeddings*

3. Setelah tokenisasi dan pemberian token-token khusus [CLS] dan [SEP], kalimat-kalimat akan disesuaikan dengan panjang maksimum yang telah ditentukan dengan mengurangi atau memberi *padding* menggunakan token khusus [PAD]. Dalam Gambar 3.4, ditunjukkan tahap pemberian padding, di mana panjang maksimum yang ditetapkan adalah 16. Jika panjang kalimat melebihi 16 token, maka beberapa token diakhir kalimat akan dihapus. Sebaliknya, jika panjang kalimat kurang dari 16 token, maka token [PAD] akan ditambahkan di akhir kalimat setelah token [SEP].



Gambar 3.4 Pengerjaan Pemberian *Padding*

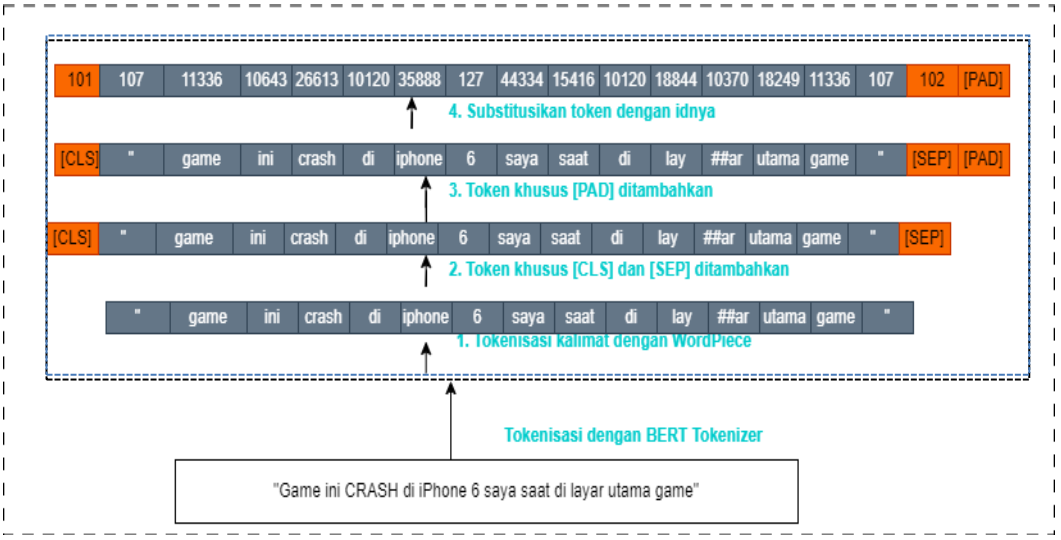
```
100 [UNK]
101 [CLS]
102 [SEP]
103 [MASK]
104 <S>
105 <T>
106 !
107 "
108 #
109 $
110 %
111 &
```

Gambar 3.5 Indeks Token

Dalam proses pelatihan model BERT, salah satu tahap yang kritis adalah mengkonversi setiap kata, sub kata, dan karakter dalam teks menjadi bilangan unik atau ID yang dapat dimengerti oleh model. Ini adalah langkah penting yang dilakukan setelah tokenisasi dan penambahan token khusus seperti [CLS] dan [SEP]. Proses ini bertujuan untuk memberikan representasi numerik bagi setiap unsur dalam teks, sehingga model BERT dapat memahaminya dengan baik. Penting untuk diingat bahwa setiap kata atau karakter dalam teks akan dihubungkan dengan ID-nya sendiri berdasarkan vocabulary yang telah dibuat sebelumnya. *Vocabulary* ini disusun dengan memperhitungkan tingkat kemunculan setiap kata dalam korpus data yang digunakan untuk pelatihan model. Sebagai contoh, jika kata "Game" muncul dalam vocabulary, maka kata tersebut akan memiliki ID tertentu, seperti 11336, sesuai dengan indeksnya dalam vocabulary. Semua kata dan sub kata dalam teks harus mengalami proses ini agar dapat dimengerti oleh model pre-trained BERT.

Dengan demikian, proses substitusi token menjadi ID ini adalah tahap yang tak terpisahkan dalam persiapan data untuk pelatihan model BERT. Selanjutnya, ID ini akan digunakan oleh model selama proses pelatihan dan saat model tersebut digunakan untuk tugas pemrosesan bahasa alami lainnya. Tahap ini merupakan fondasi penting dalam memungkinkan model BERT untuk memahami dan menganalisis teks dengan baik, dan itulah mengapa langkah ini sangat penting dalam pemahaman bagaimana BERT bekerja.

Setelah tahap tokenisasi dan penambahan token khusus [CLS] dan [SEP], setiap kalimat akan dijadikan bilangan unik atau ID sesuai dengan *vocabulary* yang telah dibuat. Proses ini melibatkan pemberian token ID yang unik pada setiap elemen dalam teks. Proses pemberian token ID ini terjadi selama pelatihan model BERT, yang menghasilkan ID khusus untuk setiap kata, subkata, dan karakter dalam kosakata. ID ini dipetakan berdasarkan indeks kata dalam kosakata, yang diatur berdasarkan seberapa sering kata tersebut muncul dalam dataset. Semua kata dan subkata perlu diubah menjadi token ID karena model BERT yang telah dilatih sebelumnya hanya dapat memproses ID token. Sebagai contoh, dalam model multibahasa, token [CLS] memiliki indeks 101, diikuti oleh token [SEP] dengan indeks 102, seperti yang ditunjukkan dalam Gambar 3.5. Proses konversi token menjadi ID ini telah ditetapkan sebelumnya, sehingga jika ada kata seperti "Game", akan diberi ID khusus, misalnya 11336. Gambar 3.6 menggambarkan langkah substitusi ID untuk setiap token dalam kalimat.



Gambar 3.6 Tahapan Substitusi Token dengan IDnya

4. Selanjutnya, langkah berikutnya adalah memberikan "*sentence embedding*" pada setiap kalimat untuk membedakan kalimat pertama dari kalimat kedua atau padding. Proses ini dilakukan dengan memberikan nilai angka 1 pada kalimat pertama dan angka 0 pada bagian yang diisi dengan *padding*. Penggunaan tokenizer memungkinkan identifikasi kalimat pertama dan bagian yang di-padding dengan mengacu pada token [SEP], yang berperan sebagai pemisah antara dua kalimat. Untuk memberikan gambaran, pada Gambar 3.7, token-token beserta "*sentence embedding*"nya diilustrasikan, di mana hanya token [PAD] yang memiliki nilai 0 sebagai "*sentence embedding*" untuk mengindikasikan bahwa bagian tersebut merupakan *padding.*

Gambar 3.7 Tahapan *Sentence Embedding*

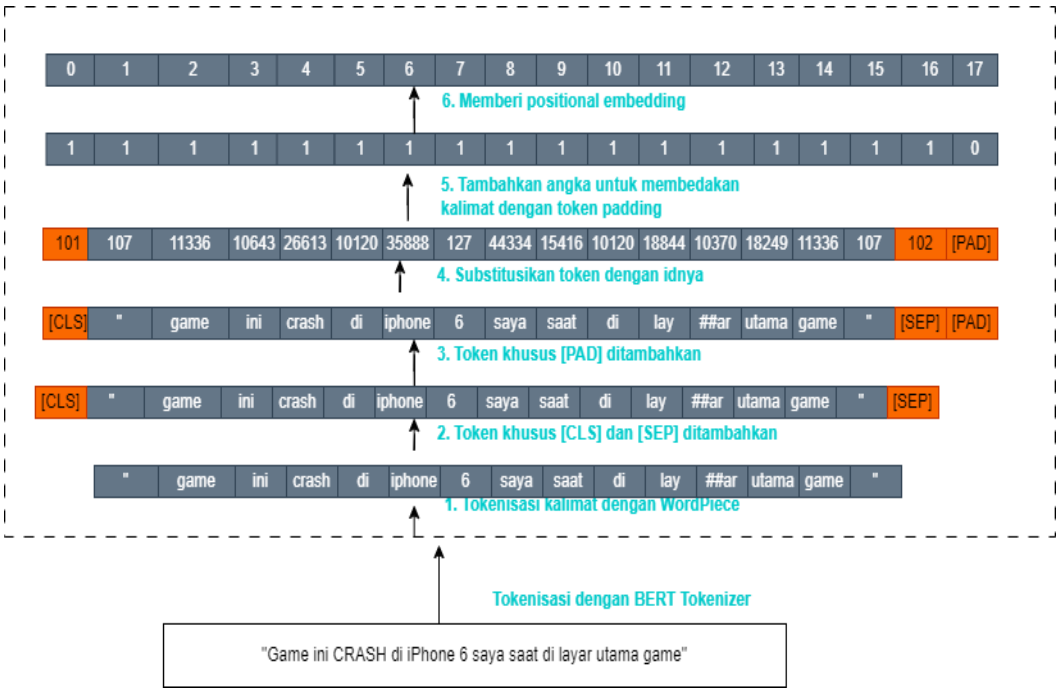
Setelah proses pemberian ID pada setiap token dalam kalimat, tahap selanjutnya adalah memberikan sentence embedding pada kalimat. Hal ini bertujuan untuk membedakan kalimat pertama dari kalimat kedua, serta membedakan antara teks aktual dengan token-token yang berfungsi sebagai padding. Proses ini dapat dilakukan dengan memberikan angka 1 pada kalimat pertama dan angka 0 pada token-token yang berperan sebagai padding. Tokenizer dalam model BERT memiliki kemampuan untuk

mengidentifikasi kalimat pertama dan mengidentifikasi token-token padding berdasarkan token [SEP], yang berfungsi untuk memisahkan dua kalimat atau tanda batas akhir kalimat. Sebagai contoh, dalam Gambar 3.7, token-token beserta sentence embedding-nya ditunjukkan, di mana hanya token [PAD] yang memiliki nilai *sentence embedding* 0 sebagai indikasi bahwa token tersebut merupakan bagian dari padding dalam kalimat. Selanjutnya, mari kita telusuri lebih lanjut bagaimana informasi ini digunakan dalam proses pelatihan model BERT.

5. Selain *sentence embedding*, tahap selanjutnya adalah menambahkan *positional embedding* untuk setiap token guna menunjukkan posisi dari tiap kata dalam kalimat. Proses ini diperlukan karena BERT tidak memiliki informasi mengenai posisi kata-kata dalam kalimat. Dengan adanya *positional embedding*, BERT dapat memahami posisi relatif setiap kata dalam kalimat. Hal ini penting karena meskipun terdapat kata yang sama pada awal dan akhir kalimat, BERT akan memperlakukan kedua token tersebut sebagai entitas yang berbeda, mengingat posisi dan konteksnya berbeda. Gambar 3.8 mengilustrasikan tahap *positional embedding*.

Gambar 3.8 Tahapan *Positional Embedding*

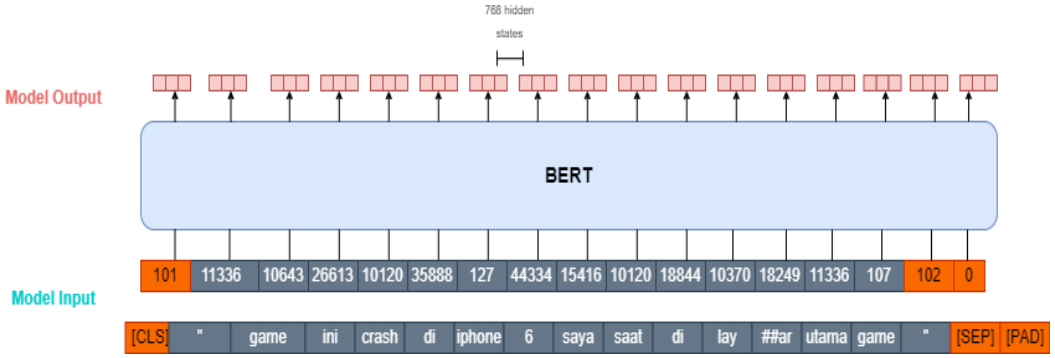
Seluruh proses yang mengubah kalimat menjadi representasi input BERT dapat diikuti pada Gambar 3.9. Tahap demi tahap, kalimat diberikan token-token khusus seperti [CLS] dan [SEP], kemudian di-cocokkan dengan bilangan unik atau ID sesuai dengan *vocabulary* yang telah dibuat. Selain itu, *positional embedding* juga ditambahkan untuk menunjukkan posisi setiap kata dalam kalimat. Semua langkah tersebut berkontribusi dalam menciptakan representasi input yang sesuai dengan BERT, yang dapat diamati pada Gambar 3.10.



Gambar 3.9 Pengerjaan Tokenisasi

Gambar 3.10 Representasi Input dan Output di BERT

BERT menerima urutan kata-kata atau kalimat sebagai input dan mengirimkannya melalui serangkaian tumpukan *encoder*. Setiap *encoder* menerapkan mekanisme *self-attention* pada input dan menghasilkan output yang kemudian diproses oleh *feed-forward network*. Proses ini berlanjut sebanyak 12 kali karena model yang digunakan dalam penelitian ini adalah BERTBASE. Setelah melewati semua *encoder*, setiap token pada setiap posisi akan menghasilkan *output* berupa vektor dengan ukuran tertentu. Pada model BERTBASE, ukuran *hidden size*-nya adalah 768, sebagaimana yang terlihat pada Gambar 3.11. Dalam proses analisis sentimen, *output* yang menjadi perhatian adalah *output* dari posisi pertama, yaitu token [CLS]. Vektor dari token [CLS] tersebut kemudian digunakan sebagai *input* untuk *classifier* yang akan digunakan dalam tugas analisis sentimen.

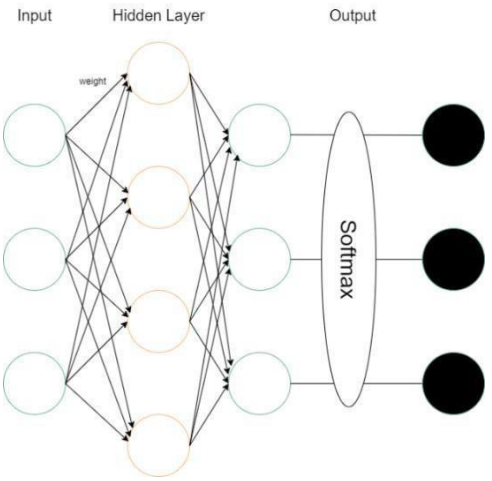


Gambar 3.11 Input dan Output dalam BERT

Pada tahap akhir dalam model BERT, langkah penting yang diperlukan untuk menghasilkan hasil yang akurat adalah penggunaan sebuah *neural network* tunggal sebagai *classifier*. Fungsi utama dari *classifier* ini adalah untuk melakukan klasifikasi teks yang telah dimasukkan ke dalam model. Pada dasarnya, ini berarti mengidentifikasi kategori atau label yang paling sesuai untuk teks yang diberikan. Untuk melakukan klasifikasi ini, model BERT menggunakan sebuah *fully connected neural network* yang disusun dengan cermat. Fungsi *softmax* digunakan pada layer output dari *neural network* ini. Pada tahap ini, output dari BERT yang digunakan untuk klasifikasi diperoleh dari vektor token khusus yang dikenal sebagai token [CLS]. Token [CLS] memiliki peran khusus dalam BERT, karena ia dianggap sebagai hasil pengumpulan rata-rata dari seluruh token kata dalam kalimat, sehingga menghasilkan vektor representasi dari keseluruhan kalimat tersebut. Pada layer terakhir dalam *classifier*, hasilnya berupa logits, yang merupakan prediksi probabilitas kasar untuk klasifikasi kalimat. *Softmax* adalah fungsi yang digunakan untuk mengubah logits ini menjadi probabilitas yang lebih mudah diinterpretasikan. Proses ini melibatkan mengambil eksponen dari tiap nilai logit sehingga total probabilitasnya adalah 1. Probabilitas yang dihasilkan oleh *softmax* ini akan berada dalam rentang antara 0 hingga angka positif. Dengan menggunakan *neural network* tunggal ini dan mengambil representasi dari token [CLS], model BERT dapat memberikan hasil yang kuat dalam tugas klasifikasi teks, yang sangat bergantung pada kemampuan model untuk memahami konteks dan makna dalam kalimat

yang diberikan.

Dalam model BERT, pencapaian yang optimal dapat diperoleh dengan menggunakan sebuah jaringan saraf tunggal sebagai alat klasifikasi. Lapisan yang digunakan untuk melakukan klasifikasi ini adalah jaringan saraf *fully connected* dengan fungsi aktivasi *softmax*, sebagaimana diilustrasikan pada Gambar 3.12. Output yang digunakan dalam klasifikasi berasal dari vektor token [CLS]. Token [CLS] dianggap sebagai hasil penggabungan atau ringkasan dari seluruh token kata dalam kalimat, yang kemudian menghasilkan vektor representasi dari seluruh kalimat tersebut. Pada lapisan terakhir dalam klasifikasi ini, logits dihasilkan, yang merujuk pada prediksi probabilitas kasar untuk mengklasifikasikan kalimat tersebut.



Gambar 3.12 Gambaran *Layer* untuk Analisis Sentimen

Berikut contoh perhitungan di dalam logits, Vektor logits $= \begin{matrix} 5 \\ (7) \\ 0 \end{matrix}$. Vektor tersebut akan diubah menjadi distribusi probabilitas sehingga dapat menghasilkan klasifikasi komentar positif, negatif dan netral. Berikut tahapan yang terjadi dalam mengubah sebuah vektor logits:

1. Hitung seluruh eksponensial dari setiap elemen pada vektor.

   $e^{z1} = e^{5} = 148.4$

   $e^{z2} = e^{7} = 1096.6$

$e^{z3} = e^0 = 1.0$

2. Normalisasi nilai dengan menjumlahkan semua eksponensial.

$$\sum_{j=1}^{k} e^{zj} = 148.4 + 1096.6 + 1.0 = 1246$$

3. Bagi eksponensial dari setiap elemen dengan normalisasi untuk memperoleh *output softmax* dari tiap elemen yang ada.

$$\sigma\,(z)_1 = \frac{148.4}{1246} = 0.1191$$

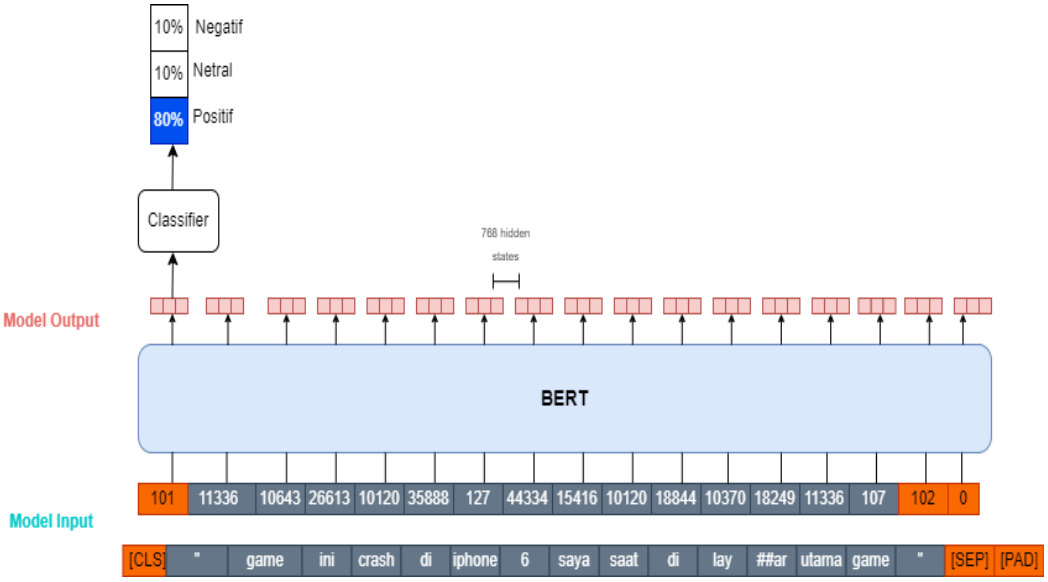
$$\sigma\,(z)_2 = \frac{1096.6}{1246} = 0.8801$$

$$\sigma\,(z)_3 = \frac{1.0}{1246} = 0.0008$$

Untuk memastikan apakah hasil prediksi dari tiap probabilitas adalah 1, maka semua probabilitas dijumlahkan.

$$\sigma\,(z)_1 + \sigma\,(z)_2 + \sigma\,(z)_3 = 0.1191 + 0.8801 + 0.0008 = 1$$

Selain itu, dalam proses klasifikasi menggunakan BERT, juga digunakan *dropout* untuk menghindari *overfitting*. *Overfitting* terjadi ketika model terlalu banyak belajar dari data pelatihan dan tidak dapat melakukan generalisasi dengan baik pada data uji. *Dropout* adalah teknik yang membantu mencegah *overfitting* dengan secara acak mengabaikan beberapa neuron selama pelatihan, sehingga model menjadi lebih *robust* dan dapat lebih umum dalam melakukan prediksi. Ilustrasi proses klasifikasi menggunakan BERT dapat dilihat pada Gambar 3.13.



Gambar 3.13 Gambaran Proses Klasifikasi Menggunakan BERT

Setelah tahap *pre-training* BERT menggunakan dataset yang telah tersedia, langkah selanjutnya adalah melakukan pelatihan (training) model. Namun, mengingat dataset yang digunakan bisa memiliki ukuran yang cukup besar, menyimpan keseluruhan dataset dalam memori secara bersamaan dapat menimbulkan kendala berupa kehabisan memori dan berpotensi menghambat kinerja program. Oleh karena itu, digunakanlah data loader untuk mengatasi permasalahan tersebut dan secara signifikan meningkatkan efisiensi selama proses pelatihan model. Data loader ini memungkinkan dataset dibagi ke dalam tiga bagian yang terpisah, yaitu dataset untuk pelatihan, validasi, dan pengujian (testing).

Seiring dengan penggunaan data loader, setiap bagian dataset dapat dimuat secara bertahap saat dibutuhkan selama proses pelatihan. Hal ini meminimalkan penggunaan memori secara berlebihan dan memastikan bahwa model dapat melakukan training dengan efisien tanpa terkendala oleh masalah memori. Dengan demikian, penggunaan data loader menjadi langkah penting dalam melanjutkan tahap pelatihan model BERT.

DataLoader, sebuah fitur yang terdapat dalam pustaka *PyTorch*, berfungsi sebagai iterator yang sangat berharga dalam proses pelatihan model. *DataLoader* memiliki peran penting dalam mengelola dataset dan mengatur pengambilan

sampel secara iteratif selama proses pelatihan. Dengan menggunakan *DataLoader*, dataset dapat diakses sebagian demi sebagian saat diperlukan, menghasilkan efisiensi dalam penggunaan memori dan mempercepat proses pelatihan secara keseluruhan. Dalam konteks ini, ketika dataset dibagi menjadi tiga bagian yang berbeda, yaitu untuk pelatihan, validasi, dan pengujian, kita juga membutuhkan tiga DataLoader yang sesuai.

Sejalan dengan pendekatan tersebut, optimizer yang digunakan dalam penelitian ini adalah AdamW, seperti yang telah disebutkan dalam paper aslinya oleh Devlin et al. (2019). Optimizer ini memiliki tujuan utama untuk mengoreksi bobot (weight) dalam model guna meningkatkan kualitas kalimat yang dihasilkan. Dengan memadukan penggunaan DataLoader dan optimizer ini, proses pelatihan model BERT dapat dilakukan dengan efisien dan akurat.

Pada tahap *fine-tuning*, model BERT disesuaikan dengan mengatur *hyperparameters*-nya. Sebagian besar *hyperparameters* yang digunakan dalam proses *fine-tuning* BERT tetap sama dengan yang digunakan saat melatih BERT secara umum. Beberapa *hyperparameters* yang signifikan dalam proses pelatihan BERT mencakup:

1. ***Batch size***: Ini adalah jumlah sampel data yang dimasukkan ke dalam jaringan sebelum dilakukan penyesuaian berat (*weight adjustment*). Semakin besar nilai *batch size*, maka waktu yang dibutuhkan untuk menyelesaikan satu batch data akan semakin lama (Osinga, 2018).
2. ***Epoch***: *Epoch* merupakan satu iterasi di mana seluruh dataset diberikan ke dalam jaringan. Selama satu *epoch*, setiap contoh data melewati jaringan untuk proses *forward pass* dan *backward pass*.
3. ***Learning rate***: *Learning rate* menentukan sejauh mana perubahan bobot (*weights*) pada jaringan saraf. Semakin tinggi nilai *learning rate, gradient* (**gradien**) akan bergerak lebih cepat menuju kondisi optimum dalam "*landscape*" fungsi objektif (*objective function*).

Dalam konteks fine-tuning, pengaturan hyperparameters ini perlu disesuaikan dengan teliti untuk mendapatkan hasil yang optimal sesuai dengan tugas yang sedang dihadapi. Selanjutnya, saya akan menjelaskan proses fine-tuning BERT dengan lebih rinci.

Devlin dan timnya memberikan rekomendasi mengenai pengaturan *hyperparameters* yang dapat menghasilkan performa yang optimal dalam tahap *fine-tuning* model BERT untuk tugas-tugas tertentu. Rekomendasi ini mencakup pengaturan *hyperparameters* yang penting untuk mencapai hasil yang optimal saat melakukan *fine-tuning* model BERT untuk tugas-tugas khusus, seperti berikut:

1. ***Batch size***: Rekomendasi adalah menggunakan nilai 16 atau 32 sebagai batch size.
2. ***Learning rate*** (Adam): Learning rate yang direkomendasikan adalah 5e-5, 3e-5, atau 2e-5.
3. ***Epoch***: Rekomendasi mencakup nilai epoch sebanyak 2, 3, atau 4.

Misalnya, jika panjang maksimal urutan kalimat yang akan dianalisis telah ditentukan, misalnya sekitar 10, hasil dari tahap preprocessing yang dilakukan pada BERT untuk mengubah input menjadi format yang dapat diterima oleh BERT dapat ditemukan dalam Tabel 3.2. Rekomendasi ini membantu dalam menentukan konfigurasi hyperparameters yang sesuai untuk tugas fine-tuning yang spesifik.

Tabel 3.2 Input pada BERT

Kalimat	`this game crash on my iphone at the game homescreen`
Kalimat yang ditokenisasi menggunakan WordPiece	`'this', 'game', 'crash', 'on', 'my', 'iphone', 'at', 'the', 'game', 'homes', '##cre', '##en'`

Tabel 3.3 Input pada BERT (lanjutan dari tabel sebelumnya)

Kalimat setelah diberi token khusus [CLS] dan [SEP] dan diberi padding [PAD]	`[CLS]', 'this', 'game', 'crash', 'on', 'my', 'iphone', 'at', 'the', 'game', 'homes', '##cre', '##en', '[SEP]', '[PAD]', '[PAD]'`
ID dari tiap token	`101, 10372, 11336, 26613, 10125, 11153, 35888, 10160, 10103, 11336, 14349, 32839, 10142, 102, 0, 0`

Berdasarkan Tabel 3.3, terdapat kalimat " this game crash on my iphone at the game homescreen ". Selanjutnya, kalimat tersebut akan melalui proses tokenisasi menggunakan WordPiece pada BERT. Jika suatu kata tidak terdapat dalam vocabulary atau Out-of-Vocabulary (OOV), kata tersebut akan dipisah menjadi sub kata (sub words). Sebagai contoh, kata "homescreen" mungkin tidak ada dalam vocabulary, sehingga kata "homescreen" akan dipisah menjadi sub kata "homes", "##cre", dan "##en". Tanda pagar (#) di awal kata menunjukkan bahwa sub kata "##ta" adalah bagian dari kata sebelumnya, yaitu "man". Setelah proses tokenisasi selesai, kalimat tersebut akan ditambahkan dengan token-token khusus seperti [CLS], [SEP], dan [PAD]. Token [PAD] ditambahkan karena panjang kalimat tidak mencapai panjang maksimum yang telah ditetapkan, yaitu 16.

3.6 Evaluasi

Tahap evaluasi bertujuan untuk mengevaluasi hasil analisis sentimen pada kalimat-kalimat dalam dataset. Akurasi tertinggi yang diperoleh selama proses pelatihan model akan menjadi nilai akurasi yang merepresentasikan kinerja model. Untuk mendapatkan prediksi dari model, digunakan metode confusion matrix, yang ditampilkan dalam Tabel 3.4. Confusion matrix digunakan sebagai alat evaluasi utama untuk melihat sejauh mana model dapat mengklasifikasikan sentimen dengan benar. *Confusion matrix* memberikan gambaran yang jelas tentang sejauh mana model berhasil dalam melakukan klasifikasi pada masing-masing kategori sentimen, termasuk *True Positive* (positif yang benar), *True Negative* (negatif yang benar), *False Positive* (positif yang salah), dan *False Negative* (negatif yang salah). Dengan menggunakan confusion matrix ini, kita dapat melakukan evaluasi yang lebih mendalam terhadap kinerja model, menghitung berbagai metrik evaluasi seperti akurasi, presisi, *recall*, serta metrik lainnya yang membantu dalam menilai seberapa baik model ini dalam memahami dan mengklasifikasikan sentimen pada teks yang dianalisis.

Tabel 3.4 Confusion Matrix

		True Class		
		Positive	*Netral*	*Negative*
Predicted Class	*Positive*	*True Positive (TP)*	*False Positive (FNt)*	*False Positive (FP)*
	Neutral	*False Neutral (FNt)*	*True Neutral (TNt)*	*False Neutral (FNt)*
	Negative	*False Negative (FN)*	*False Negative (FNt)*	*True Negative (TN)*

Confusion matrix adalah alat evaluasi yang terdiri dari empat kategori penting, yaitu *True Positive* (TP), *False Positive* (FP), *True Neutral* (TNt), *False Neutral* (FNt), *True Negative* (TN), dan *False Negative* (FN). *True Positive* (TP) terjadi ketika kalimat dengan sentimen positif telah dengan benar diprediksi sebagai sentimen positif oleh model. Sebaliknya, *False Positive* (FP) adalah ketika kalimat dengan sentimen positif diprediksi sebagai sentimen netral atau negatif oleh model, meskipun sebenarnya positif. *True Neutral* (TNt) terjadi ketika kalimat dengan sentimen netral telah dengan tepat diprediksi sebagai sentimen netral oleh model. Di sisi lain, *False Neutral* (FNt) adalah ketika kalimat dengan sentimen netral diprediksi sebagai sentimen positif atau negatif oleh model, padahal sebenarnya netral.

Kemudian, *True Negative* (TN) adalah ketika kalimat dengan sentimen negatif telah dengan benar diprediksi sebagai sentimen negatif oleh model. Terakhir, *False Negative* (FN) terjadi ketika kalimat dengan sentimen negatif diprediksi sebagai sentimen netral atau positif oleh model, padahal sebenarnya negatif. Dengan adanya confusion matrix ini, kita memperoleh pemahaman yang lebih mendalam tentang kinerja model dalam melakukan klasifikasi sentimen untuk masing-masing kategori, yang sangat penting dalam proses evaluasi model.

Setelah mendapatkan nilai confusion matrix, kita dapat menghitung nilai akurasi, presisi (precision), recall, dan F-measure. Nilai akurasi merupakan persentase dari input yang berhasil diprediksi dengan benar oleh neural network. Akurasi akan semakin baik jika nilai loss menurun. Presisi menghitung persentase

input yang diidentifikasi dengan benar oleh sistem, misalnya, sistem memberi label positif pada input yang memang benar-benar positif. Recall menghitung persentase input yang berhasil diidentifikasi dengan benar oleh sistem. Sedangkan F-measure adalah rata-rata harmonik dari presisi dan recall.

Rumus untuk menghitung nilai akurasi dan F-measure diberikan dalam Persamaan 3.1 dan Persamaan 3.4. Sementara itu, rumus untuk menghitung presisi dan recall dihitung untuk setiap kategori sentimen, seperti yang dijelaskan pada Persamaan 3.2 dan 3.3. Evaluasi ini membantu kita dalam menilai kinerja model dan efektivitasnya dalam melakukan klasifikasi sentimen pada dataset yang ada. *Precision, recall,* dan *F-measure* sangat penting dalam evaluasi model BERT karena mereka memberikan informasi tentang keakuratan, kelengkapan, dan keseimbangan model dalam mengklasifikasikan sentimen atau teks lainnya. Dengan menggunakan ketiga metrik ini, kita dapat memahami secara lebih baik sejauh mana model BERT dapat melakukan tugas klasifikasi dengan akurasi dan kelengkapan yang diinginkan.

$$Accuracv = \frac{TP + TN}{TP + FP + TN + FN} \quad \text{.....(3.1)}$$

$$Precision_{\ Positive} = \frac{TP}{TP+FP} \quad \text{.....(3.2)}$$

$$Recall_{Positive} = \frac{TP}{TP + FN} \quad \text{.....(3.3)}$$

$$F - measure = 2 * \frac{Precision * Recall}{Precision + Recall} \quad \text{.....(3.4)}$$

BAB 4
IMPLEMENTASI DAN PENGUJIAN

Bab implementasi dan pengujian dalam konteks analisis sentimen menggunakan algoritma BERT membahas langkah-langkah yang dilakukan untuk menerapkan model BERT dalam memprediksi sentimen pada teks atau kalimat.

4.1 Implementasi Sistem

4.1.1 Spesifikasi Perangkat Keras

Untuk dapat menerapkan dan menguji analisis sentimen pada review Aplikasi PUBG Mobile pada *AppStore*, diperlukan perangkat keras dengan spesifikasi sebagai berikut.:

1. Processor Intel® Core™ i7-9750H
2. Memory RAM 8 GB
3. SSD dengan kapasitas 500 GB
4. GPU GTX 1650

Untuk memenuhi kebutuhan perangkat keras yang dibutuhkan, penulis juga menggunakan layanan cloud gratis dari Google yang disebut Google Colab. Di dalam Google Colab, penulis memanfaatkan GPU seperti Tesla sebagai akselerator perangkat keras untuk membantu proses implementasi dan pengujian. Alasan mengadopsi pendekatan ini adalah karena penulis bekerja dengan model yang telah dilatih sebelumnya yang memiliki ukuran yang sangat besar.

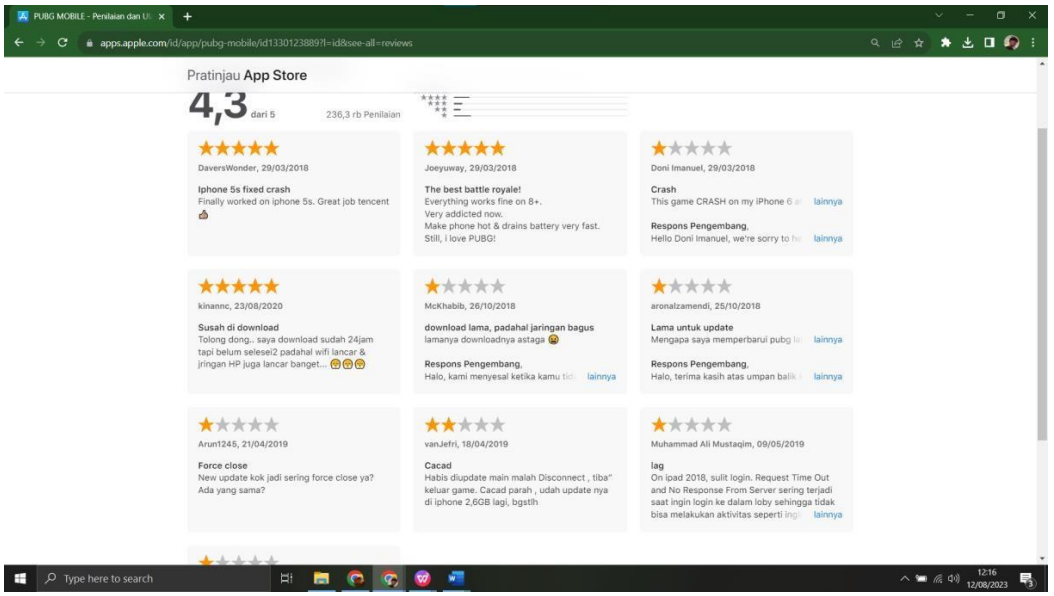
4.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 11 Home 64 *bit operating system.*
2. *Google Colab*
3. *Python*3
4. *Library*: torch, *transformer*s, numpy, nltk, matplotlib, re, sklearn, seaborn,keras ,Sastrawi.

4.2 Implementasi *Web Scraping*

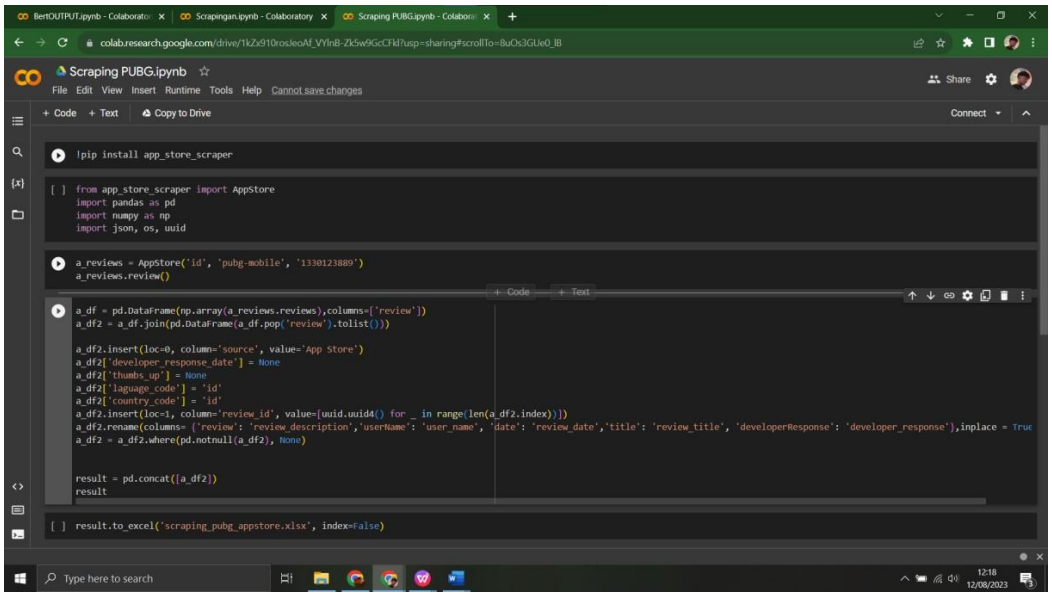
Hal pertama yang untuk melakukan implementasi *Web Scraping* penulis akan membuka URL PUBGM pada website resmi *APP Store* seperti berikut *https://apps.apple.com/id/app/pubg-mobile/id1330123889?l=id* dan akan muncul beberapa review atau komentar pada halaman tersebut yang dapat dilihat pada Gambar 4.1. Lalu penulis melakukan *scraping* data di *Google Colab* dengan memasukkan url *web* sebagai sumber untuk dilakukannya *scraping* data yang dapat dilihat pada Gambar 4.2. lalu setelah dilakukan *scraping* penulis men-download hasil data yang sudah di *scraping* yang dapat dilihat pada Gambar 4.3. Komentar yang diambil berjumlah 3715 data.



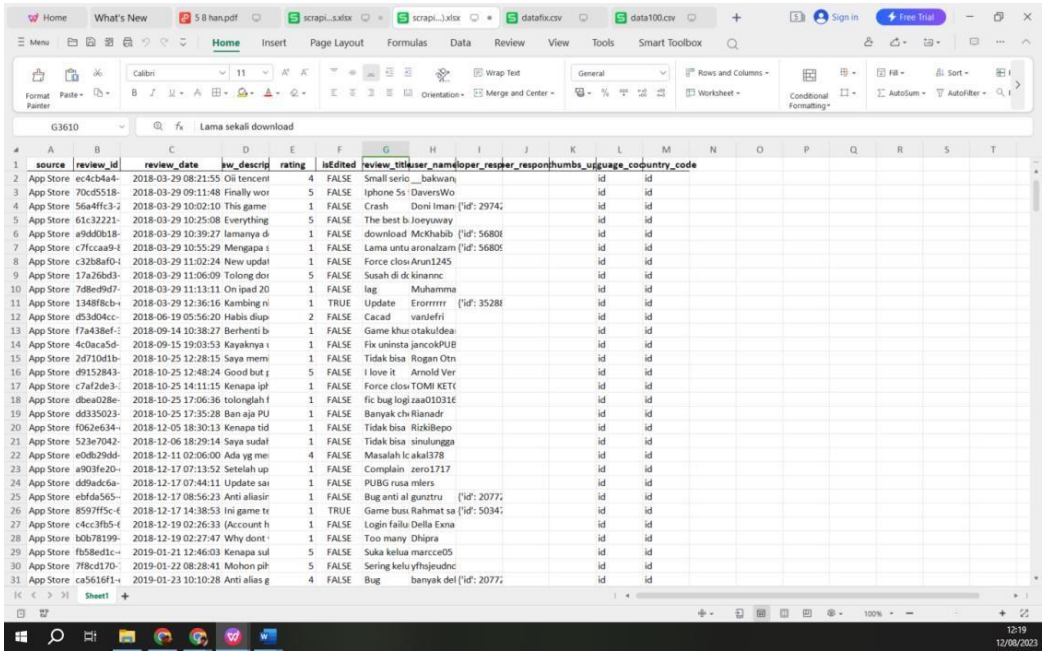
Gambar 4.1 Web yang Dipakai untuk Scraping Data

Setelah berhasil melakukan scraping data dari halaman aplikasi PUBG Mobile di website resmi *App Store*, langkah berikutnya adalah melakukan pengolahan data di Google Colab. Di sana, penulis memasukkan URL web sebagai sumber data yang akan di-scrape, seperti yang terlihat pada Gambar 4.2. Proses ini memungkinkan penulis untuk mengakses data dari website dan mengekstrak komentar atau ulasan yang diperlukan untuk analisis sentimen. Setelah berhasil melakukan scraping data, langkah selanjutnya adalah mendownload hasil data yang telah diambil dari website, sebagaimana terlihat pada Gambar 4.3. Dalam proses ini, penulis berhasil mengumpulkan sejumlah besar data komentar, yang berjumlah ribuan data.

Dengan demikian, proses scraping dan pengolahan data ini menjadi langkah awal dalam mempersiapkan dataset yang diperlukan untuk analisis sentimen terhadap aplikasi PUBG Mobile. Selanjutnya, mari kita telaah lebih lanjut mengenai langkah-langkah selanjutnya dalam penelitian ini.



Gambar 4.2 Proses Scraping Memakai *App Store* Scraper



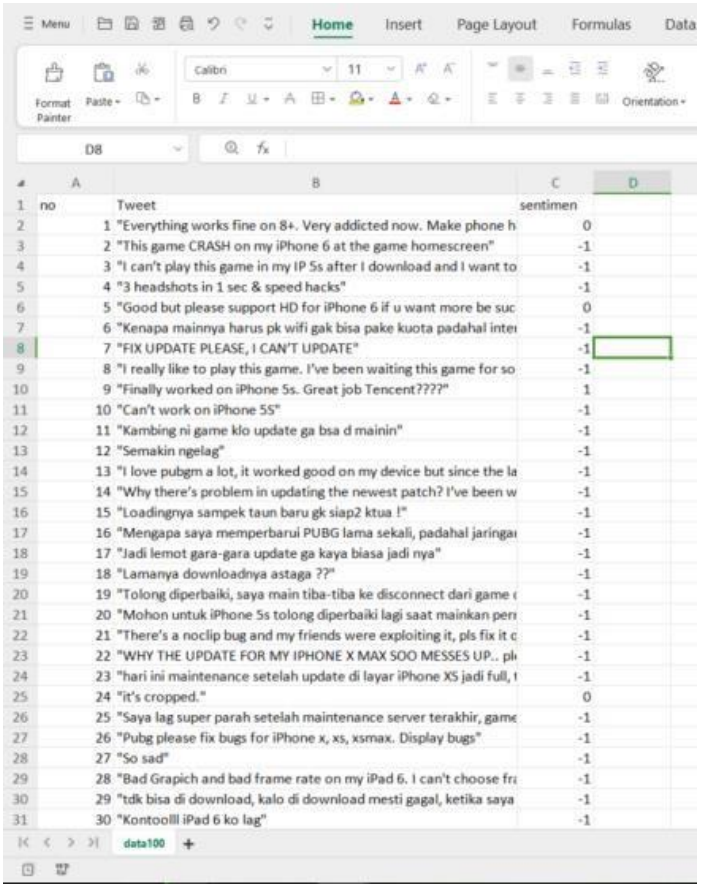
Gambar 4.3 Dataset Hasil Scraping

4.3 Labelisasi *Dataset*

Tahap labelisasi *dataset* dalam algoritma BERT adalah mengganti label sentimen positif, negatif, dan netral dengan angka yang sesuai. Dalam kasus ini, sentimen

positif akan diganti menjadi angka 1, sentimen negatif menjadi angka -1, dan sentimen netral menjadi angka 0. Jumlah komentar dalam *dataset* yang digunakan adalah 3715 komentar.

Dalam tahap labelisasi ini, setiap komentar pada *dataset* akan diperiksa label sentimennya. Jika komentar memiliki sentimen positif maka label nya ialah 1. Jika komentar memiliki sentimen negatif maka labelnya akan diganti menjadi -1. Sedangkan jika komentar memiliki sentimen netral, labelnya akan diganti menjadi 0. Proses ini dilakukan untuk setiap komentar dalam *dataset* dapat dilihat pada Gambar 4.4. Setelah labelisasi *dataset* selesai, *dataset* siap digunakan untuk proses *preprocessing dataset*.



no	Tweet	sentimen
1	"Everything works fine on 8+. Very addicted now. Make phone h	0
2	"This game CRASH on my iPhone 6 at the game homescreen"	-1
3	"I can't play this game in my IP 5s after I download and I want to	-1
4	"3 headshots in 1 sec & speed hacks"	-1
5	"Good but please support HD for iPhone 6 if u want more be suc	0
6	"Kenapa mainnya harus pk wifi gak bisa pake kuota padahal inte	-1
7	"FIX UPDATE PLEASE, I CAN'T UPDATE"	-1
8	"I really like to play this game. I've been waiting this game for so	-1
9	"Finally worked on iPhone 5s. Great job Tencent????"	1
10	"Can't work on iPhone 5S"	-1
11	"Kambing ni game klo update ga bsa d mainin"	-1
12	"Semakin ngelag"	-1
13	"I love pubgm a lot, it worked good on my device but since the la	-1
14	"Why there's problem in updating the newest patch? I've been w	-1
15	"Loadingnya sampek taun baru gk siap2 ktua !"	-1
16	"Mengapa saya memperbarui PUBG lama sekali, padahal jaringa	-1
17	"Jadi lemot gara-gara update ga kaya biasa jadi nya"	-1
18	"Lamanya downloadnya astaga ??"	-1
19	"Tolong diperbaiki, saya main tiba-tiba ke disconnect dari game	-1
20	"Mohon untuk iPhone 5s tolong diperbaiki lagi saat mainkan per	-1
21	"There's a noclip bug and my friends were exploiting it, pls fix it c	-1
22	"WHY THE UPDATE FOR MY IPHONE X MAX SOO MESSES UP.. pl	-1
23	"hari ini maintenance setelah update di layar iPhone XS jadi full, t	-1
24	"it's cropped."	0
25	"Saya lag super parah setelah maintenance server terakhir, game	-1
26	"Pubg please fix bugs for iPhone x, xs, xsmax. Display bugs"	-1
27	"So sad"	-1
28	"Bad Grapich and bad frame rate on my iPad 6. I can't choose fr	-1
29	"tdk bisa di download, kalo di download mesti gagal, ketika saya	-1
30	"Kontoolll iPad 6 ko lag"	-1

Gambar 4.4 Hasil Dataset yang Sudah Dilabelisasi

4.4 Preprocessing Dataset

Tahap preprocessing dalam penelitian ini melibatkan beberapa proses utama. Pertama, terdapat langkah case folding, yang dapat dilihat dalam detail pada Gambar 4.5. Selanjutnya, ada proses Data Cleaning yang dijelaskan dengan rinci

pada Gambar 4.6. Tahap berikutnya adalah tokenisasi, yang dicontohkan pada Gambar 4.7. Dilanjutkan dengan tahap penghapusan stopwords, yang terperinci pada Gambar 4.8. Terakhir, dalam proses normalisasi, penelitian mengikuti pedoman yang tertuang dalam Gambar 4.9. Semua tahapan ini memiliki peran penting dalam mempersiapkan data untuk analisis sentimen yang akurat. Karena data yang sebelumnya 3715 data menjadi 2526 akibat melalui tahap *preprocessing.*

Selanjutnya, data yang telah melewati berbagai tahap *preprocessing* yang disebutkan sebelumnya telah siap untuk diolah lebih lanjut dalam analisis sentimen. Tahap preprocessing ini penting untuk memastikan kualitas data yang baik sebelum masuk ke dalam model analisis sentimen, sehingga hasil analisis yang dihasilkan dapat lebih akurat dan relevan dengan tujuan penelitian. Mari lanjutkan untuk melihat bagaimana tahap analisis sentimen dilakukan terhadap data yang telah diproses dengan baik ini.

4.4.1 Case Folding

Pada tahap ini Case folding dilakukan dengan membuat semua huruf besar (uppercase) pada dataset menjadi huruf kecil (lowercase). Tahap ini dilakukan agar semua karakter pada dataset menjadi sama, yaitu menggunakan huruf kecil.

	Tweet	review_text
0	"Everything works fine on 8+. Very addicted no...	everything works fine on very addicted now ma...
1	"This game CRASH on my iPhone 6 at the game ho...	this game crash on my iphone at the game home...
2	"I can't play this game in my IP 5s after I do...	i can t play this game in my ip s after i dow...
3	"3 headshots in 1 sec & speed hacks"	headshots in sec speed hacks
4	"Good but please support HD for iPhone 6 if u ...	good but please support hd for iphone if u wa...
...	...	...
3710	Makin kesini makin aneh ini game	makin kesini makin aneh ini game
3711	Sudah log in tidak mau masuk	sudah log in tidak mau masuk
3712	Lama bener internet kenceng... masa ia download ...	lama bener internet kenceng masa ia download h...
3713	Abis evend ini bikin evend tranformens??	abis evend ini bikin evend tranformens
3714	"SORRY I GIVE 1 STAR BECAUSE THERE ARE STILL M...	sorry i give star because there are still man...

Gambar 4.5 Output Case Folding

4.4.2 Data Cleaning

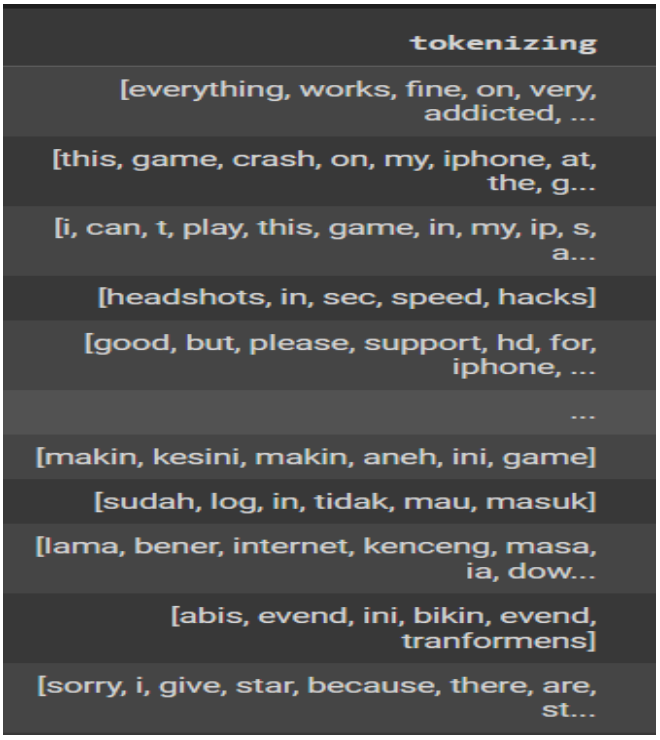
Pada tahap ini, kalimat-kalimat pada dataset dibersihkan dari segala sesuatu yang dapat mempengaruhi hasil dari analisis seperti kata dengan karakter yang berulang dua atau lebih, link, username (@username), hashtag (#), angka, simbol-simbol, spasi berlebih, tanda baca, dan angka.

	Tweet	review_text
0	"Everything works fine on 8+. Very addicted no...	everything works fine on very addicted now ma...
1	"This game CRASH on my iPhone 6 at the game ho...	this game crash on my iphone at the game home...
2	"I can't play this game in my IP 5s after I do...	i can t play this game in my ip s after i dow...
3	"3 headshots in 1 sec & speed hacks"	headshots in sec speed hacks
4	"Good but please support HD for iPhone 6 if u ...	good but please support hd for iphone if u wa...
...	...	...
3710	Makin kesini makin aneh ini game	makin kesini makin aneh ini game
3711	Sudah log in tidak mau masuk	sudah log in tidak mau masuk
3712	Lama bener internet kenceng... masa ia download ...	lama bener internet kenceng masa ia download h...
3713	Abis evend ini bikin evend tranformens??	abis evend ini bikin evend tranformens
3714	"SORRY I GIVE 1 STAR BECAUSE THERE ARE STILL M...	sorry i give star because there are still man...

Gambar 4.6 Output Data Cleaning

4.4.3 Tokenizing

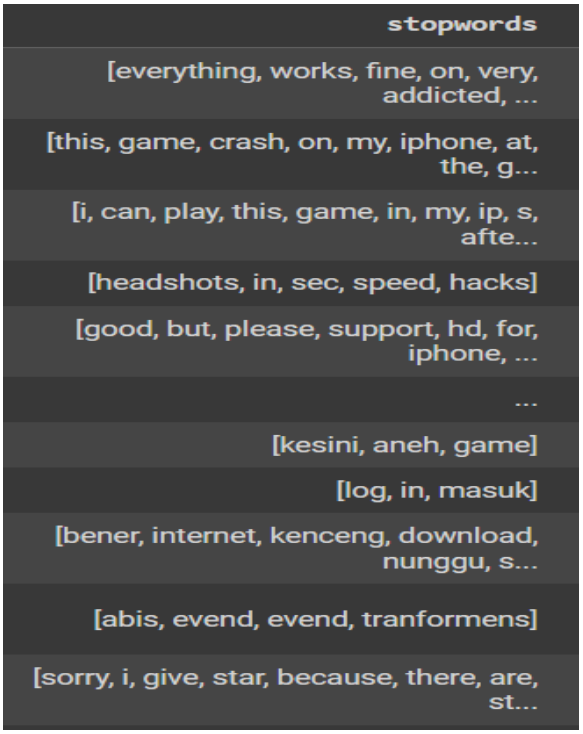
Pada *tokenizing* proses yang dilakukan untuk memecah kalimat-kalimat menjadi potongan kata-kata, tanda baca, dan ekspresi bermakna lainnya sesuai dengan ketentuan Bahasa yang digunakan.



Gambar 4.7 Output Tokenizing

4.4.4 Stopwords Removal

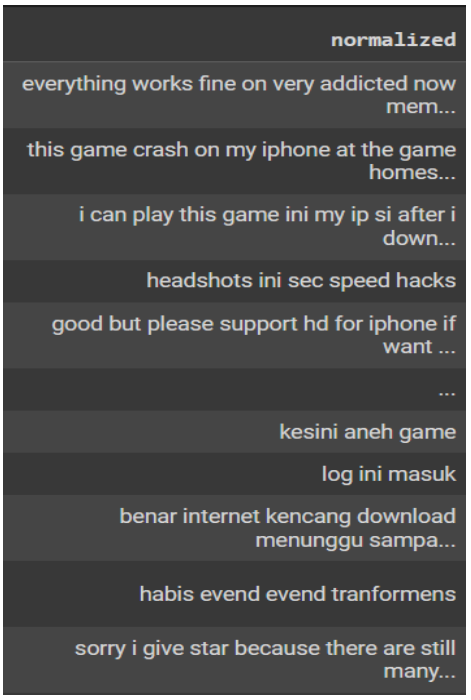
Stopwords Removal adalah proses yang dilakukan untuk menghapus kata-kata yang tidak memiki arti.



Gambar 4.8 Output Stopwords Removal

4.4.5 Normalisasi

Pada tahap normalisasi dataset, data yang tidak konsisten akan disesuaikan ke dalam format seragam agar dapat diolah dengan lebih mudah. Normalisasi dilakukan untuk menghapus variasi dan perbedaan yang tidak relevan dalam teks untuk analisis sentimen, seperti tanda baca, perbedaan huruf besar dan kecil, serta kata-kata yang tidak memiliki signifikansi dalam konteks sentimen.



Gambar 4.9 Output Normalisasi Menggunakan Kamus

4.5 Split Dataset

Sebelum melakukan proses klasifikasi, dataset dibagi menjadi tiga kelompok utama: data latih (training), data validasi, dan data uji (testing). Data latih digunakan untuk melatih model, sementara data validasi berperan dalam menghindari overfitting yang mungkin terjadi pada jaringan saraf tiruan. Terakhir, data uji digunakan sebagai tahap pengujian akhir untuk mengevaluasi sejauh mana jaringan yang telah dilatih dengan data latih dapat memberikan hasil yang akurat. Tahap pembagian dataset ini diilustrasikan dengan jelas dalam Gambar 4.10. Dengan pembagian dataset yang tepat, kita dapat mengoptimalkan kinerja model dan mengukur keefektifannya dalam mengklasifikasikan sentimen pada data yang belum pernah dilihat sebelumnya.

```
train_input, test_input, train_labels, test_labels = train_test_split(input_ids,
                                                                      labels,
                                                                      random_state=2017,
                                                                      test_size=0.2)
train_mask, test_mask, _, _ = train_test_split(attention_mask,
                                               labels,
                                               random_state=2017,
                                               test_size=0.2)

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_mask,
                                                     random_state=2018,
                                                     test_size=0.15)
```

Gambar 4.10 Proses Splitting Dataset

Dengan dataset yang telah terbagi dengan baik menjadi data latih, data validasi, dan data uji, kita dapat melanjutkan ke tahap klasifikasi sentimen. Langkah ini memungkinkan kita untuk mengggunakan model yang telah dilatih pada data latih untuk mengidentifikasi sentimen dalam data baru dengan tingkat akurasi yang tinggi. Selanjutnya, mari kita telusuri bagaimana model ini akan diterapkan dan hasil evaluasinya terhadap data uji dalam penelitian ini.

4.6 Implementasi BERT

Untuk melatih model, diperlukan penggunaan data loader yang akan memungkinkan iterasi pada setiap *dataset*. Hal ini bertujuan untuk mengelola penggunaan memori saat proses pelatihan, sehingga tidak perlu memuat seluruh *dataset* ke dalam memori secara bersamaan. Selain itu, dibutuhkan juga blok-blok yang dibuat untuk membentuk Data Loader yang akan menghasilkan komentar-komentar yang telah ditokenisasi. Komentar dan sentimen memiliki panjang maksimum sebesar 400 kata. Untuk analisis sentimen, lapisan tambahan digunakan. Lapisan tersebut menggunakan *Dropout* dengan probabilitas 0.1, mengikuti rekomendasi pada penelitian BERT. Penulis juga melakukan *fine-tuning* menggunakan beberapa *hyperparameter* berikut, yang diambil berdasarkan rekomendasi untuk BERT:

1. *Epoch*: 7
2. *Batch Size: 16*
3. *Learning rate*: 2e-5

Penulis melakukan pemilihan *hyperparameter* di atas berdasarkan beberapa pertimbangan. *Batch size* dengan nilai 16 dipilih karena semakin kecil *batch size* maka waktu untuk menyelesaikan satu *batch* juga semakin lama. Selain itu, penggunaan *learning rate* sebesar 2e-5 dipilih karena dapat membantu BERT dalam mengatasi masalah catastrophic forgetting. *Catastrophic forgetting* (penghapusan yang tragis) adalah isu yang muncul dalam pembelajaran mesin ketika model yang sudah terlatih untuk memahami suatu tugas lupa atau menghapus pengetahuannya tentang tugas sebelumnya ketika sedang mempelajari tugas yang baru. Permasalahan ini umumnya terjadi dalam konteks *transfer learning*, di mana model yang telah dipersiapkan dengan dataset awal hendak digunakan untuk tugas yang berbeda.

Penulis melakukan eksperimen menggunakan tiga variasi jumlah *epoch* untuk menentukan jumlah *epoch* yang optimal. Gambar 4.11 menampilkan akurasi yang dihasilkan dengan 3 *epoch*, sementara Gambar 4.12 menampilkan akurasi dengan 5 *epoch*. Gambar 4.13 menunjukkan akurasi dengan 7 *epoch*.

	precision	recall	f1-score	support
0	0.44	0.24	0.31	85
1	0.79	0.65	0.71	100
2	0.87	0.96	0.91	558
accuracy			0.84	743
macro avg	0.70	0.62	0.65	743
weighted avg	0.81	0.84	0.82	743

Gambar 4.11 Akurasi dengan Diperoleh dengan 3 Epoch

	precision	recall	f1-score	support
0	0.34	0.28	0.31	85
1	0.80	0.67	0.73	100
2	0.88	0.93	0.90	558
accuracy			0.82	743
macro avg	0.67	0.63	0.65	743
weighted avg	0.81	0.82	0.81	743

Gambar 4.12 Akurasi yang Diperoleh dengan 5 Epoch

	precision	recall	f1-score	support
0	0.35	0.28	0.31	85
1	0.85	0.63	0.72	100
2	0.88	0.94	0.91	558
accuracy			0.83	743
macro avg	0.69	0.62	0.65	743
weighted avg	0.81	0.83	0.82	743

Gambar 4.13 Akurasi yang Diperoleh dengan 7 Epoch

Dalam penelitian ini, penulis menggunakan 7 *epoch* untuk melakukan analisis sentiment. Pada Gambar 4.14 menggambarkan proses *training* yang dilakukan dalam percobaan pertama.

```
======= Epoch 1 / 3 =======
Training...
Batch    40 of   158.     Elapsed: 0:00:44
Batch    80 of   158.     Elapsed: 0:01:27
Batch   120 of   158.     Elapsed: 0:02:12
   Average training loss: 0.62
   Training epoch took: 0:02:53
Running Validation...
   Accuracy: 0.82
   Validation took: 0:00:11
======= Epoch 2 / 3 =======
Training...
Batch    40 of   158.     Elapsed: 0:00:44
Batch    80 of   158.     Elapsed: 0:01:28
Batch   120 of   158.     Elapsed: 0:02:12
   Average training loss: 0.42
   Training epoch took: 0:02:54
Running Validation...
   Accuracy: 0.84
   Validation took: 0:00:11
======= Epoch 3 / 3 =======
Training...
Batch    40 of   158.     Elapsed: 0:00:44
Batch    80 of   158.     Elapsed: 0:01:28
Batch   120 of   158.     Elapsed: 0:02:12
   Average training loss: 0.30
   Training epoch took: 0:02:54
Running Validation...
   Accuracy: 0.84
   Validation took: 0:00:11
Training complete!
```

Gambar 4.14 Proses *Training* dan Evaluasi Dataset pada Percobaan Pertama.

Klasifikasi *User Experience* pada penelitian ini dilakukan dengan melakukan labelisasi secara manual pada tiap komentar yang ada. Setiap aspek memiliki *dataset*nya masing-masing kemudian akan dilakukan *training* pada *dataset* yang sudah di labelisasi. Karena aspek di penelitian ini ada 4, maka *training* model pada BERT-nya juga dilakukan 4 kali sehingga dapat menghasilkan 4 model yang berbeda dari tiap aspek. Setelah dilakukan traning setiap model aspek yang sudah di *amount* ke *drive* tadi di panggil ke dalam satu kodingan utama sehingga dapat menghasilkan *multiclass classification* pada aspek user experience

Untuk melakukan *training* pada masing-masing aspek penulis menggunakan *dataset* yang sudah dipisah berdasarkan masing-masing aspek pada Gambar 4.15, Gambar 4.16, Gambar 4.17, Gambar 4.18 akan ditunjukkan masing-masing akurasi dari setiap pelatihan model pada aspek.

	precision	recall	f1-score	support
0	0.99	1.00	0.99	463
1	1.00	0.88	0.94	42
accuracy			0.99	505
macro avg	0.99	0.94	0.97	505
weighted avg	0.99	0.99	0.99	505

Gambar 4.15 Akurasi Dari Aspek Grafis

	precision	recall	f1-score	support
0	0.93	0.98	0.95	381
1	0.91	0.77	0.84	124
accuracy			0.93	505
macro avg	0.92	0.88	0.90	505
weighted avg	0.93	0.93	0.92	505

Gambar 4.16 Akurasi Dari Aspek Performa

	precision	recall	f1-score	support
0	0.97	0.99	0.98	243
1	0.99	0.97	0.98	262
accuracy			0.98	505
macro avg	0.98	0.98	0.98	505
weighted avg	0.98	0.98	0.98	505

Gambar 4.17 Akurasi Dari Aspek Jaringan

	precision	recall	f1-score	support
0	0.96	1.00	0.98	202
1	1.00	0.97	0.98	303
accuracy			0.98	505
macro avg	0.98	0.99	0.98	505
weighted avg	0.98	0.98	0.98	505

Gambar 4.18 Akurasi Dari Aspek Gameplay

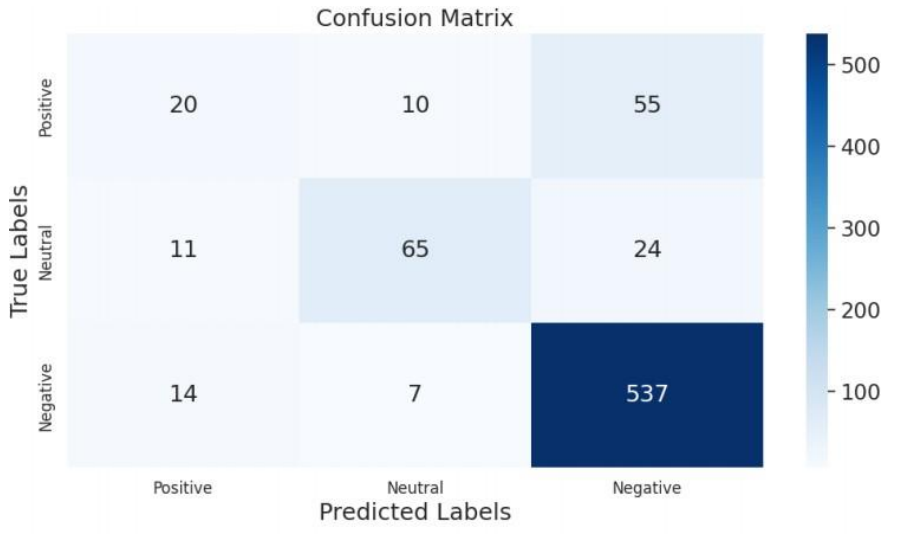
Pada Gambar 4.19 merupakan contoh sebuah kalimat “Setelah update tidak dapat bermain dengan 60 fps” yang diklasifikasikan ke dalam sentiment positif, negative atau netral dan juga berdasarkan aspeknya.

```
Sentimen negatif
and included in
---
Aspek Jaringan
---
Aspek grafis
```

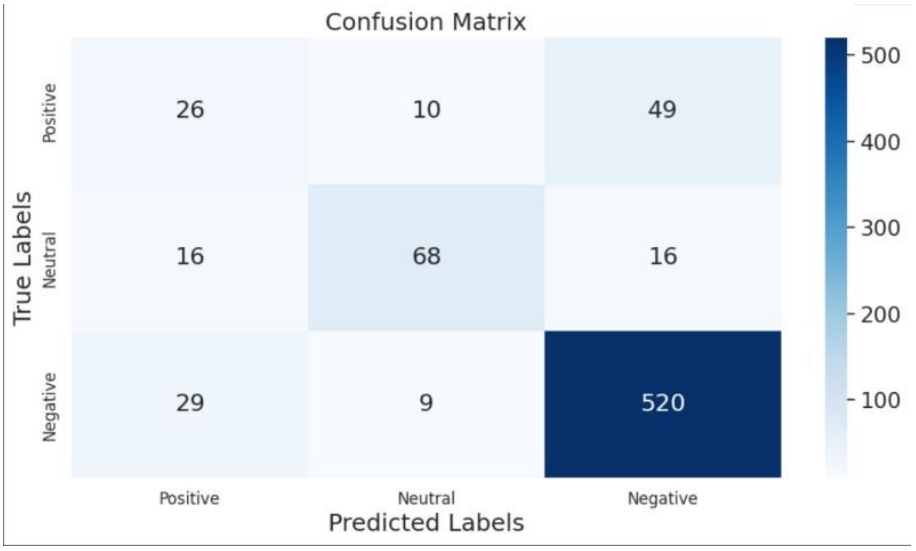
Gambar 4.19 Contoh Klasifikasi sebuah kalimat

4.7 Evaluasi

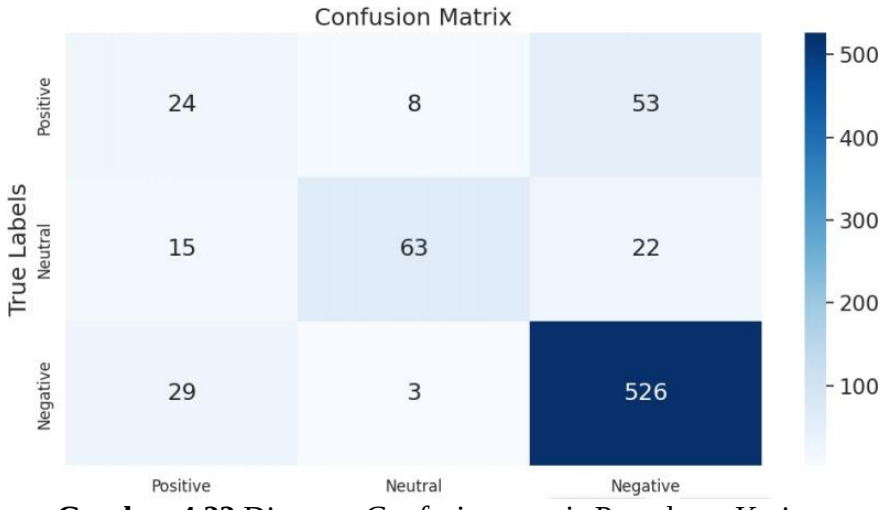
Setelah melakukan proses *training* pada BERT, model dengan akurasi terbaik diuji menggunakan *dataset testing*. Setelah memperoleh hasil akurasi, model kemudian digunakan untuk memprediksi menggunakan *dataset testing*. Gambar 4.20, Gambar 4.21, dan Gambar 4.22 secara berurutan menampilkan hasil diagram *confusion matrix*.



Gambar 4.20 Diagram *Confusion matrix* Percobaan Pertama

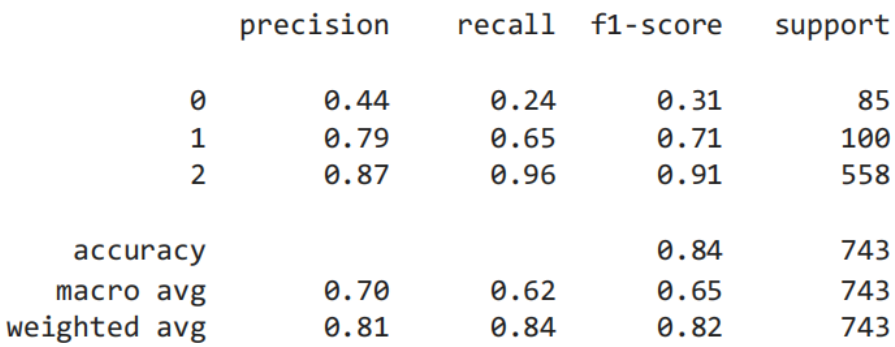


Gambar 4.21 Diagram *Confusion matrix* Percobaan Kedua



Gambar 4.22 Diagram *Confusion matrix* Percobaan Ketiga

Sistem menghadapi tantangan dalam mengklasifikasikan sentimen komentar-komentar. Hasil dari diagram *confusion matrix* dalam tiga percobaan menunjukkan bahwa sistem mengalami kesulitan dalam mengategorikan komentar yang memiliki sentimen positif dan netral. Namun, sebaliknya, sistem berhasil dengan baik dalam mengidentifikasi komentar yang memiliki sentimen negatif. Salah satu penyebab utama masalah ini adalah ketidakseimbangan jumlah komentar di antara tiga kategori sentimen, yaitu negatif, netral, dan positif. Terdapat ketidakproporsionalan yang signifikan antara jumlah komentar yang bersentimen negatif dibandingkan dengan jumlah komentar yang memiliki sentimen positif dan netral. Kondisi ini memiliki dampak pada kinerja model dalam mengklasifikasikan sentimen dengan akurasi yang diinginkan. Dari hasil diagram *confusion matrix* dalam tiga percobaan yang dilakukan, terlihat bahwa sistem menghadapi kesulitan dalam mengklasifikasikan komentar yang memiliki sentimen positif dan netral. Namun, sistem berhasil dengan baik dalam mengidentifikasi komentar yang memiliki sentimen negatif. Penyebab utama masalah ini adalah ketidakseimbangan jumlah komentar di antara ketiga kategori sentimen, yaitu negatif, netral, dan positif. Terdapat ketidakproporsionalan yang signifikan antara jumlah komentar yang bersentimen negatif dibandingkan dengan jumlah komentar yang memiliki sentimen positif dan netral. Hal ini dapat memengaruhi kinerja model dalam mengklasifikasikan sentimen dengan benar. Untuk mengetahui tingkat akurasi tiap percobaan dapat dilihat pada gambar 4.23, 4.24 dan 4.25.



	precision	recall	f1-score	support
0	0.44	0.24	0.31	85
1	0.79	0.65	0.71	100
2	0.87	0.96	0.91	558
accuracy			0.84	743
macro avg	0.70	0.62	0.65	743
weighted avg	0.81	0.84	0.82	743

Gambar 4.23 *Accuracy* Dari Percobaan Pertama

	precision	recall	f1-score	support
0	0.34	0.28	0.31	85
1	0.80	0.67	0.73	100
2	0.88	0.93	0.90	558
accuracy			0.82	743
macro avg	0.67	0.63	0.65	743
weighted avg	0.81	0.82	0.81	743

Gambar 4.24 *Accuracy* Dari Percobaan Kedua

	precision	recall	f1-score	support
0	0.35	0.28	0.31	85
1	0.85	0.63	0.72	100
2	0.88	0.94	0.91	558
accuracy			0.83	743
macro avg	0.69	0.62	0.65	743
weighted avg	0.81	0.83	0.82	743

Gambar 4.25 *Accuracy* Dari Percobaan Ketiga

Setelah menguji dataset, ditemukan bahwa akurasi secara keseluruhan menggunakan BERT pada percobaan pertama, kedua, dan ketiga adalah 84%, 82%, dan 83% secara berturut-turut. Berdasarkan data diatas membuktikan tidak selalu harus epoch banyak agar akurasi meningkat. Sehingga rata-rata akurasi dari analisis sentimen dengan BERT adalah 83%. Perbedaan akurasi sistem dipengaruhi oleh pengacakan dataset saat pembagian menjadi dataset training, testing, dan evaluasi pada setiap percobaan.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Hasil pengujian implementasi model Bahasa BERT dalam analisis sentimen menunjukkan beberapa kesimpulan sebagai berikut

1. Berdasarkan hasil analisis, dapat disimpulkan bahwa para pemain PUBG *Mobile* (PUBGM) di Indonesia masih banyak yang mengungkapkan tentang aspek performa.
2. Hasil analisis sentimen menggunakan metode *Bidirectional Encoder Representations from Transformers* (BERT) menghasilkan akurasi sebesar 84%, 82%, dan 83% dalam tiga percobaan yang berbeda. Pada percobaan ini, digunakan *hyperparameters* seperti *batch size* sebesar 16, *learning rate* sebesar 2e-5, dan *epoch* dengan variasi 3, 5, dan 7.
3. Dari hasil pengujian dengan tiga setting *epoch* yang berbeda yaitu 3, 5, dan 7, ditemukan bahwa *epoch* 3 memberikan hasil yang paling baik dalam analisis sentimen. Hal ini menunjukkan bahwa lebih banyak *epoch* tidak selalu menghasilkan akurasi yang lebih tinggi.
4. Dapat disimpulkan bahwa kualitas dan jumlah data dalam dataset sangat memengaruhi akurasi yang diperoleh saat mengimplementasikan model BERT. Semakin baik kualitas dataset, maka tingkat akurasi cenderung meningkat. Selain itu, penambahan jumlah data berkualitas juga berkontribusi pada peningkatan akurasi dalam analisis sentimen.

5.2 Saran

Adapun saran yang dapat dipertimbangkan untuk penelitian selanjutnya adalah:

1. Dataset yang digunakan dalam analisis sentimen menggunakan model BERT memiliki distribusi yang seimbang antara sentimen positif, negatif, dan netral. Ini memastikan bahwa model memiliki cukup data untuk melatih dan menghasilkan hasil yang seimbang dalam mengklasifikasikan berbagai jenis sentimen.
2. Untuk mendapatkan hasil yang lebih baik dalam analisis sentimen, dapat

dipertimbangkan untuk menggunakan model BERT yang telah dioptimalkan untuk bahasa Indonesia seperti IndoBERT atau IndoBERT . Kombinasi penggunaan model BERT yang sesuai dengan bahasa target dan dataset yang lebih besar dan seimbang dapat meningkatkan kualitas analisis sentimen.

3. Salah satu faktor yang dapat meningkatkan akurasi dalam analisis sentimen adalah penggunaan dataset yang lebih besar. Dengan dataset yang lebih besar, model memiliki akses ke lebih banyak variasi sentimen, yang dapat meningkatkan kemampuannya untuk mengklasifikasikan dengan lebih tepat.

DAFTAR PUSTAKA

B. Liu. (2012). Sentiment Analysis and Opinion Mining. Rafael: Morgan & Claypool Publishers, https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis- and-OpinionMining.pdf

Badoni, P., Katal, A., Reddy, M., Bhargava, M. (2022). Graphics vs Gameplay: A Comparative Analysis in Gaming. 2022 2nd International Conference on Intelligent Technologies (CONIT). 24-26.

Biswas, J., Rahman, M.M., Biswas, A.A., Khan, M.A.Z., Rajbongshi, A. and Niloy, H.A. (2021). Sentiment analysis on user reaction for online food delivery services using bert model. In 2021 *7th International Conference on Advanced Computing and Communication Systems* (ICACCS) (Vol. 1, pp. 1019-1023). IEEE. https://ieeexplore.ieee.org/abstract/document/9441669/

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). 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*, 4171–4186. https://doi.org/10.48550/arxiv.1810.04805

Desta, P. W. (2023). Analisis Sentimen Kepuasan Pemain PUBG Berdasarkan Komentar di Media Sosial. http://elibrary.almaata.ac.id/2426/

Dey, L., & Mirajul, H. (2008). Opinion mining from noisy text data. *in Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data* (AND- 2008). https://www.researchgate.net/publication/225360972_Opinion_Mining_from_ Noisy_Text_Data

Geetha, M. P., & Karthika Renuka, D. (2021). Improving the performance of aspect based sentiment analysis using fine-tuned Bert Base Uncased model. *International Journal of Intelligent Networks*, *2*. https://doi.org/10.1016/j.ijin.2021.06.005

Hassani, H., Beneki, C., Unger, S., Mazinani, M.T. and Yeganegi, M.R. (2020). Text mining in big data analytics. *Big Data and Cognitive Computing*, 4(1), p.1. https://www.mdpi.com/619168

Hoang, M., Bihorac, O.A. and Rouces, J. (2019). Aspect-based sentiment analysis using bert. *In Proceedings of the 22nd nordic conference on computational linguistics* (pp. 187-196). https://aclanthology.org/W19-6120/

Kusnadi, R., Yusuf, Y., Andriantony, A., Yaputra, R. A., & Caintan, M. (2021). Analisis Sentimen Terhadap Game Genshin Impact Menggunakan Bert. Rabit: *Jurnal Teknologi dan Sistem Informasi Univrab*, 6(2), 122-129. http://jurnal.univrab.ac.id/index.php/rabit/article/view/1765

Kusuma, R.M.R.W.P. and Yustanti, W., 2021. Analisis Sentimen Customer Review Aplikasi Ruang Guru dengan Metode BERT (Bidirectional Encoder Representations from Transformers). Journal of Emerging Information System and Business Intelligence (JEISBI), 2(3). https://ejournal.unesa.ac.id/index.php/JEISBI/article/view/41567

Munikar, M., Shakya, S., & Shrestha, A. (2019). Fine-grained Sentiment Classification using BERT. *International Conference on Artificial Intelligence for Transforming Business and Society, AITB 2019*. https://doi.org/10.1109/AITB48515.2019.8947435

Nacke, L., & Mirza-Babaei, P. (2018). Games User Research: A Case Study Approach. CRC Press.

Nugroho, K. S., Sukmadewa, A. Y., Wuswilahaken Dw, H., Bachtiar, F. A., & Yudistira, N. (2021). BERT Fine-Tuning for Sentiment Analysis on Indonesian Mobile Apps Reviews. *ACM International Conference Proceeding Series*, 258– 264. https://doi.org/10.1145/3479645.3479679

Pappas, P., Mikalef, M., Giannakos, P., Kourouthanassis. (2019). Explaining User Experience in Mobile Gaming Applications: An fsQCAApproach. *Internet Research.* https://www.emeraldinsight.com/doi/full/10.1108/IntR-12-2017-0479

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

Tan, A.H. (2000) Text Mining: The State of the Art and the Challenges. *Kent Ridge DigitalLabs*. https://www.researchgate.net/publication/2471634_Text_Mining_The_state_of_the_art_and_the_challenges

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-December*, 5999–6009.

LAMPIRAN – 1

Scrapping *Appstore*

```
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")

!pip install app_store_scraper
from app_store_scraper import AppStore
import pandas as pd
import numpy as np
import json, os, uuid

a_reviews = AppStore('id', 'pubg-mobile', '1330123889')
a_reviews.review()
a_df =
pd.DataFrame(np.array(a_reviews.reviews),columns=['review'])
a_df2 = a_df.join(pd.DataFrame(a_df.pop('review').tolist()))

a_df2.insert(loc=0, column='source', value='App Store')
a_df2['developer_response_date'] = None
a_df2['thumbs_up'] = None
a_df2['laguage_code'] = 'id'
a_df2['country_code'] = 'id'
a_df2.insert(loc=1, column='review_id', value=[uuid.uuid4()
for _ in range(len(a_df2.index))])
a_df2.rename(columns= {'review':
'review_description','userName': 'user_name', 'date':
'review_date','title': 'review_title', 'developerResponse':
'developer_response'},inplace = True)
a_df2 = a_df2.where(pd.notnull(a_df2), None)

result = pd.concat([a_df2])
result
result.to_excel('scraping_pubg_appstore.xlsx', index=False)
```

BERT

```
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")

!pip install transformers

import pandas as pd

df = pd.read_csv("data100.csv",sep=";", encoding='cp1252')
df.shape

df['sentimen'] = pd.to_numeric(df['sentimen'],
errors='coerce').fillna(0).astype(int)
df['sentimen'] = df['sentimen'].replace(-1, 2)

df

sentences = df.Tweet.values
labels = df.sentimen.values

from transformers import BertTokenizer

print("Loading BERT Tokenizer")

tokenizer = BertTokenizer.from_pretrained('bert-base-

multilingual-uncased', do_lower_case=True)
df.sample(5)

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

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

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

input_ids = []
# encoded_sent = tokenizer.encode(
#         str(sent),
```

```
#         add_special_tokens = True)

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

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

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

from keras.utils import pad_sequences

MAX_LEN = 400

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]

attention_mask = []

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

  attention_mask.append(att_mask)

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=0.2)
train_mask, test_mask, _, _ =
train_test_split(attention_mask,
                                                labels,

random_state=2017,
```

```
test_size=0.2)
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_mask,

random_state=2018,

test_size=0.15)

import numpy as np

print("== Train ==")

print("Input: ", train_input.shape)

print("Label: ", train_labels.shape)

print("Mask: ", np.array(train_mask).shape)


print("\n== Validation ==")

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

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

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

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

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

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

batch_size = 16

train_data = TensorDataset(train_input, train_mask,
train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data,
sampler=train_sampler, batch_size=batch_size)

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

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

     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()

params = list(model.named_parameters())

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

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

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

print("==== Output Layer ====")
```

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

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


from transformers import get_linear_schedule_with_warmup

epochs = 3

total_steps = len(train_dataloader) * epochs

scheduler = get_linear_schedule_with_warmup(optimizer,

num_warmup_steps = 0,

num_training_steps = total_steps)

import numpy as np

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

import time
import datetime

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

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!")
import matplotlib.pyplot as plt
%matplotlib inline

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()

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

model.eval()

prediction, true_labels = [], []

for batch in test_dataloader:
  batch = tuple(t.to(device) for t in batch)
  b_input_ids, b_input_mask, b_labels = batch

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

  logits = outputs[0]

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

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

print(" DONE.")

from sklearn.metrics import matthews_corrcoef

flat_prediction = [item for sublist in prediction for item
in sublist]

flat_prediction = np.argmax(flat_prediction,
axis=1).flatten()

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

mcc = matthews_corrcoef(flat_true_labels, flat_prediction)

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

from sklearn.metrics import accuracy_score

acc = accuracy_score(flat_true_labels, flat_prediction)

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

from google.colab import drive
import torch
drive.mount('/content/drive')
%cd /content/drive/MyDrive/BertDrive
```

```
main_path = "/content/drive/MyDrive/BertDrive"
torch.save(model.state_dict(), 'model_bert_sentimen.pt')

review = "gamenya banyak bug dan ngelag"
encoded_review = tokenizer.encode_plus(


    review,
    max_length=400,
    padding='max_length',
    truncation=True,
    return_tensors='pt'
)
input_ids = encoded_review['input_ids'].to(device)
attention_mask = encoded_review['attention_mask'].to(device)
model = BertForSequenceClassification.from_pretrained("bert-
base-multilingual-uncased", num_labels=3)
model.classifier = torch.nn.Linear(in_features=768,
out_features=3, bias=True)
model.load_state_dict(torch.load('model_bert_sentimen.pt'))
model.cuda()
model.eval()
with torch.no_grad():
    logits = model(input_ids, attention_mask=attention_mask)
sentiment = np.argmax(logits[0].detach().cpu().numpy())
if sentiment == 0:
    print("Sentimen netral")
elif sentiment == 1:
    print("Sentimen positif")
elif sentiment == 2:
    print("Sentimen negatif")

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

# Flatten the predictions_promotion and true labels
y_pred = [np.argmax(pred) for batch_pred in prediction for
pred in batch_pred]
y_true = [label for batch_label in true_labels for label in
batch_label]

# Create the confusion matrix
cm = confusion_matrix(y_true, y_pred)

# Create the heatmap
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g' )
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')

ticks = ['Positive', 'Neutral', 'Negative']
```

```
plt.xticks([0.5, 1.5, 2.5], ticks, fontsize=12)
plt.yticks([0.5, 1.5, 2.5], ticks, fontsize=12)

plt.show()

from sklearn.metrics import classification_report
print (classification_report(y_true, y_pred))
```

LAMPIRAN – 2

Hanif Misbah Ananda

Jl.Pimpinan No.15
Medan,Sumatera Utara
08970709342
hanif.ananda90@gmail.com

EDUCATION

Universitas Sumatera Utara, Medan — *Sarjana* 2019 - NOW

SMAN 6 Medan, Medan — *Sekolah Menengah Atas* 2016 - 2019

SMPN 2 Medan, Medan — *Sekolah Menengah Pertama* 2013 - 2016

Harapan 1, Medan — *Sekolah dasar kelas 4-6* 2010-2013

Harapan Mandiri, Medan — *Sekolah dasar kelas 1-3* 2007-2010

PROJECTS

- **Aplikasi sederhana pemesanan hotel menggunakan android studio**
- **Program membuat quiz sederhana menggunakan adobe animate**
- **Web sederhana menggunakan html,css,dan php yang terhubung ke database sql**
- **Program Kalkulator sederhana menggunakan C++**
- **Program Kasir sederhana menggunakan C#**

SKILLS

Mengetahui beberapa bahasa pemrograman seperti,phython,java ,c++,c#,html,php,sql ,css

LANGUAGES

Bahasa,English

WORK EXPERIENCE

Intern Quality Assurance at PT. Telkom Indonesia
July 2022 - September 2022