

BAB II

TINJAUAN PUSTAKA

2.1 Analisis Sentimen (*Sentiment Analysis*)

2.1.1 Definisi Analisis Sentimen

Analisis sentimen, atau sering disebut sebagai penambangan opini (*opinion mining*), merupakan salah satu bidang kajian dalam *Natural Language Processing* (NLP) yang berfokus pada ekstraksi, identifikasi, dan klasifikasi opini yang terkandung dalam suatu teks. Secara filosofis, bidang ini berupaya untuk menjembatani kesenjangan antara komunikasi manusia yang subjektif dengan sistem komputasi yang objektif. Tujuan utamanya adalah untuk memahami kecenderungan sikap, emosi, atau opini publik terhadap suatu subjek, produk, atau layanan tertentu, sehingga hasil analisis tersebut dapat digunakan sebagai basis pengambilan keputusan yang strategis.

2.1.2 Tingkat Analisis Sentimen (*Level of Analysis*)

Dalam aplikasinya, analisis sentimen dapat dilakukan pada berbagai tingkatan, mulai dari tingkat dokumen (*document-level*), tingkat kalimat (*sentence-level*), hingga tingkat aspek (*aspect-level*). Penelitian ini menggunakan pendekatan analisis sentimen tingkat kalimat (*sentence-level sentiment analysis*). Pendekatan ini dipilih karena setiap ulasan pengguna di Google Play Store umumnya terdiri dari satu atau beberapa kalimat yang mewakili satu opini utuh mengenai aplikasi TikTok. Pada tingkat ini, setiap ulasan dianggap sebagai satu kesatuan unit informasi yang kemudian akan diklasifikasikan ke dalam satu kategori polaritas tertentu. Metode ini memberikan efisiensi yang lebih baik dalam memetakan opini pengguna secara langsung tanpa harus membedah detail aspek teknis di dalam ulasan tersebut.

2.1.3 Polaritas Sentimen

Polaritas sentimen merupakan proses penentuan arah emosi atau opini yang terkandung dalam sebuah teks. Dalam penelitian ini, klasifikasi polaritas dibagi menjadi tiga kelas utama, yaitu Positif, Negatif, dan Netral.

1. Sentimen Positif: Mewakili opini yang mengekspresikan kepuasan, apresiasi, atau pengalaman menyenangkan pengguna terhadap fitur aplikasi. Contoh: *"Aplikasi sangat membantu dan sangat mudah digunakan."*
2. Sentimen Negatif: Mewakili opini yang mengekspresikan keluhan, ketidakpuasan, atau kendala teknis yang dialami pengguna. Contoh: *"Aplikasi sering force close dan sangat mengganggu saat digunakan."*
3. Sentimen Netral: Mewakili opini yang tidak menunjukkan keberpihakan emosional yang kuat. Biasanya berisi pertanyaan, fakta, saran yang bersifat objektif, atau ulasan yang memiliki bobot emosi seimbang antara positif dan negatif sehingga tidak dominan ke arah manapun. Contoh: *"Aplikasi sudah terupdate ke versi terbaru."*

Klasifikasi tiga kelas ini memungkinkan model untuk memberikan gambaran yang lebih detail mengenai respons pengguna dibandingkan dengan klasifikasi biner (hanya positif dan negatif), sehingga memberikan spektrum pemahaman yang lebih kaya mengenai persepsi pengguna terhadap TikTok.

2.2 Prapemrosesan Teks (*Text Preprocessing*)

Data ulasan pengguna yang diperoleh melalui *web scraping* dari media sosial seperti TikTok memiliki karakteristik yang sangat tidak terstruktur (*unstructured data*). Teks tersebut sering kali mengandung berbagai *noise* seperti singkatan, kesalahan penulisan, simbol-simbol yang tidak relevan, hingga penggunaan huruf kapital yang tidak konsisten. Prapemrosesan teks merupakan tahap krusial untuk mentransformasi teks mentah menjadi format yang lebih bersih dan standar. Tujuannya adalah untuk meningkatkan kualitas representasi data sehingga model *Deep Learning* dapat mengekstraksi fitur semantik dengan lebih akurat. Tanpa prapemrosesan, model akan menganggap berbagai variasi penulisan kata yang sama sebagai entitas yang berbeda, yang pada akhirnya akan mendistorsi performa klasifikasi.

2.2.1 Case Folding

Case folding adalah tahap penyelarasan karakter dengan mengubah seluruh huruf dalam teks menjadi huruf kecil (*lowercase*). Hal ini dilakukan karena dalam komputasi teks, karakter huruf besar dan kecil memiliki nilai *ASCII* yang berbeda. Dengan menyamakannya, model tidak akan menganggap kata "TikTok", "TIKTOK", dan "tiktok" sebagai tiga entitas yang berbeda, sehingga dimensi kosa kata menjadi lebih efisien.

2.2.2 Cleansing

Tahap *cleansing* bertujuan untuk membersihkan teks dari elemen-elemen yang tidak mengandung informasi sentimen (*non-informative features*). Elemen-elemen ini mencakup tautan (*URL*), *username* (seperti @nama), tagar (#), tanda baca, angka, serta emoji. Pembersihan ini dilakukan menggunakan *Regular Expression* (RegEx) untuk memastikan bahwa data yang masuk ke model benar-benar berupa teks murni yang merepresentasikan opini pengguna.

2.2.3 Normalization

Normalisasi merupakan tahap yang sangat krusial dalam konteks ulasan TikTok. Pengguna aplikasi sering kali menggunakan bahasa gaul (*slang*) dan singkatan yang tidak baku (seperti "bgt" untuk "banget", "ngelag" untuk "lambat"). Tahap ini menggunakan kamus translasi (*dictionary mapping*) untuk mengubah kata-kata tidak baku tersebut menjadi kata baku sesuai dengan kaidah bahasa Indonesia. Langkah ini sangat membantu model IndoBERT untuk melakukan pemetaan makna secara lebih presisi karena model bahasa tersebut dilatih menggunakan korpus bahasa baku.

2.2.4 Stopword Removal

Stopword removal merupakan tahap prapemrosesan teks yang bertujuan untuk membuang kata-kata umum (*common words*) yang memiliki frekuensi kemunculan sangat tinggi dalam suatu bahasa, namun tidak memberikan kontribusi makna atau muatan sentimen yang signifikan terhadap suatu kalimat. Kata-kata tersebut umumnya berupa kata depan, kata hubung, dan kata ganti, seperti "yang", "di", "dan", "dari", "ke", "adalah", atau "ini". Secara linguistik, kehadiran *stopword* di dalam korpus teks cenderung menambah dimensi kosakata (*vocabulary size*) tanpa memberi nilai diskriminatif terhadap proses klasifikasi. Jika kata-kata ini tidak dihilangkan, model berisiko mengalokasikan sebagian kapasitas pembelajarannya untuk mempelajari pola kata yang sebenarnya tidak relevan dengan tugas klasifikasi sentimen, sehingga dapat memperlambat proses konvergensi dan menambah *noise* pada representasi fitur. Dalam konteks bahasa Indonesia, proses *stopword removal* umumnya memanfaatkan daftar kata baku (*stopword list*) yang telah dikurasi sebelumnya. Salah satu pustaka (*library*) yang lazim digunakan untuk keperluan ini adalah Sastrawi, yaitu pustaka pemrosesan bahasa alami khusus bahasa Indonesia yang menyediakan daftar *stopword* baku serta dapat dikombinasikan dengan daftar kata tambahan sesuai kebutuhan domain penelitian,

misalnya istilah-istilah umum pada ulasan aplikasi yang tidak membawa muatan sentimen. Proses ini bekerja dengan mencocokkan setiap token hasil tokenisasi terhadap daftar *stopword*; apabila suatu token ditemukan cocok, maka token tersebut akan dieliminasi dari rangkaian kalimat sebelum masuk ke tahap ekstraksi fitur oleh model.

2.3 Natural Language Processing (NLP) dan Deep Learning

2.3.1 Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan irisan bidang ilmu komputer, kecerdasan buatan, dan linguistik yang berfokus pada interaksi antara komputer dan bahasa manusia. Tantangan utama dalam NLP adalah sifat bahasa manusia yang ambigu, kontekstual, dan dinamis. Mesin tidak secara langsung memahami kata-kata sebagaimana manusia; oleh karena itu, NLP berperan sebagai jembatan yang mentransformasi teks menjadi representasi numerik yang dapat diproses oleh sistem komputasi. Melalui NLP, mesin dilatih untuk mengenali pola, struktur tata bahasa, dan makna semantik di balik rangkaian kata, yang memungkinkan komputer untuk melakukan tugas-tugas cerdas seperti klasifikasi sentimen. Sebagaimana dijelaskan oleh Hotho et al. (2005), NLP merupakan bidang interdisipliner yang beririsan dengan text mining, information retrieval, machine learning, dan linguistik komputasional, yang secara bersama-sama berperan penting dalam proses ekstraksi pengetahuan dari data teks tidak terstruktur.

2.3.2 Deep Learning dalam Pemrosesan Data Tidak Terstruktur

Dalam pengembangan sistem klasifikasi sentimen, terdapat perbedaan mendasar antara *Machine Learning* konvensional dan *Deep Learning*. *Machine Learning* konvensional umumnya bergantung pada teknik *feature engineering* manual, di mana peneliti harus menentukan sendiri fitur kata atau frasa mana yang dianggap penting sebelum dimasukkan ke dalam model. Pendekatan ini sering kali kurang efektif saat dihadapkan pada data ulasan media sosial yang bersifat tidak terstruktur (*unstructured data*), seperti ulasan TikTok yang penuh dengan variasi bahasa informal. Sebaliknya, *Deep Learning* menawarkan keunggulan dalam kemampuan ekstraksi fitur secara otomatis (*automated feature extraction*). Berbeda dengan *Machine Learning* konvensional yang membutuhkan campur tangan manusia dalam pemilihan fitur, *Deep Learning* mampu mempelajari hierarki fitur secara langsung dari data mentah. Model ini

sangat efektif untuk memproses data yang tidak terstruktur, karena kemampuannya dalam menangkap dependensi kontekstual yang kompleks melalui lapisan-lapisan neuron tiruan (*neural networks*).

2.4 Arsitektur Model: IndoBERT

2.4.1 Konsep *Transformer*

Arsitektur *Transformer* diperkenalkan melalui penelitian seminal berjudul "*Attention is All You Need*" oleh Vaswani (2017). Berbeda dengan arsitektur tradisional berbasis *Recurrent Neural Networks* (RNN) atau *Long Short-Term Memory* (LSTM) yang memproses kata secara berurutan (*sequential*), *Transformer* menggunakan mekanisme *Self-Attention*. Mekanisme ini memungkinkan model untuk memberikan bobot kepentingan yang berbeda pada setiap kata dalam kalimat, terlepas dari jarak posisi kata tersebut. Hal ini membuat *Transformer* mampu menangkap hubungan ketergantungan jarak jauh (*long-range dependencies*) dalam teks dengan lebih efisien dan jauh lebih cepat melalui proses komputasi paralel.

2.4.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT merupakan model bahasa yang dikembangkan berdasarkan struktur *Encoder* dari *Transformer*. Keunggulan utama BERT terletak pada sifatnya yang *bidirectional* atau dua arah. Dalam model bahasa tradisional (seperti GPT atau model *autoregressive* lainnya), teks biasanya diproses dari kiri ke kanan. BERT mematahkan batasan tersebut dengan mempelajari konteks kata melalui representasi sisi kiri dan sisi kanan secara bersamaan. Dengan teknik *Masked Language Modeling* (MLM), BERT secara efektif mampu memahami makna sebuah kata berdasarkan keseluruhan konteks kalimat di sekitarnya, bukan hanya dari kata-kata yang mendahuluinya saja. Hal ini memberikan kedalaman pemahaman semantik yang jauh lebih akurat dibandingkan model yang bersifat satu arah.

2.4.3 IndoBERT

IndoBERT merupakan varian dari model BERT yang telah menjalani proses *pre-training* menggunakan korpus teks bahasa Indonesia yang masif, mencakup dokumen dari Wikipedia Indonesia, berita, dan berbagai sumber teks daring lainnya. Karena BERT asli (versi *multilingual*) dilatih dengan berbagai bahasa sekaligus, performanya sering kali tidak maksimal untuk bahasa yang spesifik. IndoBERT hadir sebagai solusi untuk

menangkap kekhasan struktur tata bahasa, kosakata, serta nuansa semantik bahasa Indonesia yang unik. Dalam penelitian ini, IndoBERT menjadi komponen utama karena kemampuan adaptasinya yang tinggi terhadap bahasa informal yang lazim ditemukan pada ulasan aplikasi. Penggunaan IndoBERT memungkinkan sistem untuk memahami nuansa sentimen dalam ulasan TikTok dengan lebih baik, bahkan ketika ulasan tersebut mengandung tata bahasa yang tidak baku, singkatan, atau frasa khas Indonesia, sehingga memberikan pondasi pemahaman konteks yang lebih solid dibandingkan model bahasa global lainnya.

2.5 Arsitektur Model: 1D-Convolutional Neural Network (1D-CNN)

2.5.1 Konsep 1D-CNN pada Teks

Secara historis, *Convolutional Neural Network* (CNN) dikenal luas sebagai arsitektur yang dominan dalam pengolahan citra digital (*computer vision*), di mana lapisan konvolusi digunakan untuk mendeteksi fitur visual seperti garis, tepi, atau bentuk pada gambar 2D. Namun, dalam penelitian ini, arsitektur tersebut diadaptasi ke dalam bentuk 1-Dimensi (1D-CNN) untuk kebutuhan pemrosesan teks. Berbeda dengan citra yang memiliki dimensi ruang (lebar dan tinggi), teks direpresentasikan sebagai deretan vektor yang berurutan. Dalam bentuk 1D-CNN, operasi konvolusi dilakukan dengan menggeser filter secara satu arah sepanjang dimensi urutan kata. Teknik ini sangat efektif untuk menangkap pola lokal atau frasa kata yang dikenal dengan istilah *n-gram*. Misalnya, filter dengan ukuran *kernel* 3 akan membaca tiga kata yang berdekatan secara simultan, sehingga model dapat mengenali hubungan semantik lokal seperti "sangat tidak puas" atau "aplikasi sangat bagus" yang merupakan indikator sentimen yang kuat. Sebagaimana dijelaskan oleh Suartika Eka Putra et al. (2016), arsitektur CNN pada dasarnya bekerja melalui dua tahap utama, yaitu tahap klasifikasi menggunakan feedforward dan tahap pembelajaran menggunakan backpropagation, di mana prinsip inilah yang kemudian diadaptasi ke dalam bentuk 1D-CNN untuk kebutuhan pemrosesan data teks pada penelitian ini.

2.5.2 Lapisan Konvolusi dan *Pooling*

Arsitektur 1D-CNN dalam penelitian ini bekerja melalui dua komponen operasional utama:

1. Lapisan Konvolusi (*Convolutional Layer*): Lapisan ini menggunakan filter dengan bobot yang dapat dipelajari (*learnable weights*). Filter tersebut bergerak

menyusuri representasi vektor kata untuk memetakan fitur-fitur lokal. Dengan menerapkan berbagai ukuran filter, model mampu mengenali beragam bentuk pola frasa, mulai dari frasa pendek hingga frasa yang lebih kompleks. Lapisan ini menghasilkan *feature map* yang merepresentasikan keberadaan pola frasa tertentu di sepanjang kalimat.

2. Lapisan *Pooling* (*Global Max Pooling*): Setelah lapisan konvolusi, fitur yang dihasilkan sering kali memiliki dimensi yang terlalu besar dan mengandung informasi yang kurang krusial. Lapisan *Global Max Pooling* diaplikasikan untuk melakukan reduksi dimensi dengan cara mengambil nilai fitur yang paling tinggi (maksimum) dari setiap *feature map*. Secara filosofis, proses ini berfungsi untuk melakukan "penyeleksian fitur," di mana model hanya mempertahankan informasi fitur yang paling dominan dan paling mewakili muatan sentimen dari seluruh ulasan.

Kombinasi antara lapisan konvolusi dan *pooling* inilah yang memberikan keunggulan pada model hibrid ini. Jika IndoBERT memberikan pemahaman konteks global secara utuh, maka 1D-CNN berperan sebagai penyaring yang mempertegas fitur-fitur lokal yang krusial bagi klasifikasi sentimen, sehingga prediksi akhir menjadi jauh lebih tajam dan objektif.

2.5.3 Fungsi Aktivasi *ReLU* (*Rectified Linear Unit*)

Dalam arsitektur jaringan saraf tiruan, fungsi aktivasi berperan penting untuk memperkenalkan sifat nonlinearitas ke dalam proses pembelajaran, sehingga model mampu mempelajari pola hubungan data yang kompleks dan tidak dapat direpresentasikan hanya melalui kombinasi linear sederhana. Salah satu fungsi aktivasi yang paling banyak digunakan pada arsitektur *Convolutional Neural Network* (CNN) adalah *Rectified Linear Unit* (ReLU).

Secara matematis, fungsi ReLU didefinisikan sebagai:

$$f(x) = \max(0, x)$$

Fungsi ini bekerja dengan cara mempertahankan nilai input apabila bernilai positif, dan mengubahnya menjadi nol apabila bernilai negatif. Dibandingkan dengan fungsi aktivasi konvensional lain seperti *Sigmoid* atau *Tanh*, ReLU memiliki keunggulan

komputasi yang lebih efisien karena operasinya yang sederhana, serta mampu mengurangi risiko *vanishing gradient*, yaitu kondisi ketika nilai gradien menjadi sangat kecil sehingga proses pembaruan bobot pada lapisan-lapisan awal jaringan menjadi sangat lambat. Dalam konteks arsitektur 1D-CNN untuk klasifikasi sentimen, ReLU diterapkan setelah operasi konvolusi untuk menyaring dan mempertegas fitur-fitur lokal yang relevan, sekaligus mempercepat proses pelatihan model.

2.5.4 Regularisasi *Dropout*

Overfitting merupakan salah satu permasalahan umum dalam pelatihan model *Deep Learning*, yaitu kondisi ketika model terlalu menyesuaikan diri (menghafal) terhadap data latih, sehingga performanya menurun secara signifikan ketika diuji terhadap data baru yang belum pernah dilihat sebelumnya (*unseen data*). Salah satu teknik regularisasi yang umum digunakan untuk mengatasi permasalahan ini adalah *Dropout*. *Dropout* bekerja dengan cara menonaktifkan sebagian neuron secara acak pada suatu lapisan jaringan saraf dengan probabilitas tertentu di setiap iterasi pelatihan (*epoch*). Dengan dinonaktifkannya sebagian neuron secara acak, jaringan dipaksa untuk tidak bergantung secara berlebihan pada kombinasi neuron tertentu, sehingga model dapat mempelajari representasi fitur yang lebih general dan tidak mudah menghafal pola spesifik pada data latih. Pada tahap pengujian (*inference*), seluruh neuron kembali diaktifkan secara penuh, namun bobot keluarannya disesuaikan (*scaled*) agar konsisten dengan proses pelatihan.

2.5.5 *Fully Connected Layer* dan Fungsi *Softmax*

Fully Connected Layer (lapisan terhubung penuh), atau sering disebut juga *Dense Layer* atau *Linear Layer*, merupakan lapisan yang menghubungkan setiap neuron pada suatu lapisan dengan seluruh neuron pada lapisan berikutnya secara penuh. Dalam arsitektur klasifikasi berbasis *Deep Learning*, lapisan ini umumnya diposisikan pada bagian akhir jaringan dan berfungsi untuk memetakan (*mapping*) representasi fitur berdimensi tinggi hasil ekstraksi lapisan-lapisan sebelumnya menjadi sejumlah nilai keluaran (*output*) yang sesuai dengan jumlah kelas target klasifikasi. Nilai keluaran mentah dari *Fully Connected Layer* (disebut sebagai *logits*) belum merepresentasikan nilai probabilitas yang dapat diinterpretasikan secara langsung. Oleh karena itu, diperlukan fungsi *Softmax* untuk mengubah nilai *logits* tersebut menjadi distribusi probabilitas, di mana jumlah seluruh probabilitas pada setiap kelas bernilai satu (1).

Secara matematis, fungsi *Softmax* untuk kelas ke- i dari total K kelas dirumuskan sebagai berikut:

$$\text{Softmax}(z_i) = (e^{(z_i)}) / \left(\sum_{j=1}^K e^{(z_j)} \right)$$

Kelas dengan nilai probabilitas tertinggi dari hasil fungsi *Softmax* inilah yang kemudian ditetapkan sebagai label prediksi akhir model. Dalam konteks penelitian ini, fungsi *Softmax* digunakan untuk menentukan keputusan akhir klasifikasi sentimen ke dalam tiga kelas, yaitu Positif, Negatif, dan Netral.

2.6 Tokenisasi dan *Word Embedding*

Proses konversi dari teks ulasan mentah berbasis karakter bahasa manusia menjadi bentuk input yang dapat dipahami oleh arsitektur *Deep Learning* melibatkan dua tahapan matematis utama, yaitu tokenisasi (*tokenization*) dan representasi kata ke dalam ruang vektor (*word embedding*).

2.6.1 Konsep Tokenisasi (*Tokenization*)

Tokenisasi adalah proses pemecahan satu rangkaian teks atau kalimat utuh menjadi potongan-potongan satuan bahasa yang lebih kecil yang disebut sebagai **token**. Token ini dapat berupa kata tunggal, frasa, maupun potongan suku kata (*sub-words*). Dalam penelitian ini, proses tokenisasi tidak menggunakan metode pemotongan spasi sederhana (*whitespace tokenizer*), melainkan menggunakan *IndoBERT Tokenizer* yang mengadopsi algoritma *WordPiece*. Keunggulan dari metode ini adalah kemampuannya untuk menangani masalah kosakata yang tidak terdaftar dalam kamus (*Out-of-Vocabulary* atau OOV). Jika sistem menemukan kata asing atau kata gaul khas ulasan TikTok yang tidak dikenali secara utuh, algoritma akan memecah kata tersebut menjadi beberapa sub-kata (misalnya, kata "ngelag" dapat dipecah menjadi token "nge" dan "##lag"). Setiap token yang dihasilkan kemudian akan dipetakan ke dalam sebuah indeks numerik (*token ID*) berdasarkan daftar kosakata (*vocabulary list*) bawaan model IndoBERT.

2.6.2 Konsep Word Embedding

Setelah teks diubah menjadi barisan indeks numerik melalui tahap tokenisasi, langkah selanjutnya adalah melakukan *Word Embedding*. *Word embedding* adalah teknik

pemetaan (*mapping*) setiap indeks token ke dalam ruang vektor berdimensi tinggi (*high-dimensional vector space*). Berbeda dengan teknik tradisional seperti *One-Hot Encoding* atau TF-IDF yang menghasilkan matriks jarang (*sparse matrix*) berdimensi besar tanpa makna kontekstual, *word embedding* merepresentasikan setiap kata sebagai vektor angka riil yang bersifat padat (*dense vector*). Pada penelitian ini, dimensi vektor yang digunakan adalah 768 dimensi, mengikuti arsitektur dasar IndoBERT. Konsep dasar representasi kata ke dalam ruang vektor ini pertama kali dipopulerkan oleh Mikolov et al. (2013) melalui Word2Vec, yang membuktikan bahwa kata-kata dengan makna serupa dapat direpresentasikan dengan vektor yang saling berdekatan dalam ruang berdimensi tinggi, sebuah prinsip yang kemudian menjadi dasar pengembangan model embedding kontekstual seperti IndoBERT.

Word embedding memiliki dua karakteristik ilmiah yang krusial bagi analisis sentimen:

1. Kedekatan Semantik (*Semantic Similarity*): Kata-kata yang memiliki arti mirip atau sering digunakan dalam konteks sentimen yang sama (seperti "bagus" dan "keren") akan memiliki nilai vektor yang berdekatan di dalam ruang dimensi tersebut.
2. Konteks Dinamis (*Contextualized Embedding*): Karena *embedding* ini dihasilkan oleh lapisan *Transformer* IndoBERT, nilai vektor suatu kata tidak bersifat statis, melainkan berubah secara dinamis mengikuti kata-kata di sekitarnya. Hal ini sangat penting untuk membedakan makna kata yang ambigu (misalnya, frasa "bisa diandalkan" dengan "bisa ular") sehingga model dapat mendeteksi polaritas sentimen ulasan TikTok secara jauh lebih presisi.

2.7 Optimasi dan Fungsi Kerugian dalam Pelatihan Model

Proses pelatihan (*training*) model *Deep Learning* pada dasarnya merupakan proses optimasi, yaitu upaya untuk menemukan kombinasi bobot (*weight*) jaringan saraf yang meminimalkan tingkat kesalahan prediksi model. Proses ini melibatkan beberapa komponen utama, yaitu algoritma pengoptimasi, laju pembelajaran, fungsi kerugian, dan jumlah putaran pelatihan.

2.7.1 Algoritma Optimasi *AdamW*

Adam (*Adaptive Moment Estimation*) merupakan salah satu algoritma optimasi yang populer digunakan dalam pelatihan model *Deep Learning* karena kemampuannya menyesuaikan laju pembelajaran secara adaptif untuk setiap parameter berdasarkan

estimasi momen pertama dan kedua dari gradien. *AdamW* (*Adam with Decoupled Weight Decay*) merupakan varian pengembangan dari algoritma Adam, yang memisahkan mekanisme *weight decay* (penyusutan bobot) dari perhitungan momentum gradien. Pemisahan ini terbukti memberikan regularisasi yang lebih baik dan stabil, sehingga AdamW banyak direkomendasikan dan menjadi standar dalam proses *fine-tuning* model bahasa berbasis *Transformer* seperti BERT dan IndoBERT, karena mampu mencegah pembaruan bobot yang terlalu ekstrem yang dapat merusak pengetahuan hasil *pre-training* model.

2.7.2 Laju Pembelajaran (*Learning Rate*)

Learning rate merupakan hyperparameter yang menentukan seberapa besar langkah pembaruan bobot jaringan saraf pada setiap iterasi pelatihan berdasarkan nilai gradien yang dihitung. Nilai *learning rate* yang terlalu besar dapat menyebabkan proses pelatihan menjadi tidak stabil atau bahkan gagal mencapai titik konvergensi, sedangkan nilai yang terlalu kecil dapat menyebabkan proses pelatihan berjalan sangat lambat atau terjebak pada solusi lokal yang belum optimal (*local minimum*). Dalam skenario *fine-tuning* model *Pre-trained Language Model* seperti IndoBERT, umumnya digunakan nilai *learning rate* yang relatif kecil. Hal ini bertujuan untuk menjaga agar representasi pengetahuan bahasa yang telah dipelajari model selama proses *pre-training* tidak mengalami kerusakan drastis (*catastrophic forgetting*) akibat pembaruan bobot yang terlalu agresif terhadap data baru yang berukuran relatif lebih kecil.

2.7.3 Fungsi Kerugian *Cross-Entropy Loss*

Fungsi kerugian (*loss function*) digunakan untuk mengukur seberapa besar penyimpangan antara hasil prediksi model dengan label aktual (*ground truth*). *Cross-Entropy Loss* merupakan fungsi kerugian standar yang digunakan pada permasalahan klasifikasi multi-kelas (*multi-class classification*), termasuk klasifikasi sentimen tiga kelas (Positif, Negatif, Netral). Secara konseptual, *Cross-Entropy Loss* menghitung jarak (*divergence*) antara distribusi probabilitas hasil prediksi model (keluaran fungsi *Softmax*) dengan distribusi probabilitas label aktual. Semakin besar penyimpangan antara prediksi dan label sebenarnya, semakin besar pula nilai kerugian yang dihasilkan. Nilai kerugian inilah yang kemudian digunakan sebagai acuan dalam proses *backpropagation* untuk memperbarui bobot jaringan saraf agar tingkat kesalahan model dapat diminimalkan pada iterasi berikutnya.

2.7.4 *Epoch* dalam Pelatihan Model

Epoch didefinisikan sebagai satu kali putaran penuh proses pelatihan yang melibatkan keseluruhan data latih (*training set*). Jumlah *epoch* merupakan salah satu hyperparameter penting yang perlu ditentukan sebelum proses pelatihan dijalankan, karena berkaitan erat dengan keseimbangan antara *underfitting* dan *overfitting*. Jumlah *epoch* yang terlalu sedikit berisiko menyebabkan model belum sepenuhnya mempelajari pola yang terkandung dalam data (*underfitting*), sedangkan jumlah *epoch* yang berlebihan berisiko menyebabkan model terlalu menghafal pola data latih sehingga kehilangan kemampuan generalisasi terhadap data baru (*overfitting*). Pada proses *fine-tuning* model berbasis *Transformer*, jumlah *epoch* yang relatif kecil (umumnya berkisar 3 hingga 5 *epoch*) telah banyak terbukti cukup untuk mencapai performa optimal, mengingat model dasar telah memiliki representasi pengetahuan bahasa yang matang dari hasil *pre-training* sebelumnya.

2.8 Evaluasi Kinerja Model (*Performance Metrics*)

Untuk mengukur tingkat keberhasilan dan keandalan dari arsitektur model *Deep Learning* yang dibangun, diperlukan sebuah standar evaluasi yang objektif. Pengujian performa model klasifikasi pada penelitian ini dilakukan dengan menggunakan alat ukur *Confusion Matrix* serta metrik evaluasi turunannya yang meliputi *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Penggunaan metrik-metrik ini merujuk pada kerangka evaluasi yang dijelaskan oleh Powers (2011), yang menekankan pentingnya memahami keterbatasan dan bias dari masing-masing metrik agar performa model klasifikasi dapat diinterpretasikan secara objektif, terutama pada kasus dataset dengan distribusi kelas yang tidak seimbang.

2.8.1 Confusion Matrix

Confusion Matrix adalah tabel kontingensi yang digunakan untuk mengukur kinerja suatu model klasifikasi dengan cara membandingkan label kelas aktual (fakta di lapangan) dengan label kelas yang diprediksi oleh model. Melalui matriks ini, kita dapat melihat jumlah data yang berhasil diprediksi dengan benar maupun data yang salah diklasifikasikan (*misclassification*). Elemen-elemen utama dalam perhitungan ini meliputi:

1. *True Positive* (TP): Jumlah data positif yang diprediksi dengan benar sebagai positif.

2. *True Negative* (TN): Jumlah data negatif yang diprediksi dengan benar sebagai negatif.
3. *False Positive* (FP): Jumlah data negatif yang salah diprediksi sebagai positif (Kesalahan Tipe I).
4. *False Negative* (FN): Jumlah data positif yang salah diprediksi sebagai negatif (Kesalahan Tipe II).

2.8.2 Metrik Evaluasi Kuantitatif

Berdasarkan nilai dari komponen *Confusion Matrix* tersebut, maka dapat dihitung nilai metrik evaluasi formal sebagai berikut:

1. Akurasi (*Accuracy*): Metrik yang menunjukkan persentase total prediksi benar (baik positif maupun negatif) dari keseluruhan jumlah data yang diuji. Akurasi dihitung menggunakan rumus berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. Presisi (*Precision*): Metrik yang mengukur tingkat ketepatan antara data yang diminta dengan hasil prediksi yang dikeluarkan oleh model. Presisi menjawab pertanyaan seberapa banyak data yang diprediksi positif benar-benar bernilai positif. Rumus presisi adalah:

$$Precision = \frac{TP}{TP + FP}$$

3. Sensitivitas (*Recall*): Metrik yang mengukur kemampuan model dalam menemukan kembali (*retrieve*) seluruh data yang berlabel positif asli. *Recall* sangat penting untuk melihat seberapa banyak data positif yang terlewat oleh model. Rumusnya adalah:

$$Recall = \frac{TP}{TP + FN}$$

4. F1-Score: Metrik yang menghitung nilai rata-rata harmonis (*harmonic mean*) antara *Precision* dan *Recall*. *F1-Score* digunakan sebagai indikator utama penentu performa model ketika dataset yang digunakan memiliki distribusi kelas yang tidak seimbang (*imbalanced dataset*). Rumus untuk menghitung *F1-Score* adalah:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$