

**ANALISIS EMOSI DALAM TEKS BAHASA INDONESIA DENGAN
PENDEKATAN BERT DAN CNN**

SKRIPSI

ATHA MAULANA

201401096



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

MEDAN

2024

**ANALISIS EMOSI DALAM TEKS BAHASA INDONESIA DENGAN
PENDEKATAN BERT DAN CNN**

SKRIPSI

**Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah
Sarjana Ilmu Komputer**

ATHA MAULANA

201401096



**PROGRAM STUDI S-1 ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA
MEDAN
2024**

PERSETUJUAN

Judul : ANALISIS EMOSI DALAM TEKS BAHASA
INDONESIA DENGAN PENDEKATAN BERT DAN
CNN

Kategori : SKRIPSI


Nama : ATHA MAULANA

Nomor Induk Mahasiswa : 201401096

Program Studi : SARJANA (S-1) ILMU KOMPUTER

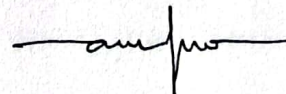
Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Komisi Pembimbing :
Pembimbing 2



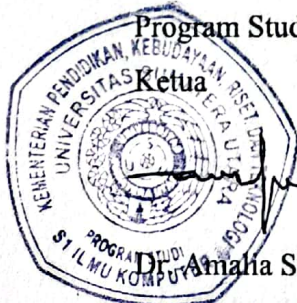
Dr. T. Henny Febriana Harumy S.Kom., M.Kom
NIP. 198802192019032016

Pembimbing 1



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

Diketahui/Disetujui Oleh
Program Studi S-1 Ilmu Komputer



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

UNIVERSITAS SUMATERA UTARA

PERNYATAAN**ANALISIS EMOSI DALAM TEKS BAHASA INDONESIA
DENGAN PENDEKATAN BERT DAN CNN****SKRIPSI**

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

Medan, 12 Juni 2024



Atha Maulana

201401096

PENGHARGAAN

Bismillahirrahmanirrahim, segala puji syukur dipanjatkan kepada Allah *Subhanahu Wa Ta'ala* atas segala nikmat iman dan islam, kesehatan serta karunia-Nya sehingga penulis mampu untuk menyusun skripsi sebagai syarat untuk mendapatkan gelar Sarjana Komputer di Program Studi S-1 Ilmu Komputer, USU. Tidak lupa shalawat serta salam tetap tercurahkan kepada Rasulullah SAW yang telah mengeluarkan umat manusia dari kegelapan menuju zaman terang benderang saat ini.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada Ibu tercinta, T. Seri Aminah atas segala bentuk perjuangan, kasih sayang, dan doa-doa yang dipanjatkan untuk penulis. Dan terima kasih juga kepada Papa saya, Alm. Ayub Khan, semoga Allah SWT menerima seluruh amal ibadah beliau serta diberikan tempat yang terbaik di sisi-Nya. Penyusunan skripsi ini tidak terlepas dari banyaknya bantuan serta bimbingan dari banyak pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Prof. Dr. Muryanto Amin S.Sos., M.Si. selaku Rektor Universitas Sumatera Utara.
2. Ibu Dr. Maya Silvi Lydia B.Sc., M.Sc. selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing Akademik yang telah memberi banyak dukungan dan motivasi kepada penulis.
3. Bapak Dr. Mohammad Andri Budiman S.T., M.Comp.Sc., M.E.M. selaku Wakil Dekan I Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara.
4. Ibu Dr. Amalia, S.T., M.T. selaku Ketua Program Studi S-1 Ilmu Komputer Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Sumatera Utara dan Dosen Pembimbing I yang telah memberi banyak masukan, arahan, motivasi, serta dukungan kepada penulis selama penyusunan skripsi ini.
5. Ibu Dr. T. Henny Febriana Harumy S.Kom., M.Kom. selaku Dosen Pembimbing II yang telah memberi banyak bimbingan dan masukan yang berharga kepada penulis selama proses penyusunan skripsi ini.
6. Ibu Dr. Ir. Elviawaty Muisa Zamzami S.T., M.T., M.M., IPU. selaku Dosen Penguji I yang telah memberikan arahan dan masukan kepada penulis.

7. Ibu Sri Melvani Hardi S.Kom., M.Kom. selaku Dosen Penguji II yang telah memberi saran serta masukan kepada penulis.
8. Seluruh Bapak dan Ibu dosen Program Studi S-1 Ilmu Komputer, yang telah meluangkan banyak waktu, tenaga, dan pikiran untuk mengajar dan memberi wawasan serta moral yang berharga, baik di bangku perkuliahan maupun setelah lulus.
9. Teristimewa kepada Orang Tua penulis Alm. Ayub Khan dan T. Seri Aminah yang telah memberikan penulis semangat, kekuatan, kesabaran, ilmu yang bermanfaat, dan doa yang tiada hentinya untuk penulis sehingga penulis dapat menjalani perkuliahan sampai dengan penyusunan skripsi.
10. Kepada Kakak dan Abang penulis Nanda Anggraini dan M. Fauzi Aulia yang telah memberikan dukungan dan nasehat kepada penulis sehingga penulis bisa menyelesaikan penulisan skripsi.
11. Sahabat seperjuangan “mujahiruddin” yang telah memberikan kenangan yang berharga dan berkesan kepada penulis selama perkuliahan sampai dengan penyusunan skripsi.
12. Faradhila Aulia Utami Tanjung yang telah menemani, memberikan semangat, dan motivasi kepada penulis selama perkuliahan hingga penyusunan skripsi.
13. “Rumjep”, “Rumwang”, “Rumja” sebagai tempat saya beristirahat dari jauhnya perjalanan serta tempat untuk mengerjakan tugas bersama-sama selama perkuliahan sampai dengan penyusunan skripsi.
14. Teman-teman stambuk 2020 yang telah menjadi teman yang baik selama perkuliahan sampai di masa akhir perkuliahan.

Dan seluruh pihak yang telah memberi dukungan serta doa baik yang tidak dapat penulis sebutkan satu per-satu. Semoga Allah *Subhanahu Wa Ta'ala* senantiasa melimpahkan keberkahan serta kebaikan dan hasil penelitian ini dapat memberi manfaat maupun inspirasi untuk kedepannya.

Medan, 12 Juni 2024

Penulis,



Atha Maulana

ABSTRAK

Emosi merupakan aspek yang penting dalam komunikasi manusia dan telah dipelajari secara luas karena menjadi bagian penting dari sifat dasar manusia. Pengenalan emosi dalam teks memungkinkan pemahaman yang lebih mendalam tentang pesan yang disampaikan, melibatkan analisis terhadap pilihan kata, tanda baca, dan panjang pesan. Pengenalan emosi dalam teks bahasa Indonesia sering kali melibatkan analisis kata-kata sifat yang menggambarkan kondisi emosi tertentu. Dengan informasi yang terus bertambah, tantangan dalam memahami emosi hanya dengan pemrosesan manual semakin kompleks. Penelitian ini memanfaatkan model gabungan dari *Bidirectional Encoder Representation from Transformer* (BERT) dan *Convolutional Neural Networks* (CNN) untuk klasifikasi teks. BERT digunakan untuk melatih model bahasa yang dapat merepresentasikan makna kata secara dinamis berdasarkan konteksnya, dan CNN digunakan untuk memprediksi *output* berdasarkan vektor semantik yang dihasilkan oleh BERT dari setiap kata dalam teks. Emosi yang dapat diklasifikasi terdiri dari emosi sedih, senang, cinta, marah, takut, dan terkejut. Dataset diambil dari situs Kaggle bernama “*Emotions*” yang berisikan 300 ribu kumpulan pesan Twitter berbahasa Inggris. Penelitian ini mendapatkan akurasi sebesar 85%, *precision* 92%, *recall* 86%, dan *F1-score* 88% dari seluruh kelas emosi.

Kata Kunci: Emosi, Pengenalan Emosi, Klasifikasi Teks, *Bidirectional Encoder Representation from Transformer* (BERT), *Convolutional Neural Networks* (CNN).

ABSTRACT

Emotion Analysis In Indonesian Texts With BERT and CNN Approaches

Emotion is an important aspect of human communication and has been widely studied as an important part of human nature. Emotion recognition in text allows for a deeper understanding of the message being conveyed, involving analysis of word choice, punctuation and message length. Emotion recognition in Indonesian texts often involves analyzing adjectives that describe certain emotional states. With the ever-growing amount of information, the challenge of understanding emotions with only manual processing is getting more complex. This research utilizes a combined model of Bidirectional Encoder Representation from Transformer (BERT) and Convolutional Neural Networks (CNN) for text classification. BERT is used to train a language model that can dynamically represent the meaning of words based on their context, and CNN is used to predict the output based on the semantic vector generated by BERT from each word in the text. The classifiable emotions consist of sadness, happiness, love, anger, fear, and surprise. The dataset is taken from the Kaggle site called "Emotions" which contains 300 thousand collections of English Twitter messages. This study obtained 85% accuracy, 92% precision, 86% recall, and 88% F1-score from all emotion classes.

Keywords: Emotion, Emotion Recognition, Text Classification, Bidirectional Encoder Representation from Transformer (BERT), Convolutional Neural Networks (CNN).

DAFTAR ISI

PERSETUJUAN	ii
PERNYATAAN.....	iii
PENGHARGAAN.....	iv
ABSTRAK	vi
ABSTRACT.....	vii
DAFTAR ISI.....	viii
DAFTAR TABEL	xi
DAFTAR GAMBAR.....	xii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah.....	4
1.3 Batasan Masalah	5
1.4 Tujuan Penelitian	5
1.5 Manfaat Penelitian	5
1.6 Metodologi Penelitian	5
1.7 Penelitian Relevan	7
1.8 Sistematika Penulisan	8
BAB 2 LANDASAN TEORI	9
2.1 Emosi	9
2.2 Analisis Emosi pada Teks	9
2.3 Klasifikasi Emosi	10
2.4 Artificial Intelligence.....	11
2.5 Natural Language Processing	11
2.5.1 Parsing	12
2.5.2 Case Folding	12

2.5.3	<i>Tokenizing</i>	12
2.5.4	<i>Stemming</i>	12
2.5.5	<i>Stopword Removal</i>	13
2.6	Neural Network	13
2.7	Deep Learning	15
2.7.1	<i>Deep Supervised Learning</i>	15
2.7.2	<i>Deep Semi-Supervised Learning</i>	16
2.7.3	<i>Deep Unsupervised Learning</i>	16
2.8	Word Embedding	16
2.8.1	<i>Embedding Layer</i>	17
2.9	Framework	17
2.10	BERT	17
2.10.1	<i>Input/Output Representation</i>	20
2.10.2	<i>Pre-Trained BERT</i>	22
2.11	Convolutional Neural Network	22
2.11.1	<i>1D Convolutional Neural Network</i>	23
BAB 3	ANALISIS DAN PERANCANGAN	26
3.1	Analisis	26
3.1.1	<i>Analisis Masalah</i>	26
3.1.2	<i>Analisis Kebutuhan</i>	26
3.2	Gambaran Umum Sistem	27
3.3	Analisis Dataset	28
3.4	Pre-Processing	31
3.4.1	<i>Case Folding</i>	31
3.4.2	<i>Tokenizing</i>	31
3.4.3	<i>Stopword Removal</i>	32
3.4.4	<i>Stemming</i>	33

3.4.5	<i>Filtering</i>	34
3.5	Representasi Fitur Teks dengan BERT	34
3.6	Klasifikasi dengan Model CNN	36
3.6.1	<i>1D Convolution</i>	37
3.7	Evaluasi	38
3.8	Menyimpan Model	38
3.9	Perancangan Sistem	38
3.3.1	<i>Diagram Umum Sistem</i>	39
3.3.2	<i>Use Case Diagram</i>	39
3.3.3	<i>Activity Diagram</i>	40
3.3.4	<i>Sequence Diagram</i>	41
3.3.5	<i>Flowchart Sistem</i>	42
BAB 4	IMPLEMENTASI DAN PENGUJIAN	44
4.1	Implementasi Sistem	44
4.1.1	<i>Halaman Utama Website</i>	44
4.1.2	<i>Hasil Klasifikasi Sistem</i>	44
4.2	Pre-Processing Dataset	45
4.3	Implementasi Ekstraksi Fitur Teks Menggunakan BERT	49
4.4	Implementasi Klasifikasi Emosi Pada Teks Dengan CNN	52
4.5	Evaluasi	57
4.6	Menyimpan dan Memanggil Model	62
4.7	Hasil Pengujian Sistem	63
BAB 5	PENUTUP	67
5.1	Kesimpulan	67
5.2	Saran	67
DAFTAR PUSTAKA	68

DAFTAR TABEL

Tabel 3. 1 Contoh Labelisasi Dataset.....	30
Tabel 3. 2 Contoh Case Folding.....	31
Tabel 3. 3 Contoh Tokenizing.....	32
Tabel 3. 4 Contoh Stopword Removal.....	32
Tabel 3. 5 Contoh Stemming.....	34
Tabel 4. 1 Tabel Nilai Hyperparameter Tuning Uji Coba Pertama dan Kedua	52
Tabel 4. 2 Tabel Hasil Metrik Evaluasi Pada Uji Coba Pertama dan Kedua.....	52
Tabel 4. 3 TP, TN, FP, dan FP Semua Kelas Emosi	62
Tabel 4. 4 Hasil Akurasi, Precision, Recall, dan F1-Score Semua Kelas Emosi.....	62
Tabel 4. 5 Tabel Pengujian Emosi pada Teks Bahasa Indonesia	63

DAFTAR GAMBAR

Gambar 2. 1 Plutchik Wheel of Emotions.....	10
Gambar 2. 2 Feedforward Neural Network.....	13
Gambar 2. 3 Persamaan Sigmoid Function, Tanh, dan ReLu	14
Gambar 2. 4 Persamaan Fungsi Softmax.....	15
Gambar 2. 5 Self Attention.....	18
Gambar 2. 6 Arsitektur Model Transformer.....	19
Gambar 2. 7 Pre-Training dan Fine-Tuning BERT.....	20
Gambar 2. 8 Representasi Input BERT	21
Gambar 2. 9 Model Arsitektur.....	24
Gambar 3. 1 Arsitektur Umum Sistem	28
Gambar 3. 2 Tahapan Pembuatan Model	28
Gambar 3. 3 Dataset “Emotions” Berbahasa Inggris	29
Gambar 3. 4 Hasil Dataset Setelah Translasi	29
Gambar 3. 5 Dataset Dengan Label Angka Berurutan	29
Gambar 3. 6 Proses Case folding	31
Gambar 3. 7 Proses Tokenizing.....	31
Gambar 3. 8 Proses Stopword Removal.....	32
Gambar 3. 9 Proses Stemming	33
Gambar 3. 10 Proses Filtering	34
Gambar 3. 11 Contoh Token yang Dikonversi Menjadi ID	35
Gambar 3. 12 Contoh Pembuatan Mask Input dan Segmen ID.....	35
Gambar 3. 13 Contoh Output Vektor dari Token ‘[CLS]’	35
Gambar 3. 14 Peta Sketsa dari Model BERT dan CNN	36
Gambar 3. 15 Fitur Vektor Disusutkan menjadi Nilai Maksimum melalui Max Pooling.....	37
Gambar 3. 16 Diagram Umum Sistem	39
Gambar 3. 17 Use Case Diagram	40
Gambar 3. 18 Activity Diagram	41
Gambar 3. 19 Sequence Diagram.....	42
Gambar 3. 20 Flowchart Sistem	43
Gambar 4. 1 Tampilan Halaman Utama.....	44
Gambar 4. 2 Hasil Klasifikasi Emosi	45
Gambar 4. 3 Case Folding dan Filtering	46
Gambar 4. 4 Hasil Case Folding dan Filtering	46

Gambar 4. 5 Tokenizing	46
Gambar 4. 6 Hasil Tokenizing.....	46
Gambar 4. 7 Stopword Removal	47
Gambar 4. 8 Hasil Stopword Removal.....	47
Gambar 4. 9 Stemming.....	47
Gambar 4. 10 Hasil Stemming	47
Gambar 4. 11 Wordcloud Setiap Kelas Emosi	48
Gambar 4. 12 Frekuensi Kata Pada Semua Kelas Emosi	48
Gambar 4. 13 Transformasi Data Input.....	49
Gambar 4. 14 Mengambil Contoh Data Latih	49
Gambar 4. 15 Menghasilkan Fitur dari Contoh-Contoh Data	50
Gambar 4. 16 Mengubah Fitur-Fitur Untuk Pelatihan Model	50
Gambar 4. 17 Pemanggilan Model BERT dan Menggunakan CUDA.....	51
Gambar 4. 18 Inialisasi Model CNN.....	53
Gambar 4. 19 Proses Feedforward Model CNN.....	55
Gambar 4. 20 Proses Menyiapkan Data Latih dan Data Validasi Untuk Model	55
Gambar 4. 21 Proses Inisialisasi Parameter-Parameter Model CNN	56
Gambar 4. 22 Proses Menghitung Metrik Evaluasi.....	57
Gambar 4. 23 Proses Pelatihan Model	57
Gambar 4. 24 Proses Evaluasi Model.....	58
Gambar 4. 25 Proses Membuat Grafik Metrik Evaluasi	59
Gambar 4. 26 Hasil Grafik Akurasi, F1-Score, Precision, dan Recall	59
Gambar 4. 27 Hasil Grafik Total Training Loss.....	60
Gambar 4. 28 Proses Membuat Grafik ROC Curve	60
Gambar 4. 29 Hasil Grafik ROC Curve	61
Gambar 4. 30 Hasil Confusion Matrix	61
Gambar 4. 31 Proses Menyimpan dan Memanggil Model.....	63

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Emosi merupakan salah satu cara manusia untuk saling berkomunikasi. Secara umum emosi sangat penting di dalam kehidupan manusia, di mana emosi merupakan salah satu faktor yang memengaruhi keputusan yang dibuat oleh manusia dengan lingkungan sosial, serta memengaruhi dalam berkomunikasi. Bahasa adalah alat komunikasi yang digunakan manusia untuk mendapatkan informasi serta berdialog. Bahasa resmi Nasional Republik Indonesia yaitu bahasa Indonesia. Bahasa Indonesia digunakan untuk menyampaikan informasi kepada orang lain, baik melalui ucapan maupun tulisan, sehingga pesan yang ingin kita sampaikan bisa dipahami dengan jelas oleh mereka (Wuwungan, 2021).

Manusia mengekspresikan emosi mereka dalam bentuk lisan dan non-lisan, seperti ucapan, ekspresi wajah, gestur tubuh, dan ekspresi menggunakan teks (Islam et al., 2021). Sekarang ini dengan adanya internet, terdapat banyak sekali data berbentuk teks (Batbaatar et al., 2019). Mengikuti perkembangan teknologi yang berkembang pesat, manusia sangat mudah untuk mengekspresikan diri pada sosial media melalui teks. Ada dua jenis penulisan yang dapat dianalisis oleh mesin yaitu formal dan informal. Konten tekstual formal berkaitan dengan puisi, novel, esai, drama, dan dokumentasi resmi/hukum, sedangkan konten tekstual informal berkaitan dengan SMS (*Short Message Service*), obrolan, dan postingan media sosial (Ahmad et al., 2020). Ketika seseorang menulis pesan teks seringkali dalam tiap kalimat mengandung emosi melalui pilihan kata, tanda baca, atau bahkan panjangnya pesan tersebut.

Pengenalan emosi adalah proses dinamis yang menargetkan keadaan emosi seseorang, yang berarti bahwa emosi yang sesuai dengan tindakan setiap individu adalah berbeda (Abdullah et al., 2021). Pengenalan emosi teks melibatkan analisis emosi yang diungkapkan dalam teks untuk mendapatkan pemahaman yang lebih mendalam mengenai isinya, termasuk sentimen, tujuan, dan ekspresi yang terkandung di dalamnya (Han et al., 2023).

Mendeteksi emosi memainkan peran penting dalam memahami perilaku dan interaksi manusia, dan dapat diterapkan dalam berbagai bidang, mulai dari lingkungan bisnis, psikologi, pendidikan, dan bidang lainnya yang membutuhkan pemahaman dan penafsiran emosi (Huang dan Lajoie, 2023). Di dalam dunia bisnis, pemahaman emosi dari teks-teks yang dikumpulkan dari media sosial, ulasan produk, atau survei dapat memberikan wawasan bagi perusahaan untuk mengambil keputusan yang baik untuk pengembangan produk, pemasaran, dan layanan pelanggan mereka. Dalam konteks kesehatan mental, analisis emosi teks dapat membantu pengidentifikasian dini masalah kesehatan mental, yang memungkinkan tindakan yang lebih cepat dan tepat untuk individu yang memerlukan bantuan.

Dalam kalimat bahasa Indonesia untuk dapat mengenali emosi bisa dilihat dari kata-kata sifat pada kalimat tersebut. Emosi seperti sedih, marah, cinta, senang, dan terkejut bisa didapatkan melalui banyaknya jumlah kata sifat yang menggambarkan keadaan emosi tersebut (Agastya dan Haryanto, 2020). Dengan banyaknya informasi yang diperoleh, sangat sulit untuk bisa memahami seluruh isi informasi yang didapat hanya dengan pemrosesan manual.

Pada abad ke-20, *Artificial Intelligence* dikembangkan lebih luas lagi dan membawa para peneliti untuk melakukan penelitian mendalam di berbagai bidang seperti NLP, *computer vision*, *machine learning*, dan *deep learning*. Bidang NLP masih belum jelas karena teknik komputasi dan linguistiknya, dimana membantu komputer memahami dan menghasilkan interaksi manusia-komputer dalam bentuk teks dan ucapan. Hal ini bertujuan untuk merancang model untuk berbagai proses seperti persepsi, sentimen, kepercayaan, dan emosi (Bharti et al., 2022).

Salah satu model *deep learning* yang cukup populer yaitu BERT, BERT merupakan jenis model bahasa yang berdasarkan arsitektur Transformer yang terkenal karena kemampuannya dalam memahami konteks dalam teks dengan baik. Terdapat dua kerangka pada BERT yaitu pra-pelatihan (*pre-training*) dan *fine-tuning*. Pada tahap pra-pelatihan, proses pelatihan model dilakukan terhadap data yang tidak berlabel dalam tugas pra-pelatihan yang terpisah. Tahap *fine-tuning*, model BERT dimulai dengan parameter yang telah dilatih sebelumnya, dan semua parameter disesuaikan dengan data berlabel dari tugas-tugas *downstream*. Setiap

tugas-tugas *downstream* memiliki model *fine-tuned* yang berbeda, walaupun dimulai oleh fitur-fitur yang sama (Devlin et al., 2019).

Pada saat ini, terdapat dua algoritma *deep learning* dasar untuk pemrosesan urutan yaitu *recurrent neural network* dan *convolutional neural network* satu dimensi (Peng dan Bao, 2020). CNN merupakan arsitektur yang sangat penting dan efisien, karena dapat mencapai akurasi prediksi yang lebih baik dan mengonsumsi lebih sedikit sumber daya komputasi (Peng dan Bao, 2020). Dalam klasifikasi teks, *sentence modelling*, dan *semantic parsing*, CNN telah direkomendasikan untuk menangani tugas-tugas NLP dan mencapai hasil yang mengesankan (Mohd et al., 2021).

Penelitian ini menggunakan model *pre-trained* BERT yang digunakan untuk memperoleh representasi kata dari teks, kemudian representasi tersebut dijadikan *input* untuk model CNN. Lalu model CNN kemudian menangkap pola lokal dan hubungan antar kata dan menghasilkan *output* berupa klasifikasi emosi.

Terdapat penelitian terdahulu yang sudah dilakukan, seperti yang dilakukan oleh Asghar, et al. mengusulkan model *deep learning* yang dinamakan Bi-LSTM (*Bidirectional Long Short-Term Memory*) dalam mengidentifikasi dan mengklasifikasi emosi pada konten teks dengan akurasi 87,66% (Asghar et al., 2022). Kemudian, penelitian selanjutnya yang dilakukan oleh R. Abas et al. mengusulkan model BERT-CNN untuk mendeteksi emosi dari teks menggunakan dua dataset yaitu SemEval-2019 Task3 (Chatterjee et al., 2019) dataset dengan akurasi 94,7% dan ISEAR (Scherer dan Wallbott, 1994) dataset dengan akurasi 76% (R. Abas et al., 2022).

Penelitian yang dilakukan oleh (Fudholi, D. H., 2021) mengklasifikasikan emosi menjadi sembilan kategori yang dimulai dari emosi marah, sedih, takut, tertarik, jijik, tertarik, netral, kaget, dan bahagia. Seluruh tipe emosi ini dapat mengekspresikan emosi manusia. Penelitian ini menggunakan model BERT (*Bidirectional Encoder Representation from Transformers*) sebagai metode klasifikasi. BERT adalah teknik berbasis *neural network* dengan arsitektur *transformers*. Model ini memungkinkan untuk mempelajari konteks kata di sekitarnya berdasarkan rangkaian kata atau kalimat. Pada penelitian ini peneliti

membandingkan akurasi antara BERT dan IndoBERT. Mereka menemukan bahwa BERT memiliki akurasi 89,1% sementara IndoBERT memiliki akurasi 76%. Penelitian ini menggunakan 2700 data dengan 9 label emosi. Meskipun terdapat perbedaan yang cukup jauh antara kedua metode, keduanya memiliki nilai akurasi yang bagus. Dengan demikian, pada penelitian ini ditemukan bahwa model BERT memiliki performa yang lebih baik untuk klasifikasi emosi pada teks dibandingkan dengan IndoBERT.

Penelitian yang dilakukan oleh Habib, et al. mengembangkan teknik klasifikasi emosi dan efektivitas pengenalan *emoticon* dari data *microblog* dengan akurasi 88% (Habib et al., 2022). Lalu penelitian yang dilakukan oleh Han, et al. mengusulkan model XLNet-BiGRU-Att, membangun XLNet dengan memanfaatkan BiGRU dan *attention* (Att) untuk mencapai peningkatan performa yang lebih baik. Penelitian ini menggunakan dua dataset yaitu IEMOCAP (Yu et al., 2020) dan CASIA (Li et al., 2017) dataset, dengan akurasi 91,71% pada IEMOCAP dan 85,71% pada CASIA (Han et al., 2023). Penelitian-penelitian tersebut melakukan analisis emosi dalam teks menggunakan dataset bahasa Inggris dan terdapat penelitian dengan menggunakan dataset bahasa Indonesia.

Pada penelitian ini, penulis memanfaatkan model *deep learning* BERT dan CNN untuk menganalisis emosi dalam teks bahasa Indonesia. Dataset diambil dari situs Kaggle dimana menyediakan akses ke ribuan dataset publik yang dapat digunakan oleh peneliti. Dataset yang dinamakan “*Emotions*” (Elgiryewithana, 2024), berisi 300 ribu kumpulan pesan Twitter berbahasa Inggris yang dianotasi secara cermat dengan enam emosi dasar (*sadness, joy, love, anger, fear, dan surprise*). Penelitian ini menggunakan enam kategori emosi yaitu emosi senang, sedih, marah, takut, cinta, dan terkejut. Dataset yang telah dikumpulkan diubah ke bahasa Indonesia kemudian akan di latih dan uji menggunakan bahasa Python.

1.2 Rumusan Masalah

Emosi adalah bagian penting dalam komunikasi. Manusia bisa menunjukkan emosi melalui kata-kata atau tulisan. Memahami emosi dalam teks tidak hanya memberikan wawasan mengenai perasaan penulis, tetapi juga dapat digunakan untuk berbagai tujuan, seperti pemahaman perilaku, analisis pasar, dan deteksi dini masalah mental. Namun, dengan jumlah data yang terus bertambah, memahami

emosi dalam teks secara manual menjadi sulit dan tidak efisien. Melalui penelitian berupa analisis emosi dalam teks bahasa Indonesia dengan pendekatan BERT dan CNN diharapkan dapat memberikan solusi yang lebih efisien dan akurat dalam mengidentifikasi dan memahami emosi dalam teks.

1.3 Batasan Masalah

Batasan-batasan masalah yang terdapat pada penelitian ini yaitu.

1. Terdapat 6 (enam) kategori emosi yaitu sedih, senang, marah, takut, cinta, dan terkejut.
2. Model dibuat dalam bahasa Python.
3. Program dirancang berbasis *web* dan dibangun menggunakan *framework* streamlit.
4. Teks kalimat yang dilatih dan diuji berbahasa Indonesia.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah untuk membangun sebuah *website* yang memudahkan mengidentifikasi emosi yang terkandung dalam teks bahasa Indonesia

1.5 Manfaat Penelitian

Manfaat dari penelitian ini diharapkan mampu.

1. Membantu para pengguna sistem untuk mengetahui emosi apa yang terkandung dalam teks pada saat menulis sebuah kalimat.
2. Mengetahui performa model BERT dan CNN dalam menganalisis emosi pada teks bahasa Indonesia.

1.6 Metodologi Penelitian

Beberapa metode yang diterapkan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

Studi literatur dilakukan untuk mencari dan mengumpulkan referensi dari berbagai sumber terpercaya dan melakukan peninjauan pustaka melalui buku-buku, jurnal, *e-book*, artikel ilmiah, makalah atau situs internet yang berhubungan dengan model BERT dan CNN.

2. Analisis dan Perancangan Sistem

Perancangan arsitektur sistem yang ingin dibuat sesuai dengan permasalahan dan perancangan sistem. Perancangan sistem akan dibuat dengan pembuatan *flowchart* dan *workflow* terhadap penelitian yang dilakukan.

3. Pencarian dan Pengumpulan Dataset

Pada tahap awal, dataset untuk klasifikasi emosi dalam bahasa Inggris diambil dari berbagai sumber, termasuk Kaggle dan sumber data lainnya yang relevan. Setiap kalimat dalam dataset sudah diberi label emosi sesuai dengan kategori emosi yang telah ditentukan, seperti emosi senang, sedih, marah, takut, cinta, dan terkejut. Kemudian, dataset akan diterjemahkan dan disesuaikan ke dalam bahasa Indonesia untuk dilakukan penelitian.

4. *Pre-processing* Data

Tahap ini dilakukan untuk membersihkan dan mempersiapkan teks agar dapat diproses oleh model BERT dan CNN. Tahapan ini meliputi *case folding* untuk mengubah semua huruf teks menjadi kecil, *tokenizing*, *stopwords removal*, *stemming*, menghapus *mentions*, *links*, dan *URLs*.

5. Membangun Model BERT dan CNN

Model akan dibuat dengan bahasa Python serta beberapa *libraries* seperti *transformers*, *torch*, dan *library* lainnya. Lalu model akan dilatih dengan data yang sudah dibuat.

6. Implementasi Sistem dan Model

Model yang telah dibuat akan digunakan untuk mengklasifikasi emosi yang terdapat dalam teks bahasa Indonesia. Pengguna dapat memasukkan kalimat teks bahasa Indonesia kedalam sistem, setelah itu sistem akan menggunakan model untuk mengklasifikasi emosi apa yang terkandung dalam kalimat tersebut. Implementasi akan dilakukan pada sistem operasi *Windows*, menggunakan *framework* streamlit berbasis Python.

7. Pengujian dan Evaluasi

Tahap ini, model yang telah dibangun dilakukan uji coba untuk melakukan analisis emosi dalam teks bahasa Indonesia. Lalu melakukan evaluasi terhadap kekurangan yang ada pada sistem.

8. Dokumentasi

Pada tahap ini, laporan serta dokumentasi hasil penelitian akan dibuat.

1.7 Penelitian Relevan

Berikut adalah penelitian terdahulu yang relevan dengan penelitian yang dilakukan, yaitu:

1. Pada penelitian Klasifikasi Emosi pada Teks dengan Menggunakan Metode *Deep Learning* (Fudholi, 2021), penelitian ini bertujuan untuk membandingkan akurasi antara BERT dan IndoBERT. Hasil penelitian ini menunjukkan BERT memiliki akurasi 89,1% sementara IndoBERT memiliki akurasi 76%. Penelitian ini menggunakan 2700 data teks bahasa Indonesia dengan 9 label emosi (marah, sedih, takut, netral, bahagia, tertarik, percaya, kaget, dan jijik). Meskipun terdapat perbedaan yang cukup jauh antara kedua metode, keduanya memiliki nilai akurasi yang baik. Penelitian penulis memiliki pendekatan yang berbeda dengan penelitian yang menggunakan BERT dan IndoBERT. Penelitian penulis menggunakan pendekatan dari BERT dan CNN untuk menganalisis emosi dalam teks bahasa Indonesia. Pendekatan ini memanfaatkan keunggulan dari kedua model, yaitu BERT yang sangat baik dalam memahami konteks dan struktur kalimat, serta CNN yang efektif dalam mengenali pola dalam data.
2. Pada jurnal BERT-CNN: *A Deep Learning Model for Detecting Emotions from Text* (R. Abas et al., 2022) mengusulkan model BERT-CNN untuk mendeteksi emosi dari teks. Penelitian ini menggabungkan model *Bidirectional Encoder Representation from Transformer* (BERT) dan *Convolutional Neural Network* (CNN) untuk klasifikasi teks, dan mendapatkan akurasi 94,7% dan *F1-score* 94% untuk dataset SemEval-2019 Task3 (Chatterjee et al., 2019), serta akurasi 76% dan *F1-score* 76% untuk dataset ISEAR (Scherer dan Wallbott, 1994). Penelitian ini juga meninjau kemajuan terbaru dalam deteksi dan analisis emosi, yang dapat dikategorikan menjadi dua yaitu pengenalan emosi berdasarkan *traditional methods* dan pengenalan emosi berdasarkan pendekatan *deep learning*.
3. Pada jurnal *A Deep Neural Network Model for the Detection and Classification of Emotions from Textual Content* (Asghar et al., 2022) membahas tentang pengenalan dan klasifikasi emosi dari konten teks menggunakan model *neural network*. Penelitian ini memperkenalkan model *deep learning*, yaitu Bi-LSTM dalam mendeteksi dan mengklasifikasi emosi pada konten teks dengan akurasi 87,66%. Tujuan dari penelitian ini untuk mengkategorikan sentimen emosional

dengan mengenali mereka dalam teks, dengan emosi memperhitungkan lima emosi utama *joy*, *sadness*, *fear*, *shame*, dan *guilt*. Hasil dari penelitian ini membantu perusahaan dalam menerapkan praktik terbaik dalam pemilihan, manajemen, dan optimalisasi kebijakan, layanan, dan informasi produk.

1.8 Sistematika Penulisan

Sistematika penulisan skripsi yang digunakan dalam penelitian ini adalah sebagai berikut:

BAB 1 PENDAHULUAN

Pada bab ini mencakup penjelasan tentang latar belakang pemilihan judul, rumusan dan batasan masalah, tujuan, manfaat, dan metodologi penelitian, penelitian relevan, dan sistematika penulisan skripsi.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan teoritis dalam penelitian dan pengembangan sistem, seperti pendahuluan penjelasan emosi, analisis emosi pada teks, NLP, *neural network*, *deep learning*, *word embedding*, BERT, dan CNN.

BAB 3 ANALISIS DAN PERANCANGAN

Bab ini menjelaskan mengenai analisis dan perancangan pada sistem dan penelitian

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini membahas tentang implementasi dan pengujian sistem yang sesuai dengan rancangan yang telah disiapkan sebelumnya.

BAB 5 KESIMPULAN DAN SARAN

Bab ini mencakup rangkuman dari setiap bab dan menarik kesimpulan yang telah dibahas, dan memberikan saran-saran yang dapat menjadi masukan untuk peneliti di masa mendatang.

BAB 2

LANDASAN TEORI

2.1 Emosi

Emosi merupakan salah satu aspek penting dalam komunikasi manusia. Secara umum emosi sangat penting di dalam kehidupan manusia, di mana emosi merupakan salah satu faktor yang memengaruhi keputusan yang dibuat oleh manusia dengan lingkungan sosial, serta memengaruhi dalam berkomunikasi. Fakta bahwa manusia sering kali mengambil keputusan dengan menggunakan emosi dibandingkan dengan logika membuatnya penting untuk memanfaatkan bahasa emosi (Sarasati dan Nurvia, 2021).

Dalam konteks berkomunikasi, pemahaman terhadap bahasa emosi harus ada untuk menciptakan hubungan yang sehat dan berkelanjutan. Emosi dapat berfungsi sebagai penghubung antar manusia dan memungkinkan untuk merasakan dan merespons satu sama lain dengan lebih baik. Dengan memperhatikan dan menghargai emosi, manusia dapat menciptakan lingkungan yang mendukung kesejahteraan mental dan emosional.

Manusia mengekspresikan emosi mereka dalam bentuk lisan dan non-lisan, seperti ucapan, ekspresi wajah, gestur tubuh, dan ekspresi menggunakan teks (Islam et al., 2021). Sekarang ini dengan adanya internet, terdapat banyak sekali data berbentuk teks (Batbaatar et al., 2019). Mengikuti perkembangan teknologi yang berkembang pesat, manusia sangat mudah untuk mengekspresikan diri pada sosial media melalui teks.

2.2 Analisis Emosi pada Teks

Analisis emosi pada teks merupakan pendekatan dengan NLP yang memiliki tujuan untuk mengidentifikasi, mengklasifikasikan, dan memahami ekspresi emosional yang terdapat dalam teks. Analisis emosi pada teks melibatkan beberapa tahap termasuk *pre-processing* teks, representasi fitur, dan pembangunan model klasifikasi. Analisis ini digunakan untuk mengekstrak dan memahami perasaan atau emosi yang tersirat dalam kata-kata dan kalimat.

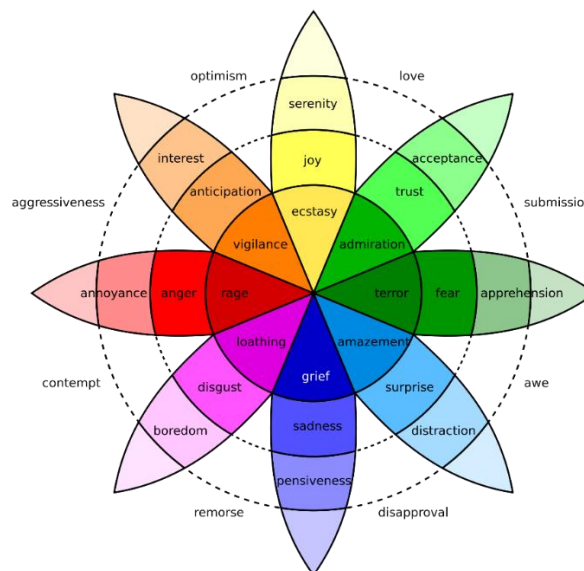
Pengenalan emosi adalah proses dinamis yang menargetkan keadaan emosi seseorang, yang berarti bahwa emosi yang sesuai dengan tindakan setiap individu adalah berbeda (Abdullah et al., 2021). Pengenalan emosi teks melibatkan analisis

emosi yang diungkapkan dalam teks untuk mendapatkan pemahaman yang lebih mendalam mengenai isinya, termasuk sentimen, tujuan, dan ekspresi yang terkandung di dalamnya (Han et al., 2023). Dalam kalimat bahasa Indonesia untuk dapat mengenali emosi bisa dilihat dari kata-kata sifat pada kalimat tersebut. Emosi seperti sedih, marah, cinta, senang, dan terkejut bisa didapatkan melalui banyaknya jumlah kata sifat yang menggambarkan keadaan emosi tersebut (Agastya dan Haryanto, 2020).

Dalam penelitian ini, model *pre-trained* BERT digunakan untuk memahami konteks dan hubungan antar kata-kata yang memungkinkan pemahaman yang lebih baik terhadap makna keseluruhan kalimat. Lalu, CNN digunakan untuk mengekstrak fitur spasial dalam teks. Dengan begitu diharapkan penggunaan kedua model dalam analisis emosi pada teks lebih komprehensif dan efektif.

2.3 Klasifikasi Emosi

Pada tahun 1980-an Robert Plutchik menempatkan sejumlah emosi ke dalam delapan kelompok penting. Setengah dari emosi-emosi ini bersifat positif, sedangkan setengahnya bersifat negatif. Robert Plutchik menjelaskan setiap emosi secara rinci dan membaginya ke dalam subkelompok, mengelompokkannya sebagai emosi sekunder dan tersier dalam mekanisme berbentuk roda.



Gambar 2.1 *Plutchik Wheel of Emotions*

Sumber (Mohsin dan Beltiukov, 2019)

Pada gambar 2.1 dapat diamati delapan kategori emosi tampak berlawanan satu sama lain, hal ini dapat dilihat pada emosi sekunder seperti *joy* yang berlawanan dengan *sadness*, *surprise* yang berlawanan dengan *anticipation*, *trust* berlawanan dengan *disgust*, dan *anger* berlawanan dengan *fear*. Penelitian ini mengambil tiga emosi positif dan tiga emosi negatif.

Emosi positif diantaranya senang, cinta, dan terkejut. Selanjutnya emosi negatif diantaranya sedih, marah, dan takut. Dengan memilih emosi-emosi ini diharapkan mampu mengatasi tantangan pemahaman emosi dalam teks secara efisien dan akurat, serta memberikan kontribusi dalam berbagai bidang, termasuk komunikasi, psikologi, dan bisnis.

2.4 Artificial Intelligence

Artificial Intelligence (AI) adalah sebuah bidang yang mengkombinasikan *computer science* dan dataset yang baik agar dapat memecahkan masalah. Secara umum *artificial intelligence* biasanya melakukan prediksi dengan mengumpulkan banyak data, menganalisis dan mengidentifikasi data, dan menggunakan pola-pola ini untuk melakukan prediksi.

Pada abad ke-20, *Artificial intelligence* dikembangkan lebih luas lagi dan membawa para peneliti untuk melakukan penelitian mendalam di berbagai bidang seperti NLP, *computer vision*, *machine learning*, dan *deep learning*. AI bisa mengerjakan suatu tugas jauh lebih baik dari manusia. Khususnya dalam tugas-tugas yang memiliki perulangan dan berorientasi pada detail. Karena proses dari kumpulan data yang sangat besar, *artificial intelligence* juga dapat memberitahu perusahaan tentang operasi yang mungkin mereka tidak tahu. Oleh karena itu, *artificial intelligence* menjadi penting mulai dari bidang pendidikan dan pemasaran hingga desain produk.

2.5 Natural Language Processing

Natural Language Processing (NLP) adalah ilmu yang fokus pada bagaimana komputer berinteraksi dengan bahasa manusia. Tujuannya membuat mesin bisa memahami, memproses, dan merespons bahasa manusia dengan cara yang bermakna. Perkembangan teknologi dalam beberapa dekade terakhir telah memungkinkan kemajuan signifikan dalam penelitian dan pengembangan NLP (Amien, 2023). Bahasa Indonesia menjadi subjek yang penting dalam penelitian

NLP karena merupakan bahasa resmi dengan jumlah penduduk terbesar keempat di dunia (Annur, 2023).

Dalam konteks bahasa Indonesia, *natural language processing* menjadi krusial untuk memahami dan memanfaatkan data teks dalam skala besar. Analisis emosi melibatkan ekstraksi sentimen yang terkandung dalam teks yang memungkinkan untuk memahami respons emosional manusia terhadap sesuatu. Dengan menggunakan teknik-teknik seperti *machine learning* dan *deep learning*, NLP dapat membantu mengidentifikasi emosi yang ada pada teks dan memperkaya pemahaman terhadap dinamika emosional dalam komunikasi manusia.

Pre-processing text adalah tahap pertama dari metode NLP pada dokumen teks. *Pre-processing text* mengubah teks yang tidak tertata menjadi data yang sesuai serta layak diproses (Katryn, 2020). Tahap *pre-processing text* melibatkan berbagai proses, diantaranya:

2.5.1 Parsing

Parsing adalah proses membagi sebuah urutan dokumen menjadi bagian-bagian tersendiri. Tujuannya untuk menghasilkan representasi struktural yang lebih struktur dan dapat dimengerti oleh mesin.

2.5.2 Case Folding

Case folding adalah proses mengonversi teks dengan huruf kecil yang berfungsi sebagai bentuk standar.

2.5.3 Tokenizing

Tokenizing, juga dikenal tahap *lexical analysis*, adalah tahap membagi teks ke dalam token. Token dapat berupa kata, frasa, atau karakter. Selain itu, proses ini menghilangkan angka dan tanda baca yang tidak dianggap berpengaruh pada proses pengolahan teks.

2.5.4 Stemming

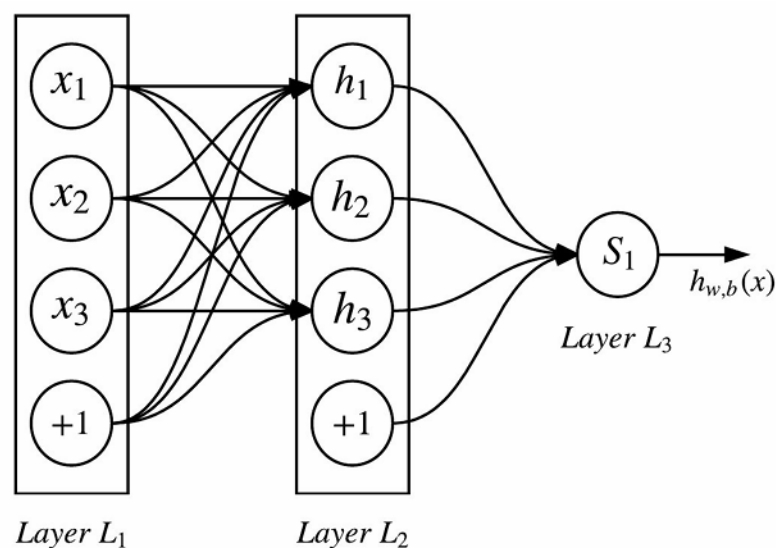
Stemming adalah tindakan mengonversi kalimat menjadi bentuk dasar atau seperti kata dalam kamus. Tujuannya adalah mengurangi variasi kata dan meningkatkan representasi pada kata.

2.5.5 Stopword Removal

Stopword removal adalah langkah menghapus istilah-istilah umum dan kurang memiliki informasi berharga, seperti konjungsi dan artikel.

2.6 Neural Network

Neural network merupakan sebuah model matematika menyalami cara berpikir manusia. Dibentuk dari banyak *neuron* yang terhubung satu sama lain dan tersusun dalam lapisan-lapisan. *Neural network* dapat mempelajari tugas-tugas tertentu, seperti klasifikasi dengan menyesuaikan bobot koneksi antar *neuron*. (Zhang et al., 2018).



Gambar 2. 2 Feedforward Neural Network

Contoh sederhana dari *feedforward neural network* bisa dilihat pada gambar 2.2 di atas, *neural network* ini terdiri dari tiga *layers* L_1 , L_2 , dan L_3 . L_1 berfungsi sebagai lapisan *input* yang menerima vektor *input* (x_1 , x_2 , x_3) dan x_{+1} . L_3 merupakan lapisan *output* yang menghasilkan vektor *output* (s_1). L_2 merupakan *hidden layer*, yang *output*-nya tidak kelihatan sebagai keluaran jaringan. Lingkaran yang ada pada L_1 merupakan elemen di *input vector*, sedangkan lingkaran pada L_2 dan L_3 merupakan *neuron*, elemen komputasi dasar dari *neural network*. Garis di antara dua *neuron* mewakili koneksi untuk aliran informasi, Setiap koneksi dihubungkan dengan sebuah *weight*, nilai yang mengendalikan sinyal antara dua *neuron*.

Neuron membaca *output* dari *neuron* di lapisan sebelumnya, memproses informasi, dan kemudian menghasilkan *output* ke *neuron* di lapisan berikutnya.

Pada gambar 2.2 jaringan netral mengubah bobot berdasarkan contoh pelatihan $(x^{(i)}, y^{(i)})$. Setelah proses pelatihan, akan membentuk hipotesis yang kompleks $h_{w,b}(x)$ yang sesuai dengan data.

Pada *hidden layer*, bisa dilihat setiap *neuron* pada L_2 mengambil *input* x_1, x_2, x_3 dan x_{+1} dari L_1 , dan *output* nilai $f(W^t x) = f(\sum_{i=1}^3 W_i x_i + b)$ dengan fungsi aktivasi f . W_i adalah bobot dari koneksi; b merupakan *bias*; dan f biasanya *non-linear*. Pilihan umum dalam f yaitu *sigmoid function*, *hyperbolic tangent function* (tanh), dan *rectified linear function* (ReLU). Persamaan adalah sebagai berikut.

$$f(W^t x) = \text{sigmoid}(W^t x) = \frac{1}{1 + \exp(-W^t x)}$$

$$f(W^t x) = \tanh(W^t x) = \frac{e^{W^t x} - e^{-W^t x}}{e^{W^t x} + e^{-W^t x}}$$

$$f(W^t x) = \text{ReLU}(W^t x) = \max(0, W^t x)$$

Gambar 2. 3 Persamaan *Sigmoid Function*, Tanh, dan ReLu

Fungsi *sigmoid* mengambil angka *real* dan menyempitkan ke nilai dalam rentang 0 dan 1. Namun, *non-linearity* dari *sigmoid* kurang populer karena aktivitas gradiennya hampir nol dan aliran informasi akan terputus. Fungsi tanh sering lebih disukai karena rentang *outputnya* berpusat pada nol, $[-1, 1]$, daripada $[0, 1]$. Fungsi ReLu memiliki aktivitas sederhana, yaitu di-*threshold* pada nol ketika *input* kurang dari 0. Dibandingkan dengan fungsi *sigmoid* dan fungsi tanh, ReLu mudah dihitung, cepat konvergen pada pelatihan, dan memberikan kinerja yang setara atau lebih baik dari *neural network* (Zhang et al., 2018).

Pada L_3 , fungsi *softmax* bisa digunakan pada *output neuron*, yang merupakan generalisasi dari fungsi logistik yang memaksimumkan vektor X berdimensi- K dari nilai *real* sembarang menjadi vektor berdimensi- K $\sigma(X)$ dari nilai *real* dalam rentang $(0, 1)$ yang berjumlah 1. Persamaan dari fungsi sebagai berikut.

$$\sigma(X)_j = \frac{e^{x_j}}{\sum_{k=1}^K e^{x_k}} \quad \text{for } j = 1, \dots, K$$

Gambar 2. 4 Persamaan Fungsi *Softmax*

Dengan menghubungkan semua *neuron*, *neural network* pada gambar 2.2 memiliki parameter (W, b), yang merupakan himpunan dari bobot ($W^{(1)}, W^{(2)}$) dan bias ($b^{(1)}, b^{(2)}$). Di sini $W_{ij}^{(l)}$ mengacu pada bobot yang menghubungkan *neuron* j di *layer* ke l dengan *neuron* i di *layer* ke- $(l+1)$ sementara $b_i^{(l)}$ adalah *bias* yang terkait dengan *neuron* i di *layer* ke- $(l+1)$.

2.7 Deep Learning

Deep learning merupakan suatu bidang dari *machine learning*, yang memiliki esensi sebagai suatu *neural network* dengan tiga atau lebih lapisan. *Neural network* ini dirancang untuk meniru cara kerja otak manusia, yang memungkinkannya belajar dari data dalam jumlah besar. Dengan mendorong pengembangan berbagai aplikasi dan *Artificial Intelligence* (AI), *deep learning* telah memungkinkan otomatisasi yang lebih baik dan pelaksanaan tugas-tugas fisik tanpa perlu campur tangan manusia.

Deep learning memiliki kemampuan dalam mempelajari suatu representasi fitur yang kompleks secara otomatis. Dalam *deep learning*, komputer akan mengembangkan pemahaman terhadap berbagai model dan melakukan klasifikasi tugas dengan memanfaatkan informasi yang terdapat dalam data yang dikumpulkan. Jenis data yang dapat digunakan bisa seperti gambar, sinyal suara, dan teks. Teknik-teknik *deep learning* dapat dibagi menjadi tiga kategori utama, yakni:

2.7.1 Deep Supervised Learning

Keunggulan dari teknik ini adalah kemampuannya untuk mengumpulkan data atau menghasilkan *output* data dari informasi yang sudah ada sebelumnya. Teknik ini berurusan dengan data berlabel. Teknik ini memiliki Kumpulan *input* dan *output* yang dihasilkan $(x_t, y_t) \sim p$. Sebagai contoh, agen menebak $\hat{y}_t = f(x_t)$ jika *input* adalah x_t dan akan mendapatkan sebagai nilai *loss*. Selanjutnya, parameter jaringan secara berulang diperbarui oleh agen untuk

mendapatkan estimasi yang ditingkatkan untuk *output* yang diinginkan. Contoh penggunaan *supervised* seperti, model *Recurrent Neural Networks* (RNNs), *Convolution Neural Networks* (CNNs), dan *Deep Neural Networks* (DNNs).

2.7.2 Deep Semi-Supervised Learning

Dalam teknik ini, proses pembelajaran didasarkan pada dataset *semi-labeled*. Salah satu keunggulan dari teknik ini adalah meminimalkan jumlah data. Mengklasifikasi dokumen teks adalah salah satu contoh aplikasi populer dari teknik *semi-supervised learning*. Karena sulitnya dalam mendapatkan dokumen teks berlabel dalam jumlah yang besar, *semi-supervised learning* sangat cocok untuk tugas klasifikasi dokumen teks.

2.7.3 Deep Unsupervised Learning

Teknik ini memungkinkan mesin belajar tanpa adanya data berlabel yang tersedia, dan mempelajari fitur-fitur atau representasi objek yang diperlukan untuk menemukan struktur atau hubungan dalam data *input*. RNNs, termasuk juga pendekatan dengan *Gated Recurrent Units* (GRUs) dan *Long Short-Term Memory* (LSTM) juga telah digunakan untuk *unsupervised learning* dalam berbagai penerapan.

2.8 Word Embedding

Word embedding adalah representasi teks yang dipelajari, kata-kata dengan makna yang mirip memiliki representasi serupa. Pendekatan ini dianggap sebagai salah satu terobosan penting dalam *deep learning* pada masalah NLP (Brownlee Disclaimer, 2017). *Word embedding* merupakan cara yang digunakan untuk mengubah kata-kata menjadi vektor nilai *real* dalam ruang vektor yang telah ditentukan sebelumnya. Setiap kata dihubungkan dalam satu vektor, sedangkan nilai-nilai vektor dipelajari melalui penggunaan *neural network*, sehingga membuat teknik ini menjadi bagian dari bidang *deep learning*.

Metode *word embedding* mempelajari representasi vektor nilai *real* untuk kosakata berukuran tetap yang telah ditentukan sebelumnya. Proses pembelajaran dapat sama seperti *neural network* pada suatu tugas, seperti klasifikasi pada teks. Terdapat tiga teknik yang bisa digunakan untuk mempelajari *word embedding* dari data teks, yaitu:

2.8.1 Embedding Layer

Embedding layer merupakan *word embedding* yang belajar secara simultan dalam model *neural network* dalam tugas-tugas NLP tertentu, seperti *language modelling* atau klasifikasi dokumen (Brownlee Disclaimer, 2017). Setiap teks harus dibersihkan dan dipersiapkan sehingga setiap kata adalah *one hot encoded*. Ukuran ruang vektor bisa 50, 100, dan 300 dimensi sesuai ditentukan oleh model. Vektor diinisialisasi dengan angka acak kecil.

One hot encoded digunakan dipetakan ke vektor kata. Jika model *multilayer perceptron* digunakan, maka vektor kata digabungkan sebelum dimasukkan sebagai *input* ke model. Jika RNN digunakan, maka setiap kata dapat diambil sebagai satu masukan dalam satu urutan. Pendekatan pembelajaran dengan *embedding layer* memerlukan banyak data dan bisa jadi lambat, namun akan mempelajari *embedding* yang ditargetkan untuk data tek tertentu dan tugas NLP (Brownlee Disclaimer, 2017).

2.9 Framework

Framework adalah struktur yang digunakan untuk mengembangkan *website* yang memudahkan dalam menulis kode (Setiawan, 2021). Penggunaan *framework* membuat pembuatan program menjadi efisien, singkat, serta terorganisir. Lalu, *framework* memiliki fungsi untuk mempercepat pembuatan *web*, meningkatkan keamanan *web*, dan mempermudah dalam memperbaiki dan merawat *website*. Penelitian ini menggunakan *framework* streamlit untuk membangun *website*.

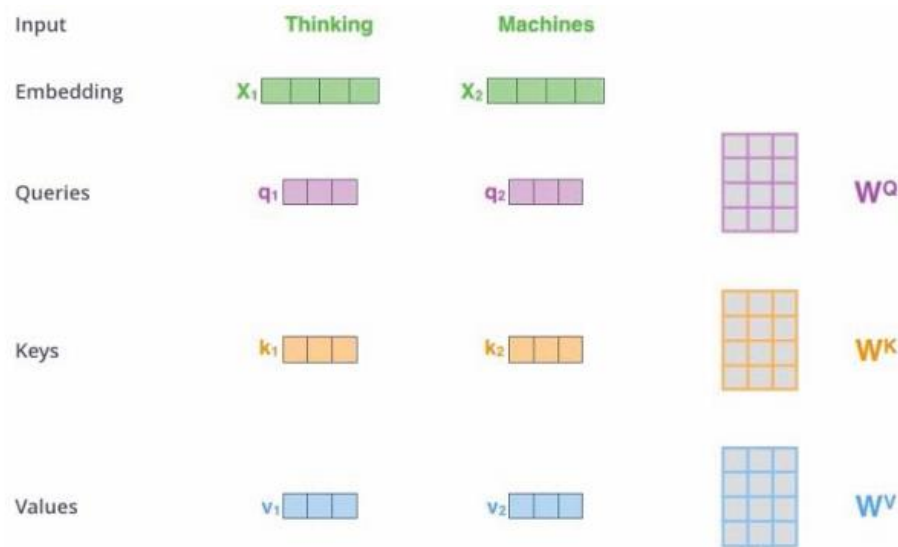
Streamlit adalah suatu kerangka kerja yang dikembangkan berbasis Python yang berfungsi untuk membuat *web* interaktif. *Framework* seperti streamlit memungkinkan pengguna menciptakan *website* dengan mudah tanpa harus menulis banyak baris kode. Streamlit khususnya dirancang untuk memudahkan penggunaan Python dalam membangun aplikasi *web* dengan tampilan menarik dan interaktif.

2.10 BERT

BERT (*Bidirectional Encoder Representation from Transformer*) adalah model bahasa *pre-trained* yang dikembangkan oleh Google untuk memahami konteks kata dalam kalimat dengan lebih baik. BERT mampu mencapai representasi fitur global dan lokal terbesar dari sebuah urutan (R. Abas et al., 2022). Arsitektur model BERT adalah *multilayer bidirectional transformer encoder*, dan

belajar untuk memprediksi representasi karakteristik dari urutan *input* (*contextualized embedding*) (Zhang, 2023).

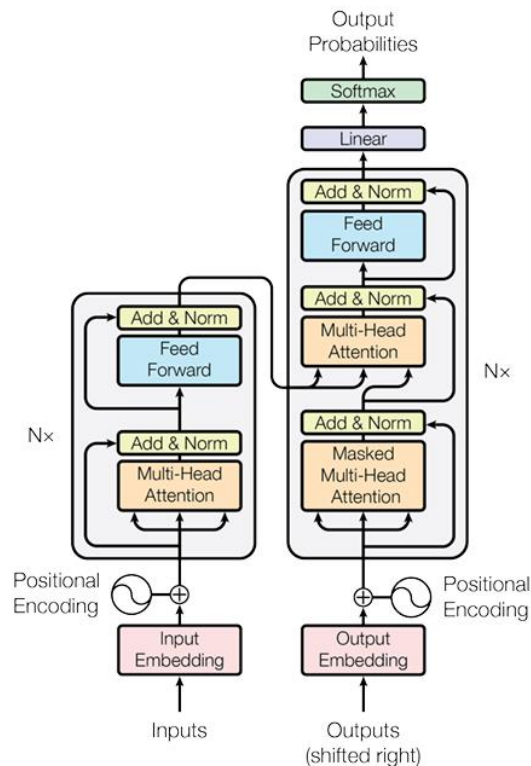
Transformer adalah modul inti yang membentuk BERT, dan mekanisme *attention* adalah bagian paling penting dari *Transformer*. Fungsi *attention* bisa dijelaskan sebagai suatu proses yang menghubungkan *query* dengan serangkaian pola *key-value* menjadi *output*, yaitu *query*, *keys*, *values*, serta *output* yang direpresentasikan sebagai vektor (Vaswani et al., 2017). Fungsi utama dari mekanisme *attention* adalah untuk memungkinkan *neural network* memfokuskan “*attention*” pada bagian *input*, yaitu untuk membedakan efek dari berbagai bagian *input* pada *output* (Zhang dan Qi, 2022).



Gambar 2. 5 *Self Attention*

Gambar di atas menunjukkan *self-attention*, setiap *self-attention* menghitung tiga vektor *query*, *key*, dan *value*. Dimensi dari setiap vektor adalah 512 dimensi, yang merupakan hasil perkalian vektor *embedding input* dengan suatu matriks. Matriks diinisialisasi secara acak pada awalnya, kemudian konten spesifik dari matriks diperoleh melalui pelatihan. Dimensi matriks adalah (64, 512) dan nilai matriks selalu diperbarui selama proses *backpropagation*. Dengan menggunakan matriks untuk menghitung matriks *query*, *key*, dan *value*, dan kemudian mengalikan nilai *embedding* dengan ketiga matriks tersebut dapat meningkatkan kecepatan perhitungan. Lalu, mengalikan matriks Q dan K yang diperoleh dengan suatu konstanta, melakukan operasi *softmax*, dan akhirnya mengalikan matriks V (Man dan Lin, 2021).

Transformer memiliki dua komponen utama dalam arsitekturnya, yaitu *encoder* dan *decoder*.



Gambar 2. 6 Arsitektur Model *Transformer*

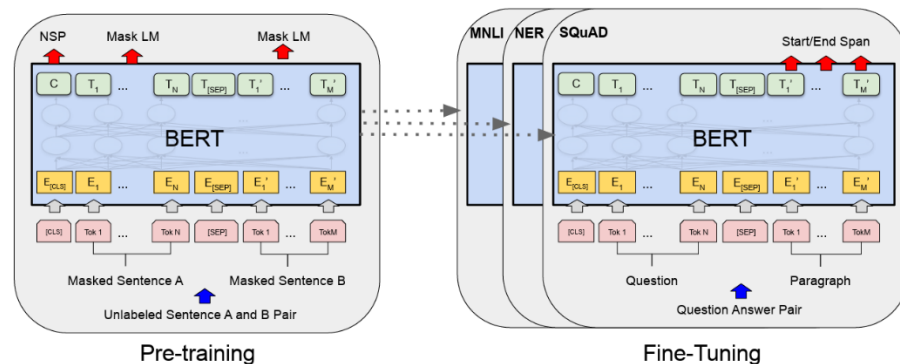
1. Encoder

Encoder pada BERT memiliki peran kunci dalam menghasilkan representasi vektor yang kontekstual dan kaya dari *input* teks. *Encoder* mencakup tumpukan $N = 6$ lapisan yang sama. Struktur *transformer* pada *encoder* BERT terdiri dari beberapa lapisan, setiap lapisan terdapat mekanisme *self-attention* yang memungkinkan model untuk menampilkan bobot dinamis untuk seluruh kata dalam kalimat. *Output* dari *sub-layer* adalah $\text{LayerNorm}(\mathcal{X} + \text{Sublayer}(x))$, dimana $\text{Sublayer}(x)$ adalah fungsi yang diimplementasikan dari *sub-layer* itu sendiri. Untuk memfasilitasi koneksi residual, semua *sub-layer* serta *embedding layer* menghasilkan *output* dengan dimensi $d_{\text{model}} = 521$ (Vaswani et al., 2017).

2. Decoder

Decoder umumnya ditemukan dalam arsitektur *transformer* untuk tugas penerjemahan mesin atau generalisasi teks di mana model harus

menghasilkan urutan kata-kata sebagai *output*. *Decoder* dapat pula berisi tumpukan $N = 6$ lapisan yang identik. *Decoder* menyisipkan *layer* ketiga yang melakukan *multi-head attention* terhadap *output* dari susunan *encoder*. *Self-attention* dimodifikasi dalam tumpukan *decoder* untuk mencegah posisi-posisi yang ada di posisi berikutnya (Vaswani et al., 2017).



Gambar 2. 7 Pre-Training dan Fine-Tuning BERT

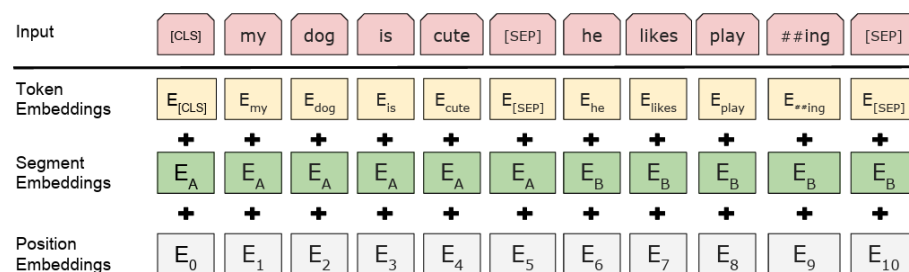
Sumber (Devlin et al., 2019)

Pre-training dan *fine-tuning* dalam BERT merupakan dua *framework* dengan fungsi yang berbeda. Selama tahap *pre-training*, kerangka kerja ini diajarkan dengan menggunakan dataset yang tidak berlabel untuk berbagai latihan. Saat dilakukan *fine-tuning*, proses inisialisasi model BERT dilakukan dengan parameter *pre-training*, kemudian semua parameter di *fine-tuning* menggunakan data berlabel dari tugas yang sesuai. Setiap tugas *downstream* mempunyai metode *fine-tuned* sendiri, walaupun awalnya menggunakan parameter *pre-trained* yang sama. (Devlin et al., 2019). Ada dua varian model *size* dalam BERT, yakni BERT_{BASE} dan BERT_{LARGE}. BERT_{BASE} memiliki 12 *layer*, *hidden size* sebesar 768, 12 *self-attention layer heads*, dan 110M parameter. Untuk BERT_{LARGE} 24 *layer*, *hidden size* sebesar 1024, 16 *self-attention layer heads*, dan 340M parameter (Devlin et al., 2019).

2.10.1 Input/Output Representation

Untuk membuat BERT menangani berbagai tugas *down-stream*, *input* representasi dapat secara jelas mewakili satu kalimat dan sepasang kalimat dalam satu urutan token. Token pertama dalam setiap urutan merupakan token

klasifikasi khusus ([CLS]). Token ini digunakan pada *hidden state* terakhir sebagai representasi urutan agregat untuk tugas klasifikasi. Pasangan kalimat digabungkan dalam satu urutan. Terdapat dua cara membedakan kalimat. Pertama, dengan memisahkan token khusus ([SEP]). Kedua, dengan menambahkan embedding yang dipelajari di setiap token yang menunjukkan itu kalimat A atau kalimat B. Seperti pada gambar 2.8, input embedding sebagai E . vektor hidden terakhir dari token khusus [CLS] sebagai $C \in \mathbb{R}^H$, dan vektor hidden terakhir untuk i^{th} sebagai input token $T_i \in \mathbb{R}^H$. Untuk token yang diberikan, representasi input dibangun dengan menjumlahkan token, segmen, dan posisi embeddings (Devlin et al., 2019).



Gambar 2. 8 Representasi *Input* BERT

Sumber (Devlin et al., 2019)

Token *embedding* bertugas mengubah kata ke dalam vektor representasi ukuran tetap. Setiap kata direpresentasikan sebagai vektor berdimensi 768 dalam kasus BERT. Teks *input* di tokenisasi sebelum melewati token *embedding layer*. Selain itu, token yang ada ditambahkan di awal [CLS] dan akhir [SEP] frasa yang telah di tokenisasi. Token-token ini merupakan representasi *input* untuk tugas klasifikasi dan untuk mengisolasi sepasang teks *input* (R. Abas et al., 2022).

BERT dapat menyelesaikan tugas-tugas NLP, termasuk klasifikasi teks untuk memberikan sepasang teks *input*. Contohnya seperti klasifikasi dua potongan teks yang secara semantic mirip. Sepasang teks *input* ini hanya dihubungkan dan dimasukkan ke dalam model yang diperkenalkan.

Position embedding digunakan untuk menjelaskan posisi kata-kata dalam sebuah kalimat. Dengan begitu, keterbatasan *transformer* yang tidak mampu menangkap informasi urutan dapat diatasi. Jadi, *positional embedding*

memungkinkan BERT untuk memahami teks *input* yang diberikan. Untuk menghasilkan representasi tunggal, representasi ini dijumlahkan secara elemen. Representasi ini menjadi representasi *input* yang melewati *encoder layer* BERT (R. Abas et al., 2022).

2.10.2 Pre-Trained BERT

BERT menggunakan dua *unsupervised tasks* pada proses *pre-trained*, yaitu:

1. Masked Language Modelling (MLM)

Masked Language Modelling adalah metode yang memungkinkan BERT untuk mempelajari bahasa dengan cara yang seimbang antara kiri dan kanan (*bidirectional learning*) (Muller, 2022). Hal ini penting untuk tugas-tugas NLP seperti klasifikasi teks, memprediksi sentimen, dan lainnya, di mana pemahaman konteks dan hubungan antara kata sangat penting.

2. Next Sentence Prediction (NSP)

Next Sentence Prediction merupakan tugas *pre-training* awal yang digunakan dalam model BERT. Tujuannya yaitu untuk membuat model memahami hubungan antara dua kalimat dalam suatu teks (Sun et al., 2021).

2.11 Convolutional Neural Network

Convolution Neural Network (CNN) merupakan jaringan saraf dengan lapisan konvolusi untuk menyaring *input* menjadi informasi yang berguna. Awalnya CNN diterapkan dalam *computer vision* untuk menangkap lokal fitur. Pada tugas NLP, vektor kata (*embeddings*) dapat dimasukkan ke 1D CNN, daripada piksel gambar, dan CNN dapat secara otomatis menangkap dan mengidentifikasi fitur-fitur penting dengan sangat akurat (Zhang, 2023). Selain itu, model CNN juga telah mencapai hasil yang sangat baik dalam *semantic parsing*, *search query retrieval*, dan *sentence modelling* (Kim, 2014).

Dalam NLP, kalimat atau dokumen teks direpresentasikan sebagai matriks, di mana setiap baris berhubungan dengan kata (karakter) yang direpresentasikan oleh *word embedding vector*. Vektor-vektor ini menangkap makna semantik dari kata-kata dalam ruang *low-dimensional*. Hal ini dapat digunakan oleh 1D CNN untuk memproses urutan *word embedding*. Mengolah urutan tersebut seperti sebuah

“gambar” satu dimensi di mana “lebar” filter sama dengan dimensi *word embedding*, dan “tinggi” (ukuran wilayah) mencakup jumlah kata yang tetap. Hal ini memungkinkan CNN untuk mempelajari pola di seluruh urutan kata, sebagaimana CNN mempelajari pola pada gambar.

2.11.1 1D Convolutional Neural Network

Arsitektur umum di mana setiap kata dalam dokumen direpresentasikan sebagai *embedding vector*, *convolutional layer* tunggal dengan m filter diterapkan, menghasilkan vektor m -dimensi untuk setiap dokumen n -gram. Vektor-vektor tersebut digabungkan menggunakan *max-pooling* yang diikuti dengan aktivasi ReLu. Hasilnya kemudian diteruskan ke *linear layer* untuk klasifikasi akhir (Jacovi et al., 2018).

Untuk n -words input teks w_1, \dots, w_n menyematkan setiap simbol menjadi dimensi vektor d , menghasilkan *word vectors* $w_1, \dots, w_n \in R^d$. Hasil dari matriks $d \times n$ kemudian masuk ke dalam *convolutional layer* di mana menerapkan *sliding window* diatas teks. Untuk setiap l -words n -gram:

$$u_i = [w_i, \dots, w_{i+l-1}] \in R^{d \times l}; 0 \leq i \leq n - l$$

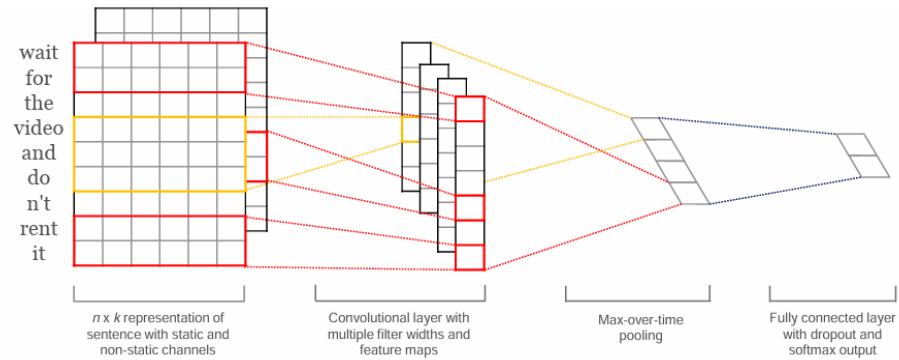
Setiap filter $f_j \in R^{d \times l}$ menghitung (u_i, f_j) . Hasil matriks dari *convolution* $\mathbf{F} \in R^{d \times l}$. Menerapkan *max-pooling* di seluruh dimensi n -gram menghasilkan $p \in R^m$ yang dimasukkan kedalam ReLu *non-linearity*. Selanjutnya, linier *fully-connected layer* $\mathbf{W} \in R^{c \times m}$ menghasilkan pembagian atas kelas-kelas klasifikasi dari kelas terkuat dikeluarkan (Jacovi et al, 2018). Secara formal:

$$u_i = [w_i, \dots, w_{i+l-1}]$$

$$F_{ij} = \langle u_i, f_j \rangle$$

$$p_j = \text{ReLu}(\max F_{ij})$$

$$\mathbf{o} = \text{softmax}(\mathbf{Wp})$$



Gambar 2. 9 Model Arsitektur

Sumber (Kim, 2014)

Model arsitektur pada gambar 2.10 di atas merupakan sedikit variasi arsitektur CNN dari Collobert et al. (2011). Misalkan $x_i \in \mathbb{R}^k$ merepresentasikan kata ke- i dalam sebuah kalimat dengan panjang n , sebagai berikut:

$$x_{1:n} = x_1 \oplus x_2 \oplus \dots \oplus x_n,$$

dimana:

\oplus = operator penggabungan

Keseluruhan, misalnya $x_{i:i+j}$ merujuk kepada penyatuan kata-kata $x_i, x_{i+1}, \dots, x_{i+j}$. Proses *convolution* mengikutsertakan penyaring $w \in \mathbb{R}^{hk}$, dimana diimplementasikan terhadap *window* h kata untuk memunculkan ciri-ciri tambahan. Contohnya nilai fitur c_i diperoleh melalui sebuah window kata-kata $x_{i:i+h-1}$ oleh:

$$c_i = f(w \cdot x_{i:i+h-1} + b)$$

dimana:

$b \in \mathbb{R} = \text{bias}$

$f = \text{non-linear function}$

Filter ini diterapkan ke setiap jendela kata yang mungkin dalam kalimat $\{x_{1:h}, x_{2:h+1}, \dots, x_{n-h+1:n}\}$ untuk menghasilkan *feature map*.

$$c = [c_1, c_2, \dots, c_{n-h+1}]$$

dengan; $c \in \mathbb{R}^{n-h+1}$.

Selanjutnya, terapkan proses *max-over-time pooling* terhadap fitur *map* dengan nilai maksimum $\hat{c} = \max\{c\}$ yang sesuai dengan filter ini. Ide di baliknya yaitu mengambil fitur terpenting, yang mempunyai nilai tertinggi di setiap fitur *map*. Pola penggabungan secara otomatis dapat bekerja pada kalimat-kalimat yang panjangnya bervariasi (Kim, 2014).

Lapisan-lapisan CNN yang digunakan diantaranya sebagai berikut:

1. *Layer Embedding*

Lapisan ini digunakan untuk mengkonversi *input* teks menjadi representasi vektor *embedding*.

2. *Convolutional Layers*

Lapisan ini terdiri dari beberapa lapisan konvolusi dengan jumlah kernel yang ditentukan dan ukuran kernel yang berbeda. Setiap kernel akan mempelajari fitur-fitur dari teks dengan ukuran kernel yang sesuai.

3. ReLu

Hasil dari operasi konvolusi akan melewati fungsi aktivasi ReLu, dengan begitu model dapat belajar dengan lebih efisien dan dapat menangkap hubungan non-linear yang kompleks dalam data *input*.

4. *Max-Pooling Layer*

Lapisan ini dilakukan untuk mereduksi dimensi data dengan mempertahankan fitur-fitur yang paling signifikan.

5. *Flattening*

Hasil dari *max-pooling* akan di-*flatten* menjadi vektor satu dimensi sebagai *input* ke lapisan *fully connected*.

6. *Fully Connected Layer*

Vektor hasil *flattening* melewati beberapa lapisan ini yang memiliki parameter-parameter yang dapat dipelajari.

7. *Output Layer*

Lapisan ini menghasilkan prediksi kelas-kelas dari data *input*.

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis

Analisis dan perancangan sistem adalah suatu pendekatan yang sistematis untuk merancang dan mengembangkan sistem. Analisis mendefinisikan kebutuhan terkait sistem yang akan dikembangkan. Dengan pendekatan yang sistematis, diharapkan sistem yang dikembangkan dapat memenuhi kebutuhan pengguna secara efektif dan efisien.

3.1.1 Analisis Masalah

Bahasa Indonesia memiliki kekayaan kata dan ekspresi yang luas untuk menyatakan emosi. Emosi dari suatu kalimat bahasa Indonesia dapat dikenali melalui kata kata sifat yang terdapat dalam kalimat tersebut. Variasi dalam makna dan konteks penggunaannya menambah kompleksitas dalam mendeteksi emosi yang terkandung dalam teks. Beberapa teks mungkin memiliki emosi yang ambigu atau campuran, sehingga sulit untuk diklasifikasikan secara tepat. Selain itu, penggunaan kata-kata sifat tidak selalu mencerminkan emosi secara langsung, karena konteks dan penekanan dalam kalimat juga dapat memengaruhi interpretasi emosi. Oleh karena itu, penelitian ini diharapkan mempermudah dalam mendeteksi emosi yang terkandung dalam teks bahasa Indonesia.

3.1.2 Analisis Kebutuhan

Analisis kebutuhan adalah proses identifikasi, pemahaman, dan spesifikasi kebutuhan yang harus dipenuhi oleh sistem yang akan dikembangkan. Analisis kebutuhan dibagi menjadi dua bagian utama, yaitu fungsional dan non-fungsional.

1. Kebutuhan fungsional

Kebutuhan fungsional adalah spesifikasi fitur dan fungsi yang mesti dimiliki setiap sistem demi mencapai tujuan utama. Penelitian ini memiliki kebutuhan fungsional utama, yaitu:

- a. Menerima teks bahasa Indonesia sebagai input.
- b. Menghilangkan elemen-elemen yang tidak relevan seperti tanda baca, simbol atau URL.
- c. Menggunakan BERT untuk menghasilkan representasi vektor dari teks.

- d. Menggunakan model CNN untuk mendeteksi emosi dari representasi vektor yang dihasilkan oleh BERT sebagai inputan.
- e. Melatih model CNN dengan data yang diberi label emosi sedih, senang, marah, cinta, takut, dan terkejut.
- f. Menghasilkan keluaran berupa probabilitas dari kelas emosi yang berbeda.
- g. Menyediakan *UI/UX* yang menarik dan mudah untuk digunakan.

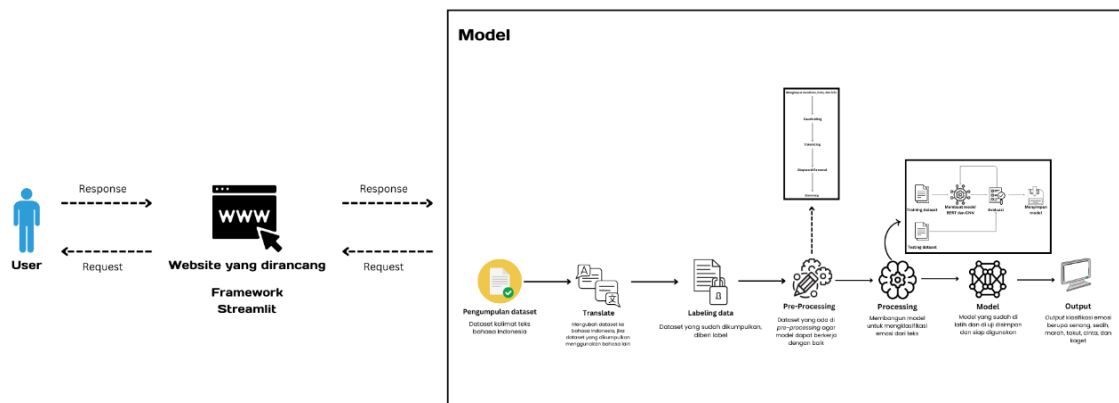
2. Kebutuhan non-fungsional

Kebutuhan non-fungsional adalah cara di mana sistem berjalan dan beroperasi dengan seharusnya. Penelitian ini memiliki beberapa kebutuhan non-fungsional, yaitu:

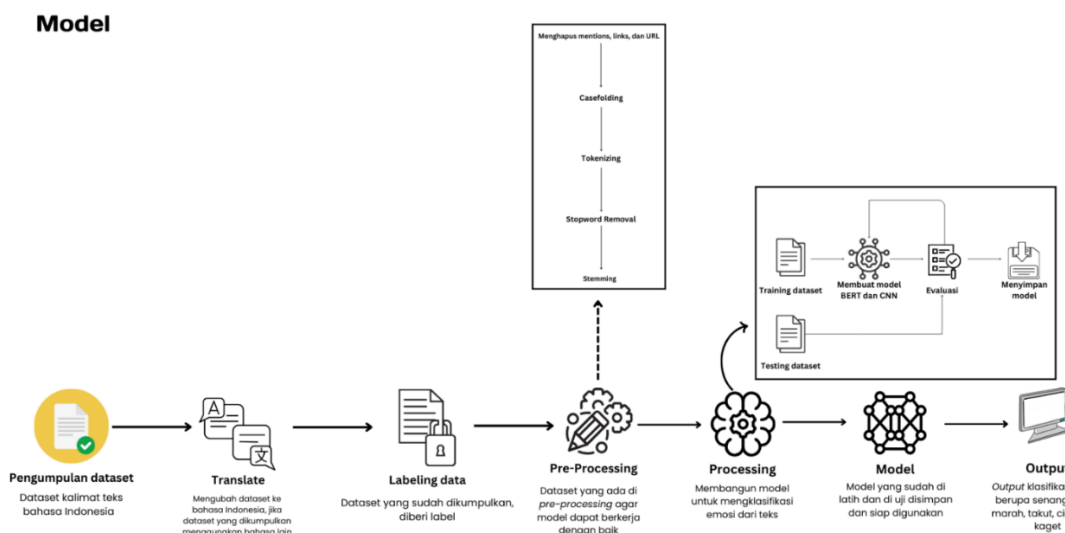
- a. Performa sistem dalam menerima inputan dan memproses data dalam waktu singkat.
- b. *User Interface* yang mudah dimengerti oleh pengguna.
- c. Hasil prediksi yang menunjukkan seberapa akurat dalam memprediksi emosi pada teks bahasa Indonesia.

3.2 Gambaran Umum Sistem

Sistem ini disusun berdasarkan analisis mendalam terhadap hasil penelitian. Tujuan utama yaitu untuk meningkatkan efisiensi dan efektivitas dari sistem. *User* akan menggunakan sistem yang berbasis *web* yang dibangun dengan *framework* streamlit, kemudian pembuatan model dimulai dari pengumpulan dataset yang diambil dari situs Kaggle. Kemudian data ditranslasi ke bahasa Indonesia, dan diberi label bilangan biner untuk setiap kelas emosi. Data tersebut akan diolah melalui proses *pre-processing* meliputi *case folding*, *tokenizing*, *stopword removal*, *stemming*, dan menghapus *mentions*, *links*, dan *URLs*. Selanjutnya, dilakukan implementasi dengan model BERT dan CNN pada data latih dan data uji sehingga didapatkan hasil evaluasi, lalu model disimpan dan *user* nantinya akan melihat *output* dari klasifikasi emosi pada teks bahasa Indonesia.



Gambar 3. 1 Arsitektur Umum Sistem



Gambar 3. 2 Tahapan Pembuatan Model

3.3 Analisis Dataset

Dataset yang digunakan pada penelitian ini merupakan teks kalimat bahasa Inggris yang diambil dari situs Kaggle yang dinamakan “*Emotions*” (Elgiriyeewithana, 2024). Dataset ini berisikan 300 ribu kumpulan pesan Twitter berbahasa Inggris yang dianotasi secara cermat dengan enam emosi dasar. Setiap entri dalam dataset ini terdiri dari segmen teks dari pesan Twitter dan label yang sesuai menunjukkan emosi yang disampaikan. Selanjutnya teks ini akan di ditranslasi ke bahasa Indonesia menggunakan Google Translate.

text,label	
0,i just feel really helpless and heavy hearted,4	
1,i've enjoyed being able to slouch about relax and unwind and frankly needed it after those last few weeks around the end of uni and the expo i have lately started to find myself feeling a bit listless which is never really a good thing,0	
2,i gave up my internship with the dmrg and am feeling distraught,4	
3,i dont know i feel so lost,0	
4,i am a kindergarten teacher and i am thoroughly weary of my job after having taken the university entrance exam i suffered from anxiety for weeks as i did not want to carry on with my work studies were the only alternative,4	
5,i was beginning to feel quite disheartened,0	
6,i would think that whomever would be lucky enough to stay in this suite must feel like it is the most romantic place on earth,2	
7,i fear that they won't ever feel that delicious excitement of christmas eve at least not in the same way i remember doing it,1	
8,i'm forever taking some time out to have a lie down because i feel weird,5	
9,i can still lose the weight without feeling deprived,0	
10,i try to be nice though so if you get a bitchy person on the phone or at the window feel free to have a little fit and throw your pen at her face,1	
11,i'm feeling a little like a damaged tree and that my roots are a little out of wack,0	
12,i have officially graduated i'm not feeling as ecstatic as i thought i would,1	
13,i feel like a jerk because the library students who all claim to love scrabble cant be bothered to participate and clearly scrabble is an inappropriate choice for a group of students whose native language isnt english,3	
14,i feel my portfolio demonstrates how eager i am to learn but some who know me better might call it annoyingly persistent,1	
15,i may be more biased than the next because i have a dependent life to take care of and to keep safe but i feel we all need to take care of ourselves as well,1	
16,i didn't feel terrific,1	
17,i miss all the others as well that feel that i wronged them and they will soon understand that i didn't,3	
18,i feel so stupid that i realise it so late,0	
19,i saunter through the airport terminals feeling that i have had an experience that renders the petty tribulations of everyday travel somehow far less significant,3	
20,i need to feel dangerous and pretty so here a striking dance pick deep in vogue minutes ago,3	

Gambar 3. 3 Dataset “Emotions” Berbahasa Inggris

teks,label	
0,aku hanya merasa sangat tidak berdaya dan berat hati,4	
1,Saya senang bisa bersantai dan bersantai dan sejujurnya membutuhkannya setelah beberapa minggu terakhir di sekitar akhir universitas dan pameran. Akhir-akhir ini saya mulai merasa sedikit lesu yang sebenarnya bukan hal yang baik,0	
2,saya berhenti magang di dmrg dan merasa putus asa,4	
3,entahlah aku merasa sangat tersesat,0	
4,saya seorang guru taman kanak-kanak dan saya benar-benar lelah dengan pekerjaan saya setelah mengikuti ujian masuk universitas. Saya menderita kecemasan selama berminggu-minggu karena saya tidak ingin melanjutkan pekerjaan. Studi adalah satu-satunya alternatif,4	
5,saya mulai merasa sangat kecewa,0	
6,menurutku siapa pun yang cukup beruntung untuk menginap di suite ini pasti merasa ini adalah tempat paling romantis di dunia,2	
7,saya khawatir mereka tidak akan pernah merasakan kegembiraan malam Natal yang nikmat, setidaknya tidak seperti yang saya ingat saat melakukannya,1	
8,saya selalu meluangkan waktu untuk berbaring karena saya merasa aneh,5	
9,saya masih bisa menurunkan berat badan tanpa merasa kekurangan,0	
10,aku mencoba untuk bersikap baik, jadi jika kamu menemui orang yang menyerahkan telepon atau di depan jendela, jangan ragu untuk bersikap sedikit fit dan melemparkan penamu ke wajahnya,1	
11,aku merasa seperti poton yang rusak dan aku sedikit rusak,0	
12,saya telah resmi lulus, saya tidak merasa gembira seperti yang saya kira,1	
13,saya merasa seperti orang brengsek karena siswa perpustakaan yang semuanya mengaku menyukai scrabble tidak mau ikut serta dan jelas scrabble adalah pilihan yang tidak pantas untuk sekelompok siswa yang bahasa ibunya bukan bahasa Inggris,3	
14,saya merasa portofolio saya menunjukkan betapa bersemangatnya saya untuk belajar, tetapi beberapa orang yang mengenal saya lebih baik mungkin menyebutnya sangat gijih,1	
15,saya mungkin lebih bias dibandingkan orang lain karena saya memiliki tanggung jawab hidup yang harus dijaga dan dijaga agar tetap aman, tetapi saya merasa kita semua juga perlu menjaga diri sendiri,1	
16,saya tidak merasa hebat,1	
17,aku juga merindukan semua orang yang merasa bahwa aku bersalah pada mereka dan mereka akan segera mengerti bahwa aku tidak bersalah,3	
18,aku merasa sangat bodoh sampai terlambat menyadarinya,0	
19,saya berjalan-jalan di terminal bandara dengan perasaan bahwa saya mempunyai pengalaman yang menjadikan kesengsaraan kecil dalam perjalanan sehari-hari menjadi tidak terlalu berarti,3	
20,saya perlu merasa berbahagia dan cantik jadi inilah pilihan tarian yang mencolok yang sedang populer beberapa menit yang lalu,3	

Gambar 3. 4 Hasil Dataset Setelah Translasi

Lalu, dataset dibagi menjadi 5 ribu per teks yang mewakili emosi yang diteliti. Jadi dataset penelitian ini berjumlah 30 ribu yang terdiri dari emosi sedih, senang, marah, cinta, takut, dan terkejut. Bentuk label dataset yang diambil melalui Kaggle yaitu angka 0 sampai dengan 5 yang merepresentasikan emosi pada suatu teks. Contohnya, jika teks emosi *sadness* akan diberi label angka 0, jika emosi *joy* akan diberi angka 1, lalu emosi *love* diberi angka 2, dan seterusnya. Untuk lebih jelasnya dapat dilihat pada gambar dibawah

text	label
i just feel really helpless and heavy hearted	4
ive enjoyed being able to slouch about relax and unwind and frankly needed it after those last few weeks around the end of uni and the expo i have lately st	0
i gave up my internship with the dmrg and am feeling distraught	4
i dont know i feel so lost	0
i am a kindergarten teacher and i am thoroughly weary of my job after having taken the university entrance exam i suffered from anxiety for weeks as i did i	4
i was beginning to feel quite disheartened	0
i would think that whomever would be lucky enough to stay in this suite must feel like it is the most romantic place on earth	2
i fear that they won't ever feel that delicious excitement of christmas eve at least not in the same way i remember doing it	1
im forever taking some time out to have a lie down because i feel weird	5
i can still lose the weight without feeling deprived	0
i try to be nice though so if you get a bitchy person on the phone or at the window feel free to have a little fit and throw your pen at her face	1
i'm feeling a little like a damaged tree and that my roots are a little out of wack	0
i have officially graduated im not feeling as ecstatic as i thought i would	1

Gambar 3. 5 Dataset Dengan Label Angka Berurutan

Proses selanjutnya yaitu melabelisasi ulang pada dataset dengan metode manual setelah teks ditranslasi ke bahasa Indonesia. Tiap teks kalimat yang mewakili salah satu emosi diberi nilai 1 dan jika tidak mewakili salah satu emosi diberi nilai 0. Contoh dataset yang sudah dilabelisasi dapat dilihat pada Tabel 3.1 di bawah ini.

Tabel 3. 1 Contoh Labelisasi Dataset

teks	sedih	senang	cinta	marah	takut	terkejut
aku selalu meluangkan waktu untuk berbaring karena aku merasa aneh	0	0	0	0	0	1
saya merasa sangat letih dan tidak antusias dengan kehidupan hampir setiap hari	1	0	0	0	0	0
aku merasakan kerinduan setiap jiwa kerinduan akan kebutuhan kita	0	0	1	0	0	0
saya merasa meriah dan memberi tahun ini jadi akan ada banyak hadiah dan kontes untuk halloween dan liburan di bulan desember	0	1	0	0	0	0
saya merasa frustrasi dengan kurangnya hal yang harus dilakukan tetapi sebagian besar waktu saya menghargai istirahat dari kehidupan sekolah pascasarjana saya yang sibuk dan sangat terhubung	0	0	0	1	0	0
aku tak pernah mengira aku akan merasa seperti ini kupikir aku akan lebih takut dari apa pun tapi ternyata tidak	0	0	0	0	1	0

3.4 Pre-Processing

Pre-Processing merupakan teknik untuk mempersiapkan teks yang tidak terstruktur menjadi data yang baik dan siap diolah. Tahapannya yaitu *case folding*, *tokenizing*, *stopwords removal*, *stemming*, menghapus *mentions*, *links*, dan *URLs*.

3.4.1 Case Folding

Case folding adalah proses mengonversi teks dengan huruf kecil yang berfungsi sebagai bentuk standar. Contohnya dapat ditemukan pada Tabel 3.2.

```
# Mengubah teks menjadi huruf kecil
text = text.lower()
```

Gambar 3. 6 Proses *Case folding*

Tabel 3. 2 Contoh *Case Folding*

Teks	Hasil <i>Case Folding</i>
Aku menyebutnya pernapasan kebahagiaan karena itu selalu membuatku merasa gembira	aku menyebutnya pernapasan kebahagiaan karena itu selalu membuatku merasa gembira
Aku belajar melepaskan sesuatu. Aku belajar merangkul orang-orang yang akan selalu mendukungku dan membuatku merasa istimewa	aku belajar melepaskan sesuatu. aku belajar merangkul orangorang yang akan selalu mendukungku dan membuatku merasa istimewa

3.4.2 Tokenizing

Tokenizing, juga dikenal tahap *lexical analysis*, adalah tahap membagi teks ke dalam token. Token dapat berupa kata, frasa, atau karakter.

```
def tokenize(text):
    tokens = nltk.tokenize.word_tokenize(text)
    return tokens

df_train['teks'] = df_train['teks'].apply(tokenize)
df_train.head()
```

Gambar 3. 7 Proses *Tokenizing*

Proses ini menggunakan *library* NLTK untuk membagi teks ke dalam token. Contoh pada tahap ini terdapat pada Tabel 3.3 di bawah ini.

Tabel 3. 3 Contoh *Tokenizing*

Teks	Hasil <i>Tokenizing</i>
aku merasa sangat aneh	[aku, merasa, sangat, aneh]
aku merasa sangat kewalahan	[aku, merasa, sangat, kewalahan]
menurutku kombo itu agak aneh	[menurutku, kombo, itu, agak, aneh]

3.4.3 *Stopword Removal*

Stopword removal adalah langkah menghapus istilah-istilah umum dan kurang memiliki informasi berharga, seperti konjungsi dan artikel. Kata-kata ini seringkali termasuk kata penghubung (seperti “dan”, “atau”, “di”, “ke”), kata ganti (seperti “saya”, “kamu”, “mereka”), dan kata-kata umum lainnya yang tidak memberikan makna khusus.

```
factory = StopWordRemoverFactory()
stopwords = factory.get_stop_words()
stopwords.append('aku') # Menambahkan kata 'aku' ke dalam daftar stopwords

def remove_stopwords(tokens):
    cleaned_text = []
    for text in tokens:
        if text not in stopwords:
            cleaned_text.append(text)
    return cleaned_text

df_train['teks'] = df_train['teks'].apply(remove_stopwords)
df_train.head()
```

Gambar 3. 8 Proses *Stopword Removal*

Library yang digunakan adalah Sastrawi untuk menghapus kata-kata yang tidak memberikan makna khusus. Proses ini bisa dilihat pada Tabel 3.4.

Tabel 3. 4 Contoh *Stopword Removal*

Teks	Hasil <i>Stopword Removal</i>
[saya, merasa, sedikit, takjub, dan, bersyukur, bisa, mendarat, di, tengah, kerumunan, seperti, itu]	[merasa, sedikit, takjub, bersyukur, mendarat, tengah, kerumunan]

[saya, merasa, mereka, pasti, terkesan, dengan, wawancara, saya, jika, mereka, mempekerjakan, saya, karena, mengetahui, bahwa, saya, akan, absen, pada, bagian, yang, sangat, penting, di, awal]	[merasa, terkesan, wawancara, mempekerjakan, mengetahui, absen, bagian, sangat, penting, awal]
[saya, merasa, sedikit, kewalahan, dengan, semua, hal, yang, perlu, diselesaikan, sebelum, conner, tiba, tetapi, saya, tahu, bahwa, entah, bagaimana, semuanya, akan, datang, bersamaan]	[merasa, sedikit, kewalahan, semua, perlu, diselesaikan, conner, tiba, tahu, entah, bagaimana, semuanya, datang, bersamaan]

3.4.4 Stemming

Stemming adalah tindakan mengonversi kalimat menjadi bentuk dasar atau seperti kata dalam kamus. Tujuannya adalah mengurangi variasi kata dan meningkatkan representasi pada kata.

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

def stemming(tokens):
    result = [stemmer.stem(text) for text in tokens]
    return result

df_train['teks'] = df_train['teks'].apply(stemming)
df_train.head()
```

Gambar 3. 9 Proses *Stemming*

Proses ini menggunakan *library* Python sastrawi dalam mencari kata dasar dan imbuhan yang ada pada teks. Contoh *stemming* dapat dilihat pada Tabel 3.5 di bawah.

Tabel 3. 5 Contoh *Stemming*

Teks	Hasil <i>Stemming</i>
[aku, selalu, meluangkan, waktu, berbaring, aku, merasa, aneh]	[aku, selalu, luang, waktu, baring, aku, rasa, aneh]
[merasa, sedikit, takjub, bersyukur, mendarat, tengah, kerumunan]	[rasa, sedikit, takjub, syukur, darat, tengah, kerumun]

3.4.5 Filtering

Proses ini merupakan pembersihan teks dari segala sesuatu yang dapat mempengaruhi hasil dari analisis, seperti menghapus *link*, *hashtag* (#), spasi yang berlebih, *mention* (@ *username*), dan menghapus *URLs* (*http/https*). *Library* Python digunakan pada proses ini diantaranya *regex* dan *string*. *Library regex* berfungsi untuk operasi *regular expression*, sedangkan *library string* digunakan untuk manipulasi *string*.

```
def preprocess_text(text):
    # Menghapus karakter tab, baris baru, back slice, dan karakter non-ASCII
    text = text.encode('ascii', 'replace').decode('ascii').replace('\t', " ").replace('\n', " ").replace('\u', " ").replace('\ ', " ")

    # Mengubah teks menjadi huruf kecil
    text = text.lower()

    # Menghapus mention, link, hashtag, dan URL yang tidak lengkap
    text = ' '.join(re.sub("([@#][A-Za-z0-9+])|(\w+:\/\/\S+)|(http\S+)|(https\S+)", " ", text).split())

    # Menghapus angka
    text = re.sub(r"\d+", "", text)

    # Menghapus tanda baca
    text = text.translate(str.maketrans("", "", string.punctuation))

    # Menghapus spasi di awal dan akhir teks
    text = text.strip()

    # Menghapus spasi berlebih
    text = re.sub("\s+", " ", text)

    return text

df_train['teks'] = df_train['teks'].apply(preprocess_text)
df_train.head()
```

Gambar 3. 10 Proses *Filtering*

3.5 Representasi Fitur Teks dengan BERT

Proses pertama yang dilakukan yaitu dengan tokenisasi teks, di mana teks dipisahkan menjadi token menggunakan *tokenizer* BERT. Lalu, jika jumlah token melebihi batas maksimal panjang *sequence* yang ditetapkan, token akan dipotong agar panjang *input* tetap konsisten. Token [CLS] ditambahkan di awal *sequence*, yang berguna untuk tugas-tugas klasifikasi, dan token [SEP] ditambahkan di akhir

sebagai pemisah jika ada lebih dari satu *sequence* yang diberikan sebagai *input*. Misal, diberi kalimat “saya sangat senang”. Kalimat ini akan di tokenisasi menjadi “[CLS]”, “saya”, “sangat”, “senang”, “[SEP]”.

Selanjutnya, setiap token dikonversi menjadi ID unik sesuai dengan indeks token dalam kamus BERT. *Mask input* digunakan untuk menandai token yang sebenarnya dengan nilai 1 dan *padding* dengan nilai 0. Lalu ID segmen digunakan oleh BERT untuk membedakan antara dua sekuens yang berbeda. Sebagai contoh dari token dikonversi menjadi ID (contoh ID), serta *mask* dan segmen *id* seperti pada gambar di bawah.

```
["[CLS]", "saya", "sangat", "senang", "[SEP]"]  
[101, 1001, 1002, 1003, 102]
```

Gambar 3. 11 Contoh Token yang Dikonversi Menjadi ID

```
input_ids: [101, 1001, 1002, 1003, 102, 0, 0, 0, 0, 0]  
input_mask: [ 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]  
segment_ids: [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Gambar 3. 12 Contoh Pembuatan *Mask Input* dan Segmen ID

Model BERT menerima ‘*input_ids*’, ‘*input_mask*’, dan ‘*segment_ids*’ sebagai *input*, kemudian menghasilkan representasi vektor (*hidden state*) melalui lapisan *transformer* untuk setiap token dalam *input*. Token [CLS] di lapisan terakhir dapat digunakan sebagai representasi dari keseluruhan sekuens untuk tugas-tugas klasifikasi.

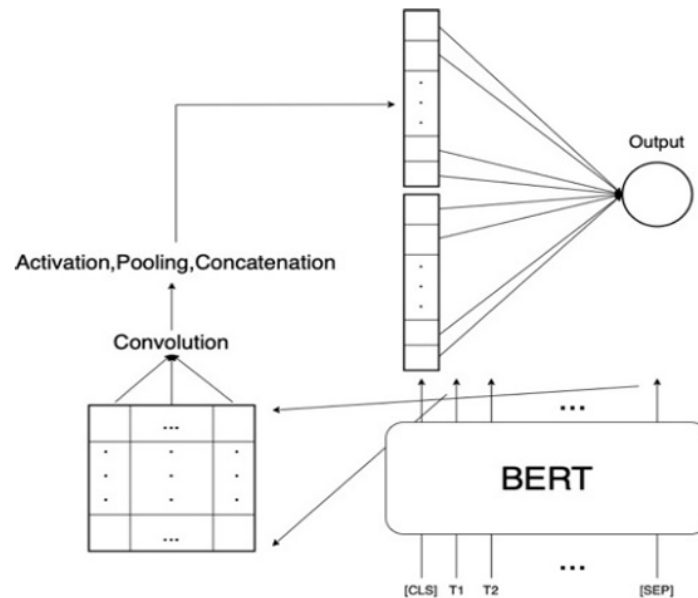
```
output_vector (dari token [CLS]): [0.2, 0.1, ..., 0.3] # Dimensi 768
```

Gambar 3. 13 Contoh *Output* Vektor dari Token ‘[CLS]’

Vektor ini kemudian dapat digunakan sebagai *input* untuk model CNN dalam tugas klasifikasi. Dengan begitu, model CNN akan memproses vektor-vektor untuk menghasilkan prediksi berdasarkan fitur yang telah dipelajari.

3.6 Klasifikasi dengan Model CNN

Setelah mendapatkan representasi fitur teks dengan BERT, *output* vektor yang dihasilkan oleh BERT akan diproses dengan model CNN untuk melakukan klasifikasi emosi.



Gambar 3. 14 Peta Sketsa dari Model BERT dan CNN

Sumber (Zhang, 2023)

Pada gambar 3.14 dapat dijelaskan sesuai dengan penelitian, pertama-tama *input* dari representasi BERT akan diproses pada *convolutional layer* CNN untuk mengekstrak fitur-fitur dari urutan kata dalam *embedding*. Setiap lapisan konvolusi menggunakan ReLu sebagai fungsi aktivasi. Kemudian diikuti dengan lapisan *max-pooling* untuk mereduksi dimensi dari fitur-fitur yang diekstraksi. Selanjutnya, hasil *max-pooling* dari setiap konvolusi diratakan menjadi satu dimensi untuk lapisan *fully connected*. Lapisan *output* terdiri dari 6 *neuron*, yang merepresentasikan kelas emosi yang diklasifikasi (sedih, senang, cinta, marah, takut, dan terkejut). Terakhir fungsi aktivasi *softmax* digunakan untuk menghasilkan distribusi probabilitas dari setiap kelas emosi. Fungsi *softmax* didefinisikan sebagai berikut.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

dimana:

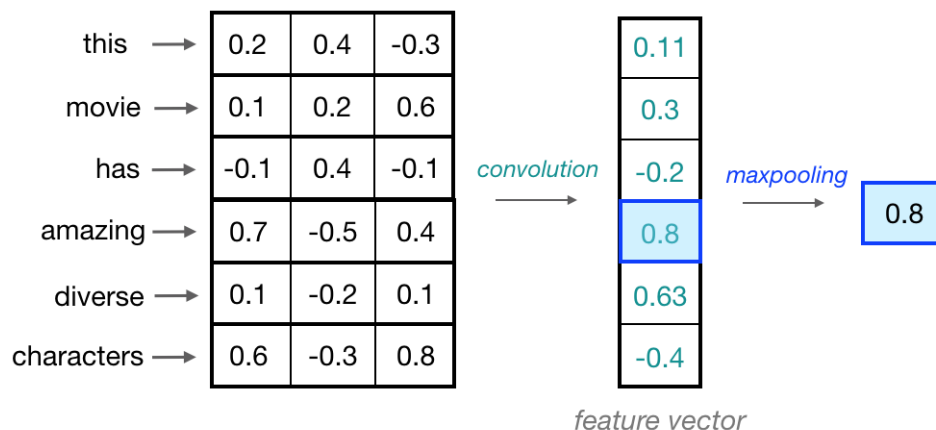
z = vektor logit

e^{z_i} = eksponensial logit kelas ke- i

$\sum_j e^{z_j}$ = jumlah eksponensial logit

3.6.1 1D Convolution

Untuk memproses seluruh rangkaian kata, kernel akan bergeser pada *word embedding* secara berurutan. Proses ini disebut *1D convolution* karena kernel hanya bergerak dalam satu dimensi. Setiap kernel akan bergerak satu per satu kebawah daftar *input embedding*, kemudian *word embedding* selanjutnya, dan seterusnya. *Output* yang dihasilkan adalah vektor fitur yang memiliki nilai sebanyak *input embedding*.



Gambar 3. 15 Fitur Vektor Disusutkan menjadi Nilai Maksimum melalui *Max Pooling*

Sumber (Camacho, 2019)

Untuk menunjukkan keberadaan fitur-fitur tingkat tinggi, diperlukan metode identifikasi yang tidak bergantung pada lokasi spesifiknya dalam urutan *input* yang lebih besar. Salah satu pendekatannya dengan menghilangkan informasi yang kurang relevan. Operasi yang umum digunakan yaitu dengan *max-pooling*, yang mempertahankan nilai maksimum dalam vektor fitur, yang diperkirakan mencerminkan fitur-fitur lokal yang paling signifikan.

Nilai-nilai maksimum yang dihasilkan dari pemrosesan setiap vektor fitur *convolutional* akan digabungkan dan diteruskan ke *fully connected layer* yang

dapat menghasilkan nilai kelas sebanyak yang dibutuhkan pada tugas klasifikasi teks.

3.7 Evaluasi

Setelah data di latih dengan model BERT dan CNN, selanjutnya akan dilakukan evaluasi untuk mengetahui kemampuan dalam menganalisis emosi pada teks bahasa Indonesia. Evaluasi yang digunakan yaitu dengan *confusion matrix*. *Confusion matrix* digunakan untuk mengevaluasi kinerja model klasifikasi dengan membandingkan nilai prediksi yang benar dan salah untuk setiap kelas. Dengan begitu, dapat dilihat seberapa baik model dalam mengklasifikasikan data ke dalam kelas yang benar dan mengidentifikasi di mana model cenderung melakukan kesalahan.

3.8 Menyimpan Model

Proses yang dilakukan pada tahap ini dilakukan setelah model telah dilatih dan dievaluasi. Penyimpanan model dilakukan dengan menyimpan parameter dan bobot dari model yang telah dilatih ke dalam *file*, dan kemudian memuatnya kembali tanpa perlu melatih ulang model tersebut.

3.9 Perancangan Sistem

Sistem yang dirancang membutuhkan beberapa diagram dengan menggambarkan spesifikasi detail tentang sistem yang direncanakan menggunakan diagram. Selain itu, dipertimbangkan alur proses dari perancangan program untuk memastikan keselarasan dan konsistensi.

3.3.1 Diagram Umum Sistem

Diagram sistem secara umum adalah gambaran visual dari suatu sistem yang menunjukkan keterkaitan dan interaksi antara elemen-elemen utamanya. Berikut adalah langkah-langkah yang terjadi dalam diagram sistem umum.

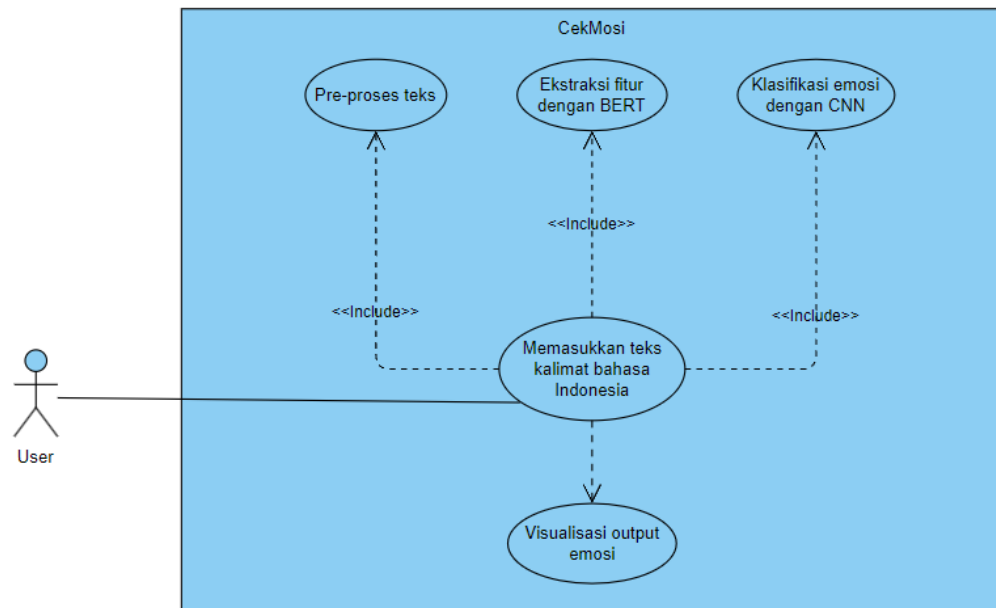


Gambar 3. 16 Diagram Umum Sistem

Berdasarkan gambar 3.16 di atas proses yang dilakukan dimulai dari *user* menginputkan kalimat teks bahasa Indonesia dan *website* akan menerima inputan berupa teks dan mengirimkan teks kalimat tersebut ke model yang sudah dimuat dalam *website*. Kemudian, model akan melakukan proses dan mengklasifikasi emosi yang terkandung pada kalimat tersebut, lalu *output* berupa kategori emosi (sedih, senang, marah, cinta, takut, dan terkejut) diberikan pada *user*.

3.3.2 Use Case Diagram

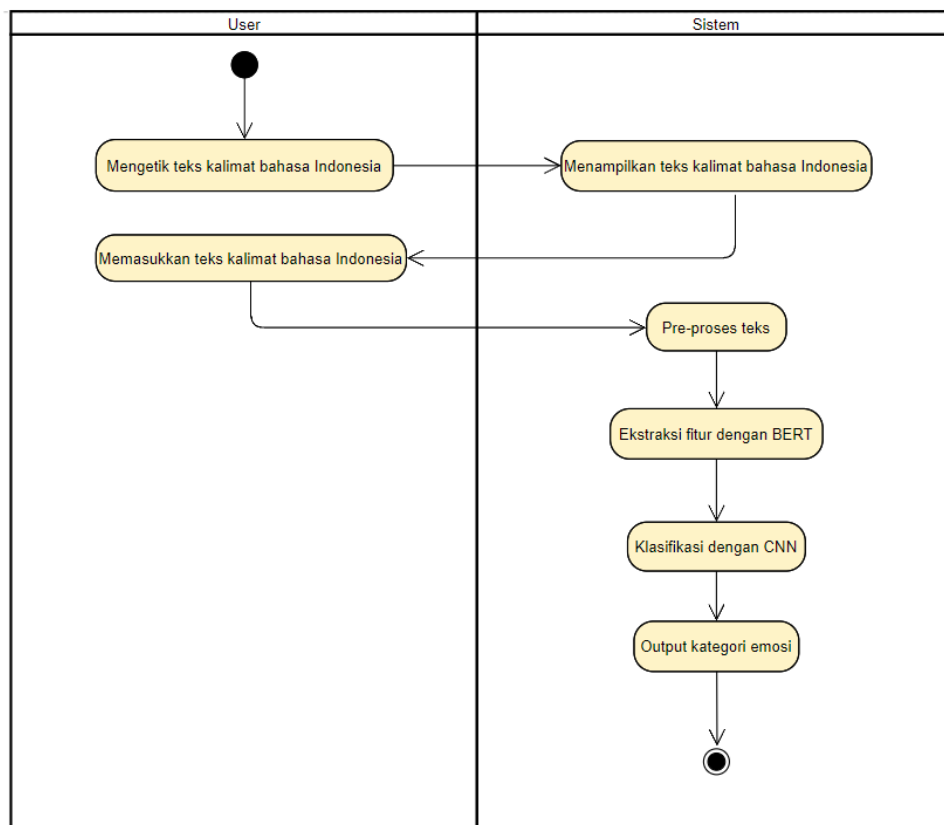
Use case diagram adalah diagram yang menggambarkan hubungan aktor tertentu (seperti *user*) dengan sebuah sistem dalam melakukan tugas tertentu. Interaksi yang dilakukan oleh *user* pada sistem ini dapat dilihat pada gambar 3.17.



Gambar 3. 17 *Use Case Diagram*

3.3.3 Activity Diagram

Activity diagram adalah diagram alur kerja yang digunakan untuk menggambarkan proses atau alur aktivitas dalam suatu sistem. Diagram ini menunjukkan urutan langkah-langkah yang diperlukan untuk menyelesaikan suatu tugas.

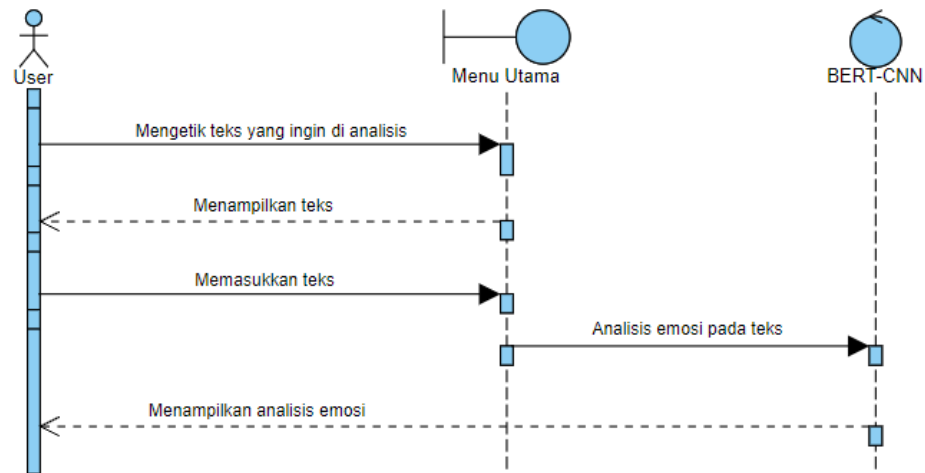


Gambar 3. 18 *Activity Diagram*

Berdasarkan gambar di atas, *activity diagram* dimulai dari user mengetik teks kalimat bahasa Indonesia yang ingin di analisis emosinya, lalu sistem akan menampilkan teks tersebut. Setelah itu, memasukkan teks tersebut dan dibersihkan dari noise. Kemudian ekstraksi fitur atau representasi vektor dari teks yang telah diproses menggunakan BERT. Lalu sistem mengklasifikasikan emosi teks tersebut menggunakan model CNN, dan sistem akan menampilkan kategori emosi teks yang telah diketik sebelumnya.

3.3.4 *Sequence Diagram*

Sequence diagram adalah alat yang digunakan untuk menggambarkan interaksi antar objek dalam sebuah sistem. Diagram ini dapat membantu memahami cara kerja sistem dan mengidentifikasi potensi masalah.

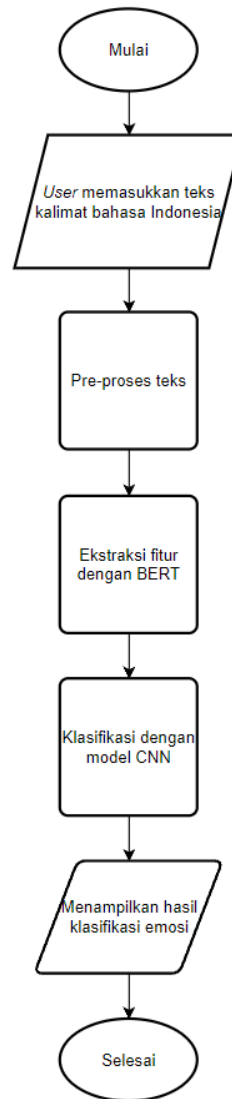


Gambar 3. 19 *Sequence Diagram*

Gambar 3.19 menjelaskan proses dari *sequence diagram* yang dimulai dari *user* mengetik teks kalimat bahasa Indonesia yang ingin di analisis, kemudian teks akan ditampilkan dan *user* memasukkan teks yang ingin di analisis. Setelah itu proses analisis dimulai dan hasilnya akan ditampilkan langsung pada menu utama.

3.3.5 *Flowchart Sistem*

Flowchart merupakan diagram yang menjelaskan alur proses dari suatu program. Tujuan utamanya untuk memvisualisasikan berbagai proses agar mudah dipahami.



Gambar 3. 20 *Flowchart Sistem*

Pada gambar di atas menjelaskan *flowchart* dari sistem yang diawali dari *user* memasukkan teks kalimat bahasa Indonesia, kemudian dilakukannya pre-proses teks untuk menghilangkan tanda baca, huruf kapital, dan spasi yang berlebihan. Lalu fitur dari kalimat bahasa Indonesia diekstraksi menggunakan BERT, setelah itu fitur yang diekstraksi diklasifikasikan menjadi salah satu dari enam kategori emosi menggunakan CNN. Terakhir *website* akan menampilkan hasil dari klasifikasi emosi seperti sedih, senang, marah, cinta, takut, dan terkejut.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Setelah tahap analisis dan perancangan, tahap selanjutnya adalah implementasi dan pengujian, spesifikasi perangkat keras dan lunak yang digunakan pada penelitian sebagai berikut:

1. *Processor* 11th Gen Intel(R) Core(TM) i5-11400H @ 2,70GHz.
2. RAM 16GB.
3. Sistem operasi *Windows 11 Home Single Language* 64-bit.
4. *Browser Google Chrome*.
5. *Cursor*.
6. Python 3.7.
7. *Library* numpy, pandas, matplotlib, nltk, sastrawi, sklearn, torch, transformers, dan streamlit.

4.1.1 Halaman Utama Website

Gambar di bawah merupakan hasil tampilan halaman utama sistem.



Gambar 4. 1 Tampilan Halaman Utama

4.1.2 Hasil Klasifikasi Sistem

Setelah user mengetik kalimat yang ingin di analisis dan menginputkan kalimat tersebut, lalu proses klasifikasi dilakukan oleh sistem. Hasilnya memperlihatkan probabilitas antar kelas emosi untuk menunjukkan bahwa sebuah kalimat tidak

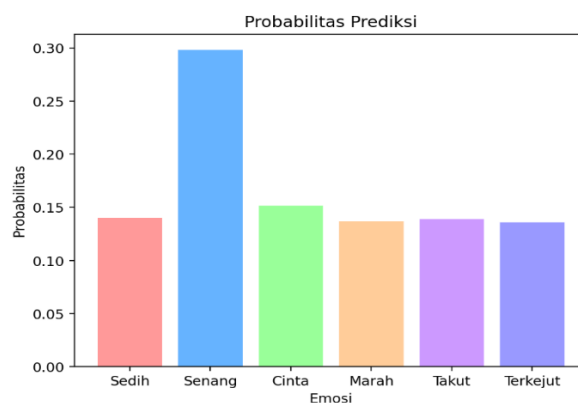
hanya terdapat satu emosi saja melainkan memiliki beberapa emosi yang terkandung pada kalimat tersebut.

Analisis Emosi Dalam Teks Bahasa Indonesia

Masukkan Kalimat:

Saya telah mampu menurunkan tiga kilo dan saya merasa lebih energik bahkan setelah saya berumur satu bulan sehingga segala sesuatunya tampak berjalan sesuai rencana

Hasil Prediksi



- Senang (29.81%)
- Cinta (15.12%)
- Sedih (13.99%)
- Takut (13.85%)
- Marah (13.65%)
- Terkejut (13.58%)

Gambar 4. 2 Hasil Klasifikasi Emosi

4.2 Pre-Processing Dataset

Tahap *Pre-Processing* dalam sistem ini meliputi sejumlah proses yaitu *case folding*, *tokenizing*, *stopword removal*, *stemming*, dan *filtering* dataset. Proses *tokenizing* dan *stemming* dilakukan dengan *library* Sastrawi.

```
def preprocess_text(text):
    # Menghapus karakter tab, baris baru, back slice, dan karakter non-ASCII
    text = text.encode('ascii', 'replace').decode('ascii').replace('\t', " ").replace('\n', " ").replace('\u', " ").replace('\\', "")

    # Mengubah teks menjadi huruf kecil
    text = text.lower()

    # Menghapus mention, Link, hashtag, dan URL yang tidak lengkap
    text = ' '.join(re.sub("([@#][A-Za-z0-9]+)|(\w+:\w+\/\w+)|(http\w+)|(https\w+)", "", text).split())

    # Menghapus angka
    text = re.sub(r"\d+", "", text)

    # Menghapus tanda baca
    text = text.translate(str.maketrans("", "", string.punctuation))

    # Menghapus spasi di awal dan akhir teks
    text = text.strip()

    # Menghapus spasi berlebih
    text = re.sub('\s+', ' ', text)

    return text

df_train['teks'] = df_train['teks'].apply(preprocess_text)
df_train.head()
```

Gambar 4. 3 Case Folding dan Filtering

	teks	sedih	senang	cinta	marah	takut	terkejut
0	aku selalu luang waktu baring aku rasa aneh	0	0	0	0	0	1
1	aku rasa sangat kewalahan	0	0	0	0	0	1
2	sangat suka baca alkitab rasa hadir tuhan yesu...	0	0	0	0	0	1
3	rasa sedikit takjub syukur darat tengah kerumun	0	0	0	0	0	1
4	aku duduk sana rasa asa aneh beberapa kemudian...	0	0	0	0	0	1

Gambar 4. 4 Hasil Case Folding dan Filtering

```
def tokenize(text):
    tokens = nltk.tokenize.word_tokenize(text)
    return tokens

df_train['teks'] = df_train['teks'].apply(tokenize)
df_train.head()
```

Gambar 4. 5 Tokenizing

	teks	sedih	senang	cinta	marah	takut	terkejut
0	[aku, selalu, meluangkan, waktu, untuk, berbar...	0	0	0	0	0	1
1	[aku, merasa, sangat, kewalahan]	0	0	0	0	0	1
2	[saya, sangat, suka, membaca, alkitab, karena,...	0	0	0	0	0	1
3	[saya, merasa, sedikit, takjub, dan, bersyukur...	0	0	0	0	0	1
4	[aku, hanya, duduk, disana, merasakan, perasaa...	0	0	0	0	0	1

Gambar 4. 6 Hasil Tokenizing

```

factory = StopWordRemoverFactory()
stopwords = factory.get_stop_words()
stopwords.append('aku') # Menambahkan kata 'aku' ke dalam daftar stopwords

def remove_stopwords(tokens):
    cleaned_text = []
    for text in tokens:
        if text not in stopwords:
            cleaned_text.append(text)
    return cleaned_text

df_train['teks'] = df_train['teks'].apply(remove_stopwords)
df_train.head()

```

Gambar 4. 7 Stopword Removal

	teks	sedih	senang	cinta	marah	takut	terkejut
0	[selalu, meluangkan, waktu, berbaring, merasa,...	0	0	0	0	0	1
1	[merasa, sangat, kewalahan]	0	0	0	0	0	1
2	[sangat, suka, membaca, alkitab, merasakan, ke...	0	0	0	0	0	1
3	[merasa, sedikit, takjub, bersyukur, mendarat,...	0	0	0	0	0	1
4	[duduk, disana, merasakan, perasaan, aneh, beb...	0	0	0	0	0	1

Gambar 4. 8 Hasil Stopword Removal

```

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

def stemming(tokens):
    result = [stemmer.stem(text) for text in tokens]
    return result

df_train['teks'] = df_train['teks'].apply(stemming)
df_train.head()

```

Gambar 4. 9 Stemming

	teks	sedih	senang	cinta	marah	takut	terkejut
0	[selalu, luang, waktu, baring, rasa, aneh]	0	0	0	0	0	1
1	[rasa, sangat, kewalahan]	0	0	0	0	0	1
2	[sangat, suka, baca, alkitab, rasa, hadir, tuh...	0	0	0	0	0	1
3	[rasa, sedikit, takjub, syukur, darat, tengah,...	0	0	0	0	0	1
4	[duduk, sana, rasa, asa, aneh, beberapa, kemud...	0	0	0	0	0	1

Gambar 4. 10 Hasil Stemming



Gambar 4. 11 Wordcloud Setiap Kelas Emosi

Frekuensi kata yang sering muncul pada teks dataset yang telah ditranslasi sebelumnya memiliki hasil cukup unik dikarenakan sistem translasi yang kurang *advance* dalam mengubah kata-kata dari bahasa Inggris ke bahasa Indonesia. Hasilnya dapat dilihat pada gambar dibawah.

```

Frekuensi Kata untuk Emosi Sedih:
[('rasa', 5028), ('sangat', 1152), ('asa', 732), ('orang', 558), ('jadi', 435), ('diri', 423), ('tahu', 416), ('laku', 414), ('buat', 398), ('lebih', 358)]
Frekuensi Kata untuk Emosi Senang:
[('rasa', 4737), ('sangat', 1086), ('lebih', 714), ('orang', 620), ('asa', 619), ('jadi', 539), ('buat', 440), ('laku', 384), ('senang', 370), ('diri', 369)]
Frekuensi Kata untuk Emosi Cinta:
[('rasa', 4650), ('sangat', 1008), ('orang', 792), ('asa', 774), ('suka', 640), ('cinta', 639), ('lebih', 514), ('jadi', 509), ('buat', 417), ('lambat', 405)]
Frekuensi Kata untuk Emosi Marah:
[('rasa', 4889), ('sangat', 1044), ('orang', 762), ('kesal', 646), ('marah', 565), ('asa', 561), ('jadi', 546), ('laku', 479), ('tahu', 415), ('diri', 401)]
Frekuensi Kata untuk Emosi Takut:
[('rasa', 5012), ('sangat', 1005), ('takut', 888), ('asa', 605), ('orang', 588), ('jadi', 531), ('sedikit', 524), ('tahu', 449), ('laku', 441), ('diri', 420)]
Frekuensi Kata untuk Emosi Terkejut:
[('rasa', 4722), ('kejut', 983), ('aneh', 977), ('sangat', 960), ('asa', 819), ('jadi', 527), ('sedikit', 489), ('biasa', 488), ('orang', 485), ('lihat', 475)]

```

Gambar 4. 12 Frekuensi Kata Pada Semua Kelas Emosi

Frekuensi kata 'rasa' menjadi yang terbanyak pada semua kelas emosi. Hal ini terjadi karena sebelum dataset ditranslasi, kata 'feel' memiliki jumlah yang sangat banyak yang mengakibatkan kata 'rasa' yang sering muncul pada dataset. Namun,

untuk kelas emosi terkejut, kata kedua yang sering muncul yaitu kata ‘kejut’ atau ‘terkejut’ sebelum proses *stemming* dilakukan yang masih sesuai dengan konteks emosinya yaitu terkejut.

4.3 Implementasi Ekstraksi Fitur Teks Menggunakan BERT

Proses ini melibatkan pembacaan data teks dari *file*, tokenisasi teks menggunakan *tokenizer* BERT yang telah dilatih sebelumnya, penyesuaian panjang token dengan *padding* jika diperlukan dan konversi token menjadi ID. Lalu, ID token, *mask input*, dan ID segmen dikonversi menjadi tensor dan dikumpulkan dalam dataset untuk dilakukan pelatihan model. Kemudian, proses ini dimasukkan ke dalam model BERT yang telah dipindahkan ke perangkat yang sesuai (GPU atau CPU). Seluruh rangkaian proses ini dilakukan pada gambar 4.13 sampai dengan 4.17.

```
class InputExample(object):
    # Inisialisasi kelas InputExample
    def __init__(self, text, labels=None):
        self.text = text # Menyimpan teks input
        self.labels = labels # Menyimpan Label dari teks jika ada

class InputFeatures(object):
    # Inisialisasi kelas InputFeatures
    def __init__(self, input_ids, input_mask, segment_ids, label_ids):
        self.input_ids = input_ids # ID token dari teks input
        self.input_mask = input_mask # Mask untuk menentukan mana token yang harus diperhatikan oleh model
        self.segment_ids = segment_ids # ID segment untuk membedakan kalimat dalam teks input
        self.label_ids = label_ids # ID Label dari teks input
```

Gambar 4. 13 Transformasi Data Input

```
def get_train_examples(train_file):
    # Membaca file data latih
    train_df = pd.read_csv(train_file)

    # Mengambil kolom teks
    text = train_df['teks'].values

    # Mengambil Label dari semua kolom selain kolom teks
    labels = train_df[train_df.columns[1:]].values

    examples = []

    # Iterasi untuk setiap baris data
    for i in range(len(train_df)):
        # Menambahkan contoh data ke dalam list
        examples.append(InputExample(text[i], labels=labels[i]))

    return examples
```

Gambar 4. 14 Mengambil Contoh Data Latih


```

def get_features_from_examples(examples, max_seq_len, tokenizer):
    # Inisialisasi list untuk menyimpan fitur
    features = []

    # Iterasi melalui setiap contoh
    for i, example in enumerate(examples):
        # Tokenisasi teks
        tokens = tokenizer.tokenize(example.text)

        # Memotong token jika melebihi batas maksimal
        if len(tokens) > max_seq_len - 2:
            tokens = tokens[: (max_seq_len - 2)]

        # Menambahkan token khusus di awal dan akhir
        tokens = ["[CLS]"] + tokens + ["[SEP]"]

        # Konversi token ke ID
        input_ids = tokenizer.convert_tokens_to_ids(tokens)

        # Membuat mask input
        input_mask = [1] * len(input_ids)

        # Membuat ID segmen
        segment_ids = [0] * len(tokens)

        # Menambahkan padding
        padding = [0] * (max_seq_len - len(input_ids))
        input_ids += padding
        input_mask += padding
        segment_ids += padding

        # Memastikan panjang sesuai dengan max_seq_len
        assert len(input_ids) == max_seq_len
        assert len(input_mask) == max_seq_len
        assert len(segment_ids) == max_seq_len

        # Konversi Label ke float
        label_ids = [float(label) for label in example.labels]

        # Menambahkan fitur ke dalam list
        features.append(InputFeatures(input_ids=input_ids,
                                     input_mask=input_mask,
                                     segment_ids=segment_ids,
                                     label_ids=label_ids))

    # Mengembalikan list fitur
    return features

```

Gambar 4. 15 Menghasilkan Fitur dari Contoh-Contoh Data

```

def get_dataset_from_features(features):
    # Mengubah input_ids dari fitur menjadi tensor
    input_ids = torch.tensor([f.input_ids for f in features], dtype=torch.long)

    # Mengubah input_mask dari fitur menjadi tensor
    input_mask = torch.tensor([f.input_mask for f in features], dtype=torch.long)

    # Mengubah segment_ids dari fitur menjadi tensor
    segment_ids = torch.tensor([f.segment_ids for f in features], dtype=torch.long)

    # Mengubah label_ids dari fitur menjadi tensor
    label_ids = torch.tensor([f.label_ids for f in features], dtype=torch.float)

    # Membuat dataset dari tensor-tensor tersebut
    dataset = TensorDataset(input_ids, input_mask, segment_ids, label_ids)

    # Mengembalikan dataset
    return dataset

```

Gambar 4. 16 Mengubah Fitur-Fitur Untuk Pelatihan Model


```

# Menentukan device yang akan digunakan, cuda atau cpu
device = torch.device(type='cuda')

# Menentukan pretrained weights yang akan digunakan
pretrained_weights = 'bert-base-uncased'

# Membuat tokenizer dari pretrained weights
tokenizer = BertTokenizer.from_pretrained(pretrained_weights)

# Membuat model dasar dari pretrained weights
basemodel = BertModel.from_pretrained(pretrained_weights)

# Memindahkan model ke device yang telah ditentukan
basemodel.to(device)

```

Gambar 4. 17 Pemanggilan Model BERT dan Menggunakan CUDA

Implementasi representasi fitur teks dengan BERT pada penelitian ini dapat dibuat sebuah yang meliputi beberapa proses dengan contoh kalimat teks “aku merasa sangat aneh”.

1. Pembacaan Data

Teks “aku merasa sangat aneh” dianggap sebagai satu baris data dalam *file* yang dibaca.

2. Tokenisasi

Teks tersebut di tokenisasi menggunakan *tokenizer* BERT yang hasilnya ["aku", "merasa", "sangat", "aneh"].

3. Penambahan Token Khusus

Token khusus [CLS] dan [SEP] ditambahkan di awal dan di akhir *array* token, sehingga menjadi ["[CLS]", "aku", "merasa", "sangat", "aneh", "[SEP]"].

4. Konversi Token ke ID

Setiap token dikonversi menjadi ID berdasarkan kamus token BERT. Contoh ID-nya adalah [101, 123, 456, 789, 321, 102].

5. Pembuatan *Mask Input* dan ID Segmen

Mask input dibuat untuk menandai token yang valid, sehingga *mask input* menjadi [1, 1, 1, 1, 1, 1], sedangkan ID segmen digunakan untuk membedakan kalimat dalam teks. Dalam contoh ini semua diatur 0 karena hanya ada satu segmen [0, 0, 0, 0, 0, 0].

6. *Padding*

Contoh jika panjang maksimal *sequence* adalah 10, maka *padding* [0] ditambahkan ke ID token, *mask input*, dan ID segmen. Sehingga bentuk ketiganya menjadi.

[101, 123, 456, 789, 321, 102, 0, 0, 0, 0]

[1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

7. Konversi ke Tensor

ID token, *mask input*, dan ID segmen dikonversi menjadi tensor dan ditampung dalam sebuah dataset.

Dengan begitu, *output* yang dihasilkan BERT merupakan vektor fitur untuk setiap token yang dijadikan *input* untuk model CNN.

4.4 Implementasi Klasifikasi Emosi Pada Teks Dengan CNN

Proses pertama yang dilakukan adalah menginisialisasi model CNN yang memiliki beberapa parameter seperti jumlah *embedding*, dimensi *embedding*, *dropout*, jumlah kernel, ukuran kernel, dan jumlah label. Selanjutnya, model mengatur parameter-parameter tersebut dan membuat *embedding layer*, *convolutional layers*, *dropout layer*, dan *classifier layer*. Proses ini dapat dilihat pada gambar 4.18.

Penelitian ini dilakukan uji coba sebanyak dua kali pada model untuk memilih nilai *hyperparameter tuning* yang memiliki angka metrik evaluasi yang cukup baik pada saat evaluasi. Pada tabel di bawah dapat dilihat perbedaan nilai dari uji coba pertama dan kedua.

Tabel 4. 1 Tabel Nilai *Hyperparameter Tuning* Uji Coba Pertama dan Kedua

No.	Jumlah Kernel	Nilai <i>Dropout</i>	<i>Learning Rate</i>	Ukuran <i>Batch</i>	<i>Epoch</i>
1	100	0.5	1e-3	12	30
2	100	0.5	4e-5	16	30

Tabel 4. 2 Tabel Hasil Metrik Evaluasi Pada Uji Coba Pertama dan Kedua

No.	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
1	71%	89%	72%	79%
2	85%	92%	86%	88%

Pada tabel 4.2 di atas dapat dilihat perbedaan yang signifikan antara uji coba pertama dan kedua. Nilai metrik evaluasi mengalami peningkatan rata-rata sebesar 13.48%. Perbedaan nilai *hyperparameter* pada kedua uji coba yaitu pada *learning*

rate dan ukuran *batch*. Hal ini cukup penting untuk diperhatikan untuk mencegah terjadinya *overfitting* dan *underfitting* pada model. *Learning rate* mempengaruhi seberapa cepat dan seberapa baik model dapat belajar dari data, lalu ukuran *batch* menentukan jumlah sampel data yang diproses sebelum model memperbarui parameter internalnya. Kedua nilai ini penting, namun perlu diperhatikan juga nilai *hyperparameter tuning* lain untuk mendapatkan performa model yang maksimal.

```
class KimCNN(nn.Module):
    # Inisialisasi model
    def __init__(self, embed_num, embed_dim, dropout=0.5, kernel_num=100, kernel_sizes=[2,3,4], num_labels=6):
        super().__init__()

        self.num_labels = num_labels
        self.embed_num = embed_num
        self.embed_dim = embed_dim
        self.dropout = dropout
        self.kernel_num = kernel_num
        self.kernel_sizes = kernel_sizes

        # Membuat layer embedding
        self.embed = nn.Embedding(self.embed_num, self.embed_dim)

        # Membuat convolutional layers
        self.convs = nn.ModuleList([nn.Conv2d(1, self.kernel_num, (k, self.embed_dim)) for k in self.kernel_sizes])

        # Membuat dropout layer
        self.dropout = nn.Dropout(self.dropout)

        # Membuat classifier layer
        self.classifier = nn.Linear(len(self.kernel_sizes) * self.kernel_num, self.num_labels)
```

Gambar 4. 18 Inialisasi Model CNN

Langkah berikutnya melakukan *feedforward* pada model. Pada proses ini *input* diubah menjadi format yang sesuai untuk dimasukkan ke dalam *convolutional layers*. Lalu melakukan operasi *convolution*, ReLu *activation*, dan *max-pooling* untuk mendapatkan fitur terbaik dari setiap *convolutional layer*. Proses ini dapat dilihat pada gambar 4.19. Dengan begitu model CNN dapat digunakan untuk melakukan klasifikasi teks.

Untuk memberikan contoh perhitungan *softmax*, asumsikan bahwa model BERT menghasilkan vektor logit berikut untuk sebuah teks.

$$z = [0.5, -1.2, 0.3, 2.0, -0.5, 1.5]$$

Vektor ini mewakili logit untuk enam kelas emosi (sedih, senang, cinta, marah, takut, dan terkejut). Langkah pertama yaitu menghitung eksponensial dari setiap logit.

$$e^{z^1} = e^{0.5} \approx 1.65$$

$$e^{z^2} = e^{-1.2} \approx 0.30$$

$$e^{z_3} = e^{0.3} \approx 1.35$$

$$e^{z_4} = e^{2.0} \approx 7.39$$

$$e^{z_5} = e^{-0.5} \approx 0.61$$

$$e^{z_6} = e^{1.5} \approx 4.48$$

Lalu menghitung total dari semua eksponensial dari setiap logit.

$$\text{sum} = 1.65 + 0.30 + 1.35 + 7.39 + 0.61 + 4.48 \approx 15.78$$

Terakhir menghitung probabilitas untuk setiap kelas.

$$\text{Softmax}_{(z_1)} = \frac{1.65}{15.78} \approx 0.10$$

$$\text{Softmax}_{(z_2)} = \frac{0.30}{15.78} \approx 0.02$$

$$\text{Softmax}_{(z_3)} = \frac{1.35}{15.78} \approx 0.09$$

$$\text{Softmax}_{(z_4)} = \frac{7.39}{15.78} \approx 0.47$$

$$\text{Softmax}_{(z_5)} = \frac{0.61}{15.78} \approx 0.04$$

$$\text{Softmax}_{(z_6)} = \frac{4.48}{15.78} \approx 0.28$$

Hasilnya adalah vektor probabilitas [0.10, 0.02, 0.09, 0.47, 0.04, 0.28] yang menunjukkan probabilitas model memprediksi setiap kelas emosi untuk teks yang diberikan. Dalam contoh ini, model mengklasifikasi bahwa teks tersebut mengungkapkan emosi marah dan diikuti dengan emosi terkejut.

```

# Forward pass
def forward(self, inputs, Labels=None):
    output = inputs.unsqueeze(1)

    # Melakukan convolution dan ReLU activation
    output = [nn.functional.relu(conv(output)).squeeze(3) for conv in self.convs]

    # Melakukan max pooling
    output = [nn.functional.max_pool1d(i, i.size(2)).squeeze(2) for i in output]

    # Menggabungkan output dari semua convolutional layers
    output = torch.cat(output, 1)

    # Menerapkan dropout
    output = self.dropout(output)

    # Mendapatkan logits
    logits = self.classifier(output)

    # Menghitung loss jika label disediakan
    if Labels is not None:
        loss_fct = nn.CrossEntropyLoss()
        loss = loss_fct(logits, Labels)
        return loss
    else:
        # Apply softmax to compute probabilities for each class
        probs = torch.softmax(logits, dim=1)
        return probs

```

Gambar 4. 19 Proses *Feedforward* Model CNN

Kemudian, dilakukan proses untuk menyiapkan data latih dan data validasi untuk model. Dataset dibagi menjadi 80% dataset latih dan 20% dataset validasi. Lalu menetapkan panjang *sequence* yang digunakan dalam tokenisasi teks dan menetapkan *batch* yang digunakan dalam proses pelatihan model. Proses ini ada pada gambar 4.20.

```

# Menetapkan panjang sequence
seq_len = 256

# Menetapkan nama file untuk data latih dan validasi
train_file = 'train.csv'
val_file = 'val.csv'

# Mendapatkan contoh data latih
train_examples = get_train_examples(train_file)

# Mendapatkan fitur dari contoh data latih
train_features = get_features_from_examples(train_examples, seq_len, tokenizer)

# Membuat dataset dari fitur data latih
train_dataset = get_dataset_from_features(train_features)

# Mendapatkan contoh data validasi
val_examples = get_train_examples(val_file)

# Mendapatkan fitur dari contoh data validasi
val_features = get_features_from_examples(val_examples, seq_len, tokenizer)

# Membuat dataset dari fitur data validasi
val_dataset = get_dataset_from_features(val_features)

# Menetapkan ukuran batch
batch = 16

# Membuat sampler dan dataloader untuk data latih
train_sampler = RandomSampler(train_dataset)
train_dataloader = DataLoader(train_dataset, sampler=train_sampler, batch_size=batch)

# Membuat sampler dan dataloader untuk data validasi
val_sampler = SequentialSampler(val_dataset)
val_dataloader = DataLoader(val_dataset, sampler=val_sampler, batch_size=batch)

```

Gambar 4. 20 Proses Menyiapkan Data Latih dan Data Validasi Untuk Model

```

# Menetapkan jumlah embedding
embed_num = seq_len

# Menetapkan dimensi embedding
embed_dim = basemodel.config.hidden_size

# Menetapkan nilai dropout
dropout = basemodel.config.hidden_dropout_prob

# Menetapkan jumlah kernel
kernel_num = 100

# Menetapkan ukuran kernel
kernel_sizes = [2, 3, 4]

# Menetapkan jumlah label
num_labels = 6

# Membuat model KimCNN dengan parameter yang telah ditetapkan
model = KimCNN(embed_num, embed_dim, dropout=0.5, kernel_num=kernel_num, kernel_sizes=kernel_sizes, num_labels=num_labels)

# Memindahkan model ke device
model.to(device)

```

Gambar 4. 21 Proses Inisialisasi Parameter-Parameter Model CNN

Pada gambar 4.21 menunjukkan proses inisialisasi model CNN dengan parameter-parameter yang telah ditetapkan sebelumnya, seperti jumlah *embedding*, ukuran kernel, dan jumlah label. Selanjutnya model tersebut dipindahkan ke *device* yang telah ditentukan untuk proses *training* dan evaluasi.

Tahap selanjutnya yaitu dengan membuat fungsi yang digunakan untuk menghitung metrik evaluasi seperti *precision*, *recall*, *f1-score*, dan *accuracy* berdasarkan label sebenarnya. Rumus untuk menghitung metrik evaluasi yaitu.

$$\text{Akurasi} = (TP + TN) / (TP + TN + FP + FN)$$

$$\text{Precision} = TP / (TP + FP)$$

$$\text{Recall} = TP / (TP + FN)$$

$$\text{F1-Score} = 2 * (\text{Presisi} * \text{Recall}) / (\text{Presisi} + \text{Recall})$$

Fungsi ‘*precision_score*’, ‘*recall_score*’, ‘*f1-score*’, dan ‘*accuracy_score*’ dari library ‘*sklearn.metrics*’ secara otomatis menghitung nilai-nilai tersebut berdasarkan label sebenarnya (‘*y_true*’) dan label prediksi (‘*y_pred*’). Proses pada gambar 4.22 di bawah merupakan proses untuk membuat perhitungan metrik evaluasi.

```
def calculate_metrics(y_true, y_pred):
    # Mengubah probabilitas menjadi label kelas dengan threshold 0.5
    y_pred_labels = (y_pred > 0.5).astype(int)

    # Menghitung skor presisi
    precision = precision_score(y_true, y_pred_labels, average='macro', zero_division=0)

    # Menghitung skor recall
    recall = recall_score(y_true, y_pred_labels, average='macro', zero_division=0)

    # Menghitung skor F1
    f1 = f1_score(y_true, y_pred_labels, average='macro', zero_division=0)

    # Menghitung akurasi
    accuracy = accuracy_score(y_true, y_pred_labels)

    # Mengembalikan nilai presisi, recall, F1, dan akurasi
    return precision, recall, f1, accuracy
```

Gambar 4. 22 Proses Menghitung Metrik Evaluasi

4.5 Evaluasi

Setelah dilakukan proses implementasi ekstraksi fitur dengan BERT dan klasifikasi teks menggunakan model CNN, selanjutnya dilakukan proses evaluasi terhadap analisis emosi dalam teks bahasa Indonesia. Proses pelatihan model menggunakan data latih dan data validasi

```
lr = 4e-5
epochs = 30
optimizer = torch.optim.Adam(model.parameters(), lr=lr)
labels = ['sedih', 'senang', 'cinta', 'marah', 'takut', 'terkejut']

accuracy_values = []
precision_values = []
recall_values = []
f1_values = []
total_losses = []

for i in range(epochs):
    # Menampilkan epoch saat ini
    print('-----EPOCH #{}-----'.format(i+1))
    print('training...')

    # Mengatur model ke mode training
    model.train()

    # Inisialisasi total loss untuk epoch ini
    total_loss = 0

    # Melakukan iterasi pada setiap batch dalam train_dataloader
    for step, batch in enumerate(train_dataloader):
        # Memindahkan batch ke device yang sesuai
        batch = tuple(t.to(device) for t in batch)

        # Mengekstrak komponen dari batch
        input_ids, input_mask, segment_ids, label_ids = batch

        # Menghitung output model tanpa melakukan perhitungan gradient
        with torch.no_grad():
            inputs_ = basemodel(input_ids, segment_ids, input_mask)

        # Menghitung loss
        loss = model(inputs_, label_ids)
        loss = loss.mean()

        # Melakukan backpropagation dan update parameter
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        # Menambahkan loss ini ke total loss
        total_loss += loss.item()
```

Gambar 4. 23 Proses Pelatihan Model

Pada gambar 4.23 di atas menunjukkan pelatihan model menggunakan optimasi “Adam” dengan *learning rate* sebesar $4e-5$ selama 30 *epoch*. Proses pelatihan dilakukan dengan mengatur model ke mode *training*, menghitung *output* model tanpa melakukan perhitungan *gradient*, menghitung *loss*, melakukan *backpropagation*, dan memperbarui parameter.

```
# Mengatur model ke mode evaluasi
model.eval()
print('evaluating...')

# Melakukan iterasi pada setiap batch dalam val_dataloader
for step, batch in enumerate(val_dataloader):
    # Memindahkan batch ke device yang sesuai
    batch = tuple(t.to(device) for t in batch)

    # Mengekstrak komponen dari batch
    val_input_ids, val_input_mask, val_segment_ids, val_label_ids = batch

    # Menghitung output model tanpa melakukan perhitungan gradient
    with torch.no_grad():
        val_inputs, _ = basemodel(val_input_ids, val_segment_ids, val_input_mask)
        probs = model(val_inputs)

    # Menyimpan label sebenarnya dan prediksi
    y_true.append(val_label_ids)
    y_pred.append(probs)

# Menggabungkan semua label sebenarnya dan prediksi
y_true = torch.cat(y_true, dim=0).float().cpu().detach().numpy()
y_pred = torch.cat(y_pred, dim=0).float().cpu().detach().numpy()

precision, recall, f1, accuracy = calculate_metrics(y_true, y_pred)
print('Precision:', precision)
print('Recall:', recall)
print('F1-score:', f1)
print('Accuracy:', accuracy)

accuracy_values.append(accuracy)
precision_values.append(precision)
recall_values.append(recall)
f1_values.append(f1)

# Inisialisasi dictionary untuk FPR, TPR, dan ROC AUC
fpr = dict()
tpr = dict()
roc_auc = dict()

# Menghitung FPR, TPR, dan ROC AUC untuk setiap label
for i, label in enumerate(labels):
    fpr[label], tpr[label], _ = roc_curve(y_true[:, i], y_pred[:, i])
    roc_auc[label] = auc(fpr[label], tpr[label])
```

Gambar 4. 24 Proses Evaluasi Model

Pada gambar 4.24 di atas, proses evaluasi model dilakukan terhadap data validasi. Pertama, model diatur ke model evaluasi dengan ‘*model.eval()*’, kemudian dilakukan iterasi pada setiap *batch* dalam ‘*val_dataloader*’. Selanjutnya, dilakukan perhitungan *output* model tanpa melakukan perhitungan *gradient*. Hasil prediksi dari model disimpan dalam ‘*y_pred*’, sedangkan label sebenarnya disimpan dalam ‘*y_true*’. Setelah itu, dilakukan perhitungan metrik evaluasi seperti *precision*, *recall*, *F1-score*, dan akurasi. Selain itu, juga dihitung *False Positive Rate* (FPR), *True Positive Rate* (TPR), dan *Area Under Curve* (AUC) untuk setiap label menggunakan *ROC curve*.

Proses selanjutnya adalah membuat grafik metrik evaluasi seperti *precision*, *recall*, *F1-score*, dan akurasi. Proses ini ada pada gambar 4.25.

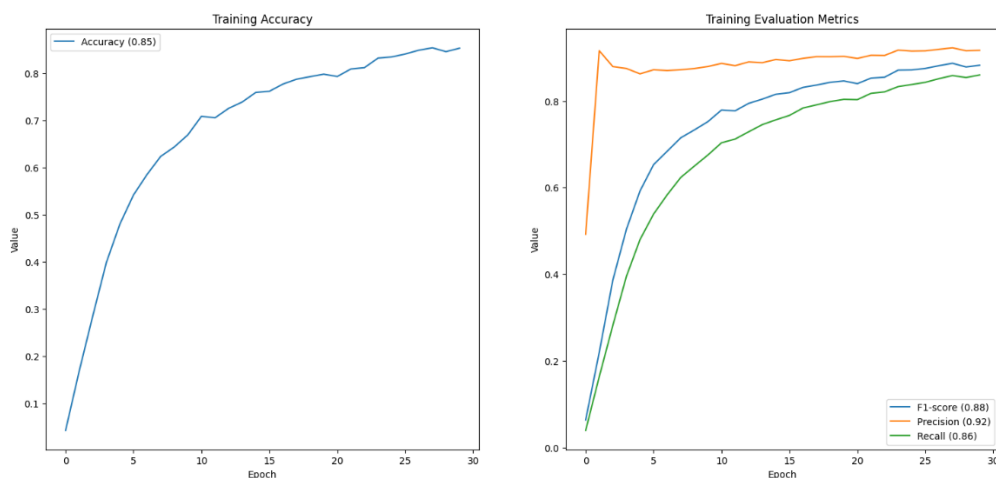
```
# Plot training loss
plt.figure(figsize=(10, 5))
plt.plot(range(1, epochs+1), total_losses[:epochs], Label='Total Training Loss')
plt.xlabel('Epoch')
plt.ylabel('Total Loss')
plt.title('Total Training Loss per Epoch')
plt.legend()

# Plot training accuracy
plt.figure(figsize=(18, 8))
plt.subplot(1, 2, 1)
plt.plot(range(epochs), accuracy_values, Label='Accuracy {:.2f}'.format(accuracy_values[-1]))
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.title('Training Accuracy')
plt.legend()

# Plot training F1-score, precision, and recall
plt.subplot(1, 2, 2)
plt.plot(range(epochs), f1_values, Label='F1-score {:.2f}'.format(f1_values[-1]))
plt.plot(range(epochs), precision_values, Label='Precision {:.2f}'.format(precision_values[-1]))
plt.plot(range(epochs), recall_values, Label='Recall {:.2f}'.format(recall_values[-1]))
plt.xlabel('Epoch')
plt.ylabel('Value')
plt.title('Training Evaluation Metrics')
plt.legend()

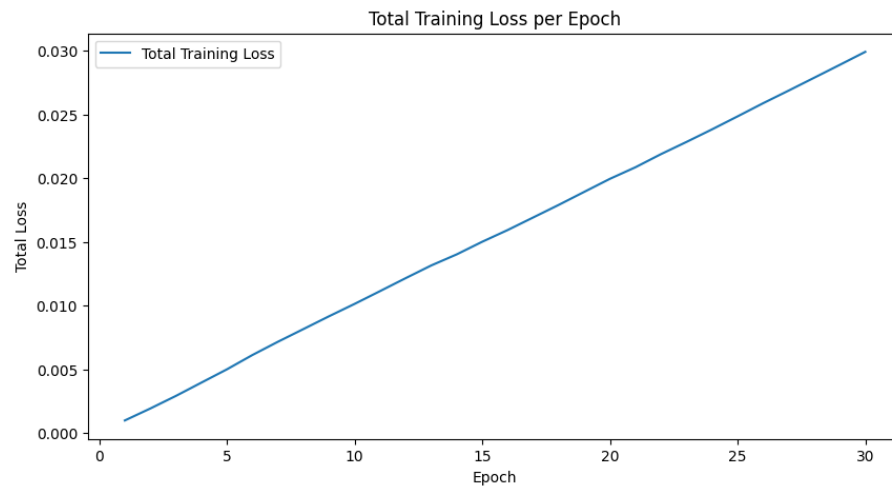
plt.show()
```

Gambar 4. 25 Proses Membuat Grafik Metrik Evaluasi



Gambar 4. 26 Hasil Grafik Akurasi, F1-Score, *Precision*, dan *Recall*

Pada gambar 4.26, akurasi pelatihan rata-rata mencapai angka 85% pada *epoch* 30. Hal ini menunjukkan bahwa model belajar untuk mengklasifikasikan data dengan cukup baik. Untuk *F1-score* sebesar 88%, *precision* 92%, dan *recall* sebesar 86%. Selanjutnya grafik yang menunjukkan total *training loss* dari setiap *epoch*. Total *training loss* pada awalnya tinggi, kemudian menurun secara bertahap. Hal ini menunjukkan bahwa, model kesulitan untuk mempelajari pola dalam data latih, tetapi kemudian lebih baik seiring dengan bertambahnya waktu pelatihan. Hal ini ada pada gambar 4.27.



Gambar 4. 27 Hasil Grafik Total *Training Loss*

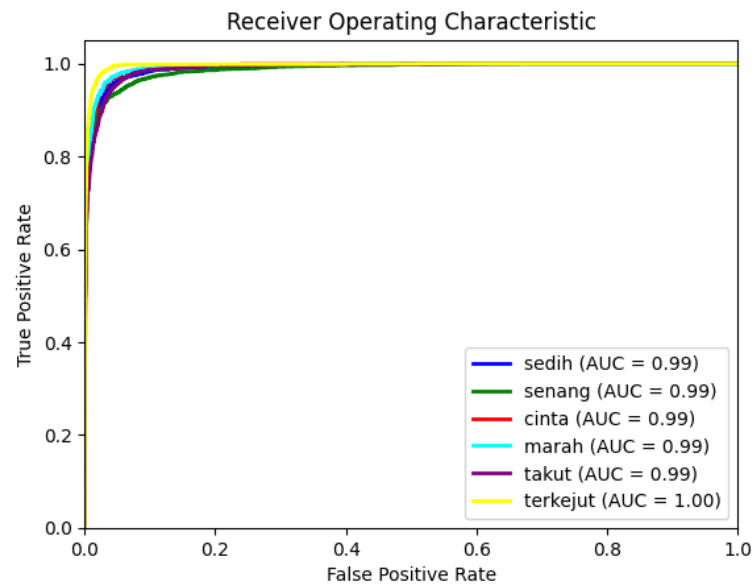
Receiver Operating Characteristic (ROC) *curve* merupakan visualisasi yang sering digunakan untuk menilai efektivitas kinerja model klasifikasi. Grafik ini menggambarkan perbandingan antara TPR dan FPR. Semakin mendekati sudut kiri atas, semakin baik kinerja model yang dievaluasi dengan kurva ROC. Gambar di bawah menunjukkan pembuatan grafik ROC *curve* dan hasil grafiknya.

```
print('ROC AUC per label:')
for label in labels:
    print(label, ': ', roc_auc[label])

# Plot the ROC AUC scores
fig, ax = plt.subplots()
colors = ['blue', 'green', 'red', 'cyan', 'purple', 'yellow']
for i, label in enumerate(labels):
    ax.plot(fpr[label], tpr[label], color=colors[i],
            lw=2, label='%s (AUC = %0.2f)' % (label, roc_auc[label]))

ax.set_xlim([0, 1])
ax.set_ylim([0, 1.05])
ax.set_xlabel('False Positive Rate')
ax.set_ylabel('True Positive Rate')
ax.set_title('Receiver Operating Characteristic')
ax.legend(loc="lower right")
plt.show()
```

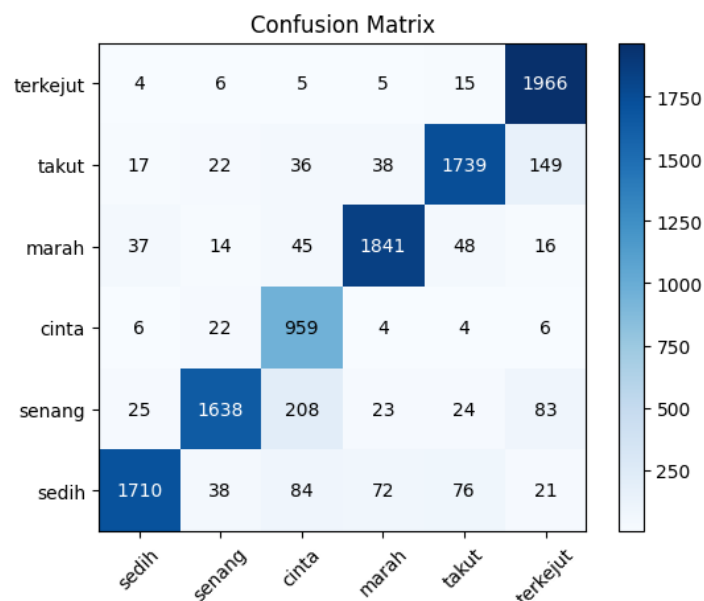
Gambar 4. 28 Proses Membuat Grafik ROC *Curve*



Gambar 4. 29 Hasil Grafik ROC Curve

Area Under Curve (AUC) merupakan metrik yang digunakan untuk mengukur kinerja klasifikasi. Semakin tinggi AUC, maka semakin baik kinerja klasifikasi. Pada gambar 4.29, semua ROC curve memiliki AUC yang tinggi, hal ini menunjukkan bahwa semua klasifikasi tersebut berkinerja baik.

Confusion Matrix merupakan tabel yang memvisualisasikan kinerja suatu model klasifikasi. Dengan begitu, dapat dilihat sejauh mana model mampu membedakan antara hasil yang benar dan salah dalam klasifikasi.



Gambar 4. 30 Hasil Confusion Matrix

Berdasarkan gambar 4.30, dapat dibuat tabel untuk melihat *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Berikut tabelnya untuk semua kelas emosi berdasarkan *confusion matrix* di atas.

Tabel 4. 3 TP, TN, FP, dan FN Semua Kelas Emosi

Kelas	TP	TN	FP	FN
Sedih	1710	3044	121	126
Senang	1638	2954	171	238
Cinta	959	4685	17	340
Marah	1841	3461	241	458
Takut	1739	3538	412	312
Terkejut	1966	2730	784	521

Perhitungan akurasi, *precision*, *recall*, dan *F1-score* untuk semua kelas emosi dapat dihitung berdasarkan informasi yang diberikan pada tabel 4.3. Hasil perhitungannya dapat dilihat pada tabel 4.4.

Tabel 4. 4 Hasil Akurasi, *Precision*, *Recall*, dan *F1-Score* Semua Kelas Emosi

Kelas	Akurasi (TP + TN) / (TP + TN + FP + FN)	<i>Precision</i> TP / (TP + FP)	<i>Recall</i> TP / (TP + FN)	<i>F1-Score</i> $2 * (\text{Presisi} * \text{Recall}) / (\text{Presisi} + \text{Recall})$
Sedih	93.41%	93.39%	93.15%	93.27%
Senang	90.98%	90.54%	87.31%	88.89%
Cinta	94.05%	97.95%	73.82%	84.19%
Marah	88.69%	88.42%	80.07%	84.03%
Takut	87.93%	80.84%	84.78%	82.70%
Terkejut	78.25%	71.49%	79.05%	75.08%

4.6 Menyimpan dan Memanggil Model

Setelah tahap evaluasi, selanjutnya model disimpan dengan nama 'model_state_dict.pt' memakai fungsi *torch.save()*. Lalu, proses pengambilan *state dictionary* yang telah disimpan sebelumnya, dan menginisialisasi model dengan parameter yang sesuai. Proses ini ada di gambar 4.31.

```

# Save the model's state dictionary to a file
torch.save(model.state_dict(), 'model_state_dict.pt')

# Load the saved state dictionary
state_dict = torch.load('model_state_dict.pt')

# Initialize a new instance of the KimCNN model
model = KimCNN(embed_num, embed_dim, dropout=0.5, kernel_num=kernel_num, kernel_sizes=kernel_sizes, num_labels=num_labels)

# Load the saved state dictionary into the model
model.load_state_dict(state_dict)

# Move the model to the device
model.to(device)

```

Gambar 4. 31 Proses Menyimpan dan Memanggil Model

4.7 Hasil Pengujian Sistem

Pengujian sistem dilakukan dengan menguji beberapa kalimat dengan menunjukkan persentase dari prediksi untuk setiap kelas emosi yang ditentukan.

Tabel 4. 5 Tabel Pengujian Emosi pada Teks Bahasa Indonesia

No	Kalimat	Emosi					
		Sedih	Senang	Cinta	Marah	Takut	Terkejut
1.	aku merasa hal itu benarbenar mengejutkannya	13%	13%	13%	13%	13%	35%
2.	aku memang merasa aneh kenapa jarang ada orang yang makan di sana	13%	13%	13%	13%	15%	33%
3.	aku merasa kewalahan tentang beberapa hal seperti membersihkan kamarku dan membereskan semuanya sebelum aku pulang lagi	13%	13%	13%	13%	14%	34%

	pada akhir pekan mendatang						
4.	saya merasa sangat terhormat ketika saya beruntung menemukan edisi yang bagus	13%	35%	13%	13%	13%	13%
4.	aku bisa bilang pada diriku sendiri aku benar - benar marah sekarang, kemarahanku secara keseluruhan sudah berkurang	13%	13%	13%	35%	13%	13%
5.	aku tidak lagi merasa seperti ibu rumah tangga yang sudah lama menderita dan tinggal di rumah sebagai ibu karena aku tidak	34%	13%	13%	13%	13%	13%
6.	saya pikir orang - orang perlu mempunyai perasaan bahwa mereka sukses dari penampilan mereka dalam arti berapa banyak perangkat elektronik yang mereka miliki di	13%	32%	14%	13%	14%	14%

	depan mereka karena ya, saya mengerti bahwa dia adalah seorang pebisnis tetapi sungguh						
7.	saya masih merasa kagum dengan perbedaan gender	13%	13%	13%	13%	13%	35%
8.	aku merasa sedikit sedih jadi kupikir baiklah kamu tidak akan menang, tapi itu mungkin menyenangkan dan eh	26%	15%	16%	14%	14%	14%
9.	aku sedang mencari jumpropeku beberapa saat yang lalu karena aku merasa bertekad untuk membuat pantatku bergerak sedikit lagi tapi sayangnya tidak bisa ditemukan, jadi di sini aku menulis blog tentang sesuatu yang sudah lama ingin kulakukan selama berminggu -	14%	30%	14%	14%	14%	14%

	minggu tapi tidak punya ruang						
10.	saya melihat grafikgrafik ini satusatunya kejutan yang saya rasakan adalah orangorang lain terkejut dengan perekonomian as yang terus terpuruk setiap kali salah satu dari tiga resep ekonom yang sudah mati itu dibalik	13%	13%	13%	13%	13%	35%

Pada tabel di atas dapat dilihat pengujian dengan kalimat teks bahasa Indonesia. Probabilitas antar kelas emosi yang paling besar ditandai dengan persentase yang diberi tanda *bold*. Dengan begitu, untuk mengetahui emosi apa yang terkandung dalam teks bahasa Indonesia dapat dilihat dengan nilai persentase paling besar diantara kelas emosi lainnya.

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan tahap implementasi yang telah dilakukan pada penelitian analisis emosi dalam teks bahasa Indonesia dengan pendekatan BERT dan CNN yang diimplementasikan berbasis *website*, dapat disimpulkan bahwa:

1. Penelitian analisis emosi dalam teks bahasa Indonesia dengan pendekatan BERT dan CNN dapat digunakan untuk mengklasifikasi emosi pada teks bahasa Indonesia.
2. Model CNN dapat digunakan dalam proses klasifikasi emosi dalam beberapa kelas yaitu sedih, senang, cinta, marah, takut, dan terkejut.
3. Translasi yang digunakan masih terlalu sederhana, dampaknya adalah hilangnya nuansa dan makna dari teks dan kesalahan dan ketidakakuratan translasi pada teks bahasa Indonesia.
4. Hasil pengujian model BERT dan CNN mendapatkan akurasi sebesar 85%, *precision* 92%, *recall* 86%, dan *F1-score* 88%.
5. Nilai akurasi tertinggi yaitu pada kelas emosi cinta sebesar 94,05%, dan nilai *F1-score* tertinggi yaitu pada kelas emosi sedih di angka 93.27%.
6. Pemilihan nilai *hyperparameter* sangat berpengaruh terhadap nilai metrik evaluasi pada model. Terbukti dengan peningkatan rata-rata sebesar 13.48%.

5.2 Saran

Saran yang dapat diberikan untuk studi lebih lanjut adalah sebagai berikut:

1. Disarankan untuk melakukan penelitian lebih lanjut untuk mendapatkan dataset yang lebih baik dan bervariasi agar model mampu mengklasifikasi emosi dengan lebih baik.
2. Disarankan alat translasi yang digunakan lebih *advance* dalam mengenali kata serta nuansa untuk mendapatkan variasi kata yang lebih banyak.
3. Disarankan untuk mencoba menambah kelas emosi agar lebih memahami emosi apa yang terkandung dalam teks.
4. Disarankan untuk mencoba algoritma lain seperti *Recurrent Neural Network*, *Long Short-Term Memory*, dan Bi-LSTM sebagai perbandingan akurasi.

DAFTAR PUSTAKA

- Abdullah, S.M., Ameen, S.Y., Sadeeq, M.A., & Zeebaree, S.R. (2021). Multimodal Emotion Recognition using Deep Learning.
- Agastya dan Haryanto, W. H. (2020). Prosiding SINTA 3 (2020) 098 Seminar Nasional Ilmu Teknik dan Aplikasi Industri (SINTA) Identifikasi Kelas Emosi Majemuk pada Kalimat Majemuk Bahasa Indonesia Menggunakan Model *Multinomial Naïve Bayes*.
- Ahmad, S., Asghar, M.Z., Alotaibi, F.M., & Khan, S. (2020). Classification of Poetry Text Into the Emotional States Using Deep Learning Technique. *IEEE Access*, 8, 73865-73878.
- Amien, M. (2023). Sejarah dan Perkembangan Teknik Natural Language Processing (NLP) Bahasa Indonesia: Tinjauan tentang sejarah, perkembangan teknologi, dan aplikasi NLP dalam bahasa Indonesia. *ArXiv*, *abs/2304.02746*.
- Annur, C.H., (2023). 10 Negara dengan Jumlah Penduduk Terbanyak di Dunia Pertengahan 2023. <https://databoks.katadata.co.id/datapublish/2023/07/28/10-negara-dengan-jumlah-penduduk-terbanyak-di-dunia-pertengahan-2023>.
- Asghar, M.Z., Lajis, A., Alam, M.M., Rahmat, M.K., Nasir, H.M., Ahmad, H., Al-Rakhami, M.S., Al-Amri, A.M., & Albogamy, F.R. (2022). A Deep Neural Network Model for the Detection and Classification of Emotions from Textual Content. *Complex.*, 2022, 8221121:1-8221121:12.
- Batbaatar, E., Li, M., & Ryu, K.H. (2019). Semantic-Emotion Neural Network for Emotion Recognition From Text. *IEEE Access*, 7, 111866- 111878.
- Bharti, S.K., Varadhaganapathy, S., Gupta, R.K., Shukla, P.K., Bouye, M., Hingaa, S.K., & Mahmoud, A. (2022). Text-Based Emotion Recognition Using Deep Learning Approach. *Computational Intelligence and Neuroscience*, 2022.
- Brownlee Disclaimer, J. (2017). Deep Learning for Natural Language Processing Develop Deep Learning Models for Natural Language in Python Acknowledgements Copyright Deep Learning for Natural Language Processing.
- Camacho, C. (2019). *CNNs for Text Classification*. https://cezannec.github.io/CNN_Text_Classification.
- Chatterjee, A., Narahari, K.N., Joshi, M., & Agrawal, P. (2019). SemEval-2019 Task 3: EmoContext Contextual Emotion Detection in Text. *International Workshop on Semantic Evaluation*.
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P.P. (2011). Natural Language Processing (Almost) from Scratch. *ArXiv*, *abs/1103.0398*.

- Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv, abs/1810.04805*.
- Elgiriye withana, N. (2024). *Emotions*.
<https://www.kaggle.com/datasets/nelgiriye withana/emotions>.
- Fudholi, D. H. (2021). Klasifikasi Emosi pada Teks dengan Menggunakan Metode Deep Learning. *Syntax Literate; Jurnal Ilmiah Indonesia*, 6(1), 546-553.
- Habib, M.A., Akhand, M.A., & Kamal, M.A. (2022). Emotion Recognition from Microblog Managing Emoticon with Text and Classifying using 1D CNN. *ArXiv, abs/2301.02971*.
- Han, T., Zhang, Z., Ren, M., Dong, C., Jiang, X., & Zhuang, Q. (2023). Text Emotion Recognition Based on XLNet-BiGRU-Att. *Electronics*.
- Huang, X., & Lajoie, S.P. (2023). Social emotional interaction in collaborative learning: Why it matters and how can we measure it? *Social Sciences & Humanities Open*.
- Islam, J., Akhand, M.A., Habib, M.A., Kamal, M.A., & Siddique, N.H. (2021). Recognition of Emotion from Emoticon with Text in Microblog Using LSTM.
- Jacovi, A., Shalom, O.S., & Goldberg, Y. (2018). Understanding Convolutional Neural Networks for Text Classification. *BlackboxNLP@EMNLP*.
- Katryn, R.G., (2020). Text Preprocessing: Tahap Awal dalam Natural Language Processing (NLP). <https://medium.com/mandiri-engineering/text-preprocessing-tahap-awal-dalam-natural-language-processing-nlp-bc5fbb6606a#:~:text=Merupakan%20tahap%20awal%20dalam%20metode,bai k%20dan%20siap%20untuk%20diolah>.
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *Conference on Empirical Methods in Natural Language Processing*
- Li, Y., Tao, J., Chao, L., Bao, W., & Liu, Y. (2017). CHEAVD: a Chinese natural emotional audio–visual database. *Journal of Ambient Intelligence and Humanized Computing*, 8, 913-924.
- Man, R., & Lin, K. (2021). Sentiment Analysis Algorithm Based on BERT and Convolutional Neural Network. *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 769-772.
- Mohd, N., Singhdev, H., & Upadhyay, D. (2021). TEXT CLASSIFICATION USING CNN AND CNN-LSTM. *Webology*.
- Mohsin, M.A., & Beltiukov, A.P. (2019). Summarizing Emotions from Text Using Plutchik's Wheel of Emotions. *Proceedings of the 7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019)*.

- Muller, B., (2022). BERT 101 State Of The Art NLP Model Explained. <https://huggingface.co/blog/bert-101>.
- Peng, C., & Bao, T. (2020). An Analysis Method for Interpretability of CNN Text Classification Model. *Future Internet*, 12, 228.
- R. Abas, A., Elhenawy, I., Zidan, M., & Othman, M. (2022). BERT-CNN: A Deep Learning Model for Detecting Emotions from Text. *Computers, Materials & Continua*.
- Sarasati, B., & Nurvia, O. (2021). EMOSI DALAM TULISAN.
- Scherer, K.R., & Wallbott, H.G. (1994). Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66 2, 310-28.
- Setiawan, R., (2021). Apa itu Framework? Developer Wajib Tahu. <https://www.dicoding.com/blog/apa-itu-framework/>.
- Sun, Y., Zheng, Y., Hao, C., & Qiu, H. (2021). NSP-BERT: A Prompt-based Few-Shot Learner through an Original Pre-training Task —— Next Sentence Prediction. *International Conference on Computational Linguistics*.
- Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. *Neural Information Processing Systems*.
- Wuwungan, K. H. A. (2021). Bahasa Indonesia Sebagai Media Komunikasi Masyarakat Indonesia. In Universitas Negeri Gorontalo. <https://mahasiswa.ung.ac.id/151420069/home/2021/3/15/bahasa-indonesia-sebagai-media-komunikasi-masyarakat-indonesia.html>.
- Yu, Y., & Kim, Y. (2020). Attention-LSTM-Attention Model for Speech Emotion Recognition and Analysis of IEMOCAP Database. *Electronics*.
- Zhang, B. (2023). A BERT-CNN Based Approach on Movie Review Sentiment Analysis. *SHS Web of Conferences*.
- Zhang, J., & Qi, H. (2022). Data Mining and Spatial Analysis of Social Media Text Based on the BERT-CNN Model to Achieve Situational Awareness: a Case Study of COVID-19. *Journal of Geodesy and Geoinformation Science*, 5(2), 38.
- Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8.