BAB II
KAJIAN LITERATUR

2.1 Penelitian Terdahulu

Berbagai penelitian sebelumnya telah dilakukan untuk mengeksplorasi efektivitas model bahasa *Transformer*, pemrosesan teks pada media sosial, serta penanganan masalah ketidakseimbangan kelas (*class imbalance*) menggunakan teknik augmentasi data [10], [15]. Pemaparan dari berbagai studi tersebut menjadi landasan berpijak untuk memosisikan penelitian ini.

Terkait dengan keandalan pemrosesan bahasa alami (NLP) pada teks berbahasa Indonesia, model IndoBERT terbukti memiliki performa yang sangat tangguh dan mengungguli metode klasifikasi tradisional. Penelitian oleh Alfatah (2024) yang mendeteksi sentimen pada 10.000 data Twitter berbahasa Indonesia membuktikan bahwa pemahaman kontekstual IndoBERT jauh lebih unggul dengan capaian akurasi 87%, mengalahkan algoritma konvensional seperti *Support Vector Machine* (SVM) yang hanya memperoleh 78% dan Naïve Bayes sebesar 73% [10]. Keunggulan arsitektur *Transformer* juga dikonfirmasi oleh Apriansyah dkk. (2025) yang membandingkan IndoBERT dan IndoRoBERTa untuk analisis sentimen pada komentar YouTube mengenai film dokumenter politik *Dirty Vote* [1]. Hasil penelitian tersebut menunjukkan bahwa IndoBERT memiliki performa yang lebih stabil dan sensitif terhadap ekspresi kritik secara eksplisit dengan perolehan akurasi rata-rata 99%, sementara IndoRoBERTa berada di angka 94%.

Dalam konteks analisis opini di platform TikTok, platform ini telah menjadi media yang krusial karena tingginya interaksi pengguna terhadap tren dan isu-isu publik [16]. Studi yang dilakukan oleh Manalu dkk. (2025) menganalisis 5.000 komentar video yang sedang tren di TikTok dengan membandingkan algoritma Naïve Bayes, SVM, dan *Random Forest* [2]. Mereka menemukan bahwa SVM sangat cocok untuk mengklasifikasikan teks pendek dan informal khas TikTok dengan akurasi 89,7%. Namun, ketika pendekatan *deep learning* dilibatkan, kinerja klasifikasi menjadi jauh lebih baik. Hal ini dibuktikan oleh Setiawan dkk. (2023) yang meneliti 50.000 ulasan aplikasi TikTok menggunakan algoritma *Long Short-Term Memory* (LSTM) dan IndoBERTweet [17]. Penelitian tersebut menunjukkan bahwa IndoBERTweet, yang merupakan model pralayang turunan BERT khusus kosakata Twitter/media sosial, mampu menghasilkan akurasi tertinggi sebesar 80%, mengungguli metode LSTM yang meraih akurasi 78%.

Lebih lanjut, tantangan terbesar dalam klasifikasi teks berbasis ulasan pengguna atau komentar publik adalah masalah ketidakseimbangan data (*class imbalance*), yang menyebabkan model kesulitan memprediksi kelas minoritas secara akurat. Untuk mengatasi kelemahan ini, teknik algoritma *Synthetic Minority Over-sampling Technique* (SMOTE) telah diuji dan terbukti efektif. Ferdiansyah dan Suryono (2025) menerapkan kombinasi IndoBERT dan optimasi SMOTE untuk mengklasifikasikan sentimen ulasan berbahasa Indonesia pada *game* Roblox [15]. Kombinasi tersebut secara signifikan berhasil memperkuat representasi kelas minoritas, sehingga model mencapai akurasi sebesar 94%, dengan nilai presisi, *recall*, dan *F1-score* yang sangat stabil di angka 99% [17]. Temuan serupa juga dihasilkan oleh A'la (2025) yang menggunakan model *IndoBERT-base-uncased* yang dipadukan dengan SMOTE untuk menganalisis 998 ulasan *game* di Google Play [12]. Hasil evaluasinya menunjukkan bahwa implementasi SMOTE sukses meningkatkan keseimbangan data, mendongkrak akurasi dari 69,5% menjadi 72,5%, serta menaikkan *F1-score* dari 0,46 menjadi 0,47.

Elgeldawi dkk. (2021) menganalisis pengaruh optimasi *hyperparameter* terhadap enam algoritma pembelajaran mesin untuk analisis sentimen teks berbahasa Arab [14]. Penelitian tersebut menerapkan *Grid Search*, *Random Search*, *Bayesian Optimization*, *Particle Swarm Optimization* (PSO), dan *Genetic Algorithm* (GA), dengan hasil terbaik diperoleh oleh *Support Vector Classifier* melalui *Bayesian Optimization* dengan akurasi 95,6208%. Temuan ini menunjukkan bahwa penyetelan *hyperparameter* secara sistematis penting dilakukan untuk memperoleh konfigurasi model yang optimal, sehingga pendekatan serupa dapat diterapkan pada MLPClassifier sebagai model pembanding dalam penelitian deteksi keberpihakan komentar TikTok [14].

Tabel 2.1 Perbandingan Penelitian Terdahulu

Peneliti	Fokus Penelitian	Metode	Objek	Hasil Penelitian
Alfatah (2024) [10]	Penerapan Model Transformer Untuk Deteksi Sentimen Pada Data Twitter Berbahasa Indonesia	IndoBERT, SVM, dan Naive Bayes	10.000 cuitan Twitter berbahasa Indonesia	Model IndoBERT terbukti sangat efektif dan mengungguli metode lain dengan tingkat akurasi sebesar 87%, sedangkan SVM hanya 78% dan Naïve Bayes 73%.

Apriansyah, dkk. (2025) [1]	Perbandingan IndoBERT dan IndoRoBERTa Untuk Analisis Sentimen Pada Film Dokumenter Dirty Vote	IndoBERT dan IndoRoBERTa	Komentar publik dari YouTube mengenai film Dirty Vote	IndoBERT memperoleh akurasi rata-rata 99%, lebih unggul dan konsisten menangkap kritik secara eksplisit dibandingkan IndoRoBERTa yang mencapai akurasi 94%.
Elgeldawi, Sayed, Galal, dan Zaki (2021)[14]	Hyperparameter Tuning for Machine Learning Algorithms Used for Arabic Sentiment Analysis	MLP, SVC, serta beberapa algoritma pembelajaran mesin lainnya; *Grid Search*, *Random Search*, Bayesian Optimization, PSO, dan GA	Teks atau ulasan berbahasa Arab	Optimasi *hyperparameter* terbuk ti berpengaruh terhadap performa klasifikasi. Hasil terbaik diperoleh SVC menggunakan Bayesian Optimization dengan akurasi 95,6208%. Penelitian ini menunjukkan bahwa pengaturan parameter model, termasuk pada MLP, penting untuk memperoleh konfigurasi klasifikasi yang optimal
Manalu, dkk. (2025) [2]	Implementasi Algoritma Klasifikasi untuk Analisis Sentimen Media Sosial Tiktok Tahun 2025	Naive Bayes, Support Vector Machine (SVM), dan Random Forest	5.000 komentar video yang sedang tren di TikTok	SVM sangat cocok untuk teks pendek dan informal seperti komentar TikTok dengan capaian akurasi tertinggi sebesar 89,7% dibandingkan algoritma lainnya.
Setiawan, dkk. (2023) [17]	Sentiment Analysis of Indonesian TikTok Review Using LSTM and IndoBERTweet Algorithm	LSTM dan IndoBERTweet	50.000 data ulasan aplikasi TikTok di Google Play Store	Model prapelatih IndoBERTweet berhasil mengungguli algoritma LSTM dengan tingkat akurasi klasifikasi sebesar 80%, berbanding 78% pada LSTM.
Ferdiansyah & Suryono (2025) [15]	Sentiment Classification of Indonesian-Language Roblox	IndoBERT yang dioptimasi dengan SMOTE	Ulasan pengguna game Roblox	Kombinasi IndoBERT dan SMOTE berhasil menangani ketidakseimbangan kelas dan mencapai tingkat akurasi 94%,

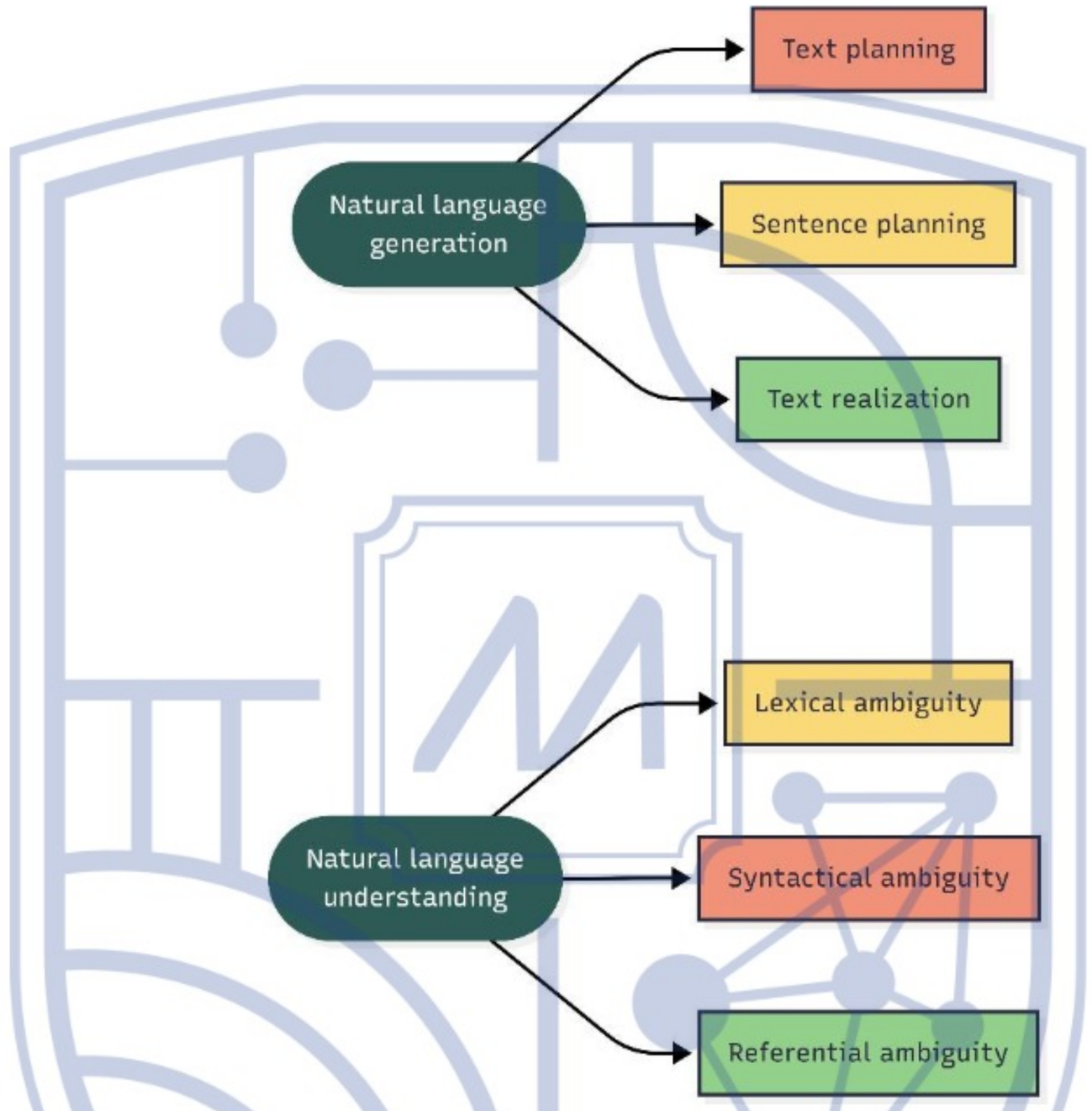
	Reviews Using IndoBERT with SMOTE Optimization		berbahasa Indonesia	dengan nilai presisi, *recall*, dan *F1-score* stabil di 99%.
A'la (2025) [12]	Optimasi Klasifikasi Sentimen Ulasan Game Berbahasa Indonesia: IndoBERT dan SMOTE untuk Menangani Ketidakseimbangan Kelas	IndoBERT-base-uncased dan algoritma SMOTE	998 ulasan aplikasi game di Google Play	Implementasi SMOTE berhasil meningkatkan keseimbangan data serta menaikkan akurasi model menjadi 72,5% dan *F1-score* dari 0,46 menjadi 0,47.

Dari narasi dan tabel literatur di atas, terlihat jelas bahwa model IndoBERT dan teknik optimasi SMOTE adalah kombinasi komputasional yang sangat andal untuk mengatasi bahasa informal media sosial dan ketimpangan data. Namun, mayoritas penelitian terdahulu hanya berfokus pada ranah analisis sentimen (mengukur positif/negatif) pada ulasan produk, film, atau game [15]. Penelitian Elgeldawi dkk. (2021) juga menunjukkan bahwa pemilihan dan optimasi *hyperparameter* merupakan tahap penting karena dapat memengaruhi kinerja algoritma klasifikasi secara signifikan. Namun, mayoritas penelitian terdahulu masih berfokus pada analisis sentimen, yaitu pengelompokan teks ke dalam kelas positif, negatif, atau netral, dan belum banyak mengkaji penggunaan MLPClassifier sebagai pembanding IndoBERT pada tugas *stance detection* berbahasa Indonesia [14]. Masih terdapat ruang kosong (celah penelitian) dalam menerapkan kombinasi arsitektur IndoBERT dan algoritma SMOTE untuk tugas *stance detection* (deteksi keberpihakan atau sikap) yang menargetkan wacana sosial-hukum yang polarisasinya sangat ekstrem, seperti pada kasus warganet TikTok merespons penetapan tersangka Hogi Minaya. Narasi ini secara langsung menegaskan bahwa penelitian ini hadir sebagai solusi mutakhir yang mengadaptasi metode-metode terbaik dari penelitian sebelumnya untuk menyelesaikan masalah pada studi kasus yang benar-benar baru.

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) merupakan bidang antardisiplin yang berada di persimpangan antara kecerdasan buatan (*Artificial Intelligence*), ilmu komputer, dan linguistik [18], [19], [20]. Tujuan utama dari NLP adalah untuk menjembatani kesenjangan komunikasi antara manusia dan mesin dengan memungkinkan komputer untuk memahami,

menafsirkan, memanipulasi, dan menghasilkan bahasa manusia [21], [22]. Seiring dengan pertumbuhan eksponensial data digital, NLP telah menjadi teknologi esensial yang memungkinkan komputer untuk memproses volume data teks yang sangat besar, baik terstruktur maupun tidak terstruktur, menjadi informasi komputasional yang bernilai dan dapat dianalisis [3].



Gambar 2.1 Bagan Cabang NLP [22]

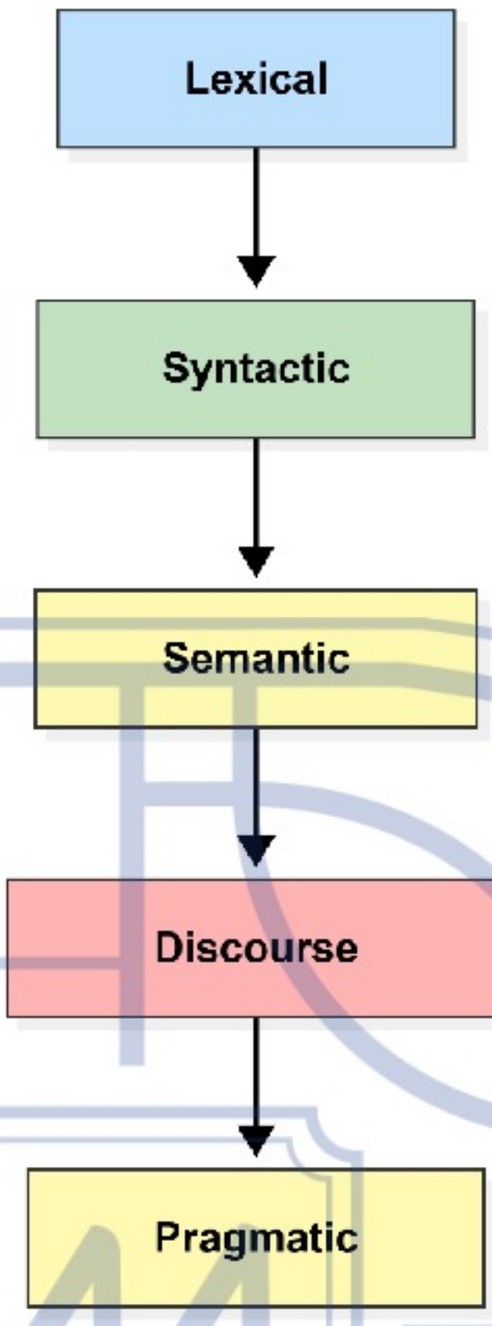
Secara konseptual dan struktural, sistem NLP umumnya dibagi menjadi dua bidang utama, yaitu *Natural Language Understanding* (NLU) dan *Natural Language Generation* (NLG) [20], [21], [22]:

1. ***Natural Language Understanding* (NLU):** Berfokus pada mesin untuk memahami makna dari teks yang diberikan. Tugas NLU sangat menantang karena harus memecahkan berbagai ambiguitas bahasa alami, seperti ambiguitas leksikal (kata dengan banyak makna), ambiguitas sintaksis (kalimat dengan struktur ganda), hingga ambiguitas semantik [22]. NLU merupakan fondasi utama dalam tugas analisis seperti *stance detection* maupun klasifikasi sentimen [10], [21].

2. ***Natural Language Generation* (NLG):** Berfokus pada kebalikannya, yaitu proses menghasilkan teks atau ucapan baru yang dapat dipahami manusia dari data yang terstruktur. Proses ini umumnya melalui tahapan perencanaan teks (*text planning*), perencanaan kalimat (*sentence planning*), dan realisasi teks [21], [22].

Dalam mengolah teks bahasa alami untuk mengekstrak makna yang akurat, sistem NLP secara teori memproses bahasa melalui lima fase hierarkis berikut [21], [22]:

1. **Analisis Leksikal dan Morfologis (*Lexical Analysis*):** Fase pertama yang membaca aliran karakter teks menjadi unit-unit yang bermakna (leksem). Fase ini membagi keseluruhan teks mentah menjadi bagian-bagian seperti paragraf, kalimat, dan kata (tokenisasi) [22].
2. **Analisis Sintaksis (*Syntactic Analysis / Parsing*):** Berfokus pada struktur gramatikal kalimat. Fase ini memeriksa tata bahasa, susunan kata, dan menunjukkan hubungan antar kata untuk memastikan bahwa urutan kalimat tersebut logis secara tata bahasa [3], [22].
3. **Analisis Semantik (*Semantic Analysis*):** Berfokus pada representasi makna dari kata, frasa, dan kalimat. Fase ini mengekstrak makna harfiah atau makna kamus yang tepat dari teks sesuai dengan konteks penggunaannya [3], [22].
4. **Integrasi Wacana (*Discourse Integration*):** Menganalisis ketergantungan makna sebuah kalimat berdasarkan kalimat-kalimat yang mendahuluinya, serta memberikan konteks untuk kalimat yang akan mengikutinya[22].
5. **Analisis Pragmatik (*Pragmatic Analysis*):** Fase terakhir dalam NLP yang menafsirkan efek atau tujuan yang dimaksudkan dari sebuah kalimat dengan menerapkan aturan konteks dunia nyata, sehingga sistem dapat membedakan antara makna harfiah dan makna kontekstual (seperti membedakan antara pertanyaan atau perintah) [22].



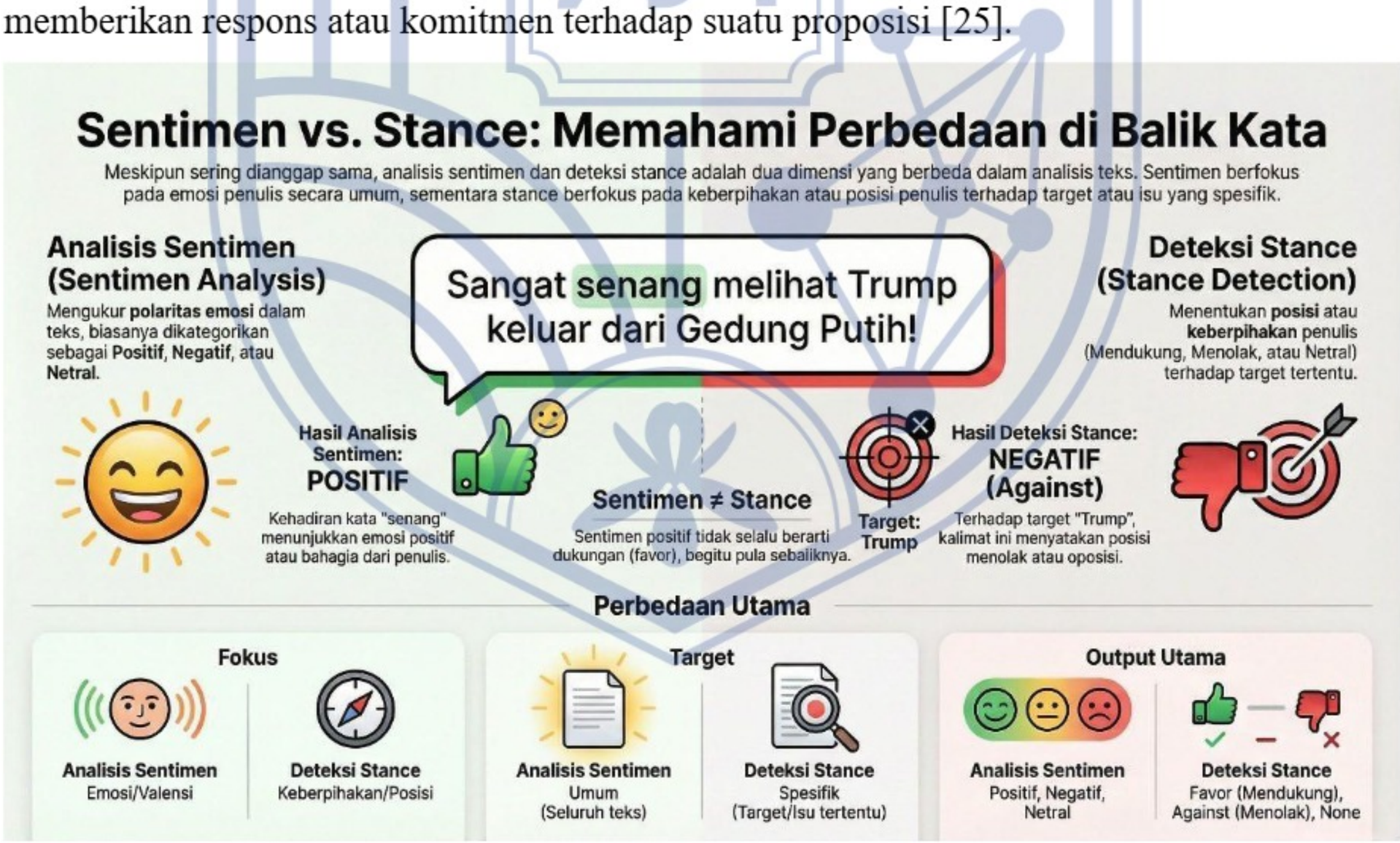
Gambar 2.2 Bagan alur fase pemrosesan bahasa [22]

Dalam konteks pemrosesan teks media sosial seperti platform TikTok, NLP menghadapi tantangan berat akibat karakteristik bahasa yang informal, struktur sintaksis yang kompleks, penggunaan variasi kosakata, serta kata gaul/singkatan (*slang*) [10], [23]. Untuk mengatasi keterbatasan metode aturan klasik (*rule-based*), algoritma NLP saat ini telah terintegrasi secara kuat dengan arsitektur *deep learning* dan *machine learning [24]*. Penerapan arsitektur berbasis Transformer (seperti model IndoBERT) telah secara drastis meningkatkan kemampuan NLP, karena model ini mampu membaca konteks secara dua arah (*bidirectional*) dan menangkap representasi tekstual yang sangat kompleks dari korpus bahasa Indonesia [3], [8]. Dengan kemajuan teknologi NLP tersebut, mesin kini mampu melakukan klasifikasi teks tingkat tinggi seperti *stance detection* secara otomatis dan objektif, menjadikannya kerangka kerja komputasional utama dalam penelitian ini.

2.3 Stance Detection dan Sentiment Analysis

Analisis sentimen, yang juga sering dikenal dengan istilah *opinion mining*, adalah sebuah metode komputasional untuk mengidentifikasi, mengekstrak, dan mengevaluasi informasi subjektif dari data tekstual guna menentukan sikap, emosi, atau opini penulis terhadap suatu subjek [3], [5]. Tujuan utama dari analisis sentimen adalah mengklasifikasikan teks berdasarkan polaritas emosinya ke dalam kategori positif, negatif, atau netral [5], [10]. Analisis ini sangat bergantung pada penangkapan valensi emosional dari kata-kata yang digunakan oleh penulis, dan telah diaplikasikan secara luas mulai dari mengukur kepuasan pelanggan hingga memantau opini publik secara umum [5].

Di sisi lain, *stance detection* (deteksi sikap) adalah tugas yang lebih spesifik dalam pemrosesan bahasa alami yang bertujuan untuk menentukan posisi, pandangan, atau keyakinan seseorang dari sebuah teks terhadap suatu target tertentu (seperti konsep, ide, kebijakan, atau tokoh), terlepas dari apakah target tersebut disebutkan secara eksplisit atau hanya tersirat [6]. Secara umum, *stance detection* mengkategorikan sikap individu ke dalam kelas keberpihakan, seperti *Favor/Support* (Mendukung), *Oppose* (Menolak), atau *None/Neutral* (Netral) [8]. *Stance* berkaitan erat dengan bagaimana seorang individu memberikan respons atau komitmen terhadap suatu proposisi [25].



Gambar 2.3 Perbandingan Sentimen dan Deteksi Sikap

Meskipun *stance detection* sering disamakan atau dianggap sebagai bagian dari analisis sentimen, berbagai literatur mutakhir secara tegas menyatakan bahwa sentimen dan *stance* adalah dua dimensi komputasional yang berbeda dan harus diukur secara independen

[25]. Perbedaan mendasar keduanya terletak pada target pengukuran dan cara ekspresi yang digunakan penulis. Tabel 2.2 berikut merangkum perbedaan utama antara kedua pendekatan tersebut:

Tabel 2.2 Perbandingan Sentiment Analysis dan Stance Detection [6]

Dimensi	Sentiment Analysis	Stance Detection
Tujuan	Mengukur polaritas emosi teks	Mengukur keberpihakan terhadap target spesifik
Output	Positif / Negatif / Netral	Favor / Against / None
Target	Subjek umum dalam teks	Target spesifik (tokoh, isu, kebijakan)
Ketergantungan konteks	Rendah - berbasis kata emosional	Tinggi - memerlukan pemahaman konteks
Sarkasme	Sering gagal mendeteksi	Dapat diidentifikasi jika konteks dipahami
Contoh	"Pelayanannya buruk" → Negatif	"Saya mendukung Hogi" → Favor

Perbedaan antara keduanya dapat diilustrasikan melalui dua kondisi utama berikut:

1. **Kehadiran Emosi yang Bertolak Belakang (Sarkasme):** Seseorang dapat menggunakan sentimen yang sangat positif untuk mengekspresikan sikap menolak atau menyerang suatu target. Sebagai contoh, kalimat *"Saya sangat senang tokoh A akhirnya dipenjara!"* memiliki polaritas sentimen yang positif karena mengandung kata 'senang', namun sebenarnya mengekspresikan stance yang negatif/menolak terhadap "tokoh A". Dalam konteks penelitian ini, komentar sarkastis terhadap aparat kepolisian di TikTok merupakan fenomena yang sangat umum terjadi [25].
2. **Ketiadaan Emosi (Sentiment-less Stance):** Sebuah teks bisa saja tidak memiliki kata-kata yang mengandung sentimen emosional sama sekali, namun memiliki stance yang sangat jelas. Misalnya, pernyataan *"Saya akan memilih kandidat B"* adalah respons dengan stance yang sangat positif (mendukung) terhadap "kandidat B", namun sama sekali tidak memuat isyarat emosi. Begitu pula komentar seperti *"Safaruddin benar"* tidak mengandung emosi eksplisit, namun secara jelas mengekspresikan keberpihakan [25].

Oleh karena itu, mengandalkan analisis sentimen saja untuk mengukur opini publik pada kasus Hogi Minaya seringkali menimbulkan bias atau kesalahan pengukuran (*measurement error*), karena posisi yang diekspresikan (sikap) sering kali berbeda dengan sentimen yang digunakan untuk menyampaikannya [25]. Hal inilah yang menjadi landasan

utama penelitian ini dalam mengadopsi pendekatan *stance detection* sebagai kerangka klasifikasi yang lebih presisi dan kontekstual.

Tabel 2.3 Sampel Teks dari Dataset Uji Semeval 2016 [25]

#	Teks	Label	Catatan
1	@ABC Bodoh adalah orang yang berbuat bodoh! Menunjukkan watak aslinya; tampaknya dia mengabaikan bahwa AS diinvasi, & dirampas, bukan ditemukan	*Against* (Kontra)	Berdasarkan pengetahuan eksternal bahwa ini tentang Trump, kemungkinan disimpulkan dari sikap mengenai narasi pendirian AS
2	@peddoc63 @realDonaldTrump Jadi menurutku Univision Adil & Seimbang. Inilah orang-orang yang sedang membantu membentuk USA!	*Against* (Kontra)	Menyimpulkan sikap terhadap Trump berdasarkan sikap positif terhadap Univision
3	Jujur aku akan lebih sering menonton #Univision sekarang, hanya untuk mendukung jaringan tersebut melawan	*Neutral* (Netral)	Mengekspresikan sikap yang sama dengan contoh #2, tetapi pemberi label tampaknya tidak memiliki informasi kontekstual yang sama saat menyimpulkan sikap

2.4 Pemanfaatan Large Language Model sebagai Alat Bantu Anotasi

Large Language Model (LLM) adalah model bahasa berbasis Transformer berparameter besar yang dilatih pada korpus teks berskala luas sehingga mampu memahami dan menghasilkan teks secara generatif dalam berbagai konteks. Kemampuan tersebut menjadikan LLM relevan tidak hanya untuk tugas generatif, tetapi juga untuk mendukung proses anotasi data dan konstruksi sumber daya linguistik dalam penelitian NLP [26].

Anotasi data secara manual merupakan proses yang memerlukan waktu, biaya, dan tenaga yang besar, serta rentan terhadap variasi penilaian antarpenanotasi. Karena itu, penggunaan LLM melalui mekanisme *prompt engineering* menjadi alternatif yang menjanjikan untuk menghasilkan label awal, daftar kata kunci, atau representasi anotasi lain secara lebih efisien. *Prompt engineering* sendiri memanfaatkan instruksi berbasis bahasa alami untuk mengarahkan model agar menghasilkan keluaran yang sesuai dengan kebutuhan tugas tanpa perlu mengubah parameter inti model [27].

Dalam konteks anotasi teks, LLM telah terbukti mampu mendukung berbagai tugas, termasuk relevansi, *stance*, topik, dan deteksi bingkai wacana. Hasil penelitian menunjukkan bahwa ChatGPT dapat mengungguli anotator manusia berbayar pada beberapa tugas anotasi teks, baik dari sisi akurasi *zero-shot* maupun kesepakatan antarpenilai. Selain itu, pada

anotasi leksikon, LLM juga dinilai mampu menjadi alat bantu yang efektif untuk menghasilkan anotasi awal yang kemudian tetap perlu disempurnakan oleh ahli domain [28].

Pada tugas yang bersifat spesifik domain, seperti *stance detection*, LLM dapat digunakan untuk membantu penyusunan kata kunci dan kriteria anotasi melalui pemberian deskripsi kasus, definisi kelas, serta contoh komentar representatif dalam bentuk *prompt*. Pendekatan ini penting karena data berlabel sering kali terbatas pada tahap awal penelitian, sehingga peneliti memerlukan alat bantu untuk mempercepat penyusunan kamus atau aturan pelabelan. Namun demikian, keluaran LLM tetap perlu diverifikasi secara manual agar sesuai dengan konteks domain dan tidak menghasilkan kata kunci yang terlalu umum atau kurang relevan [29].

Berdasarkan pertimbangan tersebut, penelitian ini memanfaatkan LLM sebagai alat bantu dalam membangun kamus kata kunci untuk proses *rule-based labeling*. LLM tidak digunakan sebagai model klasifikasi utama, melainkan sebagai instrumen awal untuk menghasilkan kandidat kata dan frasa yang merepresentasikan Kelas 0. Selanjutnya, daftar tersebut diseleksi dan divalidasi secara manual oleh peneliti sebelum digunakan sebagai acuan pelabelan data komentar TikTok kasus Hogi Minaya [30].

2.5 Arsitektur Transformer dan Mekanisme Attention

Model *Transformer* pertama kali diperkenalkan oleh Vaswani dkk. pada tahun 2017 untuk mengatasi keterbatasan yang ada pada arsitektur jaringan saraf sekuensial konvensional, seperti *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM) [31]. Kelemahan utama RNN adalah sifat pemrosesannya yang harus dilakukan kata demi kata secara berurutan, yang menyebabkan proses komputasi menjadi lambat dan kesulitan dalam menangkap ketergantungan konteks jarak jauh (*long-term dependencies*) akibat masalah *vanishing gradient* [8]. Arsitektur Transformer secara revolusioner menyelesaikan masalah ini dengan memproses seluruh urutan masukan secara paralel menggunakan mekanisme *Self-Attention*, tanpa menggunakan elemen rekursi (RNN) sama sekali [23].

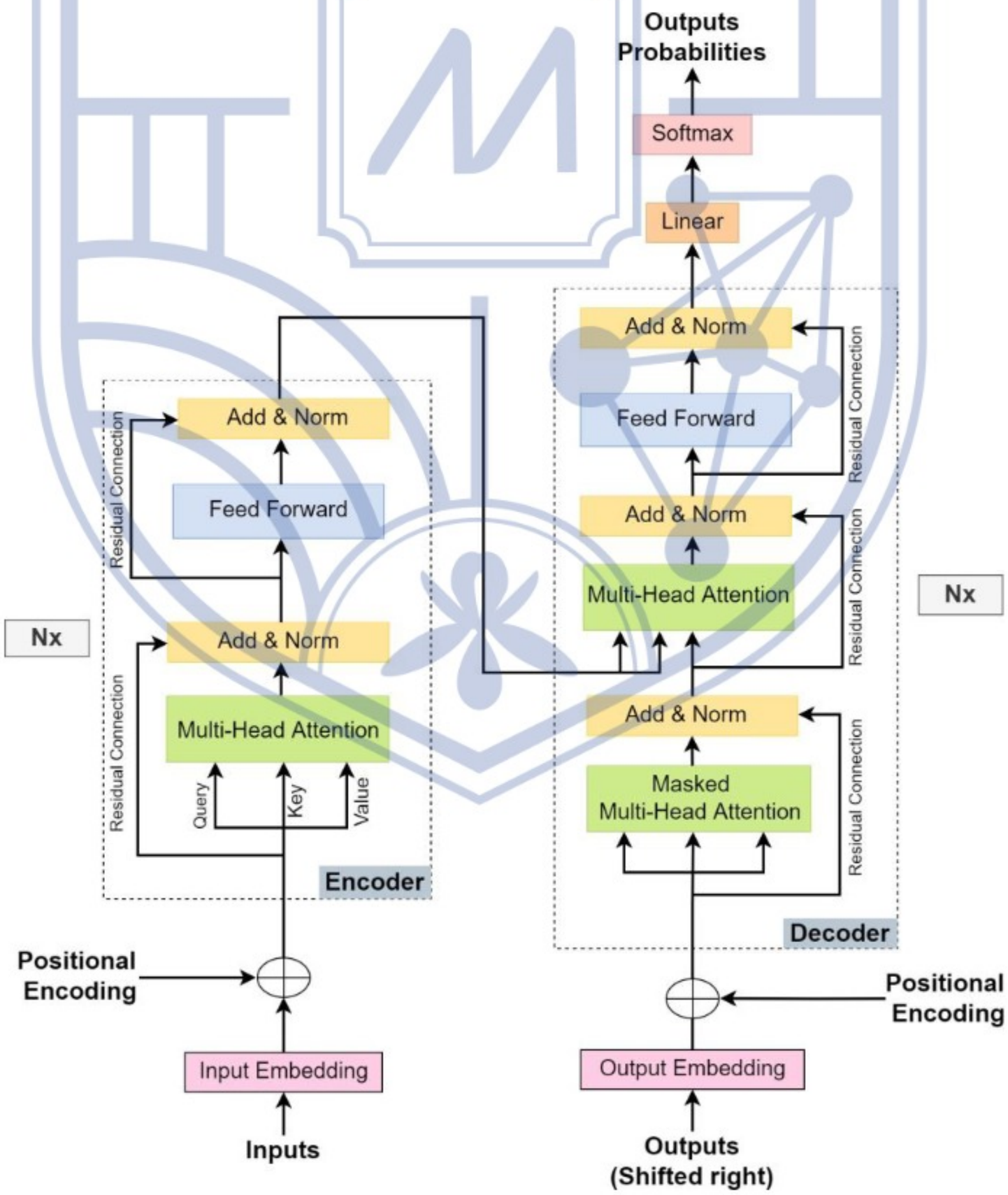
Inti dari kinerja Transformer adalah mekanisme *Scaled Dot-Product Attention*. Mekanisme ini mengevaluasi seberapa penting sebuah kata (token) terhadap kata-kata lainnya di dalam sebuah kalimat [31], [32]. Komputasi *attention* melibatkan tiga matriks parameter utama: matriks *Query* (Q), *Key* (K), dan *Value* (V) yang membawa representasi

dari setiap masukan [31]. Formula matematis untuk menghitung bobot *attention* ini dinyatakan sebagai berikut:

$$AAAAAAAAAAAAAAAA(QQ, KK, VV) = ssAAssAAssssss\left(\frac{QQQQ^{TT}}{dd_{kk}}\right)VV \quad (1)$$

Di mana matriks Q dan K memiliki dimensi dd_{kk}, dan perkalian dot-product di antara keduanya akan menghasilkan skor yang dinormalisasi dengan fungsi *Softmax* untuk menentukan nilai probabilitas/bobot perhatian setiap kata.

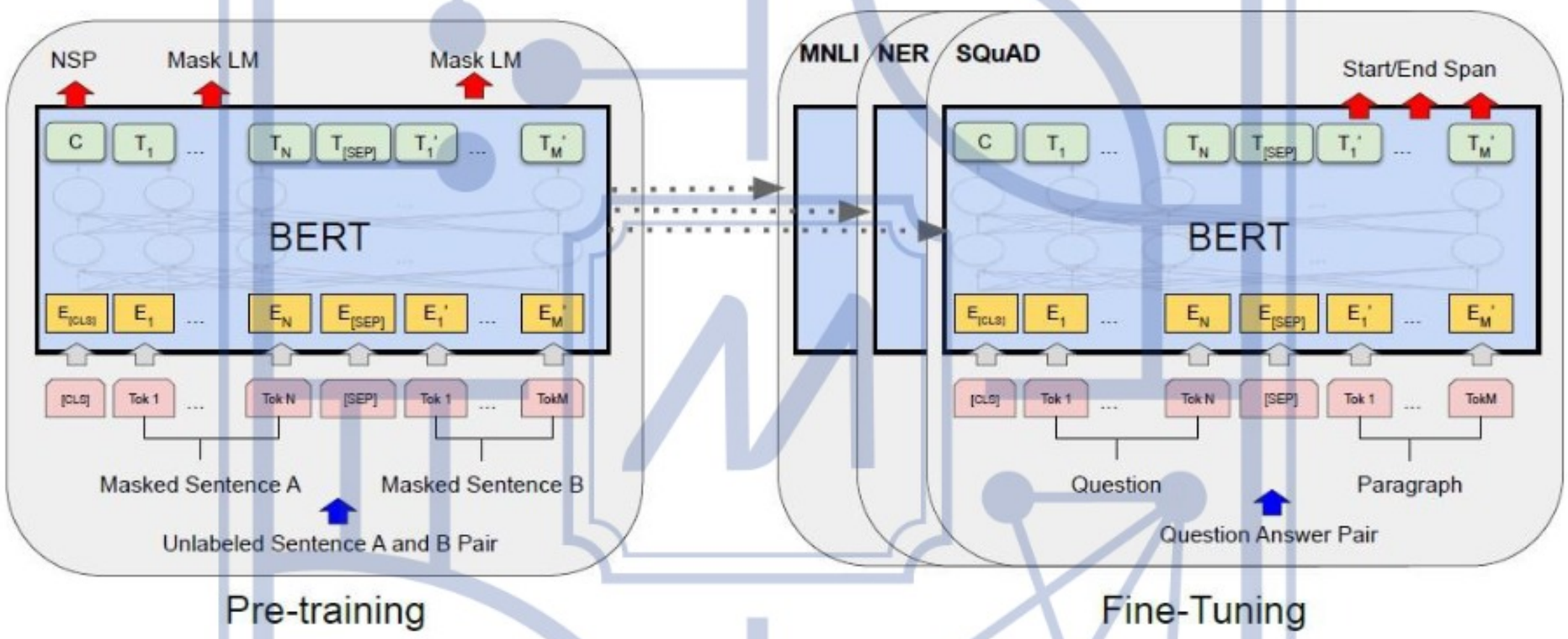
Transformer tidak hanya menerapkan satu *attention*, melainkan menggunakan *Multi-Head Attention*. Mekanisme ini menjalankan fungsi *attention* secara paralel dalam beberapa "kepala" (*heads*) yang berbeda [32]. Penggunaan banyak *head* ini memungkinkan jaringan saraf untuk mempelajari dan menangkap berbagai karakteristik serta hubungan teks (misalnya hubungan sintaksis di satu *head*, dan hubungan semantik di *head* lainnya) dari posisi urutan yang berbeda-beda secara bersamaan [23]. Output dari seluruh *head* kemudian digabungkan (*concatenated*) untuk diproses ke lapisan selanjutnya.



Gambar 2.4 Diagram Arsitektur Transformer [31]

2.6 Model BERT

Berpijak pada keberhasilan arsitektur Transformer, tim peneliti Google AI pada tahun 2018 memperkenalkan model bahasa baru bernama BERT [33]. Secara arsitektural, BERT dibangun secara eksklusif menggunakan tumpukan modul *Encoder* dari Transformer [31], [33]. Terobosan terbesar BERT yang membuatnya mengungguli model bahasa lain pada masanya adalah kemampuannya untuk mempelajari representasi teks secara *bidirectional* (dua arah) secara mendalam. Hal ini berarti BERT mengevaluasi konteks kata dari sisi kiri dan kanannya secara bersamaan di seluruh lapisan, tidak seperti model tradisional yang hanya membaca teks dari kiri-ke-kanan [33].



Gambar 2.5 Ilustrasi Arsitektur BERT [31]

Gambar 2.5 mengilustrasikan arsitektur BERT yang terdiri dari tumpukan lapisan Transformer Encoder yang diproses secara paralel. Pada sisi masukan, BERT menerima urutan token yang telah dilengkapi dengan token khusus [CLS] di posisi pertama dan token [SEP] sebagai pemisah kalimat. Setiap token direpresentasikan melalui penjumlahan tiga jenis *embedding*, yaitu Token *Embedding*, *Segment Embedding*, dan *Position Embedding*. Representasi gabungan ini selanjutnya diproses melalui seluruh lapisan *Transformer Encoder* yang masing-masing terdiri dari mekanisme *Multi-Head Self-Attention* dan jaringan *Feed-Forward*. Keluaran akhir berupa vektor representasi kontekstual untuk setiap token, di mana vektor pada posisi token [CLS] secara khusus digunakan sebagai representasi agregat keseluruhan kalimat untuk keperluan tugas klasifikasi.[33].

Untuk mencapai pemahaman bahasa yang komprehensif, BERT dilatih tanpa pengawasan (*unsupervised*) menggunakan dua strategi prapelatihan (*pre-training*) utama:

1. **Masked Language Model (MLM):** Untuk melatih model secara dua arah tanpa membiarkan sebuah kata menebak dirinya sendiri secara langsung, BERT secara acak menyembunyikan (*masking*) sekitar 15% dari token/kata masukan. Model kemudian dipaksa untuk memprediksi kata yang ditutupi tersebut (token [MASK]) hanya dengan mengandalkan pemahaman dari konteks kata-kata di sekitar kiri dan kanannya [33].
2. **Next Sentence Prediction (NSP):** Banyak tugas NLP (seperti tanya-jawab atau inferensi bahasa) memerlukan pemahaman tentang hubungan antarkalimat. Untuk ini, BERT dilatih dengan menyajikan pasangan kalimat A dan B. Sebanyak 50% data, kalimat B adalah kalimat yang benar-benar mengikuti A (*IsNext*), dan 50% sisanya adalah kalimat acak (*NotNext*). Model bertugas memprediksi hubungan kelogisan kedua kalimat ini [33].

Pemahaman terhadap arsitektur BERT di atas menjadi fondasi penting bagi penelitian ini. Secara khusus, mekanisme ekstraksi vektor *[CLS]* dari lapisan terakhir BERT merupakan komponen inti yang akan diimplementasikan pada tahap pemodelan di Bab 3, di mana representasi tersebut digunakan sebagai fitur masukan bagi MLPClassifier. Penerapan konkret dari arsitektur ini diuraikan secara rinci pada Sub-bab 3.7.1.

2.6.1 Representasi Input pada BERT

Agar dapat menangani berbagai jenis tugas NLP, BERT menggunakan format representasi masukan yang sangat spesifik dan tidak ambigu. Masukan untuk BERT dikonstruksi dari penjumlahan tiga jenis *embeddings*:

- **Token Embeddings:** Representasi kata dasar yang menggunakan algoritma WordPiece dengan perbendaharaan 30.000 kata [33].
- **Segment Embeddings:** Penanda khusus yang dipelajari untuk membedakan apakah sebuah token merupakan bagian dari Kalimat A atau Kalimat B [33].
- **Position Embeddings:** Karena BERT tidak memproses teks secara berurutan layaknya RNN, representasi ini ditambahkan untuk memberi tahu model tentang informasi urutan dan posisi setiap kata di dalam sekuens masukan [33].

Selain itu, BERT secara otomatis menyematkan dua token khusus pada setiap masukan: token [CLS] diletakkan di awal kalimat, yang nilai vektor akhirnya akan digunakan sebagai representasi agregat dari seluruh urutan untuk tugas klasifikasi akhir. Sementara token [SEP] digunakan sebagai pemisah antara kalimat satu dan kalimat lainnya [33].

2.6.2 Model IndoBERT

IndoBERT adalah model bahasa berbasis arsitektur *Transformer* yang secara khusus dikembangkan dan dilatih untuk memahami representasi linguistik bahasa Indonesia [10]. Model ini mengadopsi arsitektur dasar dari BERT (Bidirectional Encoder Representations from Transformers) dan dilatih murni sebagai *masked language model* [15]. Kehadiran IndoBERT sangat krusial karena model bahasa multibahasa (seperti mBERT) seringkali memiliki keterbatasan dalam menangkap nuansa, struktur morfologi, dan kosakata khas yang spesifik pada bahasa Indonesia, terutama dalam ragam bahasa informal [15]. Seperti halnya BERT, IndoBERT dilatih untuk memahami konteks kalimat secara dua arah (*bidirectional*) menggunakan dua objektif prapelatihan utama, yaitu *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP) [13].

Ketangguhan IndoBERT dalam memecahkan masalah *Natural Language Processing* (NLP) sangat dipengaruhi oleh skala dan variasi data teks yang digunakan pada tahap prapelatihan. Terdapat beberapa inisiatif utama dalam pengembangan IndoBERT [13], [34], [35]:

1. **IndoNLU:** Varian ini dilatih menggunakan dataset berskala raksasa yang disebut Indo4B [35]. Korpus Indo4B mengumpulkan sekitar 4 miliar kata dan lebih dari 250 juta kalimat yang diekstraksi dari beragam sumber [34]. Kekuatan utama dari dataset ini adalah variasinya yang mencakup bahasa formal (seperti artikel berita dan Wikipedia) hingga bahasa kolokial atau informal (teks media sosial, blog, dan *subtitle* video) [36].
2. **IndoLEM:** Model ini dilatih menggunakan sekitar 220 juta kata yang bersumber dari korpus web Indonesia, berita, dan Wikipedia [34].
3. **IndoBERTweet:** Untuk menangani teks yang spesifik pada media sosial, Koto et al. (2020) pengembang IndoBERT juga merilis IndoBERTweet, yaitu model pralayang yang dilatih menggunakan 409 juta token kata dari *dataset* Twitter berbahasa Indonesia [17], [34].

Bergantung pada kebutuhan kompleksitas komputasi dan pemahaman teks, IndoBERT memiliki beberapa variasi arsitektur. Rincian varian model IndoBERT dapat dilihat pada Tabel 2.4 berikut

Tabel 2.4 Varian dan Jenis Model IndoBERT [35]

Jenis Model Indobert	Arsitektur & Jumlah Parameter	Deskripsi & Kegunaan
Indobert-Base	12 *layer transformer*, 12 *attention heads*, ~110 - 125 juta parameter.	Ini adalah model fundamental dan varian dasar yang paling umum digunakan. Komputasinya lebih ringan namun tetap tangguh untuk tugas ekstraksi fitur teks pada ukuran data menengah.
Indobert-Large	24 *layer transformer*, 16 *attention heads*, ~335 - 340 juta parameter.	Model berukuran besar yang membutuhkan komputasi lebih tinggi. Varian ini mampu menangani data teks yang lebih masif dan umumnya memberikan akurasi klasifikasi yang lebih tinggi.
Indobert-Lite (Base & Large)	~11.7 juta parameter (*base*) hingga 17.7 juta parameter (*large*).	Merupakan varian berbasis ALBERT (*A Lite BERT*) untuk bahasa Indonesia. Model ini mereduksi parameter secara signifikan agar komputasi dan memori lebih efisien, namun tetap mempertahankan akurasi yang bersaing.
Indobertweet	12 *layer transformer*, 12 *attention heads*, ~125 juta parameter.	Model pralayang yang dilatih secara khusus menggunakan lebih dari 400 juta token kata dari dataset Twitter berbahasa Indonesia. Sangat ideal untuk menganalisis bahasa gaul (*slang*) dan teks informal di media sosial.
Indobert-Base-P1	Sama seperti *Base*	Model *base* Fase 1 (p1) yang dilatih menggunakan teknik *transfer learning* pada dataset besar yang bersumber dari berita, Wikipedia, dan media sosial.
Indobert-Base-P2	Sama seperti *Base*	Model *base* Fase 2 (p2) merupakan kelanjutan dari p1. Model ini dilatih lebih lanjut pada dataset yang lebih kompleks

		untuk memberikan pemahaman semantik dan klasifikasi dokumen yang lebih akurat.

Penerapan IndoBERT sangat relevan dengan variabel dalam penelitian ini, yakni klasifikasi opini (*stance detection*) pada kolom komentar TikTok terkait kasus penegakan hukum Hogi Minaya. Komentar warganet di media sosial dicirikan oleh struktur kalimat yang berantakan, padat akan bahasa gaul (*slang*), singkatan, serta makna-makna implisit seperti sarkasme. Arsitektur IndoBERT yang telah menyerap miliaran kata informal dari dataset Indo4B memiliki keunggulan absolut dalam menangkap fitur semantik serta relasi kompleks antar kata dalam struktur teks media sosial [15], [36]. Pemahaman representasi teks yang kaya dan kontekstual ini menjadikan IndoBERT instrumen klasifikasi berbasis *deep learning* paling ideal untuk menerjemahkan teks tidak terstruktur menjadi identifikasi keberpihakan (Pro, Kontra, atau Netral) secara akurat [15].

2.7 Class Imbalance dan Algoritma SMOTE

Pada konteks klasifikasi teks berbasis komentar media sosial, keberhasilan model pembelajaran mesin tidak hanya ditentukan oleh kualitas arsitektur model dan representasi fitur yang digunakan, tetapi juga oleh distribusi data latih yang digunakan dalam proses pelatihan. Salah satu tantangan yang paling umum dan sering kali diabaikan dalam pembangunan model klasifikasi pada data dunia nyata adalah permasalahan ketidakseimbangan kelas (*class imbalance*) [15]. Permasalahan ini muncul ketika jumlah sampel antar kelas dalam sebuah dataset terdistribusi secara sangat tidak proporsional, sehingga secara langsung memengaruhi kemampuan model dalam mengenali kelas yang memiliki jumlah data lebih sedikit. Pada kondisi tersebut, model cenderung bias terhadap kelas mayoritas dan gagal memprediksi kelas minoritas secara akurat, yang tercermin dari rendahnya nilai F1-Score dan Recall pada kelas minoritas [12]. Untuk mengatasi permasalahan tersebut, berbagai teknik penyeimbangan data telah dikembangkan, salah satu yang paling terbukti efektif adalah algoritma *Synthetic Minority Over-sampling Technique* (SMOTE). Sub-bab berikut akan menguraikan secara mendalam konsep class imbalance beserta mekanisme kerja algoritma SMOTE sebagai solusi yang diterapkan dalam penelitian ini.

2.7.1 Class Imbalance

Ketidakseimbangan kelas (*class imbalance*) adalah suatu kondisi dalam prapemrosesan *machine learning* di mana distribusi jumlah sampel (data) antar kelas dalam sebuah *dataset* sangat tidak proporsional [11]. Dalam kondisi ini, satu kelas memiliki jumlah data yang secara masif lebih besar (kelas mayoritas), sementara kelas lainnya memiliki jumlah data yang sangat sedikit (kelas minoritas) [15].

Ketika dihadapkan pada *dataset* yang tidak seimbang, algoritma klasifikasi standar seringkali mengalami penurunan performa yang drastis. Hal ini terjadi karena algoritma pada umumnya dirancang untuk mengoptimalkan akurasi secara keseluruhan, sehingga model akan terbias (cenderung) untuk selalu menebak kelas mayoritas dan mengabaikan kelas minoritas[11]. Pada banyak kasus, algoritma klasifikasi bahkan menganggap sampel dari kelas minoritas tersebut sebagai *noise* (derau) atau *outlier* semata [36]. Dampaknya, meskipun nilai akurasi klasifikasi secara keseluruhan terlihat sangat tinggi (misalnya 97%), nilai tersebut bersifat semu karena model sebenarnya gagal total dalam mengenali contoh dari kelas minoritas [37].

2.7.2 Synthetic Minority Over-sampling Technique (SMOTE)

Untuk menangani masalah ketidakseimbangan kelas, pendekatan umum yang sering digunakan adalah *undersampling* (mengurangi data kelas mayoritas) dan *oversampling* (menambah data kelas minoritas) [9]. Namun, metode *undersampling* berisiko membuang informasi yang berguna, sedangkan *random oversampling* tradisional yang hanya menduplikasi data minoritas secara acak sangat rentan menyebabkan model mengalami *overfitting* [11].

Sebagai solusi yang lebih mutakhir, Chawla dkk. pada tahun 2002 mengusulkan algoritma SMOTE (*Synthetic Minority Over-sampling Technique*) [12]. Konsep utama dari SMOTE adalah mengatasi *overfitting* dengan cara menciptakan sampel-sampel minoritas sintetik (buatan) yang baru, alih-alih hanya menduplikasi sampel yang sudah ada [15]. SMOTE beroperasi pada "ruang fitur" (*feature space*) data, bukan pada "ruang data" (*data space*) itu sendiri [34]. Dengan mensintesis data baru di sepanjang segmen garis yang menghubungkan antara satu data minoritas dengan tetangganya, algoritma ini secara efektif memperkuat dan memperluas wilayah keputusan (*decision boundary*) dari kelas minoritas sehingga model dapat melakukan generalisasi dengan lebih baik [11].

2.7.3 Mekanisme Kerja

Prosedur pembuatan data sintetik pada algoritma SMOTE dilakukan melalui tahapan interpolasi linear sebagai berikut [11]:

1. **Pemilihan Sampel:** Algoritma memilih sebuah sampel dari kelas minoritas secara acak (misalnya ss_{ii}).
2. **Pencarian Tetangga Terdekat:** Algoritma kemudian mencari sejumlah KK tetangga terdekat (*K-Nearest Neighbors*), yang secara *default* biasanya menggunakan $KK = 5$, dari sampel ss_{ii} tersebut yang juga merupakan anggota dari kelas minoritas yang sama.
3. **Interpolasi:** Algoritma memilih secara acak salah satu dari tetangga KK tersebut (misalnya ss_{ZZii}). Selanjutnya, dihitung selisih vektor fitur antara sampel ss_{ii} dengan ss_{ZZii}.
4. **Sintesis Data Baru:** Nilai selisih tersebut kemudian dikalikan dengan sebuah angka acak ($\delta\delta$) yang berada dalam rentang 0 hingga 1. Hasil perkalian ini kemudian ditambahkan kembali ke vektor fitur dari sampel asli ss_{ii}.

Secara matematis, rumusan pembentukan data baru (sintetik) oleh algoritma SMOTE dapat dituliskan sebagai:

$$x_{new} = x_i + \delta \times (x_k - x_i) \quad \text{.....(2)}$$

Di mana

- x_{new} adalah sampel sintetik baru yang dihasilkan,
- $x_i$adalah sampel kelas minoritas asli,
- x_{zi}adalah salah satu tetangga terdekat dari x_i dan
- δ adalah angka acak ($0 \leq \delta \leq 1$) [11].

2.7.4 Penerapan SMOTE pada Pemrosesan Teks (NLP)

Pada awalnya, algoritma SMOTE dirancang untuk beroperasi pada data terstruktur yang bersifat numerik berdimensi rendah (seperti data medis atau finansial) [12]. Penerapan SMOTE pada data teks (seperti komentar media sosial) memiliki kompleksitas dan tantangan yang jauh lebih tinggi. Hal ini dikarenakan data teks memiliki dimensi spasial yang sangat besar (terdiri dari ribuan hingga jutaan kosakata) dan mengandung pola linguistik yang tidak linier, seperti sarkasme, dialek, atau makna ambigu yang sulit direplikasi secara sintetik secara langsung dari teks mentah [12].

Oleh karena itu, dalam domain *Natural Language Processing* (NLP), SMOTE tidak diterapkan secara langsung pada karakter huruf atau kata. Teks harus terlebih dahulu ditransformasikan menjadi representasi vektor numerik (*text embeddings*) [34]. Vektor

numerik yang merepresentasikan konteks kalimat (seperti vektor keluaran dari IndoBERT atau TF-IDF) inilah yang kemudian akan menjadi "ruang fitur" bagi algoritma SMOTE untuk melakukan interpolasi dan mensintesis fitur teks buatan [34].

Dalam analisis wacana polemik hukum (studi kasus Hogi Minaya) pada kolom komentar TikTok, opini publik sangat terpolarisasi. Komentar yang membela Hogi Minaya pada Kelas 0 merupakan kelas minoritas, sedangkan Kelas 1 yang mencakup komentar *Non-Support Hogi* merupakan kelas mayoritas. Oleh karena itu, SMOTE diterapkan pada Kelas 0 untuk menambah representasi kelas minoritas dalam data latih [15]. Jika ketimpangan ini dibiarkan, model klasifikasi dapat lebih cenderung memprediksi Kelas 1 sebagai kelas mayoritas dan kurang mampu mengenali komentar Kelas 0 sebagai kelas minoritas. Penggunaan algoritma SMOTE pada tahap prapemrosesan ruang fitur diyakini mampu menyeimbangkan jumlah fitur antar kelas secara proporsional. Dengan memperbanyak representasi sintetik Kelas 0 sebagai kelas minoritas, SMOTE membantu model mempelajari pola komentar yang mendukung Hogi secara lebih memadai. Namun, SMOTE tidak secara langsung membedakan komentar *Oppose*, *Neutral*, dan *Ambiguous* yang masih tergabung dalam Kelas 1 [12], [15].

2.8 Text Preprocessing (Prapemrosesan Teks)

Data teks yang bersumber dari platform media sosial seperti TikTok pada umumnya merupakan data mentah yang tidak terstruktur (*unstructured data*) dan mengandung banyak derau (*noise*) [9], [15], [16]. Karakteristik komentar warganet sering kali menggunakan bahasa sehari-hari, singkatan, serta penyisipan simbol yang tidak memiliki makna komputasional [17]. Oleh karena itu, prapemrosesan teks (*text preprocessing*) menjadi tahapan yang sangat krusial dalam *Natural Language Processing* (NLP) [9], [16]. Tujuan utama dari tahapan ini adalah untuk membersihkan, menormalkan, dan mentransformasikan teks mentah tersebut menjadi format yang lebih terstruktur, sehingga kualitas data meningkat dan siap untuk diolah secara optimal oleh algoritma pembelajaran mesin seperti IndoBERT [5], [16].

Tabel 2.5 Contoh Hasil Data Cleaning [7]

Step	Hasil
Hapus Duplikat	Menghilangkan data tweet yang sama untuk menghindari redundansi.

Tweet Mentah	Cuma dari Jakarta Timur ke depok doank udah seminggu gak jalan-jalan nih paket @anteraja_id dianter keong apa gimana?
Case Folding	cuma dari jakarta timur ke depok doank udah seminggu gak jalan-jalan nih paket @anteraja_id dianter keong apa gimana?
Hapus Karakter	cuma dari jakarta timur ke depok doank udah seminggu gak jalan jalan nih paket dianter keong apa gimana
Normalisasi	cuma dari jakarta timur ke depok saja sudah seminggu tidak jalan jalan nih paket diantar keong apa bagaimana
Cek KBBI	cuma dari jakarta timur ke depok saja sudah seminggu tidak jalan jalan nih paket diantar keong apa bagaimana
Tokenizing	['cuma', 'dari', 'jakarta', 'timur', 'ke', 'depok', 'saja', 'sudah', 'seminggu', 'tidak', 'jalan', 'jalan', 'nih', 'paket', 'diantar', 'keong', 'apa', 'bagaimana']
Filtering	['jakarta', 'timur', 'depok', 'saja', 'sudah', 'seminggu', 'tidak', 'jalan', 'jalan', 'paket', 'diantar', 'keong']
Stemming	['jakarta', 'timur', 'depok', 'saja', 'sudah', 'minggu', 'tidak', 'jalan', 'jalan', 'paket', 'antar', 'keong']
Tweet Bersih	jakarta timur depok saja sudah minggu tidak jalan jalan paket antar keong

Secara umum, proses prapemrosesan teks pada data ulasan atau komentar berbahasa Indonesia melalui beberapa tahapan sistematis sebagai berikut:

1. ***Cleaning* (Pembersihan Data),** merupakan tahap awal yang bertujuan untuk membersihkan teks dari berbagai karakter dan elemen yang tidak relevan untuk analisis sentimen maupun *stance* [9], [15]. Pada tahap ini, berbagai *noise* dihilangkan dari dalam dokumen teks, yang mencakup penghapusan *mention* atau *username* (misalnya @user), tautan URL, *hashtag* (#), emoji, serta kode HTM [5], [15]. Selain itu, karakter-karakter khusus (seperti $, %, &), angka yang tidak relevan, spasi berlebih, serta seluruh tanda baca juga turut dihapus [9], [38].
2. ***Case Folding,*** adalah tahapan penyeragaman bentuk huruf, di mana seluruh karakter huruf kapital di dalam teks dikonversi menjadi huruf kecil (*lowercase*) [5], [9], [38]. Proses ini memastikan bahwa huruf dari "A" sampai "Z" memiliki bentuk yang konsisten [38]. Tujuan utamanya adalah untuk menghindari kesalahan analisis yang diakibatkan oleh perbedaan kapitalisasi huruf, sehingga model algoritma tidak menganggap kata yang sama namun ditulis berbeda (misalnya "Hukum", "HUKUM", dan "hukum") sebagai entitas kata yang berbeda [9], [38].

3. ***Normalization*** **(Normalisasi),** Dalam konteks pemrosesan komentar media sosial, pengguna sangat sering menulis menggunakan bahasa tidak baku, bahasa gaul (*slang*), atau menyingkat kata [10], [12]. Tahap normalisasi bertujuan untuk mentransformasikan kata-kata tidak baku atau singkatan tersebut menjadi bentuk standar atau bahasa Indonesia yang baku sesuai dengan ejaan Kamus Besar Bahasa Indonesia (KBBI) [7], [16].
4. ***Tokenizing*** **(Tokenisasi).** Tokenisasi adalah proses memotong atau memecah susunan kalimat teks yang telah bersih menjadi unit-unit bagian yang lebih kecil, yang disebut sebagai token [9], [17], [38]. Pemecahan kalimat ini pada umumnya dilakukan berdasarkan pemisah berupa spasi di antara kata [9], [16]. Dengan melakukan tokenisasi, teks yang panjang menjadi data terstruktur berupa kumpulan kata individual, yang memudahkan proses pembobotan atau transformasi data ke dalam bentuk vektor numerik oleh model NLP. Perlu dicatat bahwa tokenisasi pada tahap ini bersifat umum dan digunakan untuk keperluan *stemming* serta *stopword removal*. Khusus untuk model IndoBERT, proses tokenisasi dilakukan kembali menggunakan *tokenizer* bawaan model (*WordPiece*) yang secara otomatis menyematkan token [CLS] dan [SEP], sebagaimana diuraikan pada Sub bab 3.7.1.
5. ***Stopword Removal*** **(Penghapusan Kata Hubung).** *Stopword removal* (atau *filtering*) adalah tahapan untuk mengeliminasi kata-kata umum yang frekuensi kemunculannya sangat tinggi namun tidak membawa informasi atau makna yang signifikan dalam menentukan polaritas sentimen atau *stance* [9], [16].
6. ***Stemming*** **(Pengakaran Kata).** *Stemming* merupakan proses morfologis untuk mengembalikan suatu kata yang memiliki imbuhan (seperti awalan, sisipan, atau akhiran) kembali menjadi bentuk kata dasarnya (*root word*) sesuai dengan struktur dan tata bahasa. Proses ini secara signifikan mengurangi variasi kosakata yang ditimbulkan oleh konjugasi, sehingga kompleksitas ruang fitur menurun. Dalam prapemrosesan teks berbahasa Indonesia, tahap *stemming* sering kali mengandalkan pustaka atau *library* bahasa pemrograman Python bernama Sastrawi [9], [10].

2.9 Metrik Evaluasi Kinerja Model

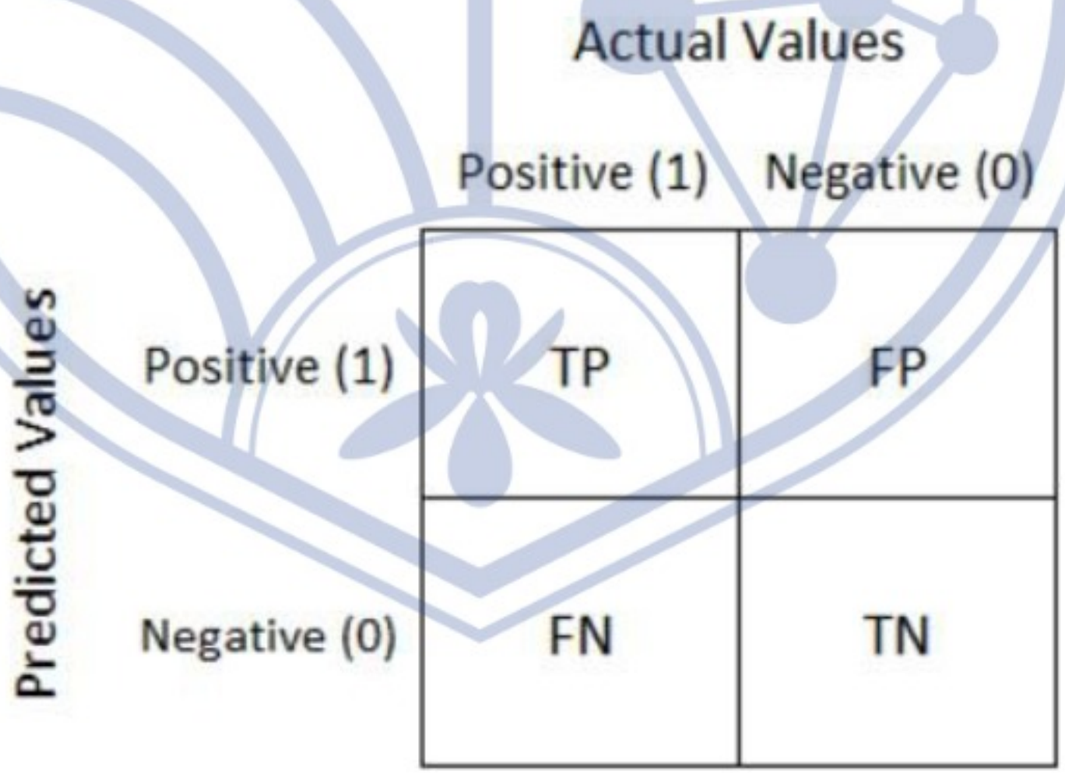
Evaluasi model merupakan tahapan kritis untuk mengukur seberapa baik model pembelajaran mesin dalam mengklasifikasikan data teks yang telah diproses. Dalam kasus klasifikasi teks pada dunia nyata, terutama yang memiliki distribusi kelas yang tidak

seimbang (*class imbalance*) seperti analisis *stance* opini publik, mengandalkan metrik akurasi saja bisa sangat menyesatkan [15]. Oleh karena itu, performa algoritma secara komprehensif dievaluasi menggunakan matriks kebingungan (*confusion matrix*) yang kemudian diturunkan menjadi empat metrik evaluasi utama: *Accuracy, Precision, Recall*, dan *F1-Score*.

2.9.1 Confusion Matrix

Confusion Matrix adalah alat evaluasi fundamental yang menyajikan tabel distribusi silang antara prediksi model dengan kelas aktual (sebenarnya). Matriks ini membantu dalam menganalisis kinerja model secara mendalam dengan memecah hasil klasifikasi ke dalam empat komponen dasar [15], [20]:

- **True Positive (TP):** Jumlah data yang secara aktual memiliki kondisi positif dan diprediksi dengan akurat sebagai kelas positif oleh model.
- **True Negative (TN):** Jumlah data yang secara aktual memiliki kondisi negatif dan secara akurat diprediksi sebagai kelas negatif oleh model.
- **False Positive (FP):** Jumlah data yang secara aktual bernilai negatif, namun secara keliru diprediksi sebagai positif oleh model.
- **False Negative (FN):** Jumlah data yang secara aktual bernilai positif, namun secara keliru diprediksi sebagai negatif oleh model.



Gambar 2.6 Tabel Confusion Matrix[12]

2.9.2 Akurasi (Accuracy)

Akurasi adalah ukuran rasio yang menunjukkan seberapa banyak prediksi kelas yang diklasifikasikan dengan benar (baik positif maupun negatif) dibandingkan dengan total keseluruhan data yang diprediksi oleh model. Persamaan matematis untuk akurasi adalah:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{..... (3)}$$

2.9.3 Presisi (Precision)

Presisi mengukur ketepatan prediksi kelas positif, yakni rasio prediksi positif yang benar (TP) dibandingkan dengan seluruh prediksi positif yang dibuat oleh model (TP + FP). Metrik ini memberikan gambaran keandalan model dalam meminimalkan prediksi *false positive*. Rumus dari presisi adalah:

$$Precision = \frac{TP}{TP+FP} \quad \text{..... (4)}$$

2.9.4 Recall (Sensitivitas)

Recall mengukur kemampuan model dalam menangkap dan menemukan kembali seluruh data yang sebenarnya positif (relevan). *Recall* dihitung dari rasio jumlah TP dibagi dengan seluruh kasus yang secara aktual benar-benar positif (TP + FN). Metrik ini penting untuk memastikan model menekan angka *false negative*. Rumus *recall* adalah:

$$Recall = \frac{TP}{TP+FN} \quad \text{..... (5)}$$

2.9.5 F1-Score

F1-Score merupakan ukuran rata-rata harmonik (*harmonic mean*) dari *Precision* dan *Recall*. Metrik ini menggabungkan dan memperhitungkan kedua nilai tersebut untuk memberikan penilaian evaluasi yang lebih seimbang. Pada *dataset* yang mengalami ketidakseimbangan kelas secara signifikan, *F1-score* menjadi metrik evaluasi yang jauh lebih representatif dan penting dibandingkan dengan akurasi semata. Rumus dari *F1-Score* adalah:

$$F1Score = \frac{2 \times Precision \times Recall}{Precision+Recall} \quad \text{..... (6)}$$