BAB 2
LANDASAN TEORI

2.1 Media Sosial X

Media sosial merupakan sarana digital yang memungkinkan pengguna untuk membuat, membagikan, serta bertukar informasi secara daring dalam berbagai bentuk, seperti teks, gambar, maupun konten multimedia. Kehadiran media sosial telah mengubah pola komunikasi masyarakat karena informasi dapat disebarkan dengan lebih cepat dan melibatkan interaksi yang lebih luas dibandingkan media konvensional [2]. Salah satu *platform* yang banyak dimanfaatkan sebagai ruang diskusi publik adalah X, yang sebelumnya dikenal dengan nama Twitter.

Sebagai *platform* berbasis *microblogging*, X memungkinkan pengguna menyampaikan informasi, opini, maupun tanggapan secara *real-time* melalui unggahan teks singkat [3]. Sifat *platform* yang terbuka membuat berbagai isu sosial, politik, dan kebijakan pemerintah dapat dengan mudah menjadi bahan diskusi publik [1]. Fitur-fitur seperti *repost*, *hashtag*, dan *trending topic* juga berperan dalam mempercepat penyebaran informasi sehingga suatu topik dapat menjangkau audiens yang lebih luas dalam waktu singkat.

X banyak digunakan sebagai sumber data dalam penelitian yang berfokus pada opini publik karena pengguna secara aktif membagikan pandangan, kritik, maupun dukungan terhadap berbagai isu yang sedang berkembang [11]. Berdasarkan data DemandSage dan DataReportal, Indonesia termasuk negara dengan jumlah pengguna X yang cukup besar, yaitu sekitar 25,2 juta pengguna pada tahun 2025 [4, 5]. Kondisi tersebut menunjukkan bahwa data yang tersedia pada *platform* X memiliki potensi untuk dimanfaatkan dalam penelitian yang berkaitan dengan analisis opini masyarakat.

Di sisi lain, data yang diperoleh dari media sosial memiliki karakteristik yang berbeda dengan dokumen formal. Pengguna sering menggunakan singkatan, *slang*, ironi, maupun ungkapan yang bersifat ambigu sehingga proses analisis menjadi lebih menantang [7]. Selain tidak terstruktur, jumlah data yang sangat besar juga menyulitkan proses analisis apabila dilakukan secara manual. Oleh sebab itu, diperlukan pendekatan komputasional melalui NLP untuk mengolah dan menganalisis data teks secara lebih efektif.

2.2 Natural Language Processing

NLP merupakan salah satu bidang dalam *Artificial Intelligence* yang berfokus pada pengolahan dan pemahaman bahasa manusia oleh komputer [6]. Pengembangan NLP melibatkan berbagai disiplin ilmu, termasuk linguistik, *machine learning*, dan *computer science*, sehingga komputer dapat memproses informasi yang disampaikan dalam bentuk teks maupun suara [8].

Seiring berkembangnya teknologi komputasi, NLP semakin banyak dimanfaatkan untuk mengolah data teks secara otomatis dalam jumlah besar [9]. Penerapannya dapat ditemukan pada berbagai tugas, seperti analisis sentimen, *machine translation*, *chatbot*, klasifikasi dokumen, *information retrieval*, dan *topic modeling* [2]. Pada penelitian media sosial, NLP berperan dalam membantu identifikasi pola, opini, maupun informasi yang terkandung dalam data teks yang dihasilkan pengguna.

Teks yang berasal dari media sosial umumnya memiliki karakteristik yang berbeda dibandingkan dokumen formal. Penggunaan bahasa tidak baku, singkatan, *slang*, hingga ungkapan yang bersifat ambigu menjadi tantangan tersendiri dalam proses analisis [7]. Oleh sebab itu, data biasanya perlu melalui beberapa tahapan pemrosesan sebelum digunakan pada tahap analisis. Tahapan tersebut meliputi *text preprocessing* seperti *case folding*, *cleansing*, *tokenization*, *normalization*, *stemming*, dan *stopword removal* [12].

Kemajuan metode berbasis *deep learning* turut mendorong perkembangan NLP modern dengan kemampuan pemahaman konteks yang lebih baik dibandingkan pendekatan sebelumnya [18]. Berbagai penelitian menunjukkan bahwa model *deep learning* mampu memberikan peningkatan performa pada beragam tugas NLP, terutama ketika diterapkan pada data media sosial yang memiliki tingkat kompleksitas bahasa yang tinggi [17].

2.3 Analisis Sentimen

Analisis sentimen merupakan salah satu bidang kajian dalam NLP yang berfokus pada identifikasi opini, emosi, maupun persepsi yang terkandung dalam suatu teks [2]. Melalui proses ini, teks dapat diklasifikasikan berdasarkan polaritas sentimennya ke dalam kategori positif, negatif, atau netral sesuai dengan konteks yang disampaikan [8].

Pemanfaatan analisis sentimen telah meluas pada berbagai sektor, mulai dari

bisnis, pemasaran, pendidikan, kesehatan, hingga evaluasi kebijakan publik [6]. Dalam konteks media sosial, analisis sentimen sering digunakan untuk mengkaji tanggapan masyarakat terhadap produk, layanan, tokoh publik, maupun kebijakan pemerintah [24]. Informasi yang diperoleh dari hasil analisis tersebut dapat digunakan sebagai bahan pertimbangan dalam proses evaluasi dan pengambilan keputusan.

Secara umum, pendekatan yang digunakan dalam analisis sentimen dapat dikelompokkan menjadi tiga kategori, yaitu *lexicon-based approach*, *machine learning approach*, dan *deep learning approach* [9]. Pendekatan berbasis leksikon menentukan polaritas sentimen berdasarkan kumpulan kata yang telah memiliki nilai atau bobot sentimen tertentu. Pendekatan ini relatif mudah diterapkan, tetapi kemampuannya dalam memahami konteks bahasa yang kompleks masih terbatas [17].

Pendekatan *machine learning* memanfaatkan algoritma klasifikasi untuk mempelajari pola sentimen dari data latih. Algoritma yang umum digunakan antara lain Naive Bayes dan Support Vector Machine (SVM) [11, 12]. Meskipun mampu menghasilkan performa klasifikasi yang cukup baik, pendekatan ini masih menghadapi kesulitan dalam menangkap hubungan kontekstual antarkata, terutama pada teks media sosial yang banyak menggunakan bahasa informal dan memiliki makna ambigu [17].

Perkembangan metode *deep learning* mendorong penggunaan model berbasis *Transformer* dalam analisis sentimen. Salah satu model yang banyak digunakan adalah BERT yang memiliki kemampuan memahami konteks kata secara dua arah dalam sebuah kalimat [18]. Berbagai penelitian melaporkan bahwa model berbasis BERT mampu memberikan performa yang lebih baik dibandingkan pendekatan *machine learning* konvensional pada tugas klasifikasi sentimen [25]. Untuk data berbahasa Indonesia, IndoBERT juga telah menunjukkan hasil yang efektif dalam melakukan klasifikasi sentimen pada berbagai data media sosial [20, 21].

Machine learning merupakan salah satu cabang *Artificial Intelligence* yang memungkinkan komputer mempelajari pola dari data dan memanfaatkannya untuk melakukan prediksi maupun klasifikasi tanpa memerlukan aturan yang dirancang secara eksplisit [8]. Dalam pengolahan teks, pendekatan ini digunakan untuk mengenali pola bahasa yang diperoleh dari data latih sehingga model dapat memberikan prediksi terhadap data yang belum pernah ditemui sebelumnya.

Berbagai algoritma *machine learning* telah diterapkan dalam penelitian

analisis sentimen, di antaranya Naive Bayes dan Support Vector Machine (SVM) [11, 12]. Naive Bayes merupakan metode klasifikasi probabilistik yang didasarkan pada Teorema Bayes dengan asumsi independensi antar fitur [12]. Sementara itu, SVM melakukan proses klasifikasi dengan membentuk *hyperplane* yang mampu memisahkan data ke dalam kelas yang berbeda [11]. Kedua algoritma tersebut banyak digunakan karena relatif efisien dan mampu memberikan hasil klasifikasi yang cukup baik pada berbagai kasus analisis teks.

Meskipun demikian, performa metode *machine learning* tradisional umumnya masih dipengaruhi oleh representasi fitur yang digunakan. Pendekatan seperti *bag-of-words* dan TF-IDF mampu merepresentasikan kemunculan kata dalam dokumen, tetapi belum sepenuhnya menangkap hubungan kontekstual maupun makna semantik antar kata [17]. Kondisi ini menjadi tantangan ketika model diterapkan pada data media sosial yang banyak mengandung bahasa informal, singkatan, serta variasi penggunaan kata.

Perkembangan *deep learning* memberikan pendekatan yang berbeda melalui pemanfaatan *Artificial Neural Network* untuk mempelajari representasi data secara otomatis dari proses pelatihan [25]. Kemampuan tersebut memungkinkan model menangkap pola yang lebih kompleks dibandingkan metode *machine learning* konvensional, termasuk hubungan kontekstual yang terdapat pada data bahasa.

Dalam bidang NLP, kemajuan *deep learning* semakin terlihat dengan hadirnya arsitektur *Transformer* yang menjadi fondasi bagi berbagai model bahasa modern, termasuk BERT [18]. Arsitektur ini dirancang untuk memahami hubungan antar kata berdasarkan konteks kalimat secara lebih efektif. Berbagai penelitian menunjukkan bahwa model berbasis *Transformer* mampu meningkatkan performa pada berbagai tugas NLP, termasuk analisis sentimen pada data media sosial [10].

2.4 Transformer

Transformer merupakan arsitektur *deep learning* yang diperkenalkan oleh Vaswani *et al.* [26] sebagai pendekatan baru dalam pemrosesan data sekuens tanpa menggunakan *Recurrent Neural Network* (RNN) maupun *Convolutional Neural Network* (CNN). Berbeda dengan arsitektur sebelumnya yang memproses token secara berurutan, *Transformer* memanfaatkan mekanisme *self-attention* untuk mempelajari hubungan antar token secara paralel sehingga mampu meningkatkan efisiensi komputasi serta menangkap ketergantungan jarak jauh dalam suatu kalimat [27].

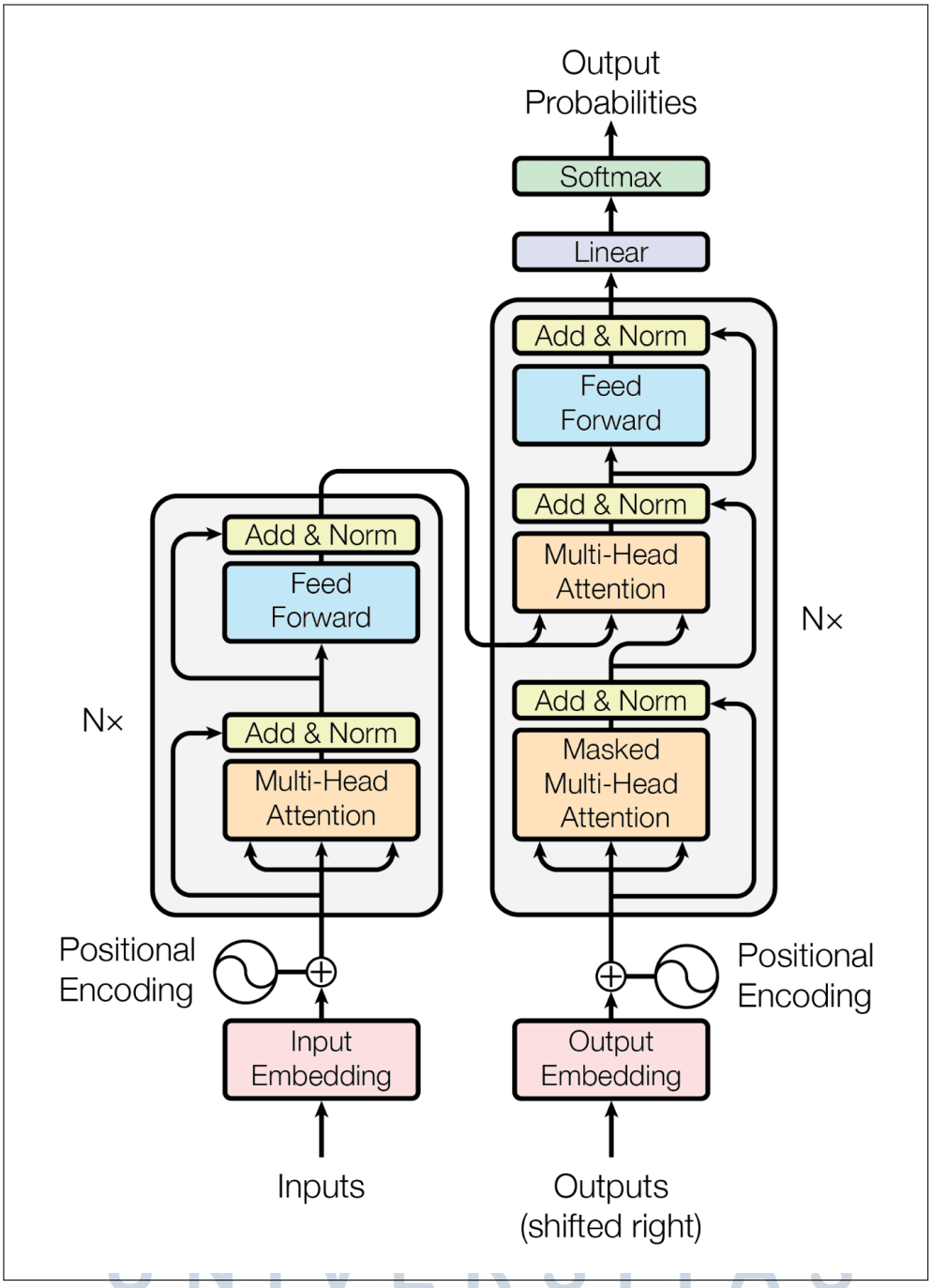
Kemampuan tersebut menjadikan *Transformer* sebagai fondasi bagi berbagai model bahasa modern, termasuk BERT, GPT, T5, dan IndoBERT. Pada penelitian ini, model IndoBERT yang digunakan dibangun berdasarkan arsitektur encoder dari *Transformer*, sehingga pemahaman terhadap mekanisme *Transformer* menjadi dasar dalam memahami proses analisis sentimen yang dilakukan [18, **?**].

Secara umum, arsitektur *Transformer* terdiri atas dua komponen utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas mengubah urutan token masukan menjadi representasi kontekstual, sedangkan *decoder* menghasilkan token keluaran berdasarkan representasi yang diperoleh dari *encoder*. Setiap *encoder* maupun *decoder* tersusun atas beberapa lapisan yang identik, di mana setiap lapisan memanfaatkan mekanisme *Multi-Head Self-Attention* dan *Feed Forward Neural Network*. Selain itu, setiap sublapisan menggunakan *Residual Connection* dan *Layer Normalization* untuk menjaga stabilitas proses pelatihan model [26, 27].

Meskipun arsitektur asli *Transformer* terdiri atas *encoder* dan *decoder*, BERT hanya memanfaatkan bagian *encoder*. Pemanfaatan *encoder* memungkinkan model menghasilkan representasi kontekstual dua arah (*bidirectional contextual representation*) sehingga setiap token dipahami berdasarkan konteks sebelum maupun sesudah token tersebut [18].

2.4.1 Arsitektur Transformer

Arsitektur *Transformer* terdiri atas dua komponen utama, yaitu *encoder* dan *decoder*. Kedua komponen tersebut disusun secara bertingkat (*stacked*) dan bekerja secara berurutan dalam memproses data sekuens. *Encoder* bertugas mengubah urutan token masukan menjadi representasi kontekstual, sedangkan *decoder* memanfaatkan representasi tersebut untuk menghasilkan urutan token keluaran. Berbeda dengan arsitektur berbasis RNN yang memproses token secara berurutan, *Transformer* memproses seluruh token secara paralel menggunakan mekanisme *attention*, sehingga mampu meningkatkan efisiensi komputasi serta menangkap hubungan antartoken secara lebih efektif [26, 27].



Gambar 2.1. Arsitektur *Transformer*

Sumber: [26]

Berdasarkan Gambar 2.1, bagian sebelah kiri merupakan *encoder*, sedangkan bagian sebelah kanan merupakan *decoder*. Pada penelitian asli, *Transformer* menggunakan enam lapisan (*layers*) *encoder* dan enam lapisan *decoder* yang disusun secara identik. Setiap lapisan pada *encoder* terdiri atas dua

sublapisan utama, yaitu mekanisme *Multi-Head Self-Attention* dan *Feed Forward Neural Network*. Adapun setiap lapisan pada *decoder* memiliki satu sublapisan tambahan berupa *Masked Multi-Head Attention* yang berfungsi untuk mencegah model mengakses informasi dari token berikutnya selama proses pembangkitan keluaran [26].

Selain kedua sublapisan tersebut, setiap lapisan pada *encoder* maupun *decoder* dilengkapi dengan *Residual Connection* dan *Layer Normalization*. Kedua mekanisme tersebut berperan dalam menjaga stabilitas proses pelatihan, mempercepat konvergensi model, serta membantu mempertahankan informasi dari lapisan sebelumnya sehingga model dapat dilatih dengan jumlah lapisan yang lebih dalam [27].

Sebelum diproses oleh *encoder*, setiap token terlebih dahulu diubah menjadi representasi numerik melalui proses *embedding*. Karena *Transformer* tidak memproses token secara berurutan, informasi mengenai posisi setiap token ditambahkan menggunakan *Positional Encoding*. Representasi hasil penjumlahan antara *embedding* dan *Positional Encoding* kemudian menjadi masukan bagi lapisan pertama pada *encoder*. Selanjutnya, keluaran dari *encoder* diteruskan ke *decoder* sebagai informasi kontekstual untuk menghasilkan token keluaran [26].

Meskipun arsitektur asli *Transformer* terdiri atas *encoder* dan *decoder*, tidak semua model bahasa memanfaatkan kedua komponen tersebut. Sebagai contoh, BERT hanya menggunakan bagian *encoder* untuk menghasilkan representasi kontekstual dua arah, sedangkan GPT hanya menggunakan bagian *decoder* untuk menghasilkan teks secara autoregresif. Oleh karena itu, komponen yang menjadi dasar berbagai model bahasa modern adalah mekanisme *attention* yang terdapat pada arsitektur *Transformer*, khususnya pada bagian *encoder* [18].

Setelah memahami arsitektur *Transformer* secara umum, pembahasan selanjutnya difokuskan pada mekanisme *Self-Attention* sebagai komponen utama yang memungkinkan setiap token mempelajari hubungan dengan token lainnya dalam suatu urutan masukan.

2.4.2 Mekanisme *Self-Attention*

Mekanisme *Self-Attention* merupakan komponen utama pada arsitektur *Transformer* yang memungkinkan setiap token dalam suatu urutan memperhatikan token lainnya untuk membentuk representasi yang lebih kontekstual. Berbeda dengan RNN yang memproses token secara berurutan, *Self-Attention* memproses

seluruh token secara bersamaan sehingga hubungan antar token dapat dihitung secara paralel. Pendekatan ini tidak hanya meningkatkan efisiensi komputasi, tetapi juga memungkinkan model menangkap hubungan antar kata yang letaknya berjauhan dalam suatu kalimat [26, 27].

Secara umum, mekanisme *Self-Attention* dimulai dengan mengubah setiap token masukan menjadi representasi numerik melalui proses *embedding*. Representasi tersebut kemudian diproyeksikan ke dalam tiga buah vektor yang berbeda, yaitu *Query* (Q), *Key* (K), dan *Value* (V). Ketiga vektor tersebut memiliki fungsi yang berbeda dalam proses perhitungan *attention*. Vektor *Query* digunakan untuk menentukan informasi yang ingin dicari oleh suatu token, vektor *Key* digunakan sebagai acuan dalam mengukur tingkat keterkaitan antar token, sedangkan vektor *Value* menyimpan informasi yang akan dikombinasikan untuk menghasilkan representasi akhir setiap token [26].

Pembentukan vektor *Query*, *Key*, dan *Value* dilakukan melalui transformasi linear terhadap representasi masukan menggunakan parameter yang dipelajari selama proses pelatihan. Rumus pembentukan ketiga vektor tersebut ditunjukkan pada Rumus 2.1.

$$Q = XW^Q, \qquad K = XW^K, \qquad V = XW^V \tag{2.1}$$

Pada Rumus 2.1, X merupakan matriks representasi token hasil proses *embedding*. Matriks W^Q, W^K, dan W^V merupakan parameter transformasi linear yang dipelajari selama proses pelatihan model untuk menghasilkan matriks *Query*, *Key*, dan *Value*. Meskipun ketiga matriks tersebut berasal dari representasi masukan yang sama, masing-masing memiliki fungsi yang berbeda sehingga model dapat menentukan hubungan antar token secara lebih efektif [27].

Sebagai ilustrasi, misalkan terdapat kalimat ”Program makan bergizi gratis mendapat dukungan masyarakat”. Ketika model memproses token ”*dukungan*”, token tersebut tidak hanya direpresentasikan sebagai sebuah *embedding*. Representasi tersebut terlebih dahulu diproyeksikan menjadi vektor *Query*, *Key*, dan *Value*. Vektor *Query* kemudian dibandingkan dengan seluruh vektor *Key* milik token lain untuk mengetahui token mana yang memiliki hubungan paling relevan. Selanjutnya, informasi pada setiap vektor *Value* akan digabungkan berdasarkan tingkat perhatian yang diperoleh sehingga dihasilkan representasi baru yang telah mempertimbangkan konteks seluruh kalimat [26, 27].

Melalui mekanisme tersebut, setiap token dapat memanfaatkan informasi dari seluruh token lain tanpa bergantung pada posisi token di dalam kalimat. Oleh karena itu, representasi yang dihasilkan oleh mekanisme *Self-Attention* mampu menggambarkan makna suatu token berdasarkan konteks keseluruhan kalimat, bukan hanya berdasarkan token yang berada di sekitarnya. Setelah vektor *Query*, *Key*, dan *Value* diperoleh, langkah selanjutnya adalah menghitung tingkat keterkaitan antar token menggunakan mekanisme *Scaled Dot-Product Attention*.

2.4.3 *Scaled Dot-Product Attention*

Setelah vektor *Query* (Q), *Key* (K), dan *Value* (V) diperoleh, langkah selanjutnya adalah menghitung tingkat keterkaitan antar token menggunakan mekanisme *Scaled Dot-Product Attention*. Mekanisme ini bertujuan untuk menentukan seberapa besar perhatian (*attention*) yang harus diberikan suatu token terhadap token lainnya. Semakin tinggi tingkat keterkaitan antara dua token, semakin besar pula kontribusi informasi yang diberikan pada proses pembentukan representasi kontekstual [26, 27].

Proses perhitungan diawali dengan melakukan operasi *dot product* antara matriks *Query* dan transpos matriks *Key*. Hasil operasi tersebut menghasilkan nilai kesamaan (*similarity score*) yang menunjukkan tingkat hubungan antara setiap pasangan token dalam suatu urutan masukan. Nilai kesamaan yang lebih besar menunjukkan bahwa kedua token memiliki hubungan yang lebih kuat.

Namun, ketika dimensi vektor *Key* semakin besar, nilai hasil *dot product* juga cenderung meningkat sehingga fungsi *Softmax* dapat menghasilkan distribusi probabilitas yang terlalu tajam. Kondisi ini menyebabkan proses pelatihan menjadi kurang stabil karena gradien yang dihasilkan menjadi sangat kecil. Oleh karena itu, nilai hasil *dot product* dibagi dengan akar kuadrat dari dimensi vektor *Key* ($\sqrt{d_k}$) sebelum diterapkan fungsi *Softmax*. Proses tersebut dikenal sebagai mekanisme *scaling* [26].

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Rumus *Scaled Dot-Product Attention* ditunjukkan pada Rumus 2.2. Pada Rumus 2.2, Q merupakan matriks *Query*, K merupakan matriks *Key*, sedangkan V merupakan matriks *Value*. Notasi K^T menyatakan transpos dari matriks *Key*, sementara d_k merupakan dimensi dari vektor *Key*. Operasi QK^T menghasilkan

matriks nilai kesamaan antar token, kemudian hasil tersebut dibagi dengan $\sqrt{d_k}$ untuk menjaga stabilitas proses pelatihan. Selanjutnya, fungsi *Softmax* mengubah nilai kesamaan menjadi distribusi probabilitas yang menunjukkan besar perhatian yang diberikan terhadap setiap token. Distribusi probabilitas tersebut kemudian dikalikan dengan matriks *Value* untuk menghasilkan representasi baru yang telah mempertimbangkan informasi dari seluruh token dalam urutan masukan [26, 27].

Melalui mekanisme tersebut, setiap token tidak hanya mempertahankan informasi yang dimilikinya sendiri, tetapi juga menggabungkan informasi dari token lain berdasarkan tingkat keterkaitannya. Dengan demikian, representasi yang dihasilkan mampu menggambarkan makna suatu token secara lebih kontekstual dibandingkan apabila hanya mempertimbangkan token secara terpisah.

Mekanisme *Scaled Dot-Product Attention* yang telah dijelaskan sebelumnya hanya menggunakan satu proses perhatian (*attention*) untuk mempelajari hubungan antar token. Pada praktiknya, arsitektur *Transformer* mengembangkan mekanisme tersebut menjadi beberapa proses perhatian yang dijalankan secara paralel melalui *Multi-Head Attention*. Pendekatan ini memungkinkan model mempelajari berbagai jenis hubungan antar token secara bersamaan sehingga representasi yang dihasilkan menjadi lebih kaya. Pembahasan mengenai mekanisme tersebut dijelaskan pada subbab berikutnya.

2.4.4 *Multi-Head Attention*

Mekanisme *Scaled Dot-Product Attention* hanya menggunakan satu proses *attention* untuk mempelajari hubungan antar token. Pada implementasinya, satu proses *attention* belum tentu mampu menangkap seluruh informasi yang terkandung dalam suatu kalimat. Oleh karena itu, arsitektur *Transformer* mengembangkan mekanisme tersebut menjadi *Multi-Head Attention*, yaitu beberapa proses *attention* yang dijalankan secara paralel. Setiap *head* menerima masukan berupa matriks *Query*, *Key*, dan *Value* yang telah diproyeksikan menggunakan parameter yang berbeda, kemudian menghitung *Scaled Dot-Product Attention* secara independen. Hasil keluaran dari seluruh *head* selanjutnya digabungkan (*concatenate*) dan diproyeksikan kembali menggunakan transformasi linear sehingga menghasilkan satu representasi keluaran yang lebih kaya dalam menangkap berbagai hubungan antar token [26, 27].

Penggunaan beberapa *attention head* memungkinkan model mempelajari hubungan sintaksis maupun semantik secara bersamaan. Sebagai contoh, satu

head dapat lebih berfokus pada hubungan antar subjek dan predikat, sedangkan *head* lainnya dapat mempelajari hubungan antar kata yang memiliki makna serupa. Dengan demikian, setiap *head* memberikan kontribusi yang berbeda dalam membentuk representasi kontekstual suatu token sehingga informasi yang diperoleh menjadi lebih beragam dibandingkan apabila hanya menggunakan satu mekanisme *attention* [27].

$$\text{MultiHead}(Q,K,V) = \text{Concat}(head_1,\ldots,head_h)W^O \tag{2.3}$$

Rumus *Multi-Head Attention* ditunjukkan pada Rumus 2.3. Pada Rumus 2.3, $head_1,\ldots,head_h$ merupakan keluaran dari setiap *attention head*, sedangkan Concat merupakan operasi penggabungan seluruh keluaran tersebut. Matriks W^O merupakan parameter transformasi linear yang dipelajari selama proses pelatihan untuk menghasilkan representasi akhir.

Setiap *attention head* dihitung menggunakan mekanisme *Scaled Dot-Product Attention* dengan parameter transformasi yang berbeda sehingga setiap *head* mampu mempelajari karakteristik hubungan antar token yang berbeda. Proses pembentukan setiap *attention head* dirumuskan sebagai berikut.

$$head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

Rumus *Attention Head* ditunjukkan pada Rumus 2.4. Pada Rumus 2.4, W_i^Q, W_i^K, dan W_i^V merupakan parameter transformasi linear yang berbeda pada setiap *head*. Perbedaan parameter tersebut memungkinkan setiap *head* mempelajari pola hubungan antar token dari sudut pandang yang berbeda sehingga representasi yang dihasilkan menjadi lebih kaya dan mampu menangkap informasi yang lebih kompleks. Representasi keluaran dari seluruh *head* kemudian digabungkan dan diteruskan ke *Feed Forward Neural Network* untuk memperkaya representasi fitur setiap token. Pembahasan mengenai mekanisme tersebut dijelaskan pada subbab berikutnya.

2.4.5 *Feed Forward Neural Network*

Keluaran yang dihasilkan oleh mekanisme *Multi-Head Attention* selanjutnya diproses menggunakan *Feed Forward Neural Network* (FFN). Komponen ini

berfungsi untuk melakukan transformasi nonlinier terhadap representasi setiap token sehingga model mampu mempelajari pola yang lebih kompleks. Berbeda dengan mekanisme *attention* yang mempelajari hubungan antar token, FFN memproses setiap token secara independen menggunakan parameter yang sama pada seluruh posisi token. Dengan demikian, representasi setiap token dapat diperkaya tanpa dipengaruhi kembali oleh token lainnya pada tahap ini [26, 27].

Secara umum, FFN terdiri atas dua lapisan *fully connected* yang dipisahkan oleh fungsi aktivasi ReLU (*Rectified Linear Unit*). Lapisan pertama bertugas memproyeksikan representasi masukan ke dimensi yang lebih tinggi untuk meningkatkan kemampuan model dalam mempelajari pola data. Selanjutnya, fungsi aktivasi ReLU diterapkan untuk memperkenalkan sifat nonlinier, kemudian hasilnya diproyeksikan kembali ke dimensi semula melalui lapisan kedua sehingga menghasilkan representasi keluaran yang akan diteruskan ke komponen berikutnya pada arsitektur *Transformer* [26].

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.5}$$

Rumus *Feed Forward Neural Network* ditunjukkan pada Rumus 2.5. Pada Rumus 2.5, x merupakan representasi token yang dihasilkan oleh mekanisme *Multi-Head Attention*. Parameter W_1 dan W_2 merupakan matriks bobot yang dipelajari selama proses pelatihan, sedangkan b_1 dan b_2 merupakan vektor bias pada masing-masing lapisan. Fungsi $\max(0, \cdot)$ menunjukkan fungsi aktivasi ReLU yang mempertahankan nilai positif dan mengubah nilai negatif menjadi nol.

Melalui proses tersebut, FFN mampu menghasilkan representasi fitur yang lebih informatif sehingga setiap token tidak hanya membawa informasi hasil mekanisme *attention*, tetapi juga memperoleh transformasi nonlinier yang membantu model mempelajari karakteristik bahasa secara lebih baik. Representasi yang telah diperkaya tersebut kemudian diteruskan menuju proses *Residual Connection* dan *Layer Normalization* untuk menjaga stabilitas pelatihan model.

2.4.6 *Residual Connection* dan *Layer Normalization*

Setiap sublapisan pada arsitektur *Transformer*, baik *Multi-Head Attention* maupun *Feed Forward Neural Network*, dilengkapi dengan mekanisme *Residual Connection* yang diikuti oleh *Layer Normalization*. Kombinasi kedua mekanisme tersebut dikenal sebagai *Add & Norm* pada arsitektur *Transformer*. Tujuannya

adalah untuk menjaga stabilitas proses pelatihan, mempertahankan informasi dari lapisan sebelumnya, serta membantu mengurangi permasalahan *vanishing gradient* ketika model memiliki banyak lapisan [26, 27].

Mekanisme *Residual Connection* bekerja dengan menjumlahkan representasi masukan dengan keluaran dari suatu sublapisan. Pendekatan ini memungkinkan informasi awal tetap dipertahankan sehingga setiap lapisan hanya perlu mempelajari perubahan atau informasi tambahan yang diperlukan. Dengan demikian, proses propagasi gradien selama pelatihan menjadi lebih stabil dan model dapat dilatih dengan jumlah lapisan yang lebih banyak tanpa mengalami penurunan performa yang signifikan.

$$y = x + \text{Sublayer}(x) \tag{2.6}$$

Rumus *Residual Connection* ditunjukkan pada Rumus 2.6. Pada Rumus 2.6, x merupakan representasi masukan pada suatu sublapisan, sedangkan $\text{Sublayer}(x)$ merupakan keluaran yang dihasilkan oleh sublapisan tersebut, seperti *Multi-Head Attention* atau *Feed Forward Neural Network*. Hasil penjumlahan keduanya menghasilkan representasi baru yang tetap mempertahankan informasi dari masukan sebelumnya.

Setelah proses penjumlahan dilakukan, representasi tersebut dinormalisasi menggunakan *Layer Normalization*. Mekanisme ini bertujuan untuk menjaga distribusi nilai aktivasi agar tetap stabil selama proses pelatihan sehingga model dapat mencapai konvergensi dengan lebih cepat. Selain itu, *Layer Normalization* juga membantu mengurangi perubahan distribusi data pada setiap lapisan yang dapat memengaruhi proses pembelajaran model [27].

$$\text{LayerNorm}(x) = \gamma \left(\frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}} \right) + \beta \tag{2.7}$$

Rumus *Layer Normalization* ditunjukkan pada Rumus 2.7. Pada Rumus 2.7, μ menyatakan nilai rata-rata dari fitur masukan, sedangkan σ^2 merupakan variansnya. Nilai ε adalah konstanta bernilai sangat kecil yang digunakan untuk menghindari pembagian dengan nol, sementara γ dan β merupakan parameter yang dipelajari selama proses pelatihan untuk mengatur hasil normalisasi.

Pada arsitektur *Transformer*, kedua mekanisme tersebut selalu diterapkan secara berurutan setelah setiap sublapisan sehingga keluaran akhirnya diperoleh

melalui proses *Add & Norm*. Proses tersebut dirumuskan sebagai berikut.

$$\text{LayerNorm}(x + \text{Sublayer}(x)) \tag{2.8}$$

Rumus *Add & Norm* ditunjukkan pada Rumus 2.8. Pada Rumus 2.8, hasil keluaran suatu sublapisan terlebih dahulu dijumlahkan dengan representasi masukannya melalui mekanisme *Residual Connection*. Selanjutnya, hasil penjumlahan tersebut dinormalisasi menggunakan *Layer Normalization* sehingga menghasilkan representasi yang lebih stabil untuk diteruskan ke lapisan berikutnya.

Kombinasi *Residual Connection* dan *Layer Normalization* menjadi salah satu komponen penting yang mendukung keberhasilan arsitektur *Transformer*. Setelah seluruh token diproses melalui mekanisme tersebut, model masih memerlukan informasi mengenai urutan token dalam kalimat. Oleh karena itu, arsitektur *Transformer* menggunakan *Positional Encoding* untuk merepresentasikan informasi posisi setiap token yang akan dibahas pada subbab berikutnya.

2.4.7 *Positional Encoding*

Salah satu perbedaan utama antara arsitektur *Transformer* dan model berbasis RNN adalah cara keduanya memproses urutan token. RNN memproses token secara berurutan sehingga informasi mengenai posisi token secara alami dipertahankan selama proses komputasi. Sebaliknya, *Transformer* memproses seluruh token secara paralel menggunakan mekanisme *attention*. Meskipun pendekatan tersebut meningkatkan efisiensi komputasi, model tidak lagi memiliki informasi mengenai urutan token dalam suatu kalimat. Oleh karena itu, diperlukan mekanisme yang mampu memberikan informasi posisi kepada setiap token sebelum diproses lebih lanjut [26, 27].

Mekanisme tersebut dikenal sebagai *Positional Encoding*, yaitu representasi posisi yang ditambahkan ke vektor *embedding* setiap token. Dengan adanya informasi posisi, model tidak hanya mengetahui hubungan antar token, tetapi juga dapat membedakan urutan kemunculan setiap token dalam suatu kalimat. Hal ini penting karena perubahan urutan kata dapat mengubah makna suatu kalimat meskipun token yang digunakan tetap sama [26].

Pada arsitektur *Transformer*, *Positional Encoding* dibentuk menggunakan fungsi sinus dan cosinus dengan frekuensi yang berbeda pada setiap dimensi *embedding*. Penggunaan kedua fungsi tersebut memungkinkan model

merepresentasikan informasi posisi setiap token sehingga hubungan posisi dalam suatu urutan tetap dapat dipertahankan meskipun seluruh token diproses secara paralel [26, 27].

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.9}$$

Rumus *Positional Encoding* menggunakan fungsi sinus ditunjukkan pada Rumus 2.9. Pada Rumus 2.9, *pos* menyatakan posisi token dalam urutan masukan, i merupakan indeks dimensi *embedding*, sedangkan d_{model} merupakan dimensi vektor *embedding* yang digunakan oleh model.

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.10}$$

Rumus *Positional Encoding* menggunakan fungsi cosinus ditunjukkan pada Rumus 2.10. Pada Rumus 2.10, parameter yang digunakan memiliki arti yang sama dengan Rumus 2.9. Perbedaannya terletak pada penggunaan fungsi cosinus pada dimensi ganjil, sedangkan fungsi sinus digunakan pada dimensi genap. Kombinasi kedua fungsi tersebut menghasilkan representasi posisi yang unik untuk setiap token sehingga model dapat membedakan urutan token tanpa memerlukan mekanisme pemrosesan secara berurutan.

Representasi posisi yang dihasilkan kemudian dijumlahkan dengan vektor *embedding* sebelum diteruskan ke lapisan pertama pada *encoder*. Dengan demikian, setiap token tidak hanya membawa informasi mengenai makna kata, tetapi juga informasi mengenai posisinya dalam suatu kalimat. Informasi tersebut selanjutnya diproses oleh mekanisme *Self-Attention* sehingga model mampu menghasilkan representasi kontekstual yang mempertimbangkan baik hubungan semantik maupun urutan token [26].

Berdasarkan uraian mengenai arsitektur *Transformer*, dapat diketahui bahwa keberhasilan model ini didukung oleh beberapa komponen utama, yaitu *Self-Attention*, *Multi-Head Attention*, *Feed Forward Neural Network*, *Residual Connection*, *Layer Normalization*, dan *Positional Encoding*. Kombinasi seluruh komponen tersebut memungkinkan *Transformer* menghasilkan representasi bahasa yang lebih kontekstual dan efisien dibandingkan pendekatan sebelumnya. Arsitektur inilah yang kemudian menjadi dasar pengembangan berbagai model

bahasa modern, termasuk BERT yang akan dibahas pada subbab berikutnya.

2.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) merupakan model bahasa berbasis arsitektur *Transformer* yang diperkenalkan oleh Devlin *et al.* [18]. Berbeda dengan model bahasa sebelumnya yang umumnya memproses teks secara satu arah (*unidirectional*), BERT menerapkan pendekatan dua arah (*bidirectional*) sehingga representasi setiap token dibentuk dengan mempertimbangkan konteks sebelum maupun sesudah token tersebut. Pendekatan ini memungkinkan model memahami hubungan antarkata secara lebih menyeluruh sehingga menghasilkan representasi bahasa yang lebih kontekstual [18].

BERT dibangun menggunakan bagian *encoder* dari arsitektur *Transformer* yang telah dibahas pada subbab sebelumnya. Berbeda dengan *Transformer* asli yang terdiri atas *encoder* dan *decoder*, BERT hanya memanfaatkan susunan *encoder* bertingkat untuk menghasilkan representasi kontekstual dari setiap token. Representasi tersebut kemudian dapat dimanfaatkan pada berbagai tugas *Natural Language Processing* (NLP), seperti klasifikasi teks, *named entity recognition*, *question answering*, maupun analisis sentimen [18, 25].

Sebelum digunakan pada suatu tugas tertentu, BERT terlebih dahulu menjalani proses *pre-training* menggunakan korpus teks dalam jumlah yang sangat besar. Pada tahap ini, model mempelajari hubungan antarkata dan antarkalimat melalui dua tugas utama, yaitu *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP). Setelah proses *pre-training* selesai, model dapat disesuaikan (*fine-tuning*) untuk berbagai tugas NLP dengan menambahkan lapisan keluaran sesuai kebutuhan tanpa mengubah arsitektur dasar BERT [18].

Keberhasilan BERT dalam menghasilkan representasi bahasa yang kontekstual mendorong pengembangan berbagai model turunannya yang disesuaikan dengan karakteristik bahasa tertentu, seperti RoBERTa, ALBERT, DistilBERT, maupun IndoBERT untuk bahasa Indonesia. Meskipun memiliki tujuan pengembangan yang berbeda, seluruh model tersebut tetap mempertahankan prinsip dasar arsitektur BERT sebagai fondasi utama dalam proses pembelajaran representasi bahasa [18, 19].

2.5.1 Arsitektur BERT

BERT dibangun berdasarkan arsitektur *Transformer* dengan hanya memanfaatkan bagian *encoder*. Berbeda dengan arsitektur *Transformer* asli yang terdiri atas *encoder* dan *decoder*, BERT tidak menggunakan *decoder* karena model ini tidak dirancang untuk menghasilkan urutan token baru, melainkan untuk membentuk representasi kontekstual dari setiap token pada teks masukan. Pendekatan tersebut memungkinkan setiap token memperoleh informasi dari seluruh token lain dalam satu kalimat sehingga hubungan semantik antar kata dapat dipelajari secara lebih menyeluruh [18].

Pada BERT, setiap token masukan terlebih dahulu diubah menjadi representasi numerik melalui proses *embedding*. Representasi tersebut kemudian diproses secara bertingkat oleh beberapa lapisan *Transformer Encoder*. Pada setiap lapisan, mekanisme *Multi-Head Self-Attention* digunakan untuk mempelajari hubungan antar token, sedangkan *Feed Forward Neural Network* melakukan transformasi nonlinier terhadap representasi yang dihasilkan. Selain itu, setiap sublapisan dilengkapi dengan mekanisme *Residual Connection* dan *Layer Normalization* sebagaimana telah dijelaskan pada Subbab 2.4.

Tidak seperti model bahasa satu arah yang hanya memanfaatkan konteks sebelum atau sesudah suatu token, BERT menggunakan mekanisme *bidirectional encoding*. Melalui pendekatan tersebut, setiap token dipelajari dengan mempertimbangkan seluruh konteks dalam kalimat secara bersamaan. Sebagai contoh, makna suatu kata yang memiliki lebih dari satu arti dapat ditentukan berdasarkan kata-kata yang muncul sebelum maupun sesudahnya. Kemampuan ini memungkinkan BERT menghasilkan representasi bahasa yang lebih kontekstual dibandingkan model yang hanya memanfaatkan konteks satu arah [18].

BERT tersedia dalam dua konfigurasi utama, yaitu BERT Base dan BERT Large. BERT Base terdiri atas 12 lapisan *Transformer Encoder*, 12 *attention heads*, dimensi *hidden state* sebesar 768, serta sekitar 110 juta parameter. Sementara itu, BERT Large memiliki 24 lapisan *Transformer Encoder*, 16 *attention heads*, dimensi *hidden state* sebesar 1024, dan sekitar 340 juta parameter. Penambahan jumlah lapisan dan parameter memungkinkan model mempelajari representasi bahasa yang lebih kompleks, meskipun membutuhkan sumber daya komputasi yang lebih besar selama proses pelatihan maupun inferensi [18].

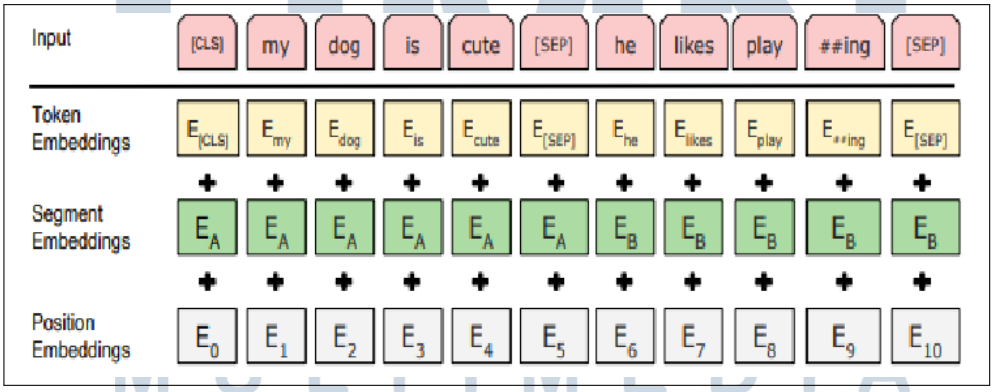
Selain arsitekturnya yang hanya memanfaatkan *encoder*, BERT juga memiliki representasi masukan yang menggabungkan informasi token, segmen

kalimat, dan posisi token sebelum diproses oleh setiap lapisan *Transformer Encoder*. Representasi masukan tersebut menjadi salah satu karakteristik penting yang membedakan BERT dari model bahasa sebelumnya.

2.5.2 *Input Representation*

Sebelum diproses oleh lapisan *Transformer Encoder*, setiap teks masukan pada BERT terlebih dahulu diubah menjadi representasi numerik yang disebut sebagai *Input Representation*. Berbeda dengan *embedding* pada model bahasa konvensional yang hanya merepresentasikan identitas kata, BERT membentuk representasi masukan dengan menggabungkan informasi token, segmen kalimat, dan posisi token dalam suatu urutan. Penggabungan ketiga komponen tersebut memungkinkan model memahami tidak hanya makna setiap token, tetapi juga urutan kemunculan token serta hubungan antar kalimat yang menjadi masukan [18].

Suatu masukan pada BERT dapat terdiri atas satu kalimat maupun sepasang kalimat. Untuk membedakan kedua kondisi tersebut, BERT menggunakan beberapa token khusus, yaitu *[CLS]* dan *[SEP]*. Token *[CLS]* ditempatkan pada awal urutan token dan berfungsi sebagai representasi keseluruhan masukan yang umumnya digunakan pada tugas klasifikasi. Sementara itu, token *[SEP]* digunakan sebagai penanda akhir kalimat atau sebagai pemisah antara dua kalimat yang diproses secara bersamaan. Dengan adanya token khusus tersebut, model dapat mengenali struktur masukan sebelum diproses oleh lapisan *Transformer Encoder* [18].



Gambar 2.2. *Input Representation* pada BERT
Sumber: [18]

Gambar 2.2 menunjukkan bahwa *Input Representation* pada BERT dibentuk melalui penjumlahan tiga komponen utama, yaitu *Token Embedding*,

Segment Embedding, dan *Position Embedding*. Selain itu, gambar tersebut juga memperlihatkan penggunaan token khusus *[CLS]* pada awal urutan token serta *[SEP]* sebagai penanda akhir kalimat atau pemisah antara dua kalimat. Ketiga komponen tersebut menghasilkan representasi masukan yang selanjutnya diteruskan ke lapisan pertama *Transformer Encoder* [18].

Token Embedding merepresentasikan identitas setiap token dalam kosakata BERT. Sementara itu, *Segment Embedding* digunakan untuk membedakan apakah suatu token berasal dari kalimat pertama (*Sentence A*) atau kalimat kedua (*Sentence B*), sedangkan *Position Embedding* memberikan informasi mengenai posisi setiap token dalam urutan masukan. Berbeda dengan *Positional Encoding* pada arsitektur *Transformer* yang menggunakan fungsi sinus dan cosinus, *Position Embedding* pada BERT dipelajari secara langsung selama proses *pre-training* sehingga nilai representasinya diperoleh melalui proses pembelajaran model [18]. Ketiga komponen tersebut kemudian dijumlahkan untuk membentuk representasi akhir yang digunakan sebagai masukan bagi setiap lapisan *Transformer Encoder*.

$$E = E_{\text{token}} + E_{\text{segment}} + E_{\text{position}} \tag{2.11}$$

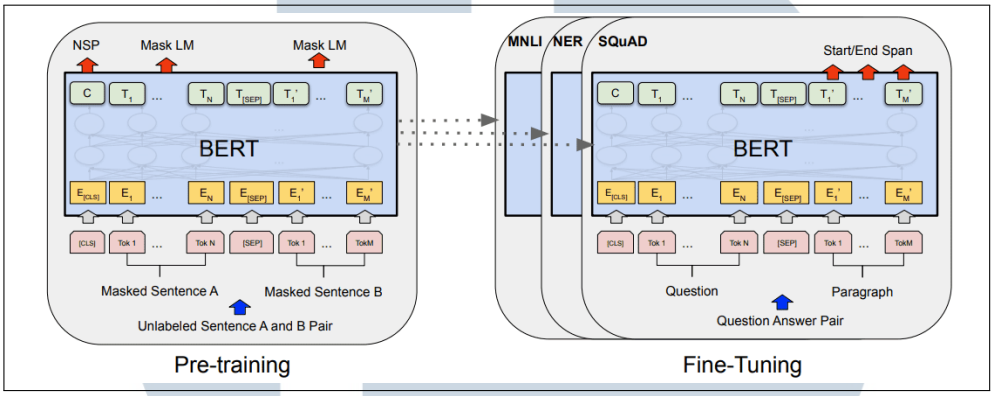
Rumus *Input Representation* BERT ditunjukkan pada Rumus 2.11. Pada Rumus 2.11, E_{token} merupakan *Token Embedding* yang merepresentasikan identitas setiap token, E_{segment} merupakan *Segment Embedding* yang menunjukkan asal token berdasarkan segmen kalimat, sedangkan E_{position} merupakan *Position Embedding* yang menyimpan informasi posisi token dalam urutan masukan. Hasil penjumlahan ketiga komponen tersebut menghasilkan representasi akhir yang digunakan sebagai masukan bagi setiap lapisan *Transformer Encoder* [18].

Melalui *Input Representation*, setiap token tidak hanya membawa informasi mengenai identitas kata, tetapi juga informasi mengenai posisi token serta segmen kalimat tempat token tersebut berada. Representasi tersebut kemudian diproses oleh mekanisme *Self-Attention* pada setiap lapisan *Transformer Encoder* sehingga BERT mampu menghasilkan representasi bahasa yang bersifat kontekstual [18].

2.5.3 *Pre-training*

Sebelum digunakan pada berbagai tugas *Natural Language Processing* (NLP), BERT terlebih dahulu menjalani proses *pre-training* menggunakan korpus teks berskala besar. Tahap ini bertujuan agar model mampu mempelajari

representasi bahasa yang bersifat umum tanpa bergantung pada tugas tertentu. Melalui proses *pre-training*, BERT mempelajari hubungan antar kata, struktur kalimat, serta konteks bahasa sehingga pengetahuan yang diperoleh dapat dimanfaatkan kembali pada berbagai tugas NLP melalui proses *fine-tuning* [18].



Gambar 2.3. Tahapan *Pre-training* dan *Fine-tuning* pada BERT

Sumber: [18]

Gambar 2.3 menunjukkan bahwa pada tahap *pre-training*, BERT dilatih menggunakan dua tugas utama, yaitu *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP). Kedua tugas tersebut dirancang agar model mampu memahami konteks kata maupun hubungan antar kalimat secara dua arah (*bidirectional*) sehingga representasi bahasa yang dihasilkan menjadi lebih kontekstual dibandingkan model bahasa sebelumnya [18].

A *Masked Language Modeling* (MLM)

Masked Language Modeling (MLM) merupakan tugas utama pada proses *pre-training* BERT yang bertujuan melatih model untuk memprediksi token yang disembunyikan berdasarkan konteks di sekitarnya. Berbeda dengan model bahasa satu arah yang hanya memanfaatkan konteks sebelum atau sesudah suatu token, MLM memungkinkan BERT memanfaatkan seluruh konteks dalam kalimat sehingga menghasilkan representasi yang bersifat *bidirectional* [18].

Pada proses MLM, sebanyak 15% token dari setiap urutan masukan dipilih secara acak sebagai kandidat prediksi. Dari token yang terpilih tersebut, sebanyak 80% diganti menjadi token *[MASK]*, 10% diganti dengan token acak dari kosakata BERT, sedangkan 10% sisanya tetap dipertahankan. Strategi tersebut bertujuan agar model tidak hanya bergantung pada keberadaan token *[MASK]*, tetapi juga

mampu menghasilkan representasi yang tetap baik ketika memproses teks nyata yang tidak mengandung token tersebut. Melalui mekanisme tersebut, BERT mempelajari hubungan antar token berdasarkan konteks di sekitarnya sehingga mampu menghasilkan representasi bahasa yang lebih kontekstual [18].

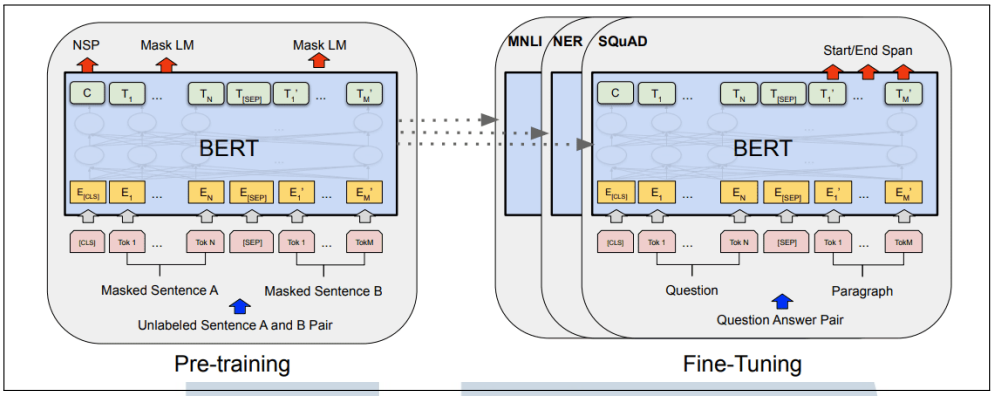
B *Next Sentence Prediction* (NSP)

Selain mempelajari hubungan antar token, BERT juga dilatih menggunakan tugas *Next Sentence Prediction* (NSP) untuk memahami hubungan antar kalimat. Pada tugas ini, model menerima dua kalimat sebagai masukan dan menentukan apakah kalimat kedua benar-benar merupakan kelanjutan dari kalimat pertama atau tidak. Sebanyak 50% pasangan kalimat merupakan pasangan yang berurutan di dalam korpus, sedangkan 50% sisanya dipasangkan secara acak sehingga tidak memiliki hubungan langsung [18].

Melalui proses tersebut, BERT mempelajari keterkaitan antar kalimat yang berguna pada berbagai tugas NLP, seperti *question answering* maupun *natural language inference*. Kombinasi antara tugas MLM dan NSP memungkinkan BERT mempelajari representasi bahasa yang kaya akan informasi kontekstual sebelum disesuaikan dengan tugas tertentu melalui proses *fine-tuning*. Tahap tersebut memungkinkan model yang telah melalui proses *pre-training* dimanfaatkan pada berbagai tugas NLP dengan hanya melakukan penyesuaian parameter menggunakan data berlabel [18].

2.5.4 *Fine-tuning*

Setelah menyelesaikan proses *pre-training*, BERT dapat disesuaikan kembali melalui proses *fine-tuning* untuk menyelesaikan berbagai tugas *Natural Language Processing* (NLP). Berbeda dengan *pre-training* yang bertujuan mempelajari representasi bahasa secara umum menggunakan korpus teks yang besar, *fine-tuning* dilakukan menggunakan dataset berlabel yang disesuaikan dengan tugas tertentu. Pada tahap ini, parameter model yang telah dipelajari selama *pre-training* diperbarui kembali sehingga mampu menghasilkan prediksi yang sesuai dengan kebutuhan tugas yang dikerjakan [18].



Gambar 2.4. Tahapan *Pre-training* dan *Fine-tuning* pada BERT

Sumber: [18]

Berdasarkan Gambar 2.4, model yang telah melalui proses *pre-training* dapat digunakan pada berbagai tugas NLP dengan menambahkan lapisan keluaran (*output layer*) sesuai dengan kebutuhan. Seluruh parameter BERT beserta lapisan keluaran tersebut kemudian dilatih kembali menggunakan data berlabel sehingga model mampu menyesuaikan representasi bahasa yang telah dipelajari terhadap karakteristik tugas yang akan diselesaikan [18].

Salah satu karakteristik utama proses *fine-tuning* adalah penggunaan arsitektur dasar yang sama untuk berbagai tugas NLP. Perbedaannya hanya terletak pada lapisan keluaran yang ditambahkan di atas representasi yang dihasilkan oleh BERT. Sebagai contoh, tugas klasifikasi teks menggunakan lapisan klasifikasi berbasis *Softmax*, sedangkan tugas *Named Entity Recognition* (NER) maupun *Question Answering* menggunakan lapisan keluaran yang disesuaikan dengan karakteristik masing-masing tugas [18, 25]. Pada tugas klasifikasi, representasi keluaran token *[CLS]* umumnya digunakan sebagai representasi keseluruhan masukan. Representasi tersebut kemudian diteruskan ke lapisan klasifikasi untuk menghasilkan probabilitas setiap kelas menggunakan fungsi *Softmax*.

$$P(y=i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{2.12}$$

Rumus *Softmax* ditunjukkan pada Rumus 2.12. Pada Rumus 2.12, z_i merupakan nilai *logit* untuk kelas ke-i, sedangkan C menyatakan jumlah seluruh kelas. Fungsi *Softmax* mengubah nilai *logit* menjadi distribusi probabilitas sehingga jumlah probabilitas seluruh kelas bernilai satu [18]. Selama proses pelatihan, hasil prediksi model kemudian dibandingkan dengan label sebenarnya menggunakan

fungsi *Cross Entropy Loss* untuk mengukur tingkat kesalahan prediksi.

$$\mathscr{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.13}$$

Rumus *Cross Entropy Loss* ditunjukkan pada Rumus 2.13. Pada Rumus 2.13, y_i merupakan label sebenarnya untuk kelas ke-i, sedangkan $\hat{y}_i$ merupakan probabilitas hasil prediksi model setelah melalui fungsi *Softmax*. Nilai *loss* digunakan sebagai dasar untuk memperbarui parameter model melalui proses *backpropagation*. Semakin kecil nilai *Cross Entropy Loss*, semakin baik kesesuaian antara hasil prediksi model dengan label sebenarnya [18].

Melalui proses *fine-tuning*, BERT dapat disesuaikan untuk berbagai tugas NLP tanpa perlu melakukan pelatihan model dari awal. Pendekatan tersebut memungkinkan model memanfaatkan pengetahuan bahasa yang telah diperoleh selama proses *pre-training* sehingga mampu menghasilkan performa yang baik pada berbagai tugas klasifikasi maupun pemahaman bahasa [18, 25].

2.6 IndoBERT

IndoBERT merupakan model bahasa pralatih (*pre-trained language model*) yang dikembangkan untuk memproses dan memahami teks berbahasa Indonesia. Model ini dibangun berdasarkan arsitektur BERT yang telah dijelaskan pada subbab sebelumnya, tetapi dilatih menggunakan korpus berbahasa Indonesia sehingga mampu menghasilkan representasi bahasa yang lebih sesuai dengan karakteristik linguistik, struktur kalimat, serta kosakata yang umum digunakan dalam bahasa Indonesia [19].

Pengembangan IndoBERT merupakan bagian dari proyek IndoLEM (*Indonesian Language Evaluation Montage*), yaitu kumpulan *benchmark*, model pralatih, dan sumber daya yang dirancang untuk mendukung penelitian *Natural Language Processing* (NLP) berbahasa Indonesia [19]. Selain itu, IndoNLU (*Indonesian Natural Language Understanding*) turut menyediakan berbagai dataset evaluasi yang digunakan untuk mengukur performa model pada beragam tugas NLP, seperti klasifikasi teks, analisis sentimen, *named entity recognition*, dan *question answering* [28]. Keberadaan sumber daya tersebut memungkinkan pengembangan model bahasa yang lebih sesuai dengan karakteristik bahasa Indonesia dibandingkan model yang dilatih menggunakan banyak bahasa secara

bersamaan.

Karena dilatih menggunakan korpus berbahasa Indonesia, IndoBERT mampu mempelajari karakteristik bahasa yang beragam, mulai dari bahasa formal hingga variasi bahasa yang umum digunakan pada media sosial, seperti *slang*, singkatan, maupun istilah tidak baku [1]. Kemampuan tersebut menjadikan IndoBERT banyak dimanfaatkan pada berbagai tugas NLP berbahasa Indonesia, khususnya analisis sentimen, karena mampu menghasilkan representasi kontekstual yang lebih baik dibandingkan model *multilingual*.

Sejumlah penelitian menunjukkan bahwa IndoBERT mampu memberikan performa klasifikasi yang lebih baik dibandingkan pendekatan *machine learning* konvensional [20]. Sebagai contoh, Manoppo *et al.* [10] melaporkan bahwa IndoBERT memperoleh akurasi sebesar 84,94% pada analisis sentimen publik terhadap kebijakan kenaikan PPN 12% di Indonesia. Temuan serupa juga dilaporkan pada penerapan IndoBERT untuk menganalisis opini masyarakat terhadap kebijakan publik maupun isu sosial yang berkembang di media sosial [21, 1].

Pada penelitian ini, model yang digunakan adalah *indobenchmark/indobert-base-p1*, yaitu salah satu model IndoBERT yang dikembangkan dalam proyek IndoLEM dan tersedia melalui pustaka Hugging Face. Model tersebut digunakan untuk melakukan klasifikasi sentimen terhadap *tweet* mengenai Program Makan Bergizi Gratis. Pemilihan model ini didasarkan pada kemampuannya dalam memahami konteks bahasa Indonesia serta karakteristik bahasa yang umum digunakan pada media sosial sehingga diharapkan mampu menghasilkan klasifikasi sentimen yang lebih akurat.

2.7 Topic Modeling

Topic modeling merupakan salah satu teknik dalam *Natural Language Processing* (NLP) yang digunakan untuk mengidentifikasi topik atau tema yang tersembunyi dalam kumpulan data teks [22]. Teknik ini bekerja dengan menemukan pola keterkaitan antar kata sehingga dapat mengelompokkan informasi yang memiliki kesamaan pembahasan ke dalam suatu topik tertentu. Melalui proses tersebut, data teks yang berjumlah besar dapat diringkas dan diorganisasikan menjadi sejumlah topik yang lebih mudah untuk dianalisis [22].

Penggunaan *topic modeling* banyak diterapkan pada penelitian yang melibatkan data teks dalam jumlah besar karena mampu membantu proses

eksplorasi informasi secara otomatis [7]. Melalui pendekatan ini, peneliti dapat memperoleh gambaran mengenai tema-tema yang dominan dalam suatu kumpulan teks tanpa harus melakukan penelaahan secara manual terhadap seluruh data.

Secara umum, *topic modeling* termasuk ke dalam pendekatan *unsupervised learning* karena proses pembentukan topik dilakukan tanpa menggunakan label yang telah ditentukan sebelumnya [23]. Hasil yang diperoleh umumnya berupa kumpulan kata yang saling berkaitan dan dapat digunakan untuk merepresentasikan suatu topik berdasarkan pola kemunculannya dalam data teks.

Penerapan *topic modeling* pada data media sosial memiliki tantangan tersendiri. Sebagian besar unggahan di media sosial bersifat singkat, tidak terstruktur, serta banyak menggunakan bahasa informal [7]. Karakteristik tersebut dapat memengaruhi kualitas topik yang dihasilkan sehingga proses identifikasi tema menjadi lebih kompleks dibandingkan pada teks yang memiliki struktur bahasa yang lebih teratur.

Perkembangan metode analisis teks mendorong munculnya berbagai pendekatan *topic modeling* yang bertujuan menghasilkan topik yang lebih representatif dan mudah diinterpretasikan [23]. Pengembangan tersebut dilakukan untuk mengatasi berbagai keterbatasan metode sebelumnya, terutama ketika diterapkan pada data media sosial yang memiliki karakteristik bahasa yang beragam dan dinamis.

2.8 BERTopic

BERTopic merupakan metode *topic modeling* yang memanfaatkan representasi *embedding* berbasis *Transformer* untuk mengidentifikasi topik dalam kumpulan dokumen teks [23]. Metode ini mengombinasikan beberapa tahapan, mulai dari pembentukan *embedding*, reduksi dimensi, *clustering*, hingga ekstraksi kata representatif untuk menghasilkan topik yang lebih mudah diinterpretasikan.

Dibandingkan metode *topic modeling* tradisional seperti *Latent Dirichlet Allocation* (LDA) yang bergantung pada pendekatan *bag-of-words*, BERTopic memanfaatkan representasi *embedding* sehingga mampu menangkap hubungan semantik antar kata dengan lebih baik [7]. Kemampuan tersebut menjadikan BERTopic lebih sesuai untuk data media sosial yang umumnya singkat, tidak terstruktur, dan banyak menggunakan bahasa informal.

Tahap awal dalam BERTopic adalah menghasilkan *embedding* dokumen menggunakan model berbasis *Transformer* [23]. Representasi tersebut kemudian

direduksi menggunakan *Uniform Manifold Approximation and Projection* (UMAP) agar struktur data dapat dipertahankan dalam dimensi yang lebih rendah. Proses ini bertujuan mempermudah tahap pengelompokan dokumen berdasarkan kemiripan representasinya.

Setelah proses reduksi dimensi dilakukan, dokumen dikelompokkan menggunakan *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN). Metode ini mengidentifikasi kelompok dokumen yang memiliki kedekatan semantik tanpa harus menentukan jumlah *cluster* sejak awal [23]. Pendekatan tersebut juga memungkinkan identifikasi data yang tidak termasuk ke dalam topik tertentu sebagai *noise*.

Representasi topik kemudian dibentuk menggunakan *class-based TF-IDF* (c-TF-IDF) untuk menentukan kata-kata yang paling mewakili setiap kelompok dokumen [23]. Melalui pendekatan tersebut, BERTopic dapat menghasilkan topik yang lebih mudah diinterpretasikan dan memiliki keterkaitan yang lebih kuat dengan isi dokumen dibandingkan metode *topic modeling* tradisional.

Sejumlah penelitian menunjukkan bahwa BERTopic mampu menghasilkan topik yang lebih koheren dibandingkan pendekatan *topic modeling* konvensional [7]. Penelitian yang dilakukan oleh Nur *et al.* berhasil mengombinasikan IndoBERT dan BERTopic untuk menganalisis opini publik pada media sosial X serta memperoleh topik yang relevan dengan sentimen pengguna [3]. Temuan tersebut menunjukkan bahwa BERTopic dapat digunakan sebagai pelengkap analisis sentimen untuk memberikan gambaran yang lebih mendalam mengenai isu yang dibahas oleh pengguna media sosial.

Pada penelitian ini, BERTopic digunakan untuk mengidentifikasi topik-topik utama yang muncul dalam diskusi masyarakat mengenai Program Makan Bergizi Gratis berdasarkan data *tweet* yang diperoleh dari media sosial X.

2.9 Evaluation Metrics

Evaluation metrics digunakan untuk untuk menilai kinerja model klasifikasi, baik yang dibangun menggunakan pendekatan *machine learning* maupun *deep learning* [11]. Pada penelitian ini, metrik evaluasi dimanfaatkan untuk mengetahui kemampuan model IndoBERT dalam mengklasifikasikan sentimen pada data *tweet* sesuai dengan label yang telah ditentukan.

Penilaian performa model dilakukan melalui *confusion matrix* [12]. Matriks ini membandingkan hasil prediksi model dengan label aktual sehingga jumlah

prediksi yang benar maupun yang keliru pada setiap kelas dapat diamati secara lebih rinci.

Nilai pada *confusion matrix* kemudian digunakan untuk menghitung beberapa metrik evaluasi, yaitu *accuracy*, *precision*, *recall*, dan *F1-score* [10]. *Accuracy* menunjukkan tingkat ketepatan model secara keseluruhan terhadap seluruh data yang dievaluasi dan dihitung menggunakan Rumus 2.14. Sementara itu, *precision* menggambarkan ketepatan prediksi ketika model menetapkan suatu kelas tertentu dan dihitung menggunakan Rumus 2.15. Selanjutnya, *recall* menunjukkan kemampuan model dalam menemukan seluruh data yang memang termasuk ke dalam kelas tersebut dan dihitung menggunakan Rumus 2.16. Untuk memberikan penilaian yang lebih seimbang, digunakan pula *F1-score* yang merupakan rata-rata harmonik antara *precision* dan *recall*, yang dihitung menggunakan Rumus 2.17.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.14}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.15}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.16}$$

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.17}$$

Keterangan: TP (*True Positive*): data yang diprediksi benar sebagai anggota suatu kelas TN (*True Negative*): data yang diprediksi benar sebagai bukan anggota suatu kelas FP (*False Positive*): data yang diprediksi sebagai anggota suatu kelas, tetapi sebenarnya bukan FN (*False Negative*): data yang sebenarnya merupakan anggota suatu kelas, tetapi diprediksi bukan anggota kelas tersebut

2.10 Cohen's Kappa

Cohen's Kappa merupakan ukuran statistik yang digunakan untuk mengukur tingkat kesepakatan (*inter-rater agreement*) antara dua anotator dalam proses pelabelan data dengan mempertimbangkan kemungkinan kesepakatan yang terjadi secara kebetulan (*chance agreement*) [29]. Berbeda dengan persentase kesepakatan sederhana, Cohen's Kappa memperhitungkan peluang kedua anotator memberikan

label yang sama secara acak sehingga menghasilkan pengukuran reliabilitas yang lebih objektif. Oleh karena itu, Cohen's Kappa banyak digunakan untuk mengevaluasi konsistensi hasil anotasi pada berbagai penelitian klasifikasi dan analisis sentimen.

$$\kappa = \frac{P_o - P_e}{1 - P_e} \tag{2.18}$$

Rumus Cohen's Kappa ditunjukkan pada Rumus 2.18. Pada rumus tersebut, P_o (*observed agreement*) merupakan proporsi kesepakatan yang diamati antara dua anotator, sedangkan P_e (*expected agreement by chance*) merupakan proporsi kesepakatan yang diperkirakan terjadi secara kebetulan. Nilai koefisien Cohen's Kappa berada pada rentang -1 hingga 1, di mana nilai yang semakin mendekati 1 menunjukkan tingkat kesepakatan yang semakin tinggi, nilai 0 menunjukkan bahwa tingkat kesepakatan tidak lebih baik dibandingkan kesepakatan yang terjadi secara acak, sedangkan nilai negatif menunjukkan tingkat kesepakatan yang lebih rendah daripada yang diharapkan secara kebetulan [29].

2.11 Program Makan Bergizi Gratis

Program Makan Bergizi Gratis (MBG) adalah salah satu program pemerintah Indonesia yang memiliki tujuan untuk meningkatkan kualitas gizi masyarakat sekaligus mendukung pembangunan sumber daya manusia menuju visi Generasi Emas 2045 [13]. Program ini dilaksanakan oleh Badan Gizi Nasional (BGN) dengan sasaran utama peserta didik, balita, ibu hamil, dan ibu menyusui. Program MBG dirancang untuk membantu mengurangi permasalahan gizi, meningkatkan kualitas kesehatan masyarakat, serta mendukung peningkatan kualitas pendidikan melalui pemenuhan kebutuhan nutrisi [16]. Selain itu, program ini juga diharapkan mampu meningkatkan kualitas sumber daya manusia Indonesia dalam jangka panjang.

Dalam implementasinya, Program MBG mendapatkan perhatian luas dari masyarakat karena memiliki cakupan program yang besar dan alokasi anggaran yang signifikan. Berdasarkan data APBN tahun 2025 yang dipublikasikan oleh Kementerian Keuangan Republik Indonesia, pemerintah mengalokasikan anggaran sebesar Rp71 triliun untuk mendukung pelaksanaan Program Makan Bergizi Gratis melalui Badan Gizi Nasional [14]. Besarnya alokasi anggaran tersebut menjadikan Program MBG sebagai salah satu kebijakan pemerintah yang memperoleh perhatian

luas dari masyarakat maupun media.

Di sisi lain, implementasi Program MBG juga memunculkan berbagai tanggapan masyarakat, baik berupa dukungan maupun kritik [15]. Sebagian pihak menilai program ini mampu meningkatkan kualitas gizi masyarakat, sedangkan pihak lain mempertanyakan efektivitas distribusi, kesiapan infrastruktur, transparansi pelaksanaan, serta keberlanjutan anggaran program [16]. Tingginya perhatian publik terhadap Program MBG menyebabkan media sosial menjadi sumber data yang relevan untuk memahami persepsi masyarakat terhadap kebijakan tersebut. Oleh karena itu, penelitian ini menggunakan data *tweet* dari media sosial X untuk menganalisis sentimen serta mengidentifikasi topik diskusi publik terkait Program Makan Bergizi Gratis.

2.12 Penelitian Terdahulu

Berdasarkan Tabel 2.1, penelitian terdahulu telah menerapkan berbagai metode analisis sentimen, mulai dari algoritma *machine learning* seperti *Support Vector Machine* (SVM) dan *Naïve Bayes* hingga model berbasis *Transformer* seperti IndoBERT. Beberapa penelitian juga mengombinasikan analisis sentimen dengan *topic modeling* menggunakan BERTopic maupun LDA. Namun, penelitian yang mengintegrasikan IndoBERT dan BERTopic untuk menganalisis opini publik terhadap Program Makan Bergizi Gratis pada media sosial X masih terbatas. Oleh karena itu, penelitian ini menerapkan kombinasi kedua metode tersebut untuk menghasilkan analisis sentimen sekaligus mengidentifikasi topik-topik utama yang mendasari opini publik.

Tabel 2.1. Penelitian Terdahulu

Peneliti & Tahun	**Judul Penelitian**	**Metode / Model**	**Dataset & Hasil Utama**	**Kelebihan dan Kelemahan**
Nur et al. (2025)	Analisis Sentimen dan Pemodelan Topik pada *Post* tentang Merek Teknologi di X Menggunakan *Fine-tuning* IndoBERT dan BERTopic	IndoBERT untuk klasifikasi sentimen; BERTopic untuk pemodelan topik	Dataset: 10.130 *post* mengenai merek Xiaomi. Hasil: Akurasi IndoBERT sebesar 79,8%, F1-score 0,699, serta BERTopic menghasilkan 16 klaster topik dengan nilai *Coherence* (C_v) sebesar 0,5437.	**Kelebihan:** Menggunakan kombinasi Transformer yang optimal untuk analisis teks media sosial. **Kelemahan:** Domain penelitian hanya berfokus pada produk teknologi sehingga sentimen relatif eksplisit.

Bersambung ke halaman berikutnya

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Peneliti & Tahun	**Judul Penelitian**	**Metode / Model**	**Dataset & Hasil Utama**	**Kelebihan dan Kelemahan**
Ishak et al. (2025)	Analisis Sentimen terhadap Pemerintahan Prabowo–Gibran Menggunakan IndoBERT dan LDA	IndoBERT dan LDA	Dataset: 195 artikel dari portal berita daring. Hasil: Sentimen didominasi negatif (80%) dan LDA menghasilkan 10 topik utama.	**Kelebihan:** Menggabungkan analisis sentimen dan pemodelan topik pada isu pemerintahan. **Kelemahan:** Menggunakan data portal berita formal dan LDA berbasis *bag-of-words*. **Research Gap:** Penelitian ini menggunakan data media sosial X serta mengganti LDA dengan BERTopic yang mampu menangkap konteks semantik teks pendek.

Bersambung ke halaman berikutnya

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Peneliti & Tahun	**Judul Penelitian**	**Metode / Model**	**Dataset & Hasil Utama**	**Kelebihan dan Kelemahan**
Geni et al. (2023)	*Sentiment Analysis of Tweets Before the 2024 Elections in Indonesia Using IndoBERT Language Models*	IndoBERT, dibandingkan dengan SVM, Naïve Bayes, dan TextBlob	Dataset: 1.000 *tweet* terkait Pemilu 2024. Hasil: IndoBERT *large-p1* memperoleh akurasi 83,5%, lebih tinggi dibandingkan SVM dan Naïve Bayes.	**Kelebihan:** Membuktikan keunggulan IndoBERT dibandingkan metode klasifikasi tradisional. **Kelemahan:** Hanya melakukan analisis sentimen tanpa identifikasi topik.

Bersambung ke halaman berikutnya

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Peneliti & Tahun	**Judul Penelitian**	**Metode / Model**	**Dataset & Hasil Utama**	**Kelebihan dan Kelemahan**
Manoppo et al. (2025)	Analisis Sentimen Publik di Media Sosial terhadap Kenaikan PPN 12% di Indonesia Menggunakan IndoBERT	*Fine-tuning* IndoBERT	Dataset: 2.581 data dari X, Instagram, dan TikTok. Hasil: Akurasi 84,94%, Precision 85,60%, Recall 84,94%, dan F1-score 84,37%.	**Kelebihan:** Memberikan performa klasifikasi yang tinggi pada isu kebijakan fiskal. **Kelemahan:** Menggabungkan data dari beberapa *platform* dan belum melakukan pemodelan topik.
Fathoni et al. (2025)	Analisis Sentimen Public Twitter terhadap Kebijakan Pemerintah Menggunakan Metode SVM (Studi Kasus: RUU TNI)	Support Vector Machine (SVM) dan TF-IDF	Dataset: 1.001 *tweet* mengenai RUU TNI. Hasil: Akurasi mencapai 94,66%.	**Kelebihan:** Memperoleh akurasi tinggi meskipun data tidak seimbang. **Kelemahan:** SVM dengan TF-IDF belum mampu menangkap hubungan kontekstual antarkata.

Bersambung ke halaman berikutnya

Tabel 2.1 Penelitian Terdahulu (Lanjutan)

Peneliti & Tahun	**Judul Penelitian**	**Metode / Model**	**Dataset & Hasil Utama**	**Kelebihan dan Kelemahan**
Harahap & Kurniawan (2024)	Analisis Sentimen Komentar terhadap Kebijakan Pemerintah Mengenai TAPERA pada Aplikasi X Menggunakan Metode Naïve Bayes	Naïve Bayes dan TF-IDF	Dataset: 1.210 *tweet* mengenai TAPERA. Hasil: Akurasi mencapai 96%, Precision 93%, Recall 93%, dan F1-score 97%.	**Kelebihan:** Komputasi ringan dengan performa klasifikasi yang baik. **Kelemahan:** Asumsi independensi fitur pada Naïve Bayes menyebabkan model kurang mampu memahami konteks kalimat.