BAB II
KAJIAN LITERATUR

2.1 Kecerdasan Buatan (*Artificial Intelligence*)

Artificial Intelligence adalah cabang ilmu komputer yang berfokus pada perancangan sistem yang mampu menjalankan tugas-tugas yang biasanya memerlukan kecerdasan manusia, seperti penalaran, pembelajaran dan pengalaman, pemahaman bahasa dan pengenalan pola [19]. Berbeda dari pemrograman konvensional yang mengandalkan aturan eksplisit yang dikodekan secara manual, sistem AI dirancang agar dapat mempelajari pola secara otomatis dari data. Perkembangan AI dimulai dari pendekatan berbasis aturan (*rule-based*) pada dekade 1950-an. Fokus ini kemudian bergeser menuju paradigma *machine learning* (ML) pada 1980-an, dan saat ini didominasi oleh *deep learning* (DL). DL merupakan subbidang ML yang menggunakan jaringan saraf tiruan berlapis banyak untuk mengekstrak representasi hierarkis dari data mentah [20].

AI mencakup sejumlah subbidang terapan, di antaranya *computer vision, speech recognition,* sistem rekomendasi, dan *Natural Language Processing* [19]. Kemajuan DL, khususnya melalui arsitektur *Transformer* dan model *pre-trained* seperti BERT, telah memberikan peningkatan performa pada berbagai tugas AI, termasuk klasifikasi teks dan analisis sentimen [21]. Penelitian ini berfokus pada NLP sebagai subbidang AI yang paling relevan dan akan dibahas lebih lanjut pada subbab 2.1.1.

2.1.1 *Natural Language Processing*

Natural Language Processing (NLP) adalah bidang interdisipliner yang menggabungkan ilmu komputer, kecerdasan buatan, dan linguistik untuk memungkinkan interaksi antara manusia dan komputer melalui bahasa alami (*natural language*). Bahasa alami adalah bahasa yang digunakan manusia dalam komunikasi sehari-hari, yang memiliki ambiguitas, variasi gaya, serta konteks pragmatis yang kompleks. NLP mengembangkan berbagai metodologi untuk menangani kompleksitas tersebut melalui tugas-tugas seperti *text summarization, part-of-speech tagging, word sense disambiguation, named entity recognition, sentiment analysis, natural language understanding,* dan *speech recognition* [22], [23].

Perkembangan teknik NLP berevolusi dari pendekatan berbasis aturan dan model statistik klasik seperti SVM yang bergantung pada fitur eksplisit dan kurang mampu

menangani variasi linguistik pada data tidak terstruktur menuju teknik representasi berbasis vektor (*word embedding*) yang mampu menangkap kemiripan semantik antar kata. Model *embeddings* klasik seperti Word2Vec (CBOW & Skip-Gram) dan GloVe mentransformasikan representasi kata dari bentuk biner (*one-hot*) menjadi vektor n-dimensi yang merefleksikan kesamaan semantik antar kata [24], [25]. Word2Vec memanfaatkan teknik *negative sampling* dan *hierarchical softmax* sedangkan GloVe menggabungkan *global matrix factorization* dan *local context window* sehingga hasil vektor merepresentasikan rasio kemunculan relatif antar kata. Namun *word embedding* klasik bersifat statis sehingga satu kata hanya memiliki satu representasi dan tidak mempertimbangkan konteks (polisemi).

Model *pre-training* berbasis *Transformer* seperti BERT mengatasi keterbatasan tersebut melalui *contextual embeddings* yang merepresentasikan kata relatif sesuai konteks kalimat sehingga mampu menangani polisemi dan konteks semantik yang lebih kaya. Pendekatan *pre-training* diikuti *fine-tuning* pada *task downstream* telah meningkatkan performa pada berbagai tugas NLP seperti klasifikasi teks, ekstraksi entitas, dan analisis sentimen [21]. Untuk kajian Bahasa Indonesia, model *pre-training monolingual* seperti IndoBERT menyediakan landasan yang lebih relevan dibanding model multibahasa, karena dilatih pada korpus besar berbahasa Indonesia dan mampu menangani fenomena linguistik lokal seperti *slang*, morfologi, dan *code-mixing* yang umum pada media sosial [26], [27].

2.1.2 Analisis Sentimen

Analisis sentimen adalah salah satu cabang dari *Natural Language Processing* (NLP) yang secara sistematis mengidentifikasi dan mengklasifikasikan opini, emosi, serta sikap subjektif yang terkandung dalam teks [28]. Klasifikasi ini pada umumnya bertujuan untuk menentukan polaritas opini yang disampaikan, yang secara tradisional dikategorikan sebagai positif, negatif, atau netral. Meskipun demikian, studi yang lebih mendalam juga dapat mencakup deteksi emosi spesifik seperti marah, sedih, atau gembira [29]. Tujuan utama dari analisis sentimen adalah untuk memahami persepsi masyarakat terhadap suatu entitas atau peristiwa melalui interpretasi bahasa yang muncul dalam komunikasi digital. Selain itu, pendekatan ini juga dimanfaatkan dalam mendukung pengambilan keputusan kebijakan dan strategi organisasi melalui penyediaan wawasan kuantitatif atas sentimen publik, terutama ketika data diperoleh dari media sosial yang menghasilkan teks autentik dalam volume besar [30].

Secara metodologis, analisis sentimen dapat dibagi ke dalam tiga pendekatan utama: (1) pendekatan berbasis lexicon (*lexicon-based*) yang mengandalkan kamus sentimen pra-definisi, (2) *machine learning* tradisional seperti Naïve Bayes atau SVM yang bergantung pada *feature engineering*, dan (3) pendekatan *deep learning* seperti CNN atau LSTM. Perkembangan metode ini menunjukkan pergeseran dan ketergantungan pada aturan dan fitur manual ke arsitektur *neural network* yang mampu mempelajari representasi data secara otomatis dari teks mentah. Model seperti CNN, LSTM dan model berbasis *Transformer* seperti BERT telah menunjukkan keunggulan dalam menangani data tidak terstruktur serta menangkap nuansa dan konteks semantik yang kompleks dalam bahasa alami, yang sering kali gagal ditangkap oleh metode tradisional [31], [32].

Analisis sentimen pada teks politik Indonesia menghadapi sejumlah tantangan linguistik dan konteks sosial yang khas. Bahasa politik di media sosial sangat dipengaruhi oleh penggunaan *slang*, campuran bahasa, singkatan, serta gaya ironi dan sarkasme yang sulit ditangkap oleh model berbasis aturan maupun fitur sederhana [33]. Selain itu, diskursus politik sering memuat ambiguitas makna, polarisasi opini yang ekstrem, serta penyebaran misinformasi, sehingga semakin menyulitkan identifikasi polaritas secara akurat. Kompleksitas ini diperparah oleh karakteristik bahasa Indonesia yang kaya bentuk informal dan tidak memiliki struktur morfologi yang selalu konsisten, sehingga representasi semantik menjadi lebih sulit dipelajari, sehingga analisis sentimen politik di Indonesia menuntut model yang mampu memahami konteks mendalam, menangkap variasi linguistik, dan tetap *robust* terhadap *noise* teks autentik dari media sosial.

Di ranah politik dan kebijakan publik, analisis sentimen telah menjadi instrumen penting untuk memantau persepsi publik secara *real-time* [30]. Pendekatan ini semakin relevan ketika digunakan untuk menganalisis fenomena "17+8 Tuntutan Rakyat", di mana platform X berfungsi sebagai ruang diskursus utama bagi warga untuk mengekspresikan opini mereka. Namun, data yang dihasilkan dari platform ini berupa teks autentik yang tidak terstruktur, informal dan *noisy* [28]. sehingga menuntut penggunaan pendekatan NLP yang mampu menangkap konteks linguistik dan dinamika sosial-politik yang kompleks sehingga analisis sentimen dengan model yang memiliki kapasitas representasi semantik tinggi menjadi krusial untuk mengekstraksi wawasan sosial secara komprehensif, memahami arah opini publik, serta memetakan respons masyarakat terhadap isu kebijakan dan tuntutan yang diajukan.

2.2 Demonstrasi dan 17 + 8 Tuntutan Rakyat

Demonstrasi adalah bentuk aksi kolektif yang ditujukan untuk menyampaikan tuntutan politik atau sosial dengan tujuan mempengaruhi kebijakan publik. Dalam kajian gerakan sosial kontemporer, demonstrasi merupakan salah satu *repertoire of contention* yang muncul ketika partisipasi formal dianggap kurang responsif. Analisis literatur menekankan pentingnya struktur peluang politik, pembentukan identitas kolektif, dan mekanisme jaringan sosial dalam mengorganisasi serta mempertahankan aksi kolektif [34], [35].

Transformasi ke ruang digital mengubah cara mobilisasi kolektif. Fenomena mobilisasi yang difasilitasi platform media sosial cenderung membentuk hubungan yang lebih longgar (*loosely connected*) antar individu melalui tagar, unggahan, dan praktik *reposting*, sehingga dapat menyebar lebih cepat tanpa adanya struktur organisasi hirarkis tradisional. Literatur mengenai kontensi digital menunjukkan bahwa mekanisme ini menurunkan biaya koordinasi dan memperbesar jangkauan aksi, tetapi menghadirkan kerentanan, seperti fragmentasi agenda, keviralan yang belum tervalidasi, serta peluang manipulasi oleh aktor non-organik. Temuan-temuan empiris dan kajian komparatif menekankan perlunya kehati-hatian dalam menafsirkan keviralan sebagai representasi opini publik secara menyeluruh [34], [36].

Kasus "17+8 Tuntutan Rakyat" yang berlangsung pada 25 Agustus - 9 September 2025 lalu merupakan mobilisasi yang sangat dipengaruhi oleh dinamika platform digital. Tuntutan tersebut, yang terdiri dari 17 tuntutan jangka pendek dan 8 tuntutan jangka panjang, mencakup isu penegakan hukum, reformasi institusional, pengembalian fungsi sipil aparat keamanan, transparansi anggaran publik, serta kebijakan ekonomi dan ketenagakerjaan yang prorakyat. Tuntutan ini menyebar cepat melalui unggahan dari *influencer*, akun aktivis, tagar, serta *repost* massal yang mendorong aksi pada sejumlah titik strategis di Jakarta dan kota-kota lain. Media daring mendokumentasikan jejak penyebaran dan isi tuntutan tersebut [37]. Meski demikian, interpretasi mengenai seberapa representatif tuntutan ini terhadap opini publik luas perlu memperhitungkan bias demografis pengguna platform, dominasi akun berpengaruh, aktivitas akun non-organik, dan perubahan akses API atau algoritma yang dapat memengaruhi apa yang tampak sebagai "suara publik" di ruang digital.

2.3 Media Sosial dan Teks Autentik

Media sosial adalah platform daring yang memfasilitasi pengguna untuk membuat, membagikan, dan saling bertukar informasi secara interaktif. Selain mempercepat difusi

informasi, platform media sosial memungkinkan mobilisasi opini dan tindakan kolektif secara cepat. Karena volume dan sifatnya yang berbasis teks, platform ini menjadi sumber data yang relevan dan masif untuk analisis berbasis teks [38].

Di sisi lain, media sosial berperan memperluas dan mempercepat diseminasi tuntutan serta argumen ke dalam wacana publik. Namun demikian, platform juga menghadirkan kerentanan, antara lain intervensi yang dilakukan aktor non-organik serta potensi penyebaran informasi yang belum terverifikasi, sehingga dapat menyebabkan ketidakseimbangan visibilitas dan representasi [34].

Unggahan di platform media sosial sebagai konten buatan pengguna (*user-generated content*) sering disebut sebagai teks autentik karena berasal dari praktik komunikasi sosial nyata pengguna. Ciri khas teks autentik media sosial meliputi bahasa tidak formal, *code-mixing*, penggunaan simbol non-verbal (*emoji*), struktur kalimat yang singkat atau terfragmentasi, serta kecenderungan pragmatis seperti ironi atau sarkasme. Karakteristik tersebut membuat teks media sosial kaya konteks sosial namun berbeda secara esensial dari teks formal atau korpus tertulis tradisional [33].

2.4 Pengumpulan dan Preprocessing Data Teks

Dalam penelitian NLP yang berbasis *deep learning,* data merupakan komponen fundamental yang menentukan keberhasilan model. Kinerja dan kemampuan generalisasi model klasifikasi secara langsung bergantung pada kualitas dan akurasi anotasi data yang digunakan untuk pelatihannya sehingga diperlukan serangkaian tahapan yang sistematis diperlukan untuk mengubah data teks mentah (*raw-text*), yang pada platform media sosial sering kali tidak terstruktur dan *noisy,* menjadi format yang terstruktur dan siap digunakan oleh model [39].

2.4.1 Pengumpulan Data

Pengumpulan data dari platform media sosial X (sebelumnya Twitter) merupakan langkah fundamental dalam analisis sentimen untuk menangkap opini public terkait isu-isu sosial politik. Data yang diperoleh dari platform ini merupakan *User-Generated Content* (UGC) yang merefleksikan pendapat beragam secara *real-time*. Teks yang dihasilkan dari lingkungan media sosial memiliki tantangan unik, seperti volume dan kecepatan data yang tinggi, konteks yang terfragmentasi, serta mengandung *noise* data yang signifikan. Teks ini seringkali bersifat tidak terstruktur, acak dan informal. Karakteristik semacam ini justru

dipandang sebagai representasi otentik dari opini dan bahasa alami pengguna, sehingga sejumlah penelitian analisis sentimen kontemporer secara sengaja mempertahankan karakter aslinya, bukan menormalkannya secara agresif [40] .

Secara teoritis, literatur ilmiah mendiskusikan dua metode utama untuk akuisisi data dari X. Metode pertama adalah melalui *Application Programming Interface* (API) resmi yang disediakan oleh platform. Meskipun merupakan pendekatan yang sah secara prosedural, metode ini memiliki keterbatasan ketat (*rate limits*) dan restriksi akses data historis yang signifikan bagi peneliti. Untuk mengatasi tantangan ini, metode kedua yaitu *web scraping* (perambanan web) yang sering diimplementasikan dalam kebutuhan akademis [41]. Seiring dengan perubahan kebijakan API dan platform, banyak penelitian akademis di Indonesia yang telah beralih menggunakan *tools crawler* yang lebih baru seperti *Tweet-Harvest.* Keunggulan *tools* ini yang terdokumentasi dalam sebuah penelitian adalah kemampuannya dalam mengumpulkan data dalam jumlah yang besar dengan hanya mengandalkan token autentikasi pengguna [42], tanpa perlu melalui proses pendaftaran developer API yang kompleks.

Meskipun data yang dikumpulkan bersifat publik, proses pengumpulan data harus tetap mematuhi prinsip etika penelitian [43]. Pada penelitian akademis yang melibatkan data publik, praktik standar etika mencakup pembatasan penggunaan data hanya untuk tujuan penelitian dan non-komersial. Selain itu, penelitian wajib melindungi privasi pengguna anonimisasi data, yaitu sebuah proses teknis untuk menghapus atau mengaburkan informasi identitas pribadi (seperti *username* dan *user ID*) sebelum data tersebut diproses, dianalisis, dan disajikan dalam publikasi.

2.4.2 *Preprocessing Data*

Preprocessing data merupakan tahapan esensial dalam alur kerja NLP yang berfungsi untuk mengonversi data mentah (*raw text*) menjadi format terstruktur yang dapat diinterpretasi oleh algoritma *machine learning* [39]. Teks yang berasal dari media sosial secara inheren bersifat *noise,* tidak terstruktur dan mengandung artefak (misalnya tautan URL, *mention* pengguna, dan emoji). Tujuan fundamental dari *preprocessing data* adalah untuk membersihkan *noise* tersebut dan menstandarisasi, sehingga model dapat fokus pada fitur-fitur linguistik yang bermakna. Proses ini secara langsung berkontribusi pada peningkatan presisi, reliabilitas, dan kinerja keseluruhan dari model analisis sentimen yang dikembangkan [44]. Tahapan umum dalam *preprocessing data* mencakup *data cleaning*,

case folding, *normalization*, *tokenization*, *stopword removal*, dan *stemming* atau *lemmatization*.

Dalam paradigma *machine learning* tradisional, khususnya yang bergantung pada rekayasa fitur manual seperti *Term Frequency-Inverse Document Frequency* (TF-IDF), pendekatan yang dominan adalah *preprocessing* agresif. Pendekatan ini melibatkan serangkaian intervensi teks yang substansial, seperti *case folding, filtering* (termasuk penghapusan *stopwords*), dan *stemming*. Rasional utama dibalik metode ini adalah untuk mereduksi dimensionalitas ruang fitur secara signifikan. Berdasarkan rasional tersebut, *stemming* digunakan untuk menormalkan varian-varian tersebut ke satu akar kata, sehingga mengurangi kompleksitas komputasi dan menyederhanakan pola yang harus dipelajari oleh model.

Namun, strategi *preprocessing* agresif justru dapat merugikan model *deep learning* kontekstual seperti IndoBERT. Penghapusan *stopwords* berisiko menghilangkan token negasi seperti "tidak" atau "bukan" yang menentukan polaritas kalimat. Demikian pula, penghapusan tanda baca atau emoji dapat mengaburkan penanda emosional yang penting. IndoBERT telah dilatih pada korpus besar (Indo4B) [42] untuk memahami bahasa secara keseluruhan, termasuk *stopwords* dan konteks implisit, sehingga pendekatan yang digunakan adalah *preprocessing* ringan yang hanya membersihkan artefak non-linguistik seperti URL, *mention* pengguna, dan karakter HTML, tanpa mengubah struktur teks. Studi empiris pada teks Bahasa Indonesia mengkonfirmasi bahwa *stemming* dan *stopword removal* tidak meningkatkan akurasi secara konsisten, dalam beberapa kasus justru menurunkan kinerja model [18].

2.4.3 Pelabelan Data

Pelabelan data adalah proses pemberian tag kategoris pada data teks mentah sebagai *ground truth* untuk model *supervised learning*. Dalam tugas analisis sentimen, skema pelabelan yang umum digunakan terdiri dari tiga kelas, yaitu Positif, Negatif dan Netral [45], meskipun jumlah dan definisi kelas dapat bervariasi sesuai tujuan penelitian. Secara metodologis, pelabelan dapat dilakukan secara otomatis menggunakan aturan atau *machine learning*, maupun secara manual oleh anotator manusia [45]. Pada teks autentik media sosial yang umumnya mengandung sarkasme, ironi, dan konteks implisit, pendekatan *lexicon-based* otomatis cenderung kurang akurat karena tidak mampu menangkap nuansa tersebut, sehingga anotasi manual oleh manusia lebih sering dipilih karena kemampuannya

menginterpretasi pola bahasa yang subtil dan konteks yang tidak dapat ditangkap sistem berbasis aturan [45]. Kualitas anotasi inilah yang pada akhirnya menentukan kinerja model klasifikasi yang dihasilkan.

Untuk menjamin konsistensi dan reliabilitas dalam proses anotasi manual, terutama dalam skenario yang melibatkan lebih dari satu anotator, pengembangan pedoman anotasi (*annotation guidelines*) yang rinci merupakan prasyarat metodologis yang esensial [46], [47]. Pedoman ini berfungsi sebagai instrumen ilmiah untuk menstandardisasi interpretasi dan mengurangi subjektivitas anotator. Pedoman yang komprehensif harus secara eksplisit mendefinisikan setiap label dan menyediakan aturan yang kaku (*rigid rules*) untuk menangani kasus-kasus ambigu yang umum ditemukan dalam teks autentik, seperti identifikasi sarkasme, teks yang murni informasional (netral), atau ekspresi sentimen campuran (*conflict sentiment*).

2.4.4 Pembagian Data

Pembagian data (*data splitting*) adalah langkah standar dalam *supervised machine learning* untuk mengevaluasi kemampuan model pada data yang belum pernah dilihat sebelumnya [48]. Tanpa pemisahan ini, model berisiko mengalami *overfitting*, yaitu kondisi di mana model hanya menghafal pola data latih namun gagal menggeneralisasi pada data baru.

Dataset umumnya dibagi menjadi tiga bagian [46]: Data Latih (*Training Set*) yang merupakan porsi terbesar (70–80%) dan satu-satunya data yang digunakan untuk menyesuaikan bobot model [49]; Data Validasi (*Validation Set*) (10–15%) yang digunakan untuk memantau performa selama pelatihan dan mendukung proses *hyperparameter tuning* serta *early stopping*; dan Data Uji (*Test Set*) (10–15%) yang hanya digunakan sekali di akhir untuk melaporkan hasil evaluasi yang tidak bias [47].

Pemilihan teknik pembagian data bergantung pada karakteristik *dataset*. Pendekatan pembagian statis yang umum, dikenal sebagai validasi *hold-out*, memiliki keterbatasan, terutama jika *dataset* berukuran kecil atau memiliki distribusi kelas yang tidak seimbang (*imbalanced*). *Dataset* analisis sentimen pada isu sosial dan politik seringkali bersifat *imbalanced*, di mana satu kelas (misalnya, netral) mungkin jauh lebih sedikit daripada kelas lainnya (misalnya, negatif). Pembagian acak pada data tersebut berisiko menghasilkan set validasi atau uji yang tidak representatif. Untuk mengatasi ini, teknik *Stratified Split* (Pembagian Terstratifikasi) harus diterapkan. Stratifikasi memastikan bahwa proporsi relatif

dari setiap kelas sentimen (persentase Positif, Negatif, dan Netral) tetap terjaga secara konsisten di semua himpunan data. Sebagai alternatif yang lebih *robust* daripada validasi *hold-out* statis, literatur sering merekomendasikan *K-Fold Cross-Validation* (Validasi Silang K-Lipatan) [18]. Dalam metode ini, data latih/validasi dibagi menjadi 'K' lipatan (*folds*). Model dilatih sebanyak K kali, di mana setiap iterasi menggunakan satu lipatan sebagai data validasi dan 'K-1' lipatan sisanya sebagai data latih [50]. Kinerja akhir model dihitung berdasarkan rata-rata dari 'K' iterasi tersebut, memberikan estimasi yang lebih stabil atas kinerja model, terutama saat *dataset* berlabel terbatas.

2.5 *Artificial Neural Network* (ANN)

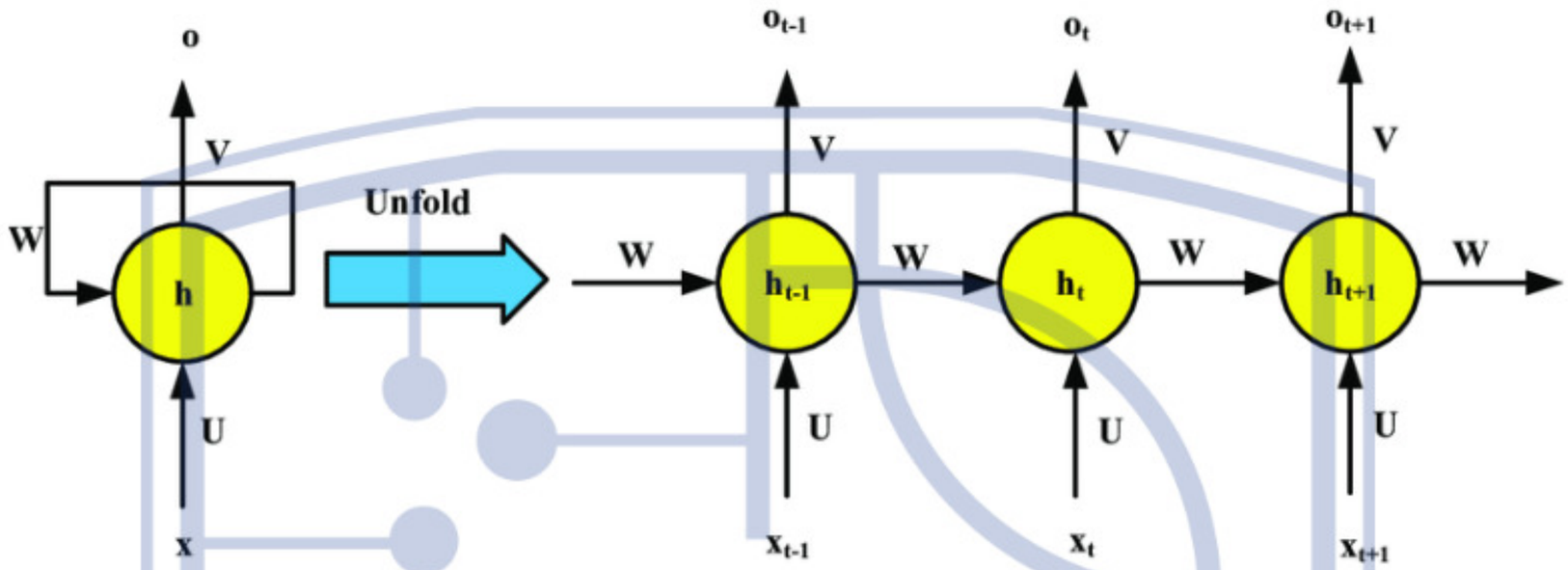
Artificial Neural Network (ANN) adalah model komputasional yang terinspirasi dari mekanisme kerja jaringan saraf biologis pada otak manusia [20]. Secara arsitektural, ANN tersusun dari unit pemrosesan yang disebut neuron buatan. Neuron ini diorganisasikan ke dalam tiga jenis lapisan, yaitu lapisan masukan (*input layer*), satu atau lebih lapisan tersembunyi (*hidden layers*), dan lapisan keluaran (*output layer*). Setiap koneksi antar lapisan memiliki bobot (*weight*) dan bias yang nilainya disesuaikan selama proses pelatihan. Fungsi aktivasi non-linear, seperti sigmoid, tanh, atau RELU, diterapkan pada keluaran tiap neuron agar model mampu mempelajari hubungan yang tidak dapat dimodelkan secara linear [20].

Proses pembelajaran ANN berlangsung melalui dua mekanisme utama: *forward propagation* untuk menghasilkan prediksi, dan *backward propagation* untuk memperbarui bobot berdasarkan nilai *loss function* menggunakan algoritma *gradient descent* [20]. Seiring meningkatnya kebutuhan pemrosesan berbagai jenis data, beberapa varian arsitektur ANN dikembangkan untuk mengatasi keterbatasan jaringan *feed forward* standar. Untuk memproses data berurutan seperti teks, tuturan, dan deret waktu, dikembangkan keluarga *Recurrent Neural Network* (RNN) beserta variannya, yang mencakup *Long Short-Term Memory* (LSTM), dan *Bidirectional LSTM* (BiLSTM) [51].

2.5.1 *Recurrent Neural Network* (RNN)

Recurrent Neural Network (RNN) adalah model *neural network* yang dirancang untuk memproses data berurutan (*sequential data*). Arsitektur RNN memungkinkan pembelajaran berulang pada *hidden layer* sehingga pola temporal dapat dipelajari dengan parameter yang dibagi sepanjang dimensi waktu (*t*). Namun, RNN konvensional mengalami

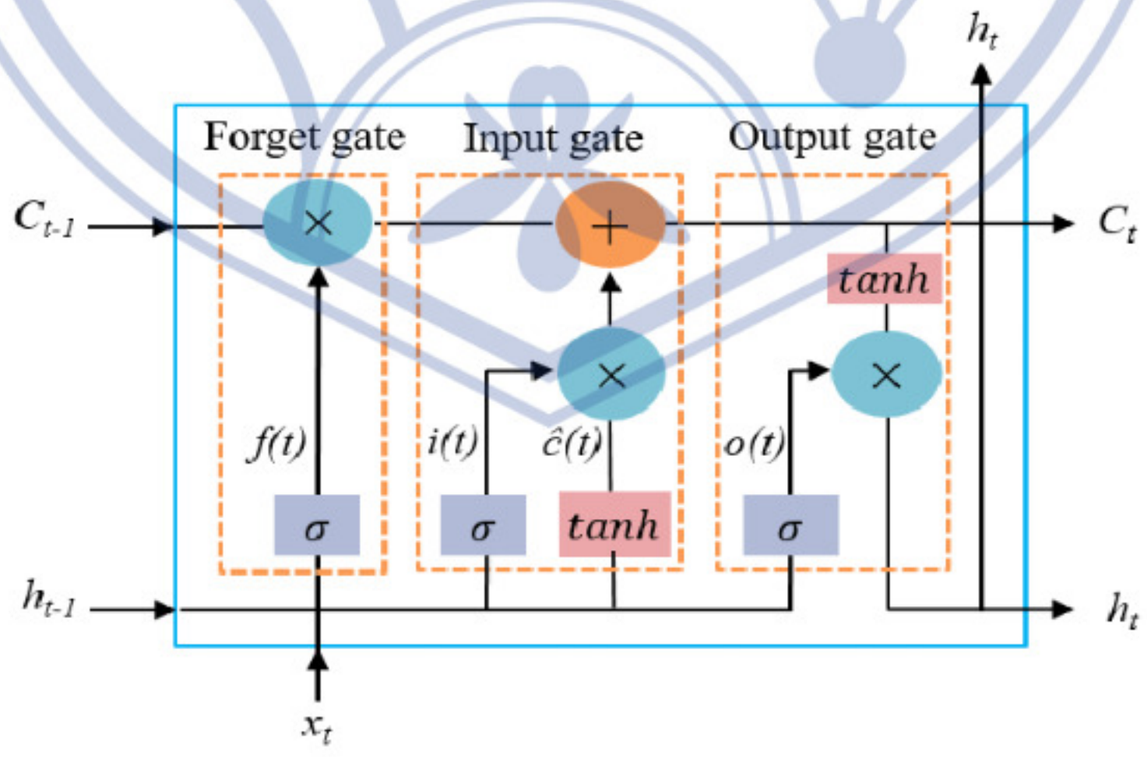
keterbatasan ketika mempelajari *long-term dependencies*, yakni masalah *vanishing* dan *exploding gradients* yang menyebabkan kesulitan dalam propagasi sinyal kesalahan melintasi banyak langkah waktu [52]. Kendala ini mendorong pengembangan beberapa turunan arsitektur RNN dikembangkan untuk mengatasi kendala ini, antara lain LSTM dan GRU [51].



Gambar 2.1 Arsitektur Dasar RNN [53]

2.5.2 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) dikembangkan untuk mengatasi keterbatasan RNN klasik dalam memodelkan dependensi jangka panjang. LSTM memperkenalkan *cell state*, serta tiga gerbang utama *forget gate*, *input gate*, dan *output gate* yang mengatur aliran informasi ke dalam, keluar dari, dan di dalam memori sel. Dengan mekanisme ini, LSTM dapat memilih informasi mana yang dipertahankan, ditulis, atau dibuang, sehingga menjaga sinyal relevan selama rentang waktu yang panjang [51].



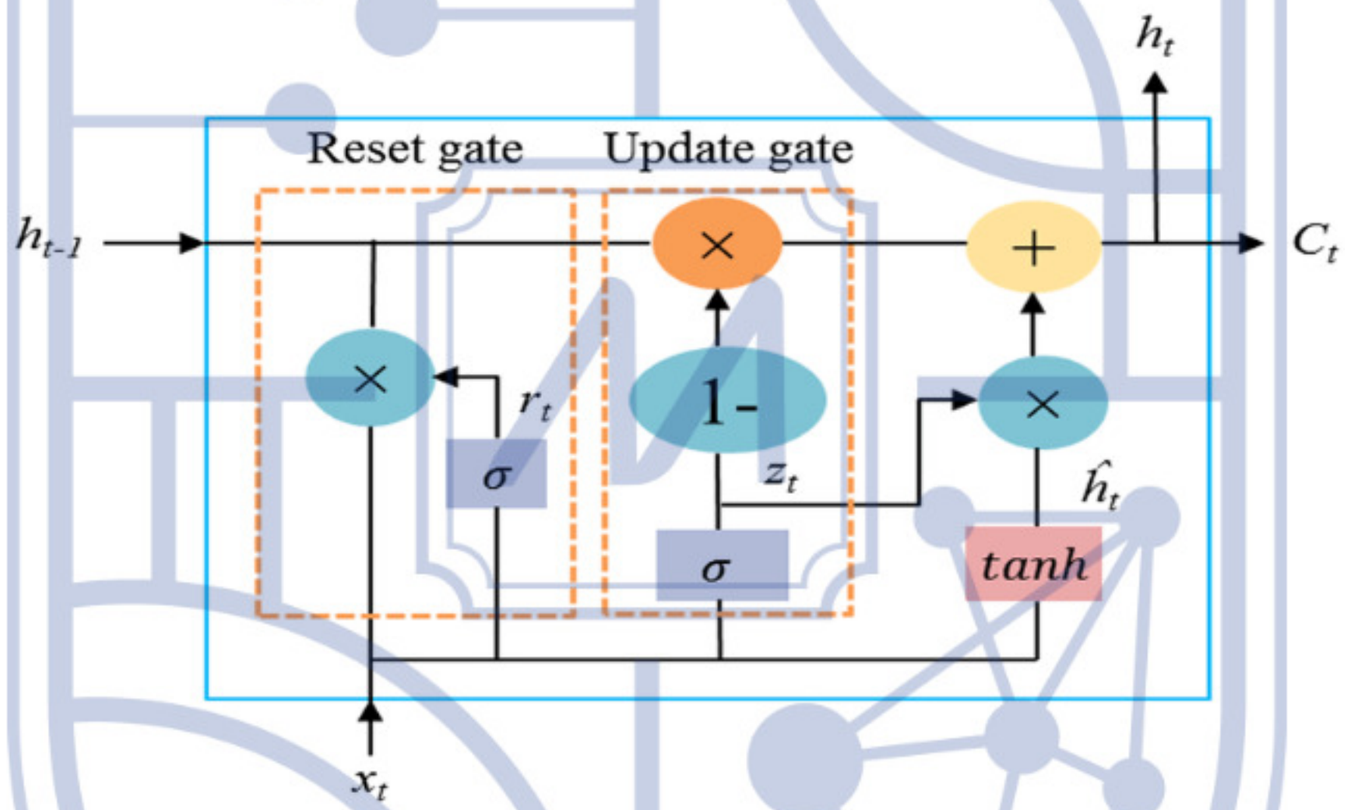
Gambar 2.2 Arsitektur LSTM [54]

Komponen dalam arsitektur LSTM sebagaimana bisa dilihat pada gambar 2.2 adalah sebagai berikut:

1. *Cell State (C_t),* sebagai penyimpanan memori jangka panjang.
2. *Forget Gate,* menentukan bagian dari $C_t - 1$ yang dipertahankan atau dilupakan.
3. *Input Gate,* filtrasi informasi yang akan ditambahkan ke C_t
4. *Output Gate,* mengontrol informasi yang dipindahkan dari C_t ke *hidden state (h_t).*

2.5.3 *Gated Recurrent Unit* (GRU)

Gated Recurrent Unit (GRU) adalah varian yang lebih sederhana dari LSTM. GRU mereduksi kompleksitas dengan hanya menggunakan dua gerbang yaitu *update gate* dan *reset gate*. serta menghilangkan pembagian eksplisit antara *cell state* dan *hidden state*. Desain ini menghasilkan jumlah parameter yang lebih sedikit dan komputasi yang lebih efisien [51].



Gambar 2.3 Arsitektur GRU [55]

Komponen dalam arsitektur GRU sebagaimana bisa dilihat pada gambar adalah sebagai berikut:

1. *Update Gate,* menentukan proporsi informasi lama dari *state* sebelumnya ($h_t - 1$) yang dipertahankan
2. *Reset Gate,* mengontrol informasi lama yang diabaikan untuk membentuk kandidat *hidden state* baru h_t.

2.5.4 *Bidirectional* LSTM (BiLSTM)

Bidirectional Long Short-Term Memory (BiLSTM) memproses urutan *input* di dua arah, *forward* (left to right) dan *backward* (*right to left*). Pada tiap posisi waktu (*t*), representasi akhir didapat dengan menggabungkan dari kedua arah, sehingga setiap token direpresentasikan dengan konteks sebelum dan sesudahnya. Pendekatan ini meningkatkan

kualitas representasi pada tugas-tugas *token-level* seperti *named-entity recognition*, *POS tagging*, dan beberapa varian klasifikasi kalimat karena memanfaatkan konteks dua arah [56].

Secara matematis, BiLSTM mengoperasikan dua jalur LSTM yang bekerja secara paralel namun berlawanan arah [56]. *Forward* LSTM memproses sekuens dari posisi pertama ke terakhir, sedangkan *backward* LSTM memproses dari posisi terakhir ke pertama. Keluaran kedua jalur pada setiap posisi waktu *t* kemudian digabungkan (*concatenated*) untuk membentuk representasi akhir setiap token, sebagaimana ditunjukkan pada Persamaan (2.1) hingga (2.6).

Setiap jalur LSTM menjalankan komputasi gerbang pada tiap langkah waktu. *Forget gate* menentukan informasi dari *cell state* sebelumnya yang dipertahankan, *input gate* memfilter informasi baru yang akan ditambahkan, *cell state* diperbarui dari hasil kedua gerbang tersebut, dan *output gate* menghasilkan *hidden state* akhir [56]:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.2}$$

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.3}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{2.4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.5}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.6}$$

Keterangan notasi pada persamaan (2.1) hingga (2.6) adalah sebagai berikut: t menyatakan indeks langkah waktu (*time step*) saat ini; x_t adalah vektor *input* pada langkah waktu t; h_{t-1} adalah *hidden state* dari langkah waktu sebelumnya; f_t adalah keluaran *forget gate* yang menentukan proporsi informasi lama yang dipertahankan; i_t adalah keluaran *input gate* yang memfilter informasi baru; $\widetilde{C}_t$ adalah kandidat *cell state* baru; C_{t-1} adalah *cell state* dari langkah waktu sebelumnya; C_t adalah *cell state* yang telah diperbarui; o_t adalah keluaran *output gate*; h_t adalah *hidden state* yang dihasilkan pada langkah waktu; W_f, W_i, W_C, W_o adalah matriks bobot *weight* masing-masing gerbang; b_f, b_i, b_C, b_o adalah vektor bias masing-masing gerbang; $\sigma(\cdot)$ adalah fungsi aktivasi sigmoid yang menghasilkan nilai dalam rentang (0, 1); $\tanh(\cdot)$ adalah fungsi aktivasi *hyperbolic tangent* yang menghasilkan nilai dalam rentang $(-1,1)$; dan $\odot$ adalah operasi perkalian *element-wise* (*Hadamard product*).

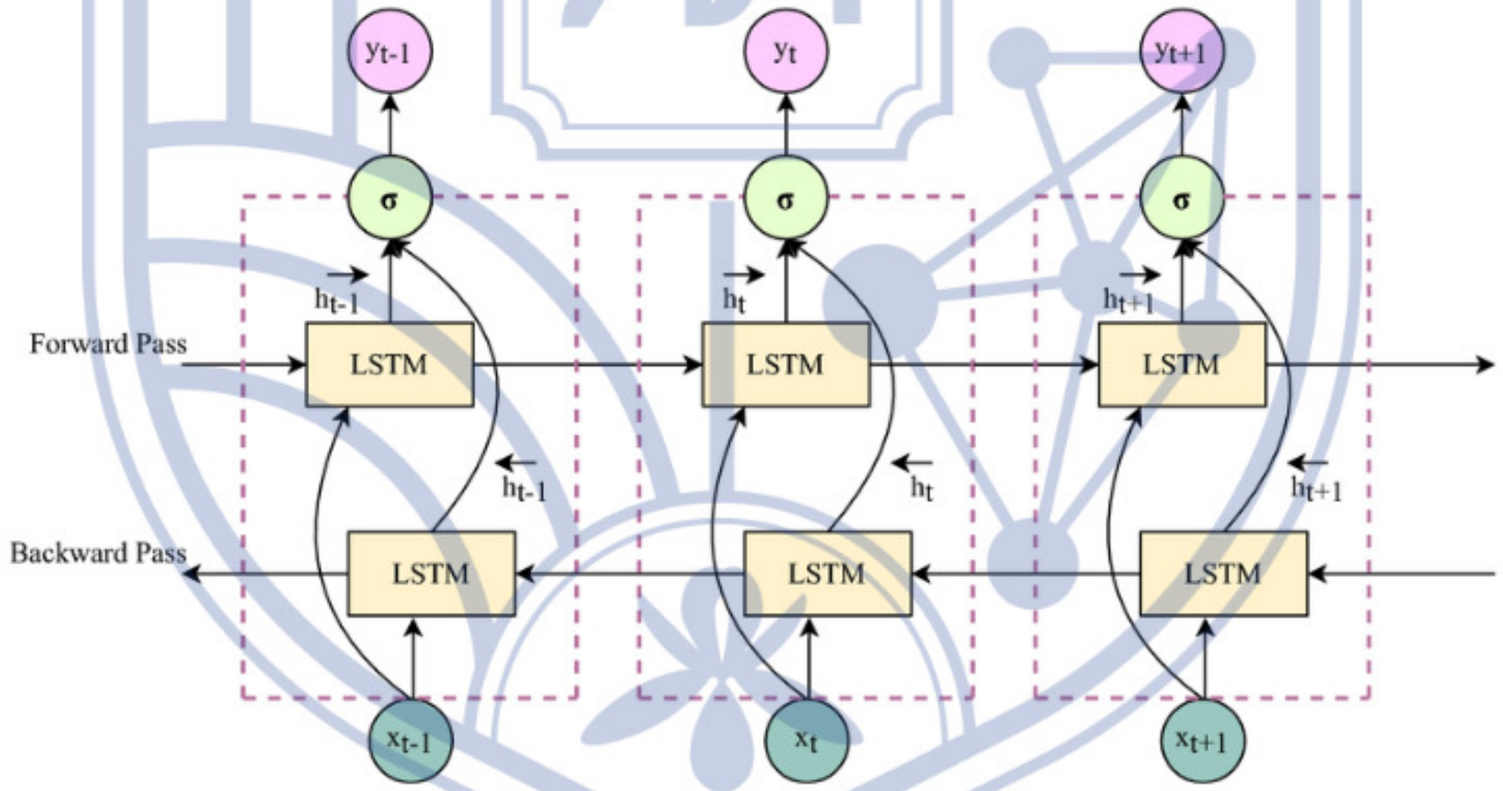
Pada BiLSTM, kedua jalur LSTM menghasilkan *hidden state* masing-masing, kemudian digabungkan:

$$\overrightarrow{h_t} = LSTM^{\rightarrow}(x_t, \overrightarrow{h_{t-1}}) \quad (2.7)$$

$$\overleftarrow{h_t} = LSTM^{\leftarrow}(x_t, \overleftarrow{h_{t+1}}) \quad (2.8)$$

$$h_t = [\overrightarrow{h_t}; \overleftarrow{h_t}] \quad (2.9)$$

Keterangan notasi pada Persamaan (2.7) hingga (2.9) adalah sebagai berikut: x_t adalah vektor *input* pada langkah waktu *t*; $\overrightarrow{h_t}$ adalah *hidden state* yang dihasilkan oleh *forward* LSTM pada langkah waktu *t*, yang memproses sekuens dari kiri ke kanan; $\overrightarrow{h_{t-1}}$ adalah *hidden state forward* dari langkah waktu sebelumnya; h_t adalah *hidden state* yang dihasilkan oleh *backward* LSTM pada langkah waktu *t,* yang memproses sekuens dari kanan ke kiri; $\overleftarrow{h_{t+1}}$ adalah *hidden state backward* dari langkah waktu berikutnya; h_t adalah representasi akhir token pada posisi *t* yang merupakan gabungan kedua arah; dan [;] menyatakan operasi *concatenation* yang menggabungkan dua vektor secara berurutan menjadi satu vektor berdimensi lebih besar.



Gambar 2.4 Arsitektur BiLSTM [57]

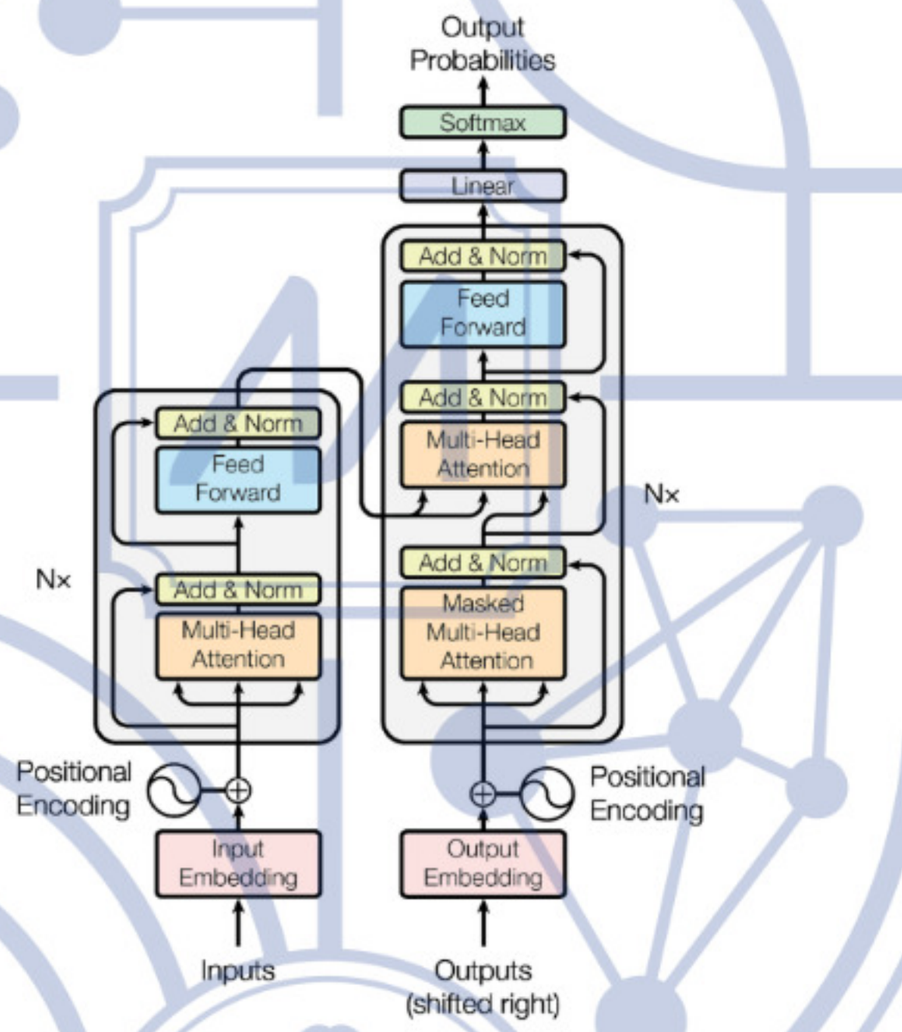
2.6 Model Berbasis *Transformer*

Model berbasis *Transformer* merupakan keluarga arsitektur *deep learning* yang diperkenalkan oleh Vaswani *et al.* dan telah merevolusi bidang NLP sejak kemunculannya [58]. Berbeda dari pendekatan sekuensial seperti RNN dan LSTM yang memproses token satu per satu, arsitektur *Transformer* memproses seluruh token secara paralel menggunakan mekanisme *self-attention*, sehingga mampu menangkap dependensi jarak jauh (*long-range dependencies*) antar kata tanpa kendala posisi [58]. Keunggulan ini menjadikan *Transformer*

sebagai fondasi dari berbagai model *pre-trained* modern, termasuk BERT dan variannya [59]. Subbab ini membahas komponen-komponen arsitektur *Transformer* secara berurutan: arsitektur dasar, mekanisme *attention*, komponen *encoder* dan *decoder*, lapisan *output*, serta model *pre-trained* dan *fine-tuning* yang dibangun di atasnya.

2.6.1 Arsitektur *Transformer*

Arsitektur *Transformer* adalah arsitektur *deep learning* yang diperkenalkan sebagai alternatif dari arsitektur sekuensial tradisional seperti *Recurrent Neural Network* (RNN) dan *Convolutional Neural Network* (CNN). Berbeda dengan model sekuensial dan konvolusi yang memproses data secara berurutan kata demi kata, *Transformer* mengadopsi mekanisme utama yang disebut *self-attention* [58], [21].



Gambar 2.5 Arsitektur Model *Transformer* [58]

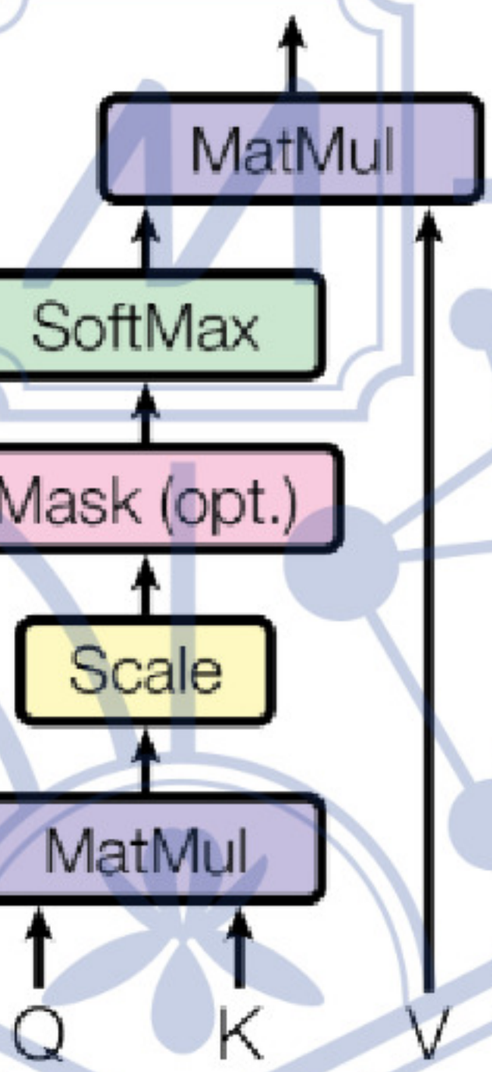
Mekanisme *self-attention* memungkinkan model melakukan pemrosesan paralel atas keseluruhan token dalam sebuah sekuens. Hal ini dicapai dengan secara dinamis menghitung relevansi bobot antara sebuah token dengan seluruh token lainnya dalam sekuens tersebut sehingga model dapat menentukan relasi antar kata tanpa terikat oleh jarak posisionalnya [58].

Model *self-attention* dapat menangkap dependensi jarak jauh (*long range dependencies*) yang membuat arsitektur *Transformer* lebih efektif daripada RNN. Arsitektur ini menjadi fondasi model *pre-trained* yang lebih besar, yang secara kolektif dikenal sebagai *Transformer-based models*. Model seperti BERT, mengubah paradigma NLP dari pelatihan model *from scratch* untuk setiap tugas, menjadi model yang telah dilatih pada data besar

(*pre-training*) dan kemudian disesuaikan (*fine-tuning*) pada tugas-tugas yang lebih spesifik [21], [59].

a. *Scaled Dot-Product Attention*

Scaled dot-product attention adalah komponen inti pada *Transformer* yang menghitung bobot atensi (*attention*) antar token dalam sebuah sekuens dengan membandingkan *vector query* (Q) terhadap *vector key* (K). Hasil dari perbandingan Q dan K dinormalisasi (*scaling*) terhadap $\sqrt{d_k}$ kemudian melewati fungsi *softmax* untuk menghasilkan bobot *perhatian*, bobot kemudian digunakan mengagregasi *vector value* (V) sehingga setiap token memperoleh representasi kontekstual yang mengukur kontribusi token lainnya dalam sekuens. Mekanisme ini memungkinkan model untuk menangkap dependensi jarak jauh secara paralel dan menjadi dasar *multi-head attention* [58].

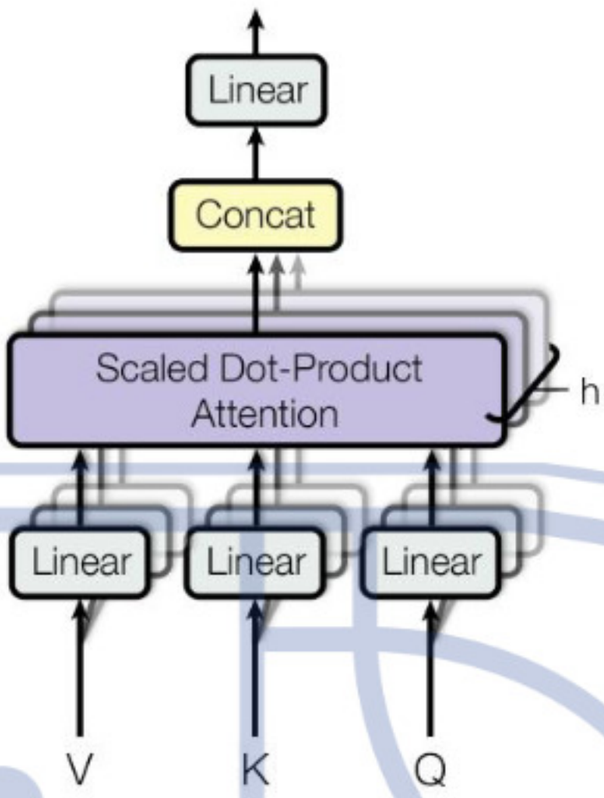


Gambar 2.6 *Scaled Dot-Product Attention* [58]

b. *Multi-Head Attention*

Multi-Head Attention (MHA) merupakan mekanisme yang memungkinkan model untuk menerapkan fungsi *attention* pada beberapa subruang representasi secara paralel. Pada MHA, setiap *head* melakukan proyeksi terpisah terhadap Q, K, dan V, kemudian hasil dari seluruh *head* digabungkan (*concatenated*) dan diproyeksikan kembali untuk menghasilkan representasi akhir. Pendekatan ini memungkinkan model untuk menangkap berbagai jenis

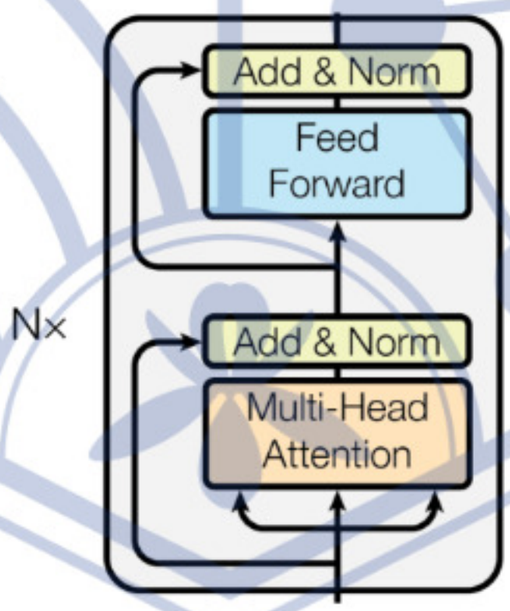
hubungan antar token dari perspektif yang berbeda secara bersamaan, sehingga memperkaya representasi kontekstual dibandingkan *attention* tunggal [58].



Gambar 2.7 *Multi-Head Attention* [58]

c. *Encoder*

Blok *encoder* berfungsi mengubah urutan token *input* (*x1, x2, ..., x_n*) menjadi representasi kontinu yang mempresentasikan makna kontekstual pada setiap token. Representasi ini kemudian digunakan oleh *decoder* atau modul *downstream* untuk menghasilkan *output* atau prediksi tugas spesifik [58], [60], [61].



Gambar 2.8 *Encoder Transformer* [58]

1. *Encoder* terdiri dari *stack* sebanyak N = 6 *layer* identik. Setiap *layer* memiliki 2 *sub-layer* yaitu:
    a. *Multi-head self-attention layer,* mekanisme ini memungkinkan setiap posisi token dalam sekuens mengagregasi informasi pada seluruh posisi lain secara paralel.

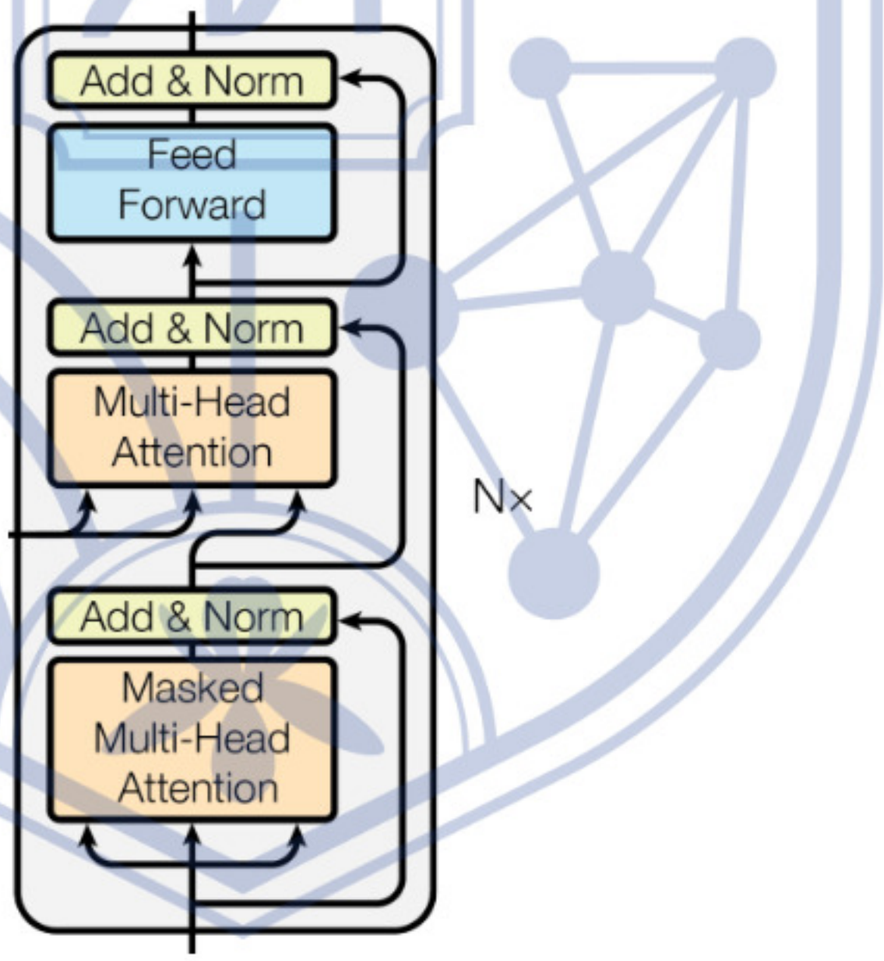
b. *Position-wise feed-forward network, feed-forward network* yang diterapkan secara independen pada setiap posisi token untuk melakukan transformasi *non-linear* fitur lokal dan memperkaya representasi setelah tahap *attention*.

2. Setiap *sub-layer* memiliki *residual connections* dan *layer normalization. Residual connection* membantu menjaga aliran gradien sehingga mempermudah konvergensi, sementara *layer normalization* menstabilkan distribusi aktivasi di tiap *sub-layer*.

d. *Decoder*

Blok *decoder* berfungsi untuk menghasilkan urutan *output* berdasarkan representasi kontekstual yang dihasilkan oleh *encoder*. Berbeda dengan *encoder*, setiap *layer* pada *decoder* menambahkan satu *sub-layer* tambahan, yaitu *masked multi-head self-attention* untuk memastikan autogresif pada saat generasi [58].

Pada *sub-layer* ini, *self-attention* dimodifikasi dengan menerapkan masking untuk mencegah model memerhatikan token-token sebelumnya sehingga fokus memerhatikan token di posisi berikutnya. Mekanisme ini memastikan bahwa proses generasi bersifat autoregresif, yaitu setiap token *output* hanya bergantung pada token-token sebelumnya.



Gambar 2.9 *Decoder Transformer* [58]

1. *Decoder* terdiri dari *stack* sebanyak N = 6 *layer* identik, setiap *layer* memiliki tiga *sub-layer* utama:
   a. *Masked multi-head self-attention layer, multi-head self-attention* yang dimasked untuk menjaga kausalitas pada proses generasi token.

b. *Multi-head encoder-decoder attention layer,* komponen yang memungkinkan *decoder* mengakses dan menyeleksi informasi relevan dari *output encoder*.
c. *Position-wise feed-forward network, feed-forward network* yang diterapkan identik pada setiap posisi untuk transformasi *non-linear*.

2. Setiap *sub-layer* dilengkapi dengan *residual connection* dan *layer normalization* untuk menjaga stabilitas pelatihan serta mempercepat konvergensi model.

e. *Output Layer*

Setelah melewati *stack decoder*, *output* berupa representasi vektor berdimensi tinggi diubah menjadi distribusi probabilitas terhadap seluruh kosakata target. *Output layer* mengonversi representasi vektor menjadi skor *logit*, kemudian menormalisasi skor tersebut menggunakan fungsi *softmax* untuk menentukan token berikutnya (*next-token prediction)* [58].

Proses ini dilakukan melalui dua tahap utama, yaitu:

1. *Linear transformation*, memproyeksikan vektor berdimensi tinggi dari *decoder* ke dimensi vektor sesuai dengan ukuran kosakata target (*vocabulary size*).
2. *Softmax function*, menormalisasi hasil proyeksi skor *logit* menjadi distribusi probabilitas untuk token berikutnya.

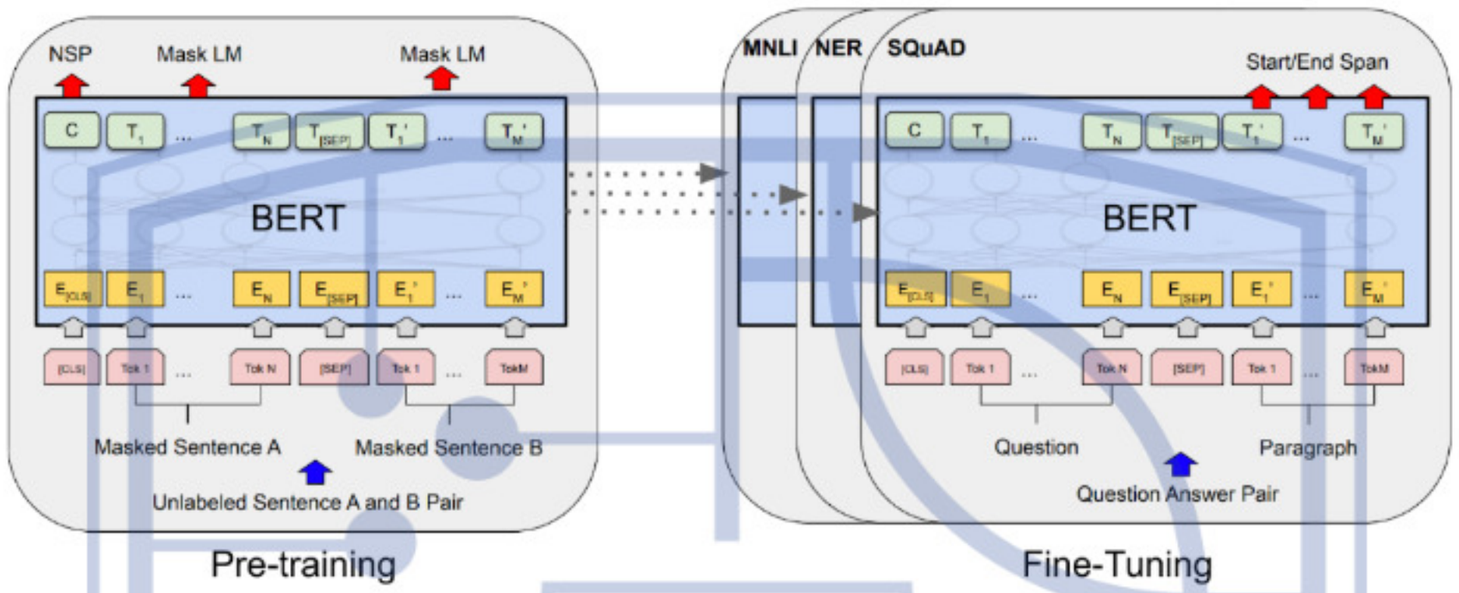
Token dengan probabilitas tertinggi umumnya dipilih sebagai *output* pada langkah tersebut, dan proses ini berlanjut secara *autoregressive* hingga keseluruhan urutan *output* terbentuk atau token akhir (*end-of-sequence*) dihasilkan.

2.6.2 *Pre-trained Model*

Pre-trained model adalah arsitektur *deep learning* yang terlebih dahulu dilatih pada korpus data berukuran besar menggunakan pendekatan *self-supervised learning*, sebelum diterapkan pada tugas spesifik [62], [63]. Pada fase *pretrained* ini, model mempelajari representasi bahasa yang umum, termasuk struktur sintaksis dan konteks semantik, sehingga tidak perlu belajar dari nol untuk setiap tugas baru. Keunggulan utama pendekatan ini adalah *transfer learning*: pengetahuan linguistik yang diperoleh dari data umum dapat ditransfer ke tugas yang memiliki data berlabel terbatas [64], [65]. Pada model berbasis *Transformer* seperti BERT, *pretrained* dilakukan pada miliaran kata untuk menangkap hubungan antar token dan antar kalimat.

A. *Bidirectional Encoder Representations from Transformers* (BERT)

Bidirectional Encoder Representation from Transformers (BERT) merupakan model *encoder-only* representasi bahasa kontekstual *bidirectional* yang dilatih atau *pre-training self-supervised*. BERT dapat di *fine-tuning* untuk tugas *downstream* atau spesifik dengan menambahkan *layer task-specific* [21].



Gambar 2.10 Arsitektur BERT [21]

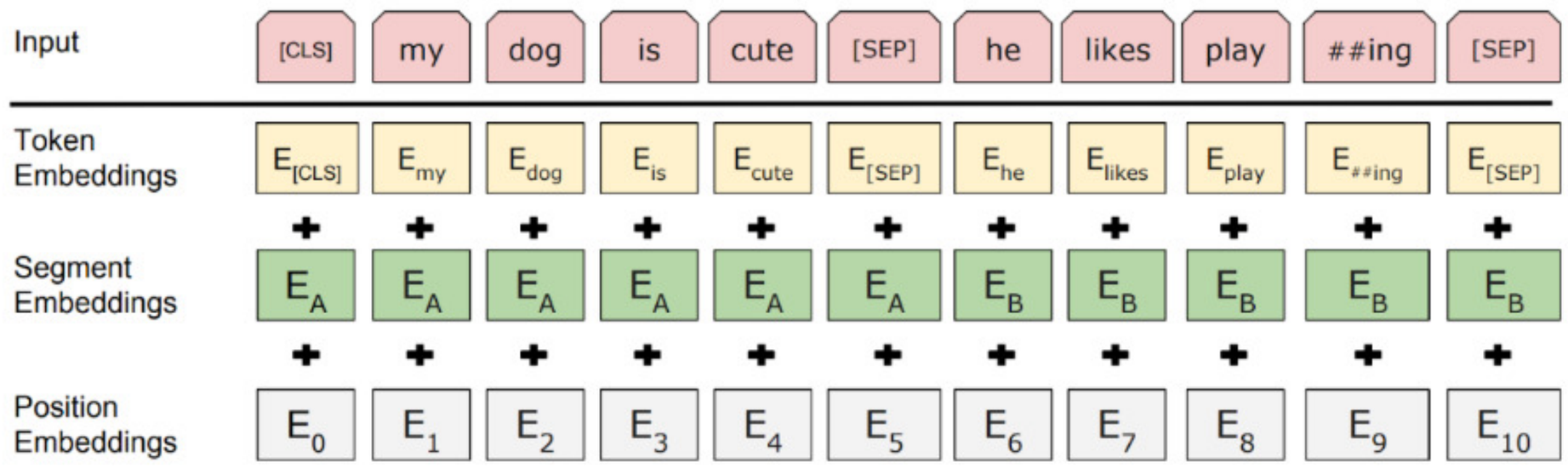
BERT terdiri dari 2 tahapan prosedur utama, yakni *pre-training* dan *fine-tuning*.

1. *Pre-training*, melatih model dalam memahami representasi bahasa secara umum menggunakan dua tujuan utama, yakni *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP).
2. *Fine-tuning*, dilakukan untuk menyesuaikan parameter hasil *pre-training* agar sesuai dengan tugas spesifik (*task-specific objective*).

Basis dari model BERT adalah *encoder-stack Transformer* yang terdiri 12 *encoder layer* dengan 768 *hidden size* dan 12 *attention head* untuk BERT Base, dan untuk BERT Large 24 *encoder layer* dengan 1024 *hidden size* dan 24 *attention head* [58].

a. *Tokenization* dan *Embedding* BERT

Tokenization merupakan proses memecah teks mentah menjadi unit-unit kecil yang disebut token, agar teks dapat diproses oleh model NLP. Tokenisasi dilakukan menggunakan *WordPiece*, sebuah metode *subword tokenization* yang memecah kata menjadi unit lebih kecil jika kata tersebut tidak ada dalam kosakata model [21]. Pendekatan ini memungkinkan BERT menangani kata-kata baru (*out-of-vocabulary*) dan variasi morfologi dengan lebih efisien dibandingkan tokenisasi berbasis kata utuh [60], [61].



Gambar 2.11 *Input* Representasi BERT [21]

Selama proses *pre-training*, BERT menggunakan *Masked Language Modeling* (MLM) dengan cara *masking* secara acak 15% token dari setiap urutan *input*. Model kemudian dilatih untuk memprediksi token *masked* tersebut dengan mempertimbangkan konteks dari kedua arah secara *bidirectional*. Strategi ini memungkinkan BERT untuk membangun pemahaman semantik yang lebih dalam terhadap struktur bahasa [21].

BERT juga menggunakan *Next Sentence Prediction* (NSP) untuk melatih kemampuan model memahami hubungan antar dua kalimat atau *sentence pair*. Pada tahap ini, model diberi dua segmen teks dan diminta memprediksi apakah segmen kedua secara logis mengikuti segmen pertama [21].

Dalam setiap urutan (*sequence*) token, BERT menambahkan token khusus, yaitu:

1. **[CLS]** (*classification token*), yang ditempatkan di awal urutan dan digunakan sebagai representasi agregat dari seluruh kalimat untuk tugas klasifikasi.
2. **[SEP]** (*separator token*), yang digunakan sebagai pemisah antar kalimat atau segmen teks.
3. **[MASK]**, yang digunakan saat pelatihan MLM untuk menggantikan sebagian token yang disembunyikan.

Setelah proses tokenisasi dan penambahan token khusus, setiap token dikonversi menjadi vektor numerik melalui tiga komponen *embedding* utama, yaitu:

1. *Token Embedding*, merepresentasikan masing-masing token dalam bentuk vektor.
2. *Segment Embedding*, membedakan segmen teks pertama dan kedua dalam *input* model.
3. *Positional Embedding*, mekanisme untuk merepresentasikan posisi token dalam kalimat sehingga model memahami struktur urutan kata.

Ketiga *embedding* tersebut dijumlahkan untuk menghasilkan representasi akhir setiap token yang kemudian diteruskan ke lapisan *encoder* BERT [21], [61].

b. *Encoder* Pada BERT

Struktur *encoder* BERT mengadopsi arsitektur *Transformer encoder-stack* yang terdiri atas beberapa lapisan (*multi-layer stack*). Tiap *layer encoder* memiliki tiga komponen utama yang secara prinsip sama seperti pada *Transformer encoder* [58].

1. *Multi-Head Self-Attention* (MHSA)

   Komponen ini memungkinkan setiap token dalam urutan memperhatikan token lain di seluruh konteks kalimat secara *bidirectional*. Melalui beberapa *attention heads*, model dapat mempelajari berbagai jenis relasi semantik dari representasi yang berbeda.

2. *Feed-Forward Network* (FFN)

   Merupakan jaringan saraf dua lapis yang bekerja secara independen pada setiap token. FFN melakukan transformasi *non-linear* terhadap representasi hasil MHSA menggunakan fungsi aktivasi GELU (*Gaussian Error Linear Unit*) untuk menghasilkan representasi yang lebih kompleks dan informatif.

3. *Residual Connection* dan *Layer Normalization*

   Residual connection menambahkan keluaran dari setiap *sub-layer* dengan *input* aslinya untuk menjaga kestabilan gradien dan mempertahankan informasi dari lapisan sebelumnya. Sementara itu, *layer normalization* berfungsi untuk menormalkan nilai aktivasi pada setiap lapisan agar proses pelatihan tetap stabil dan konvergen dengan baik [58], [60].

B. IndoBERT

IndoBERT merupakan model bahasa BERT yang dilatih secara khusus pada korpus berbahasa Indonesia. Model ini dikembangkan untuk mengatasi keterbatasan model multibahasa seperti *BERT* dalam memahami karakteristik linguistik dan morfologis Bahasa Indonesia, seperti imbuhan, reduplikasi, serta struktur kalimat yang khas [21], [66].

IndoBERT menggunakan konfigurasi yang identik dengan *BERT-Base*, yakni terdiri dari 12 *hidden layers*, 768 dimensi vektor, 12 *attention heads*, dan FFN 3.072 [21]. Perbedaan utama IndoBERT terletak pada tahap *pre-training*, di mana model ini dilatih secara *monolingual* menggunakan korpus Bahasa Indonesia yang besar dan bervariasi.

Korpus pelatihan IndoBERT bersumber dari *Indonesian Language Evaluation Montage* (IndoLEM) yang berisi ~220 juta kata hasil ekstraksi dari berbagai domain, seperti Wikipedia, berita, dan situs web umum berbahasa Indonesia yang telah melalui *preprocessing* dan normalisasi [66], [67]. Pendekatan ini memungkinkan IndoBERT menghasilkan representasi semantik yang lebih akurat untuk konteks Bahasa Indonesia

dibandingkan model multibahasa, terutama dalam tugas-tugas seperti klasifikasi teks, analisis sentimen, dan ekstraksi informasi [67].

2.6.3 *Fine-Tuning*

Fine-tuning merupakan proses adaptasi model *pre-trained model* terhadap tugas spesifik menggunakan *dataset* yang lebih kecil dan berorientasi pada domain tertentu. Teknik ini memungkinkan model untuk mempertahankan pengetahuan umum yang diperoleh dari *pretrained* sekaligus menyesuaikan representasi internalnya agar lebih relevan dengan konteks baru [68]. *Fine-tuning* telah terbukti secara empiris meningkatkan performa model tanpa memerlukan pelatihan dari awal, serta mempercepat konvergensi pembelajaran [69]. Melalui proses ini, hanya sebagian parameter yang diperbarui, sehingga efisiensi komputasi dapat tetap terjaga sembari mempertahankan generalisasi dari *pre-trained model* [70].

Pendekatan ini efektif dalam meningkatkan ketahanan dan akurasi model lintas domain. Wang *et al.* memperkenalkan metode *Pre-trained Model Guided Fine-Tuning* yang mampu meningkatkan *robustness* terhadap serangan *adversarial* dalam skenario *zero-shot* [68]. Sementara itu, Jia *et al.* menegaskan bahwa *parameter-efficient fine-tuning* (PEFT) menawarkan peningkatan efisiensi pelatihan hingga 90% tanpa penurunan kinerja yang signifikan [70]. Temuan ini menjadikan *fine-tuning* menjadi pilar utama dalam strategi *transfer learning* modern, yang berfungsi memperkuat kemampuan generalisasi dan meningkatkan kinerja model terhadap berbagai tugas spesifik.

2.7 Arsitektur Hibrida IndoBERT-BiLSTM

Model hibrida IndoBERT-BiLSTM dipilih untuk memadukan representasi kontekstual *pretrained* dengan pemodelan sekuensial yang krusial dalam menangani teks autentik (*user-generated content*) yang bersifat *noise*. IndoBERT berfungsi sebagai *contextual feature extractor* yang menghasilkan *embeddings* sensitif terhadap morfologi, imbuhan, *code-mixing*, dan variasi non-baku Bahasa Indonesia, sedangkan BiLSTM memodelkan dependensi temporal lokal yang secara signifikan memengaruhi polaritas sentimen. Dalam penelitian ini IndoBERT digunakan sebagai *pretrained encoder* yang di *fine-tuning*, kemudian keluaran *embeddings* diproses oleh lapisan BiLSTM, dilanjutkan dengan mekanisme *pooling* dan lapisan klasifikasi untuk memprediksi label sentimen.

Cara kerja IndoBERT-BiLSTM sebagai berikut:

1. *Dataset splitting*. Bagi *dataset* menjadi *training*, *validation*, dan *test set*.
2. *Preprocessing*. *Preprocessing* ringan dilakukan untuk menjaga autentisitas dan integritas semantik teks.
3. *Tokenization*. Tokenisasi korpus menggunakan *tokenizer* IndoBERT sehingga teks diubah menjadi token.
4. *Contextual encoding*. Token diproses *encoder* IndoBERT untuk menghasilkan *contextual embeddings*.
5. *Sequential modelling*. BiLSTM memodelkan urutan *embedding* untuk menangkap dependensi temporal eksplisit, seperti negasi, intensifikasi, dan relasi antarklausa.
6. *Pooling* and *classification*. Melakukan *pooling* dari keluaran BiLSTM, kemudian diproses *dense classifier* untuk menghasilkan skor kelas.
7. *Output*. Menghasilkan prediksi label sentimen keluaran akhir.

Penelitian terdahulu menunjukkan bahwa penggabungan model *Transformer pretrained* dengan BiLSTM mampu meningkatkan kualitas representasi dan akurasi pada tugas klasifikasi emosi dan sentimen pada korpus berbahasa Indonesia. Meskipun IndoBERT menghasilkan *contextual embeddings*, integrasi dengan BiLSTM terbukti mampu memperkaya pemodelan dependensi sekuensial yang lebih baik. Kombinasi ini dilaporkan mampu meningkatkan akurasi dan ketahanan pada teks media sosial berbahasa Indonesia yang bersifat autentik dan berderau [35], [7], [8].

2.8 *Hyperparameter Tuning*

Kinerja model *deep learning*, termasuk arsitektur hibrida IndoBERT-BiLSTM sangat dipengaruhi oleh konfigurasi *hyperparameter*. Berbeda dari parameter model seperti bobot dan bias yang dipelajari dari data, *hyperparameter* adalah nilai yang ditetapkan sebelum pelatihan dimulai [71]. Pemilihan kombinasi *hyperparameter* yang tepat menentukan keseimbangan antara kemampuan model untuk mempelajari pola dari data latih dan kemampuannya menggeneralisasi ke data baru. Berikut adalah *hyperparameter* utama yang memegang peranan kritis dalam penelitian ini:

1. *Optimizer*. Merupakan algoritma yang digunakan untuk memperbarui parameter model (bobot) guna meminimalkan fungsi kerugian (*loss function*). Untuk model berbasis *Transformer* seperti IndoBERT, literatur sangat merekomendasikan penggunaan AdamW (*Adam with Weight Decay*). AdamW merupakan varian dari *optimizer* Adam yang memisahkan peluruhan bobot (*weight decay*) dari pembaruan gradien, yang terbukti

secara empiris mampu meningkatkan stabilitas pelatihan dan generalisasi pada model bahasa besar [72].

2. *Learning Rate*. Mengontrol seberapa besar langkah yang diambil model saat memperbarui bobot. Pada *fine-tuning* IndoBERT, nilai ini biasanya diset sangat kecil (misalnya 2×10^5 hingga 5×10^5) untuk mencegah kerusakan pada fitur linguistik yang telah dipelajari sebelumnya (*catastrophic forgetting*) [73].
3. *Batch Size*. Jumlah sampel data yang diproses model sebelum memperbarui bobot internalnya. Ukuran batch berpengaruh pada stabilitas gradien dan kecepatan konvergensi. Pemilihan ukuran *batch* yang tepat penting karena *batch* yang terlalu besar berpotensi menurunkan generalisasi model, sedangkan *batch* yang terlalu kecil dapat menyebabkan estimasi gradien menjadi kurang stabil [74].
4. *Epoch*. Jumlah iterasi total di mana model melihat keseluruhan *dataset* latih. Jumlah *epoch* harus ditala (*tuned*) dengan hati-hati menggunakan teknik *early stopping* untuk menghindari *overfitting*. Penentuan jumlah *epoch* yang optimal erat kaitannya dengan mekanisme regularisasi seperti *early stopping*, yang berfungsi menghentikan pelatiahn begitu performa pada data validasi berhenti membaik guna mencegah *overfitting* [75].
5. LSTM *Hidden Size*. Spesifik pada arsitektur hibrida, *hyperparameter* ini menentukan jumlah unit memori tersembunyi dalam lapisan BiLSTM. Nilai ini merepresentasikan kapasitas model dalam menangkap fitur kontekstual dari urutan *embedding* yang dihasilkan IndoBERT. Ukuran *hidden state* yang lebih besar memberikan kapasitas representasi yang lebih tinggi, namun juga meningkatkan risiko *overfitting* serta beban komputasi, sehingga pemilihannya perlu mempertimbangkan keseimbangan antara kapasitas model dan ukuran *dataset* yang tersedia [76].
6. *Dropout Rate* & *Weight Decay*. Merupakan mekanisme regularisasi untuk mencegah *overfitting*. *Dropout* bekerja dengan menonaktifkan sebagian *neuron* secara acak selama pelatihan, sedangkan *Weight Decay* memberikan penalti pada bobot model yang terlalu besar agar model tetap sederhana dan tangguh (*robust*). Kedua teknik regularisasi ini terbukti secara empiris efektif dalam meningkatkan kemampuan generalisasi model *deep learning* dengan mengurangi ketergantungan model terhadap fitur-fitur spesifik pada data latih [72], [77].

Mengingat banyaknya kombinasi variabel tersebut, proses *hyperparameter tuning* diperlukan untuk mencari konfigurasi optimal.

2.9 Metrik Evaluasi Model

Evaluasi kinerja model merupakan tahap penting untuk menilai sejauh mana model mampu melakukan prediksi secara akurat terhadap data uji (*ground truth*), diperlukan metrik kuantitatif yang dapat menggambarkan kinerja model secara objektif dan komprehensif. Penggunaan satu metrik tunggal seperti *accuracy* sering kali tidak cukup untuk memberikan gambaran performa yang representatif, terutama apabila distribusi kelas tidak seimbang, sehingga diperlukan evaluasi yang lebih menyeluruh dengan berbagai metrik diperlukan untuk memahami kekuatan dan kelemahan model secara mendalam [78].

1. *Accuracy*

   Accuracy mengukur proporsi prediksi yang benar terhadap total jumlah sampel dalam *dataset* pengujian. Metrik ini memberikan gambaran umum mengenai seberapa sering model mengklasifikasikan teks secara benar.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.10}$$

   Keterangan:

   TP = True Positive, prediksi benar untuk kelas positif

   TN = True Negative, prediksi benar untuk kelas negatif.

   FP = False Positive, prediksi salah untuk kelas positif.

   FN = False Negative, prediksi salah untuk kelas negatif.

2. *Precision*

   Precision mengukur proporsi prediksi positif yang benar dari seluruh *instance* yang diprediksi sebagai positif oleh model. Metrik ini menilai sejauh mana model tepat dalam memprediksi kelas tertentu. Nilai *precision* yang tinggi menunjukkan bahwa ketika model memprediksi teks sebagai positif, prediksi tersebut memiliki tingkat keandalan tinggi.

$$Precision = \frac{TP}{TP + FP} \tag{2.11}$$

3. *Recall*

   Recall atau sensitivitas mengukur sejauh mana model berhasil mengenali semua *instance* aktual dari suatu kelas target. Metrik ini menghitung rasio antara jumlah prediksi positif

yang benar terhadap total *instance* positif aktual. Nilai *recall* yang tinggi menunjukkan kemampuan model dalam mendeteksi sebagian besar *instance* positif.

$$Recall = \frac{TP}{TP + FN} \tag{2.12}$$

4. *F1-Score*

F1-Score merupakan metrik gabungan yang menghitung rata-rata harmonik dari *precision* dan *recall*, yang berguna untuk menilai keseimbangan antara kedua metrik tersebut. Nilai *F1-Score* yang tinggi mengindikasikan bahwa model memiliki keseimbangan baik antara ketepatan dan kelengkapan dalam melakukan klasifikasi.

$$F1 = 2\,x\,\frac{Precision\,x\,Recall}{Precision + Recall} \tag{2.13}$$

5. *Macro F1-Score*

Macro F1-Score merupakan metrik evaluasi pada tugas klasifikasi multikelas yang dihitung dengan cara mengambil rata-rata dari nilai *F1-Score* pada setiap kelas. Dalam pendekatan ini, setiap kelas diberikan bobot yang sama tanpa mempertimbangkan jumlah sampel pada masing-masing kelas, sehingga *Macro F1-Score* sangat sesuai digunakan pada *dataset* yang memiliki distribusi kelas tidak seimbang (*imbalanced*), karena metrik ini tidak didominasi oleh kelas mayoritas.

$$Macro\,F1 = \frac{1}{K}\sum_{i=1}^{K} F1_i \tag{2.14}$$

6. *Confusion Matrix*

Confusion Matrix merupakan representasi tabular yang menunjukkan jumlah prediksi benar dan salah untuk setiap kelas. Matriks ini menyajikan nilai secara eksplisit, sehingga memudahkan analisis kesalahan klasifikasi antar kelas. Struktur umum *confusion matrix* untuk klasifikasi biner ditunjukkan pada Tabel 2.1.

Tabel 2.1 *Confusion Matrix*

	Positive Predict	**Negative Predict**
Actual Positive	TP	FN
Actual Negative	FP	TN

2.10 Penelitian Terdahulu

Penelitian mengenai analisis sentimen dalam domain politik dan kebijakan publik terus berkembang seiring meningkatnya aktivitas politik digital di media sosial. Platform seperti X kini menjadi ruang strategis dalam memetakan opini publik dan dinamika wacana politik. Studi Rauniyar *et al.* (2023) menunjukkan bahwa diskursus politik di media sosial, memainkan peran penting dalam memetakan opini publik secara langsung [79]. Studi tersebut menegaskan bahwa percakapan publik, penggunaan *hashtag*, dan aktivitas digital masyarakat mampu merepresentasikan sikap politik serta respons terhadap isu kebijakan. Meskipun fokus utama penelitian tersebut adalah penyusunan *dataset* untuk konteks pemilu Nepal, temuan penelitian mereka menegaskan bahwa analisis sentimen berbasis media sosial merupakan pendekatan yang esensial dalam studi politik modern. Melalui pemetaan ekspresi publik di platform seperti X, analisis ini mampu menangkap dinamika sikap politik masyarakat terhadap isu dan peristiwa kontemporer secara lebih cepat dan terukur.

Penerapan analisis sentimen pada isu politik di Indonesia juga dilakukan oleh Hidayatullah *et al.* (2021) yang mengkaji opini publik mengenai *Indonesian Presidential Election 2019* menggunakan data percakapan Twitter [5]. Dalam penelitian tersebut, para peneliti membandingkan sejumlah pendekatan *neural network*, termasuk CNN, LSTM, CNN-LSTM, GRU-LSTM, dan *Bidirectional* LSTM. Selain itu sebagai perbandingan dengan model *deep learning*, peneliti juga melatih *dataset* menggunakan algoritma *machine learning* tradisional lainnya, yaitu *Support* SVM, *Logistic Regression* (LR), dan *Multinomial* Naïve Bayes (MNB). Ditemukan bahwa *Bidirectional* LSTM mencapai kinerja terbaik dengan akurasi 84,60%. Arsitektur sekuensial seperti BiLSTM mampu menangkap konteks dan dinamika bahasa Indonesia dalam percakapan politik digital secara lebih efektif dibandingkan model klasik, menandai keunggulan model sekuensial dalam tugas klasifikasi sentimen teks informal.

Penelitian yang dilakukan oleh Angelo *et al*. (2025) mengkaji dinamika sentimen publik terhadap kebijakan pemindahan Ibu Kota Negara (IKN), khususnya pada masa transisi kepemimpinan politik [6]. Studi ini memanfaatkan 9.451 data cuitan dari platform X yang kemudian dianalisis menggunakan berbagai metode pembelajaran mesin tradisional, seperti Naïve Bayes, SVM, AdaBoost, XGBoost, dan LightGBM, serta dibandingkan dengan model berbasis *Transformer*, yakni IndoBERT. Hasil penelitian menunjukkan bahwa IndoBERT secara konsisten mengungguli seluruh model tradisional pada metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Temuan tersebut menegaskan superioritas IndoBERT dalam

menangkap pola linguistik yang kompleks dan bernuansa dalam teks media sosial yang bersifat informal. Selain itu, hasil penelitian ini memberikan implikasi praktis bagi pembuat kebijakan, terutama dalam memahami respons publik serta merumuskan strategi komunikasi yang adaptif dalam implementasi kebijakan berskala nasional.

Selaras dengan temuan tersebut, penelitian eksperimental yang dilakukan oleh Kurniasih *et al.* (2022) menunjukkan bahwa berbagai tahapan *preprocessing* teks yang umum digunakan ternyata tidak memberikan pengaruh signifikan terhadap kinerja model ketika memanfaatkan *embedding* BERT dan arsitektur *deep learning* yang tepat. Studi tersebut menguji berbagai model, termasuk LSTM, BiLSTM, CNN, GRU, dan MLP, serta menemukan bahwa performa model seperti BiLSTM relatif stabil baik dengan maupun tanpa *preprocessing*. Bahkan, kombinasi *embedding* BERT dengan model *deep learning* menghasilkan performa klasifikasi terbaik [80]. Temuan ini mengindikasikan bahwa model berbasis *Transformer* telah mampu memahami konteks linguistik pada teks informal tanpa memerlukan *preprocessing* yang agresif, sehingga mendukung pemilihan pendekatan *preprocessing* ringan dalam penelitian ini.

Efektivitas pendekatan hibrida berbasis *pretrained Transformer* dengan model sekuensial juga diperkuat oleh studi yang menguji berbagai kombinasi model hibrida yang memadukan representasi kontekstual IndoBERT dengan jaringan sekuensial seperti BiLSTM dan *variants* lain dalam tugas klasifikasi emosi dan sentimen [7]. Salah satu konfigurasi yang diuji adalah hibrida IndoBERT-BiLSTM, dan terbukti menghasilkan akurasi klasifikasi yang baik pada *dataset* bahasa Indonesia yang beragam. Meskipun penelitian ini tidak secara eksplisit mengkaji domain politik atau kebijakan publik, konteks data yang digunakan berupa teks autentik dari pengguna yang memiliki variasi ekspresi, struktur kalimat tidak baku, dan nuansa bahasa spontan menunjukkan kemampuan model hibrida untuk menangkap pola linguistik yang kompleks pada teks yang tidak terstruktur. Keberhasilan pendekatan *hybrid* pada *sentiment analysis* ini memberikan dasar empiris yang kuat untuk menerapkan model hibrida serupa pada domain politik dan kebijakan publik, terutama mengingat belum adanya studi yang secara eksplisit menguji arsitektur IndoBERT-BiLSTM terhadap fenomena politik kontemporer seperti "17+8 Tuntutan Rakyat". Penelitian ini turut memperluas aplikasi model hibrida ke ranah politik dan memberikan kontribusi metodologis dalam memahami sentimen publik terhadap isu kebijakan yang kompleks dan dinamis.