BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Kajian terhadap penelitian terdahulu sangat penting untuk memberikan landasan teoritis, memperkuat relevansi penelitian, dan mengidentifikasi celah yang dapat diisi oleh penelitian ini. Selain itu, penelitian terdahulu membantu penulis memahami perkembangan topik serupa dan menentukan kontribusi yang diharapkan. Tabel 2.1 menyajikan rangkuman beberapa penelitian sebelumnya yang menjadi referensi utama dalam pengembangan penelitian ini.

Tabel 2.1 Tabel Penelitian Terdahulu

Penulis	Judul	Jurnal	Metode	Hasil
Li S, Wang R [25]	*Cryptocurrency Price Prediction Using LSTM and FEDformer Enhanced by Sentiment Analysis*	*IEEE Access (2025)*	CryptoBERT, FinBERT, BERT, VADER	Model CryptoBERT mencapai RMSE 1.770, MAPE 1,90%, dan R^2 0,9874, menunjukkan performa terbaik dan mengungguli VADER, TextBlob, BERT, serta FinBERT.
Kulakowski M, Frasincar F, Cambria E [24]	*Sentiment Classification of Cryptocurrency-Related Social Media Posts*	*IEEE Intelligent Systems (2023)*	CryptoBERT, BERTweet, FinBERT, BERT, VADER, LUKE	Model CryptoBERT menunjukkan performa terbaik berdasarkan metrik F1-score, precision, dan recall, serta mengungguli LUKE, VADER, BERT, FinBERT, dan BERTweet.
Kim G, Kim M [27]	*CBITS: Crypto BERT Incorporated Trading System*	*IEEE Access (2023)*	CryptoBERT, mBERT, XLM-RoBERTa	Model CryptoBERT mencapai akurasi 81,42% dan menunjukkan performa terbaik, mengungguli mBERT (79,83%) serta XLM-RoBERTa (80,63%) dalam analisis sentimen *crypto*.

Penulis	Judul	Jurnal	Metode	Hasil
Bello A, Ng S, Leung M [32]	*A BERT Framework to Sentiment Analysis of Tweets*	*Sensors (2023)*	BERT-CNN, BERT-RNN, BERT	Model BERT-CNN menunjukkan performa terbaik dengan akurasi 93,37%, diikuti BERT-RNN 92,21%, serta mengungguli model dasar BERT (85,61%) dengan peningkatan kinerja masing-masing sebesar 7,76% dan 6,6%.
Hossain M [31]	*Multi task opinion enhanced hybrid BERT model for mental health analysis*	*Scientific Reports (2025)*	BERT-CNN, BERT-BiGRU, BERT	Model BERT-CNN mencapai akurasi 92,13% dan menunjukkan performa terbaik, mengungguli BERT-BiGRU (87,55%) serta BERT (90,69%) dalam analisis kesehatan mental berbasis teks, dengan peningkatan kinerja sebesar 1,44%.
Kaur K, Kaur P [38]	*BERT-CNN: Improving BERT for Requirements Classification using CNN*	*Procedia Computer Science (2023)*	BERT-CNN, BERT	BERT-CNN mencapai akurasi 85% dan menunjukkan peningkatan performa dibandingkan BERT (81%), dengan kenaikan kinerja sebesar 4% dalam klasifikasi kebutuhan.
He A, Abisado M [36]	*Text Sentiment Analysis of Douban Film Short Comments Based on BERT-CNN-BiLSTM-Att Model*	*IEEE Access (2024)*	BERT, BERT-CNN, BERT-BiLSTM, BERT-CNN-BiLSTM-At	Model BERT-CNN-BiLSTM-Att mencapai akurasi tertinggi sebesar 95,7%, menunjukkan performa terbaik dibandingkan BERT-CNN (92,5%) dan BERT-

Penulis	Judul	Jurnal	Metode	Hasil
				BiLSTM (91,7%). Selain itu, BERT-CNN juga menunjukkan kinerja lebih baik dibandingkan model dasar BERT (91,0%).
Zamani N, Kamaruddin N [34]	*Crypto-sentiment Detection in Malay Text Using Language Models with an Attention Mechanism*	*Journal of Information Systems Engineering and Business Intelligence (2023)*	BERT-GRU, BERT-GRU + Attention	Model BERT-GRU dengan Attention menunjukkan peningkatan kinerja yang signifikan dengan Adjusted R^2 sebesar 0,575, dibandingkan model tanpa Attention sebesar 0,426.
Chi D, Huang T [35]	*Research on sentiment analysis of hotel review text based on BERT-TCN-BiLSTM-attention model*	*Array (2025)*	BERT-TCN-BiLSTM, BERT-TCN-BiLSTM-Attention	Model BERT-TCN-BiLSTM dengan Attention menunjukkan peningkatan performa dengan F1-score sebesar 90,40%, dibandingkan model tanpa Attention (86,10%), atau meningkat sekitar 4,3%.
Thekkekara J, Yongchareon S [37]	*An Attention-Based BERT–CNN–BiLSTM Model for Depression Detection from Emojis in Social Media Text*	*Big Data and Cognitive Computing (2025)*	BERT-CNN-BiLSTM, BERT-CNN-BiLSTM + Attention	Model BERT-CNN-BiLSTM dengan Attention meningkatkan akurasi menjadi 97,12% dibandingkan model tanpa Attention (95,47%) dalam deteksi depresi berbasis emoji pada teks media sosial.

Berdasarkan berbagai penelitian terdahulu pada Tabel 2.1, integrasi analisis sentimen berbasis model bahasa pada domain *cryptocurrency* terbukti berkontribusi signifikan terhadap peningkatan akurasi prediksi harga maupun pengambilan keputusan dalam sistem trading. Penelitian yang memanfaatkan model bahasa yang dipra-latih secara khusus pada korpus *cryptocurrency* menunjukkan bahwa

pendekatan domain-spesifik mampu menangkap istilah, jargon, dan dinamika komunikasi pasar kripto dengan lebih baik dibandingkan model bahasa umum maupun metode berbasis leksikon. Hal ini terlihat dari capaian model CryptoBERT yang memperoleh RMSE sebesar 1,770, MAPE 1,90%, dan R^2 sebesar 0,9874 pada tugas prediksi harga *cryptocurrency*, sehingga mengungguli *VADER*, *TextBlob*, BERT, dan FinBERT [25]. Temuan ini menunjukkan bahwa kualitas representasi bahasa yang lebih sesuai dengan konteks pasar kripto dapat meningkatkan presisi prediksi secara nyata. Pada tugas klasifikasi sentimen unggahan media sosial terkait *cryptocurrency*, model domain-spesifik juga memperlihatkan keunggulan yang konsisten, di mana CryptoBERT memberikan performa terbaik berdasarkan metrik *F1-score*, *precision*, dan *recall* dibandingkan BERTweet, FinBERT, BERT, VADER, dan LUKE [24]. Hasil ini menegaskan bahwa penyesuaian model terhadap *domain cryptocurrency* tidak hanya bermanfaat pada prediksi harga, tetapi juga pada pemahaman polaritas sentimen dalam data sosial yang bersifat informal dan dinamis. Lebih lanjut, penerapan sentimen berbasis CryptoBERT dalam sistem *trading* juga menunjukkan hasil yang kompetitif, dengan akurasi sebesar 81,42%, lebih tinggi dibandingkan mBERT yang mencapai 79,83% dan XLM-RoBERTa sebesar 80,63% [27]. Perbedaan performa tersebut mengindikasikan bahwa model yang dilatih secara spesifik pada *domain cryptocurrency* lebih efektif dalam memahami konteks linguistik pasar dibandingkan model multibahasa umum, sehingga lebih relevan untuk mendukung strategi *trading* berbasis sentimen.

Di sisi lain, berbagai penelitian juga menyoroti bahwa model BERT dasar masih memiliki keterbatasan ketika diterapkan pada teks media sosial yang cenderung singkat, ambigu, padat makna, dan kaya konteks implisit. Untuk mengatasi permasalahan tersebut, sejumlah penelitian mengembangkan arsitektur hibrida dengan menggabungkan BERT dan lapisan jaringan saraf tambahan, terutama *Convolutional Neural Network* (CNN) dan *Recurrent Neural Network* (RNN). Integrasi CNN dilakukan untuk memperkuat ekstraksi fitur lokal, sedangkan RNN, GRU, atau BiGRU digunakan untuk menangkap ketergantungan sekuensial antar kata. Pada analisis sentimen *tweet*, model BERT-CNN berhasil mencapai akurasi 93,37%, sedangkan BERT-RNN memperoleh 92,21%; keduanya secara signifikan melampaui BERT dasar yang hanya mencapai 85,61% [32]. Kenaikan tersebut

menunjukkan bahwa pengayaan arsitektur setelah representasi kontekstual BERT dapat membantu model memahami pola sentimen yang tidak tertangkap optimal oleh *encoder transformer* saja. Dalam konteks analisis kesehatan mental, model BERT-CNN juga menunjukkan performa tertinggi dengan akurasi 92,13%, lebih baik dibandingkan BERT yang memperoleh 90,69% dan BERT-BiGRU yang mencapai 87,55% [31]. Hasil ini memperlihatkan bahwa lapisan konvolusional tidak hanya efektif pada data media sosial umum, tetapi juga pada teks yang memuat opini dan kondisi psikologis yang lebih halus. Pada klasifikasi *requirement* perangkat lunak, pendekatan serupa menghasilkan peningkatan yang konsisten, di mana BERT-CNN mencapai akurasi 85%, sedangkan BERT dasar berada pada 81% [38]. Temuan ini memperkuat argumentasi bahwa kombinasi BERT dan CNN mampu memperkaya representasi semantik dengan cara menangkap pola lokal yang relevan terhadap tugas klasifikasi, bahkan pada *domain* yang berbeda dari analisis sentimen konvensional.

Pengembangan selanjutnya dilakukan dengan menambahkan mekanisme *attention* ke dalam arsitektur hibrida untuk meningkatkan kemampuan model dalam memfokuskan perhatian pada fitur atau kata yang paling relevan terhadap polaritas sentimen. Dalam pendekatan ini, BERT umumnya digunakan untuk menghasilkan representasi semantik dinamis, CNN untuk mengekstraksi fitur lokal, dan BiLSTM, GRU, atau TCN untuk memodelkan dependensi sekuensial dan konteks yang lebih luas, sementara attention berfungsi menyeleksi informasi penting yang paling berpengaruh terhadap keputusan klasifikasi. Pada analisis ulasan singkat film, model BERT-CNN-BiLSTM-Att mencapai akurasi tertinggi sebesar 95,7%, melampaui BERT-CNN yang memperoleh 92,5%, BERT-BiLSTM sebesar 91,7%, dan BERT dasar sebesar 91,0% [36]. Perbedaan ini menunjukkan bahwa kombinasi ekstraksi fitur lokal, pemodelan urutan, dan perhatian kontekstual dapat saling melengkapi untuk menghasilkan klasifikasi yang lebih presisi. Pola serupa juga terlihat pada analisis teks berbahasa Melayu, di mana penambahan *attention* pada BERT-GRU meningkatkan nilai Adjusted R^2 dari 0,426 menjadi 0,575 [34]. Pada ulasan *hotel*, model BERT-TCN-BiLSTM-Attention memperoleh *F1-score* sebesar 90,40%, lebih tinggi dibandingkan model tanpa *attention* yang hanya mencapai 86,10% [35]. Sementara itu, dalam deteksi depresi berbasis teks media sosial dan

emoji, model BERT-CNN-BiLSTM dengan *Attention* mencapai akurasi 97,12%, meningkat dari 95,47% pada model tanpa *attention* [37]. Hasil-hasil tersebut menunjukkan bahwa *attention mechanism* memberikan kontribusi nyata dalam meningkatkan sensitivitas model terhadap konteks, emosi, dan kata-kata penting, terutama pada teks yang kompleks, pendek, atau sarat nuansa. Secara keseluruhan, rangkaian penelitian tersebut memperlihatkan bahwa model bahasa *domain*-spesifik seperti CryptoBERT unggul dalam memahami bahasa khusus *cryptocurrency*, sedangkan pengembangan arsitektur hibrida berbasis CNN, RNN, BiLSTM, GRU, TCN, dan *attention* terbukti efektif untuk meningkatkan kualitas ekstraksi fitur serta akurasi klasifikasi sentimen pada berbagai *domain*.

Berdasarkan berbagai penelitian terdahulu yang telah dirangkum pada Tabel 2.1, dapat disimpulkan bahwa model bahasa berbasis *transformer* yang dilatih secara khusus pada domain *cryptocurrency* mampu memberikan performa yang lebih baik dalam analisis sentimen dibandingkan model bahasa umum. Sejumlah penelitian juga menunjukkan bahwa pengembangan arsitektur hibrida dengan mengintegrasikan BERT dengan lapisan *Convolutional Neural Network* (CNN) serta mekanisme *attention* dapat meningkatkan kemampuan model dalam mengekstraksi fitur lokal dan memahami konteks semantik yang lebih kompleks pada teks. Namun demikian, pemanfaatan CryptoBERT dalam penelitian sebelumnya umumnya masih digunakan secara langsung untuk klasifikasi sentimen tanpa pengembangan arsitektur tambahan yang dapat memperkaya ekstraksi fitur teks. Di sisi lain, penelitian yang menggabungkan CNN dan mekanisme *attention* umumnya masih diterapkan pada model BERT umum dan belum banyak diterapkan pada model bahasa yang dilatih secara khusus pada domain *cryptocurrency*. Selain itu, sebagian besar penelitian terdahulu masih berfokus pada peningkatan performa klasifikasi sentimen tanpa mengevaluasi hubungan antara skor sentimen yang dihasilkan model dengan pergerakan harga *cryptocurrency*. Oleh karena itu, penelitian ini mengusulkan pendekatan yang mengintegrasikan CryptoBERT dengan lapisan CNN sebagai arsitektur hibrida serta menambahkan mekanisme *attention* untuk meningkatkan kemampuan model dalam mengekstraksi fitur dan memahami konteks sentimen pada teks terkait *cryptocurrency*. Selanjutnya, skor sentimen yang dihasilkan model akan dianalisis keterkaitannya dengan data harga

cryptocurrency aktual menggunakan uji korelasi *Spearman* guna mengidentifikasi hubungan antara dinamika sentimen publik dan pergerakan harga pasar. Pendekatan ini diharapkan dapat memberikan kontribusi dalam pengembangan model analisis sentimen berbasis domain *cryptocurrency* sekaligus memberikan pemahaman yang lebih baik mengenai keterkaitan antara sentimen publik dan fluktuasi harga aset kripto.

2.2 Tinjauan Teori

2.2.1 Cryptocurrency

Cryptocurrency adalah bentuk aset digital atau mata uang virtual yang menggunakan teknik kriptografi untuk mengamankan transaksi serta beroperasi di atas teknologi blockchain, yaitu sistem buku besar terdistribusi (*distributed ledger*) yang bersifat transparan, tidak dapat diubah (*immutable*), dan tidak dikendalikan oleh otoritas pusat [39]. Konsep *cryptocurrency* pertama kali direalisasikan melalui peluncuran Bitcoin pada tahun 2008 oleh individu atau kelompok anonim dengan nama samaran Satoshi Nakamoto, yang mempublikasikan *whitepaper* berjudul "*Bitcoin: A Peer-to-Peer Electronic Cash System*" [40]. Implementasi jaringan Bitcoin dimulai pada tahun 2009 dan menjadi tonggak lahirnya sistem pembayaran digital terdesentralisasi yang memungkinkan transaksi dilakukan secara langsung antar pengguna (peer-to-peer) tanpa perantara seperti bank [41]. Dalam literatur keuangan digital, *cryptocurrency* dipandang sebagai inovasi moneter berbasis teknologi yang menggabungkan kriptografi, teori jaringan, dan mekanisme konsensus untuk menciptakan sistem keuangan alternatif yang lebih terbuka dan global.

Secara fungsional, *cryptocurrency* digunakan sebagai alat pembayaran digital, instrumen investasi spekulatif, penyimpan nilai (*store of value*), serta sarana transfer lintas negara dengan biaya relatif lebih rendah dan waktu transaksi lebih cepat dibanding sistem perbankan konvensional [42]. Seiring perkembangan ekosistem *blockchain*, *cryptocurrency* juga mendukung implementasi *smart contracts*, keuangan terdesentralisasi (*Decentralized Finance/DeFi*), *token non-fungible* (NFT), serta aplikasi *Web3* [43]. Berdasarkan karakteristiknya, *cryptocurrency* dapat diklasifikasikan menjadi beberapa jenis, antara lain: (1) *coin*

utama (*native coins*) seperti Bitcoin dan Ethereum yang memiliki *blockchain* sendiri; (2) altcoin, yaitu alternatif selain Bitcoin; (3) *stablecoin* yang nilainya dipatok pada aset tertentu seperti dolar AS untuk mengurangi volatilitas; serta (4) *utility* dan *governance* token yang digunakan dalam ekosistem aplikasi terdesentralisasi [44].

2.2.2 Platform *Twitter*

Twitter merupakan salah satu media sosial berbasis *microblogging* yang memungkinkan pengguna untuk membagikan pesan singkat, atau dikenal sebagai "tweet," dengan batas hingga 280 karakter [45]. Platform ini dirancang untuk mendukung komunikasi yang cepat dan interaktif, menjadikannya media utama dalam penyebaran informasi secara langsung dan waktu nyata [46]. Seiring perkembangannya, Twitter telah mengalami transformasi signifikan, dari sekadar ruang diskusi sosial hingga menjadi alat strategis untuk pemasaran dan penelitian [47] Kemampuan *platform* ini untuk terhubung dengan layanan lain cenderung meningkatkan popularitasnya dalam berbagi konten lintas kanal. Twitter juga memainkan peran penting dalam analisis politik platform, khususnya terkait dengan pengelolaan data pengguna dan pengendalian konten [48]. Fungsinya yang beragam dalam berbagai bidang seperti pendidikan, kesehatan, dan penelitian menunjukkan fleksibilitasnya sebagai ruang interaksi sosial sekaligus alat untuk memperkuat dampak penelitian [49]. Sebagai ekosistem digital, Twitter tidak hanya menjadi platform komunikasi, tetapi juga sumber data yang sangat bernilai bagi studi sosial dan teknologi.

2.2.3 Sentiment Analysis

Sentiment Analysis, atau analisis sentimen, merupakan cabang dari pemrosesan bahasa alami (*Natural Language Processing*) yang berfokus pada identifikasi, ekstraksi, dan klasifikasi opini atau emosi yang terkandung dalam teks [50]. Secara konseptual, analisis sentimen bertujuan untuk menentukan orientasi polaritas suatu pernyataan, seperti positif, negatif, atau netral, terhadap entitas, peristiwa, maupun topik tertentu [51]. Analisis sentimen dipandang sebagai bagian dari opinion mining yang tidak hanya menilai polaritas, tetapi juga mempertimbangkan intensitas emosi, subjektivitas, serta konteks linguistik yang

memengaruhi makna suatu ujaran [52]. Penerapannya mencakup berbagai bidang, seperti evaluasi ulasan produk, analisis opini publik, pemantauan media sosial, dan pengambilan keputusan berbasis persepsi pengguna. Seiring kemajuan arsitektur berbasis *Transformer* seperti BERT dan GPT, kemampuan model dalam memahami konteks, ambiguitas, serta hubungan semantik antar kata semakin meningkat [53]. Perkembangan ini mendorong analisis sentimen menjadi lebih adaptif, akurat, dan mampu menangani variasi bahasa yang dinamis di berbagai domain aplikasi.

2.2.4 Deep Learning

Deep Learning adalah bidang pembelajaran mesin yang menggunakan jaringan saraf tiruan multilayer untuk mengekstrak dan mempelajari representasi data hierarkis [54]. Teknologi ini meniru cara otak manusia memproses informasi dengan membangun struktur berlapis yang terdiri dari *neuron* buatan, di mana setiap lapisan mampu menangkap pola-pola kompleks pada data masukan [55]. Dalam penerapannya, *deep learning* sangat efektif untuk menangani tugas-tugas seperti pengenalan gambar, pemrosesan bahasa alami, dan pengenalan suara, berkat kemampuannya untuk mempelajari fitur secara otomatis tanpa memerlukan rekayasa fitur manual [56]. Model-model seperti *Convolutional Neural Networks* (CNN), *Recurrent Neural Networks* (RNN), dan *Transformer* adalah beberapa contoh arsitektur populer dalam *deep learning* yang telah merevolusi berbagai bidang seperti kesehatan, keuangan, dan teknologi informasi [57]. Keberhasilan *deep learning* sangat bergantung pada ketersediaan data yang besar dan sumber daya komputasi yang mumpuni, yang memungkinkan pelatihan model dalam skala besar untuk menghasilkan prediksi yang akurat dan berdaya guna [58].

2.2.5 Data Pre-processing

Data preprocessing adalah langkah awal yang esensial dalam menganalisis data, khususnya dari platform seperti *Twitter*, untuk memastikan bahwa data yang digunakan memiliki kualitas tinggi, relevansi, dan siap untuk dianalisis [59]. Data di Twitter memiliki karakteristik unik, seperti teks singkat (280 karakter), banyaknya penggunaan simbol (*@mention*, *#hashtag*), *URL*, emotikon, hingga singkatan yang memerlukan penanganan khusus [60]. *Pre-processing* data Twitter

bertujuan untuk mengubah data mentah menjadi format yang terstruktur, bersih, dan lebih mudah dipahami oleh model analitik [61]. Langkah-langkah utama dalam data preprocessing untuk data Twitter antara lain [59], [61]:

1) Penghapusan Teks Tidak Relevan

   Tahap ini bertujuan untuk menghilangkan elemen-elemen teks yang tidak berkontribusi terhadap makna utama tweet, seperti *URL*, *mention* (*@username*), serta elemen tertentu dari *hashtag* (*#topic*). *URL* umumnya hanya berupa tautan eksternal, sedangkan mention merujuk pada akun pengguna lain dan tidak selalu berkaitan langsung dengan isi pesan. *Hashtag* dapat dihapus atau diekstraksi tergantung pada kebutuhan analisis. Proses ini membantu memfokuskan analisis pada konten inti teks.

2) Penormalan Teks

   Penormalan dilakukan untuk menjaga konsistensi data. Proses ini mencakup pengubahan seluruh huruf menjadi huruf kecil (*case folding*) serta konversi slang atau singkatan ke dalam bentuk standar. Misalnya, kata "u" diubah menjadi "*you*" dan "*gr8*" menjadi "*great*". Penormalan bertujuan agar model dapat mengenali kata dengan representasi yang seragam dan mengurangi variasi token yang tidak perlu.

3) Pembersihan Data

   Tahap pembersihan meliputi penghapusan karakter khusus, tanda baca berlebihan, serta simbol yang tidak relevan dengan analisis. Contohnya, penggunaan tanda seru berulang seperti "!!!" dapat disederhanakan atau dihapus sesuai kebutuhan penelitian. Proses ini bertujuan untuk mengurangi *noise* dalam data sehingga meningkatkan kualitas representasi teks.

4) Tokenisasi

   Tokenisasi merupakan proses pemecahan teks menjadi unit-unit yang lebih kecil, biasanya berupa kata atau token. Sebagai contoh, kalimat "*I love AI*!" akan diubah menjadi ["*I*", "*love*", "*AI*"]. Tahap ini penting karena sebagian besar metode analisis teks bekerja pada level token.

5) *Stopword Removal*

Stopword adalah kata-kata umum yang sering muncul namun tidak memberikan makna signifikan dalam analisis, seperti "and", "the", atau "yang". Dengan menghapus *stopword*, teks menjadi lebih fokus pada kata-kata yang memiliki nilai informatif lebih tinggi. Namun, dalam konteks tertentu seperti analisis sentimen, beberapa *stopword* yang memengaruhi makna (misalnya "tidak" atau "not") perlu dipertimbangkan untuk tetap dipertahankan.

6) Lematisasi atau *Stemming*

Lematisasi dan *stemming* bertujuan untuk mengubah kata ke bentuk dasarnya. Stemming menghilangkan imbuhan untuk mendapatkan akar kata, misalnya "*running*" menjadi "*run*", sedangkan lemmatisasi mempertimbangkan konteks sehingga kata seperti "*better*" dapat diubah menjadi "*good*". Proses ini membantu mengurangi variasi morfologis sehingga meningkatkan konsistensi analisis.

7) Pengenalan dan Ekstraksi Entitas Khusus

Karakteristik unik Twitter seperti hashtag, mention, dan emoji dapat diperlakukan sebagai fitur tambahan dalam analisis. Hashtag sering merepresentasikan topik tertentu, mention menunjukkan interaksi antar pengguna, dan emoji dapat mengandung informasi sentimen. Oleh karena itu, entitas-entitas ini dapat diekstraksi dan dimanfaatkan sebagai variabel dalam proses analisis lanjutan.

2.2.6 Data Splitting

Data Splitting adalah proses pembagian dataset menjadi beberapa subset, biasanya terdiri atas data *training*, *validation*, dan *testing*, untuk memastikan model pembelajaran mesin (*machine learning*) dapat dilatih dan dievaluasi secara optimal [62]. Subset training digunakan untuk melatih model, *subset validation* untuk menyesuaikan hyperparameter, dan subset testing untuk mengukur kinerja model pada data baru yang tidak pernah dilihat sebelumnya [63]. Teknik data splitting yang umum meliputi *holdout method*, di mana dataset dibagi secara langsung (misalnya, 70% *training* dan 30% *testing*), dan *k-fold cross-validation*, yang membagi data menjadi k bagian dan melatih model k kali untuk

mendapatkan evaluasi yang lebih stabil [64]. *Data splitting* yang tepat sangat penting untuk mencegah *overfitting* atau *underfitting*, sehingga model dapat memprediksi dengan baik pada data nyata di luar dataset pelatihan [65].

2.2.7 Metrik Evaluasi

Metrik evaluasi adalah alat yang digunakan dalam menilai kinerja model pembelajaran mesin dengan membandingkan hasil prediksi model terhadap data yang telah dilabeli secara benar [66]. Pemilihan metrik evaluasi yang tepat sangat penting karena secara langsung mempengaruhi cara model dinilai dan dioptimalkan [67]. Metrik ini bervariasi tergantung pada jenis masalah yang dihadapi, seperti klasifikasi, regresi, *clustering*, atau tugas-tugas khusus lainnya seperti segmentasi gambar atau pemrosesan bahasa alami. Metrik yang umum digunakan dalam konteks sentimen analisis meliputi *accuracy*, *loss*, dan *execution* time, yang memberikan gambaran lengkap tentang keandalan, efisiensi, dan performa komputasi model [68].

1. *Accuracy*

   Accuracy merupakan metrik evaluasi yang digunakan untuk mengukur persentase jumlah prediksi yang benar dibandingkan dengan keseluruhan data yang dievaluasi. Metrik ini banyak diterapkan dalam tugas klasifikasi, baik biner maupun multikelas, karena sifatnya yang sederhana dan mudah dipahami. *Accuracy* memberikan gambaran umum mengenai kemampuan model dalam mengklasifikasikan data secara tepat dengan menghitung proporsi prediksi yang sesuai, yaitu kombinasi antara *True Positive* dan *True Negative*, terhadap seluruh observasi. Meskipun demikian, *accuracy* memiliki keterbatasan, terutama pada dataset yang tidak seimbang (*imbalanced*), karena nilai yang tinggi tidak selalu menunjukkan bahwa model mampu mengenali seluruh kelas dengan baik, khususnya kelas minoritas. Dalam penggunaannya, interpretasi *accuracy* perlu dilakukan secara hati-hati dan umumnya disertai dengan metrik evaluasi lainnya. Persamaan *accuracy* dapat dilihat pada Persamaan (2.1).

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}\ x\ 100\% \qquad (2.1)$$

Rumus 2.1 Rumus *Accuracy*

Berikut penjelasan dari persamaan (2.1):

a. TP *(True Positive)*: Kasus positif yang benar-benar teridentifikasi sebagai positif.
b. TN *(True Negative)*: Kasus negatif yang benar-benar teridentifikasi sebagai negatif.
c. FP *(False Positive)*: Kasus negatif yang salah teridentifikasi sebagai positif.
d. FN *(False Negative)*: Kasus positif yang salah teridentifikasi sebagai negatif.

2. *F1-Score*

F1-Score merupakan metrik evaluasi yang banyak digunakan dalam tugas klasifikasi untuk menilai performa model dengan mempertimbangkan keseimbangan antara precision dan recall [69]. Metrik ini sangat relevan ketika dataset bersifat tidak seimbang (*imbalanced*) atau ketika kesalahan *False Positive* dan *False Negative* sama-sama penting untuk diminimalkan [70]. Berbeda dengan *accuracy* yang menilai proporsi prediksi benar secara keseluruhan, F1-Score berfokus pada kualitas prediksi terhadap kelas target karena mengukur kemampuan model dalam menghasilkan prediksi yang tepat (precision) sekaligus menangkap sebanyak mungkin data positif yang benar (recall) [71]. Untuk mendefinisikan F1-Score, digunakan komponen pada *confusion matrix*, yaitu *True Positive (TP), False Positive (FP), False Negative (FN), dan True Negative (TN*) [72]. *F1-Score* didefinisikan sebagai *harmonic mean* dari *precision* dan *recall*, sehingga model akan memperoleh nilai tinggi hanya jika keduanya sama-sama tinggi [67]. Persamaan *F1-Score* tersebut dapat dilihat pada persamaan (2.2) [68].

$$F1\ Score = \frac{2TP}{2TP+FP+FN} \tag{2.2}$$

Rumus 2.2 Rumus *F1-Score*

Nilai *F1-Score* berada pada rentang 0 hingga 1 (atau 0% hingga 100%), di mana nilai yang lebih tinggi menunjukkan performa model yang lebih baik dalam menyeimbangkan ketepatan prediksi positif dan kemampuan

menangkap kelas positif [66]. Dalam bentuk klasifikasi multi-kelas, terdapat *Macro-F1* dimana nilai F1 dihitung untuk setiap kelas secara terpisah kemudian dirata-ratakan tanpa mempertimbangkan proporsi jumlah data pada masing-masing kelas, sehingga setiap kelas memiliki bobot yang sama dalam evaluasi. Perhitungan Macro-F1 dapat dilihat pada Persamaan (2.3) [69].

$$Macro\ F1 = \frac{1}{k}\sum_{k-1}^{K} F1_k \tag{2.3}$$

Rumus 2.3 Rumus *Macro F1*

3. *Loss*

Loss merupakan fungsi objektif yang digunakan untuk mengukur tingkat kesalahan antara hasil prediksi model dan nilai aktual pada data [73]. Dalam proses pelatihan model *machine learning* dan *deep learning*, *loss* berperan sebagai indikator utama yang menentukan arah pembaruan bobot melalui proses optimasi [74]. Semakin kecil nilai loss, semakin dekat prediksi model terhadap label yang sebenarnya, sehingga model dianggap memiliki performa yang lebih baik [75]. Jenis loss yang digunakan bergantung pada karakteristik tugas yang dikerjakan, misalnya *Mean Squared Error* untuk regresi dan *Cross-Entropy Loss* untuk klasifikasi [76]. Pada klasifikasi biner, yang hanya memiliki dua kelas, label positif umumnya direpresentasikan dengan nilai 1 dan label negatif dengan nilai 0. Model menghasilkan probabilitas untuk masing-masing kelas, seperti probabilitas kelas positif dan probabilitas kelas negatif, yang kemudian dibandingkan dengan label sebenarnya untuk menghitung besar kesalahan [77]. Perhitungan *loss* dapat dilihat pada Persamaan (2.4) [75].

$$L(y,p) = -(y\ \log(p) + (1-y)\log(1-\mathrm{p})) \tag{2.4}$$

Rumus 2.4 Rumus *Loss*

Berikut penjelasan dari persamaan (2.4):

a. $L(y,p)$: Loss untuk satu sampel.

b. y: Label sebenarnya, yang bernilai 1 untuk kelas positif dan 0 untuk kelas negatif.
c. p: Probabilitas prediksi untuk kelas positif ($y = 1$) yang dihasilkan oleh model.

4. *Execution Time*

Execution time (waktu eksekusi) adalah metrik evaluasi yang mengukur waktu yang diperlukan oleh suatu algoritma atau model untuk menyelesaikan tugas tertentu [78]. *Execution time* dinyatakan dalam satuan waktu, seperti detik (s), milidetik (ms), atau mikrodetik (μs), tergantung pada kebutuhan evaluasi [79]. *Execution time* tidak memiliki rumus universal yang baku karena biasanya tergantung pada implementasi perangkat lunak dan perangkat keras. Namun, dalam konteks pengukuran, *execution time* dihitung dengan mencatat waktu mulai dan waktu selesai dari proses tertentu [80]. Persamaan metrik *execution time* dapat didefinisikan sebagai persamaan (2.5) [81].

$$Execution\ Time = End\ Time - Start\ Time \qquad (2.5)$$

Rumus 2.5 Rumus *Execution Time*

2.2.8 Correlation Test

Uji korelasi merupakan teknik analisis statistik yang digunakan untuk mengidentifikasi dan mengukur kekuatan serta arah hubungan antara dua variabel [82]. Dalam berbagai publikasi ilmiah, uji korelasi dimanfaatkan untuk mengevaluasi keterkaitan antarvariabel sebelum dilakukan pemodelan lanjutan, serta untuk menguji hipotesis mengenai hubungan asosiatif antar konstruk penelitian. Secara umum, koefisien korelasi berada pada rentang −1 hingga +1, di mana nilai positif menunjukkan hubungan searah dan nilai negatif menunjukkan hubungan berlawanan arah [83]. Semakin mendekati nilai absolut 1, semakin kuat hubungan antarvariabel tersebut. Berbeda dengan analisis regresi, korelasi tidak mengindikasikan hubungan kausalitas, melainkan hanya mengukur derajat asosiasi statistik [84]. Pada penelitian ini digunakan *correlation test* sebagai berikut:

1. *Spearman Correlation*

Korelasi *Spearman* atau *Spearman Rank Correlation* merupakan metode korelasi non-parametrik yang banyak digunakan dalam penelitian empiris ketika data tidak memenuhi asumsi normalitas atau ketika variabel yang dianalisis berskala ordinal [85]. Koefisien korelasi *Spearman* dilambangkan dengan ρ (rho) dan dihitung berdasarkan selisih peringkat antar pasangan observasi [86]. Secara matematis, formulasi koefisien *Spearman* dinyatakan sebagai Persamaan (2.6) [87].

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2-1)} \tag{2.6}$$

Rumus 2.6 Rumus *Spearman Correlation*

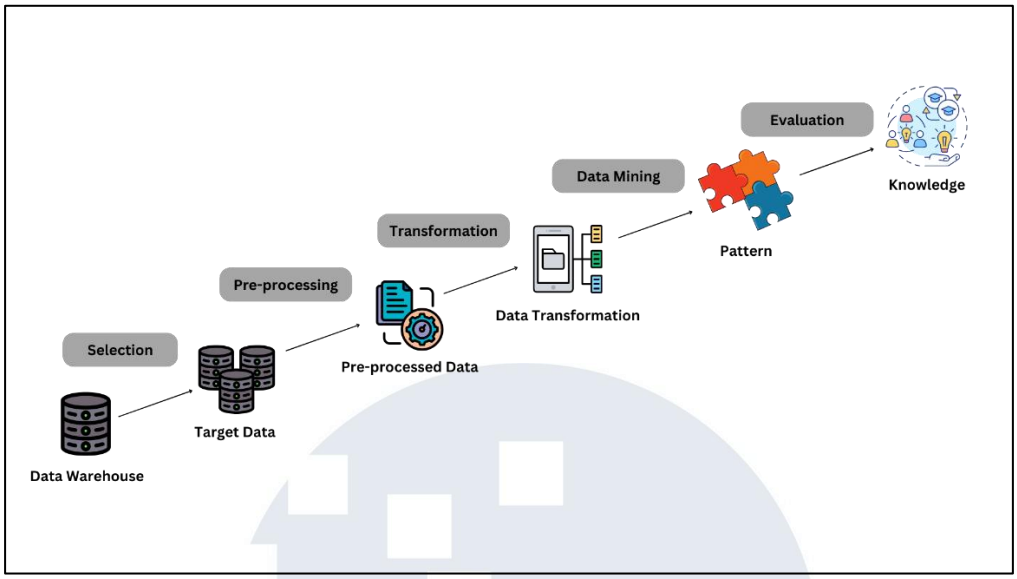
Berikut penjelasan dari persamaan (2.6):

a. ρ (rho) merupakan koefisien korelasi *Spearman* yang menunjukkan arah dan kekuatan hubungan antarvariabel, dengan rentang nilai −1 hingga +1.
b. d_i adalah selisih antara peringkat dua variabel pada observasi ke-i setelah data diubah menjadi ranking.
c. Σd_i^2 menunjukkan jumlah kuadrat selisih peringkat seluruh pasangan data dalam sampel.
d. n menyatakan jumlah sampel atau banyaknya pasangan observasi yang dianalisis.

2.3 Framework dan Algoritma

2.3.1 *Knowledge Discovery in Databases* (KDD)

Knowledge Discovery in Database (KDD) adalah proses yang digunakan untuk mengekstrak pengetahuan yang dapat dipahami dan berguna dari data besar, terutama data yang tidak terstruktur atau semi-terstruktur [88]. Proses KDD melibatkan beberapa tahapan yang terintegrasi untuk memperoleh pola atau informasi yang dapat dipakai untuk pengambilan keputusan atau prediksi [89]. KDD bukan hanya tentang mengumpulkan data, tetapi juga melibatkan analisis dan pemahaman yang mendalam terhadap data yang ada. Diagram alur *Knowledge Discovery in Database* dapat dilihat pada Gambar 2.1.



Gambar 2.1 *Framework* KDD

Gambar 2.1 menggambarkan alur dalam framework *Knowledge Discovery in Databases* (KDD) yang melibatkan lima tahapan utama: *Selection, Pre-processing, Transformation, Data Mining,* dan *Evaluation*. Setiap tahapan ini tidak bersifat linier, melainkan dapat berlangsung secara iteratif, yang memungkinkan kembali ke langkah sebelumnya jika hasil yang diperoleh tidak memenuhi harapan atau perlu diperbaiki. Proses ini memberi fleksibilitas untuk memperbaiki atau menyempurnakan data dan analisis pada setiap tahapannya. Berikut penjelasan lebih lanjut mengenai masing-masing tahap dalam *framework* KDD [90]:

1. *Selection*

   Pada tahap ini, data yang relevan dengan tujuan analisis dipilih dari berbagai sumber data yang ada. Proses ini melibatkan identifikasi data yang dibutuhkan serta pemilihan subset yang akan digunakan dalam analisis lebih lanjut. Pemilihan data yang tepat sangat penting untuk memastikan bahwa hasil yang diperoleh relevan dengan masalah yang ingin diselesaikan.

2. *Pre-processing*

   Data yang telah dipilih sering kali mengandung kekurangan seperti nilai yang hilang, duplikat, atau inkonsistensi. Tahap pembersihan data bertujuan untuk memperbaiki masalah tersebut dengan menghapus atau mengganti

data yang tidak valid, serta menyaring data yang tidak diperlukan, sehingga dataset menjadi lebih bersih dan siap untuk dianalisis.

3. *Transformation*

   Setelah data dibersihkan, proses transformasi dilakukan untuk mengubah data ke dalam format yang lebih sesuai untuk analisis atau model pembelajaran mesin. Ini dapat mencakup normalisasi, penggabungan fitur, atau pengurangan dimensi untuk menyederhanakan data tanpa mengorbankan informasi penting yang terkandung di dalamnya.

4. *Data Mining*

   Data Mining adalah tahap inti di mana algoritma analitik diterapkan untuk mengekstrak pola, tren, atau hubungan yang tersembunyi dalam data. Teknik yang digunakan bisa berupa klasifikasi, regresi, clustering, atau asosiasi, tergantung pada tujuan analisis. Hasil dari proses ini adalah pengetahuan eksplisit yang dapat digunakan untuk membuat keputusan atau merumuskan hipotesis lebih lanjut.

5. *Evaluation*

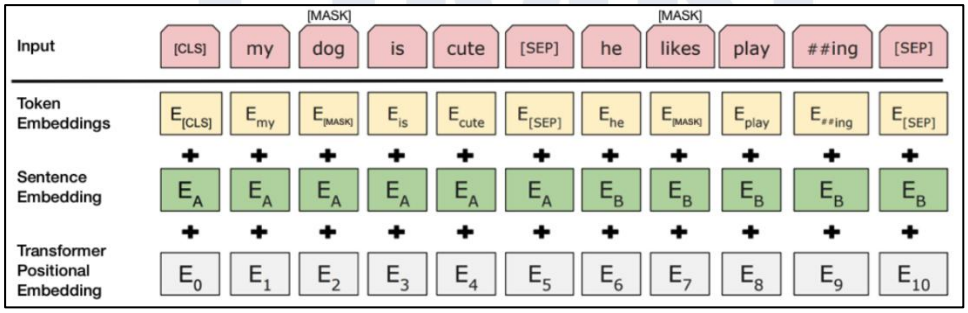
   Setelah penambangan data, hasil yang diperoleh dievaluasi untuk menentukan kualitas dan kegunaannya. Evaluasi dilakukan dengan mengukur kinerja model atau pola yang ditemukan menggunakan metrik yang sesuai, seperti akurasi, presisi, atau recall. Tahap ini memastikan bahwa pengetahuan yang ditemukan benar-benar bermanfaat dan relevan dengan tujuan yang ingin dicapai.

Hasil akhir dari framework KDD (Knowledge Discovery in Databases) adalah pengetahuan atau knowledge yang dapat diterapkan untuk mendukung pengambilan keputusan atau memberikan wawasan baru. Pengetahuan ini dapat berupa informasi yang terstruktur, model prediktif, atau temuan yang mendalam tentang data yang digunakan [91]. Pengetahuan yang dihasilkan dari proses KDD ini sering digunakan untuk meningkatkan pemahaman terhadap fenomena

tertentu, merumuskan kebijakan, atau mengoptimalkan proses bisnis di berbagai bidang, mulai dari penelitian hingga aplikasi industry.

2.3.2 *Bidirectional Encoder Representation from Transformers* (BERT)

BERT (*Bidirectional Encoder Representations from Transformers*) adalah model berbasis transformer yang dirancang untuk memahami konteks kata secara lebih mendalam dengan memproses informasi dari arah kiri ke kanan dan kanan ke kiri secara bersamaan [92]. Pendekatan ini memungkinkan BERT untuk menangkap makna kata dalam konteks yang lebih luas, mengatasi masalah polisemi (kata yang memiliki banyak arti tergantung konteks), dan memahami hubungan antar kata dengan lebih akurat [93]. Sebagai contoh, kata "bisa" dapat dimaknai sebagai "dapat" atau "racun", tergantung konteks kalimatnya, yang sulit dilakukan oleh model-model sebelumnya [94]. BERT telah terbukti efektif dalam meningkatkan performa berbagai tugas NLP dengan sedikit modifikasi. Dengan melatih model pada dataset tertentu, BERT mampu menyelesaikan berbagai tugas seperti menjawab pertanyaan, klasifikasi teks, dan analisis sentimen [95]. Keunggulan lainnya adalah kemampuannya untuk memproses hingga 11 bahasa, menjadikannya fleksibel dan dapat diterapkan dalam berbagai aplikasi [96]. Secara keseluruhan, BERT mengubah paradigma pemrosesan bahasa alami dengan memberikan pemahaman konteks yang lebih baik, serta efisiensi dan fleksibilitas yang tinggi dalam berbagai tugas NLP [36].

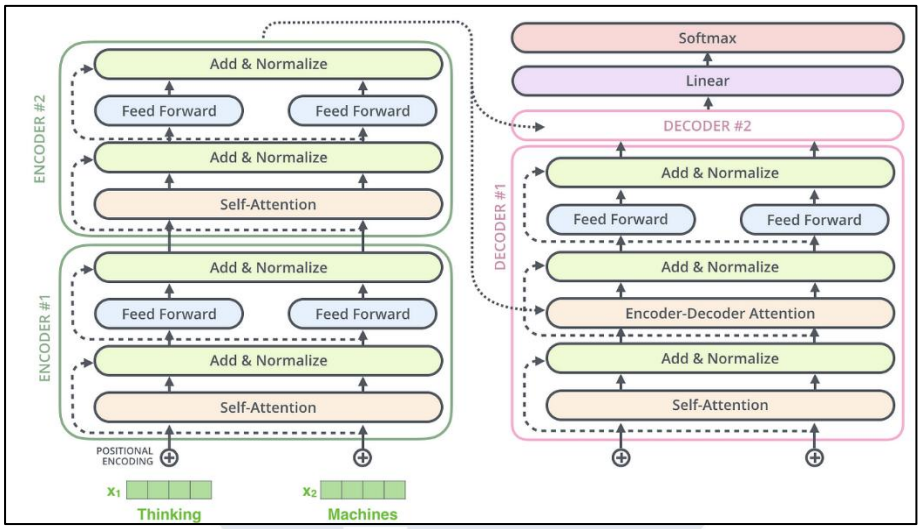


Gambar 2.2 Struktur *Embedding* dalam Model BERT

Sumber: [97]

Struktur embedding dalam model BERT, seperti yang ditunjukkan pada Gambar 2.2, melibatkan tiga komponen utama: token *embedding*, positional embedding, dan *sentence embedding*. Token *embedding* mengonversi setiap token

menjadi vektor yang mewakili makna semantik, sementara positional embedding memberikan informasi urutan posisi token dalam kalimat. Sentence embedding membedakan kalimat pertama dan kedua dalam input. Ketiga jenis embedding ini dijumlahkan untuk membentuk representasi akhir setiap token, yang kemudian diproses lebih lanjut oleh model untuk tugas pemahaman bahasa alami [97].



Gambar 2.3 Arsitektur *Transformer* dengan Lapisan *Encoder* dan *Decoder*.
Sumber: [98]

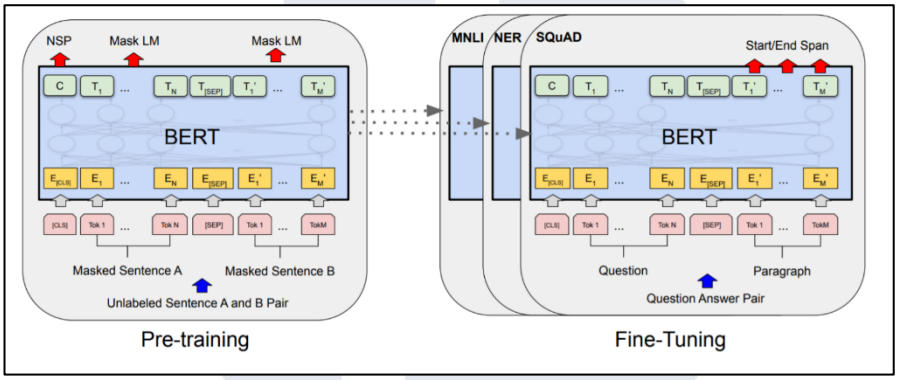
Gambar 2.3 menunjukkan keseluruhan arsitektur *Transformer*, yang terdiri dari lapisan *Encoder* dan *Decoder*, namun BERT hanya menggunakan bagian *Encoder*. Model BERT (*Bidirectional Encoder Representations from Transformers*) menggunakan arsitektur *Transformer* yang dilatih secara *bidirectional*, memungkinkan model untuk memahami konteks sebuah kata dengan memperhatikan kata-kata di sekitarnya, baik yang ada di sebelah kiri maupun kanan [98]. Model ini berbeda dengan ELMo dan GPT, yang hanya membaca teks dalam satu arah [99]. BERT dilatih menggunakan *Masked Language Model* (MLM), di mana model belajar mengisi kata yang hilang berdasarkan konteks sekitar tanpa menggunakan label, menjadikannya model *unsupervised* [100]. Output BERT berupa distribusi probabilitas yang dihitung dengan fungsi *Softmax* [101].

Dalam proses klasifikasi teks, BERT memanfaatkan token khusus yang diberi tanda [CLS] di awal input [35]. Token ini berfungsi untuk menghitung

representasi teks (h), yang kemudian dikalikan dengan bobot tersembunyi lainnya untuk melatih model dalam menjalankan tugas tertentu (W). Selanjutnya, untuk mengklasifikasikan teks, BERT menambahkan lapisan *Softmax* pada 18 lapisan *Encoder* terakhirnya untuk menghitung probabilitas dari label kelas [101]. Rumus untuk fungsi *Softmax* dapat dilihat pada Persamaan (2.7) [102].

$$p(c|\boldsymbol{h}) = softmax(W\boldsymbol{h} + \boldsymbol{b}) \tag{2.7}$$

Rumus 2.7 Rumus *Softmax* untuk Klasifikasi Teks *BERT*



Gambar 2.4 Arsitektur *Pre-training* dan *Fine-tuning* BERT

Sumber: [97]

Gambar 2.4 menunjukkan dua kerangka kerja utama yang ada dalam BERT, yaitu *pre-training* dan *fine-tuning*, yang masing masing memiliki fungsi berbeda namun saling melengkapi. Pada kerangka kerja *pre-training*, model dilatih menggunakan data tanpa label atau unlabeled data melalui sejumlah tugas *pre-training* yang dirancang untuk mempelajari representasi bahasa secara umum. Proses ini bertujuan membangun pemahaman kontekstual yang mendalam terhadap struktur dan makna bahasa berdasarkan pola kemunculan kata dalam korpus berskala besar. Sementara itu, pada kerangka kerja *fine-tuning*, model BERT diinisialisasi menggunakan parameter hasil *pre-training* tersebut, kemudian disesuaikan lebih lanjut dengan data berlabel sesuai kebutuhan tugas *downstream* tertentu. Meskipun seluruh model berangkat dari parameter awal yang sama hasil *pre-trained*, setiap tugas *downstream* seperti klasifikasi teks, analisis sentimen, atau question answering memerlukan proses *fine-tuning* yang spesifik, baik dari segi penyesuaian lapisan akhir maupun konfigurasi parameter

pelatihannya [97]. Tabel 2.2 merupakan *pseudocode* algoritma pre-trained dalam BERT yang menggambarkan alur proses pelatihan awal tersebut secara sistematis.

Tabel 2.2 *Pseudocode* Algoritma *Pre-Trained* BERT

No	Langkah
1	Muat data teks besar (misalnya *Wikipedia* atau *BookCorpus*) untuk digunakan dalam pelatihan model.
2	Tokenisasi teks menjadi *subword* menggunakan algoritma seperti *WordPiece*.
3	Bagi data teks yang sudah ditokenisasi menjadi batch kecil agar lebih mudah diproses model.
4	Inisialisasi model BERT berbasis arsitektur *transformer* dengan lapisan *encoder*.
5	Tentukan *optimizer* (misalnya *Adam*) dan fungsi loss (misalnya *Cross-Entropy*) untuk pembaruan bobot.
6	Lakukan *loop* utama untuk pelatihan model, mengulang *epoch* hingga *konvergen*.
7	Secara acak pilih 15% token dalam setiap *batch* untuk *dimask* (menggunakan token [MASK]).
8	Siapkan pasangan kalimat untuk tugas prediksi urutan kalimat *(Next Sentence Prediction)*.
9	Lakukan *forward pass* melalui model dengan input untuk kedua tugas (MLM dan NSP).
10	Hitung *loss* untuk tugas MLM (prediksi token yang dimask).
11	Hitung *loss* untuk tugas NSP (apakah kalimat kedua mengikuti kalimat pertama).
12	Total *loss* adalah penjumlahan dari MLM *loss* dan NSP *loss*.
13	Lakukan *backpropagation* untuk memperbarui bobot model berdasarkan total *loss*.
14	Perbarui bobot model dengan *optimizer*.
15	Lacak kemajuan pelatihan, seperti mencatat *loss* setiap beberapa langkah.
16	Setelah selesai melatih model, simpan model yang telah *pre-trained* untuk digunakan selanjutnya.

Pseudocode pada Tabel 2.2 menjelaskan langkah-langkah dalam pre-training BERT, sebuah model berbasis *Transformer* untuk pemahaman bahasa alami. Proses dimulai dengan pemuatan data teks dalam jumlah besar yang kemudian ditokenisasi menggunakan algoritma seperti *WordPiece* untuk mengubah teks menjadi representasi subword agar mampu menangani kosakata yang beragam. Token yang dihasilkan dikonversi menjadi token ID dan diproses dalam bentuk batch untuk pelatihan. BERT dilatih menggunakan pendekatan *self-supervised learning* melalui dua tugas utama, yaitu *Masked Language Modeling* (MLM), di mana 15% token dimask secara acak dan diprediksi berdasarkan konteks dua arah, serta *Next Sentence Prediction* (NSP), yang bertujuan memprediksi hubungan logis antara dua kalimat. Parameter model diperbarui menggunakan *backpropagation* dengan *optimizer Adam* untuk meminimalkan fungsi loss gabungan dari kedua tugas tersebut [103]. Proses ini seringkali dilakukan selama beberapa *epoch* hingga model mampu menghasilkan

representasi bahasa yang kontekstual dan kaya informasi, kemudian model disimpan untuk tahap *fine-tuning* pada tugas spesifik. Tabel 2.3 merupakan *pseudocode* algoritma *fine-tuning*.

Tabel 2.3 *Pseudocode* Algoritma *Fine-Tuning* BERT

No	Langkah
1	Muat model BERT yang telah *pre-trained*.
2	Muat dataset tugas spesifik (misalnya, klasifikasi teks atau analisis sentimen).
3	Tokenisasi dataset menggunakan *tokenizer* BERT yang sudah dilatih sebelumnya.
4	Siapkan *batch input* dari dataset untuk pelatihan.
5	Tentukan *task-specific head* (misalnya, lapisan klasifikasi untuk klasifikasi teks).
6	Gabungkan model BERT dengan *task-specific head*.
7	Tentukan *optimizer* (misalnya *Adam*) dan fungsi *loss* yang sesuai dengan tugas (misalnya, *Cross-Entropy* untuk klasifikasi).
8	Lakukan loop pelatihan untuk beberapa epoch, dengan langkah berikut: 1. Lakukan *forward pass* pada *input batch* menggunakan model BERT yang digabung dengan *task-specific head*. 2. Hitung *loss* berdasarkan output model dan label sebenarnya. 3. Lakukan *backpropagation* untuk memperbarui bobot model. 4. Perbarui bobot model menggunakan *optimizer*.
9	Lacak kemajuan pelatihan, misalnya dengan mencatat *loss* dan akurasi setiap *epoch*.
10	Setelah pelatihan selesai, evaluasi model menggunakan data validasi atau pengujian.
11	Simpan model yang telah *fine-tuned* untuk digunakan dalam aplikasi atau prediksi lebih lanjut.

Pseudocode pada Tabel 2.3 menggambarkan proses *fine-tuning* BERT, yang dimulai dengan memuat model BERT yang telah *pre-trained* dan kemudian melatihnya pada dataset tugas spesifik, seperti klasifikasi teks. Selama proses *fine-tuning*, model BERT dihubungkan dengan *task-specific head*, seperti lapisan klasifikasi untuk tugas tertentu. *Optimizer* dan fungsi *loss* disesuaikan dengan tugas, dan pelatihan dilakukan dengan beberapa *epoch* menggunakan teknik *backpropagation* untuk memperbarui bobot model. Setelah *fine-tuning*, model dapat dievaluasi dan disimpan untuk digunakan pada aplikasi nyata [104].

CryptoBERT dapat dipahami sebagai varian BERT yang diadaptasi khusus untuk domain cryptocurrency melalui pendekatan domain-adaptive pretraining (sering disebut post-training), yaitu melanjutkan proses pre-training pada korpus teks kripto (misalnya percakapan media sosial) sebelum dilakukan fine-tuning untuk tugas tertentu seperti klasifikasi sentimen [27]. Secara teori,

keunggulan utama CryptoBERT dibanding BERT umum terletak pada kemampuannya mengurangi domain mismatch: model menjadi lebih peka terhadap jargon kripto, simbol/ticker, singkatan, gaya bahasa informal, serta pola ekspresi sentimen yang khas di komunitas kripto, sehingga representasi kontekstual yang dihasilkan lebih relevan untuk tugas downstream di domain tersebut. Dalam implementasinya, CryptoBERT umumnya memulai dari model berbasis BERT yang sudah kuat pada teks media sosial (misalnya menggunakan tokenisasi byte-level BPE), kemudian melakukan post-training dengan tugas Masked Language Modeling (MLM) dengan cara memask sebagian token (umumnya sekitar 15%) dan memprediksi token asli berdasarkan konteks dua arah. Objektif pelatihan ini meminimalkan Loss MLM yang secara umum dirumuskan sebagai Persamaan (2.8) [24].

$$\mathcal{L}_{MLM}(\theta) = -\sum_{i \in M} \log P_\theta(x_i \mid \tilde{x}) \tag{2.8}$$

Rumus 2.8 Rumus *Loss Masked Language Modeling*

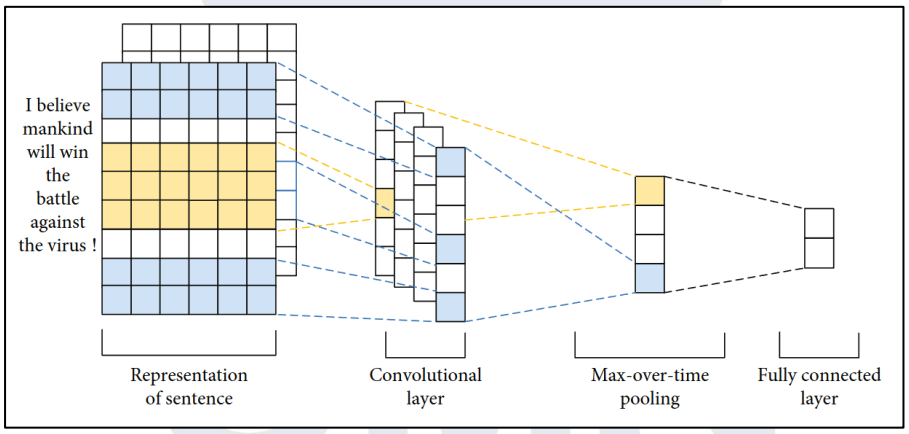
Berikut penjelasan dari Persamaan (2.8):

a. $\mathcal{L}_{MLM}(\theta)$: Nilai loss Masked Language Modeling yang digunakan sebagai fungsi objektif dalam proses pelatihan model.
b. θ: Parameter atau bobot model (misalnya bobot CryptoBERT) yang diperbarui selama proses training.
c. M : Himpunan indeks token yang dimask (sekitar 15% dari total token dalam satu sekuens).
d. x_i : Token asli pada posisi ke-i yang harus diprediksi oleh model.
e. $\tilde{x}$: Input teks yang telah dimodifikasi dengan proses masking.
f. $P_\theta(x_i \mid \tilde{x})$: Probabilitas prediksi token asli x_i berdasarkan konteks dari input yang telah dimask.

2.3.3 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah salah satu arsitektur pembelajaran mendalam yang pertama kali diperkenalkan oleh LeCun et al. pada tahun 1998, dirancang khusus untuk menangani data spasial seperti gambar dan

data berurutan [105]. CNN dikenal sebagai jaringan saraf *feedforward* yang efektif dalam mengenali pola lokal pada data dengan memanfaatkan sifat persepsi lokal dan pembagian bobot [106]. Hal ini memungkinkan pengurangan jumlah parameter, sehingga meningkatkan efisiensi pelatihan model tanpa kehilangan informasi penting. CNN juga memiliki keunggulan dalam mendeteksi fitur hierarkis, dimulai dari pola sederhana hingga pola kompleks [107]. Dalam konteks klasifikasi teks, CNN bekerja pada representasi numerik teks yang telah diubah menjadi word embedding, sehingga setiap kalimat direpresentasikan sebagai matriks berukuran panjang urutan kata × dimensi *embedding* [108]. Melalui mekanisme konvolusi satu dimensi (1D *convolution*), model dapat mendeteksi pola lokal seperti frasa penting atau kombinasi kata (*n-gram*) yang berkontribusi terhadap makna kalimat. Dalam pengembangannya, CNN terdiri dari tiga jenis lapisan utama, yaitu *convolutional layer*, *pooling layer*, *fully connected* layer [109]:



Gambar 2.5 Arsitektur *CNN*

Sumber: [110]

1. *Convolutional Layer*

   Lapisan konvolusi bertugas mengekstrak fitur-fitur utama dari data input melalui penerapan filter pada area tertentu dari data. Filter ini, yang terdiri dari bobot-bobot terlatih, bertanggung jawab untuk mendeteksi pola penting seperti tepi, tekstur, atau kurva pada data. Proses ini dilakukan secara berulang pada seluruh area input, menghasilkan representasi peta fitur (*feature map*) yang mencerminkan karakteristik signifikan dari data asli

[110]. Dalam konteks teks, filter ini diterapkan pada jendela kata untuk menghasilkan representasi baru berdasarkan fitur semantik yang ditemukan. Representasi dari lapisan konvolusi dapat dilihat pada Gambar 2.5. Rumus operasi konvolusi satu dimensi (*1D Convolution*) dapat dilihat pada Persamaan (2.9) [111].

$$c_i = f(W \cdot X_{i:i+h-1} + b) \quad (2.9)$$

Rumus 2.9 Rumus Operasi Konvolusi Satu Dimensi

Berikut penjelasan dari Persamaan (2.8):

a. c_i : Nilai fitur (feature) pada posisi ke-i hasil konvolusi.
b. $X_{i:i+h-1}$: Submatriks input dari posisi ke-i hingga i+h−1.
c. W : Matriks bobot filter (kernel) berukuran $h \times d$.
d. h : Ukuran filter (jumlah kata dalam satu jendela).
e. b : Bias.

2. *Pooling Layer*

Lapisan pooling berfungsi untuk mengurangi dimensi spasial dari peta fitur tanpa mengubah kedalaman volume data, sehingga mengurangi jumlah parameter sekaligus kompleksitas model. Proses ini juga membantu mengurangi risiko overfitting. Salah satu metode yang sering digunakan adalah max pooling, yang memilih nilai terbesar dari sekumpulan elemen dalam area tertentu untuk mewakili fitur tersebut. Dengan demikian, lapisan ini menyederhanakan data dan mempertahankan informasi paling signifikan yang mendukung proses klasifikasi selanjutnya. Representasi dari lapisan pooling dapat dilihat pada Gambar 2.5.

3. *Fully Connected Layer*

Lapisan fully connected menghubungkan setiap neuron dalam lapisan saat ini ke semua neuron di lapisan berikutnya, memungkinkan integrasi fitur yang telah diekstrak untuk membentuk prediksi akhir. Lapisan ini juga mendukung regularisasi model melalui fungsi dropout, yang secara acak menonaktifkan sejumlah neuron selama pelatihan untuk mencegah overfitting. Pada akhirnya, fungsi aktivasi seperti softmax digunakan untuk

mengklasifikasikan data ke dalam kelas yang telah ditentukan. Representasi dari lapisan fully connected dapat dilihat pada Gambar 2.5.

2.3.4 Attention Mechanism

Attention mechanism adalah pendekatan dalam pembelajaran mesin, khususnya *deep learning*, yang memungkinkan model fokus pada elemen data yang paling relevan untuk tugas tertentu [112]. Mekanisme ini pertama kali diterapkan dalam penerjemahan mesin, di mana model memilih kata atau frasa penting dari sumber bahasa yang relevan dengan kata dalam bahasa target [113]. Dalam *Natural Language Processing* (NLP), *attention* digunakan untuk memahami hubungan kontekstual dalam teks panjang, seperti pada model Transformer (misalnya, BERT dan GPT) yang menggunakan *self-attention* untuk menangkap dependensi antar kata di seluruh teks [114]. Mekanisme ini juga diadopsi dalam *Computer Vision* untuk memprioritaskan fitur penting dalam gambar, meningkatkan akurasi deteksi objek dan segmentasi [115].

Dengan fleksibilitas untuk menangani data tak terstruktur dan efisiensi dalam memahami hubungan kompleks, *attention mechanism* telah menjadi komponen kunci dalam model pembelajaran modern. Salah satu bentuk attention yang paling kuat adalah *self-attention*, yang unggul dalam menangkap hubungan kontekstual antar elemen dalam data, bahkan ketika elemen-elemen tersebut berjauhan secara posisi [116]. Dalam konteks penelitian hybrid *Transformer*-CNN, penerapan *self-attention* dapat memberikan nilai tambah yang signifikan. Self-attention pada Transformer membantu memahami konteks global dalam teks secara mendalam dengan menangkap hubungan antar kata meskipun berjauhan dalam urutan kalimat, sementara CNN melengkapi dengan analisis pola lokal melalui ekstraksi fitur berbasis jendela kata (*n-gram*) [117]. Pada arsitektur hybrid, mekanisme *Attention* umumnya diterapkan setelah proses ekstraksi fitur oleh CNN atau pada tahap integrasi representasi, sehingga berfungsi untuk memberikan bobot kepentingan pada fitur-fitur yang telah dihasilkan CNN [118]. Persamaan *Self-Attention* dapat didefinisikan pada Persamaan (2.8) [119].

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V \tag{2.8}$$

Rumus 2.10 Rumus *Self Attention*

Berikut penjelasan dari Persamaan (2.8):

a. Q (Query): Vektor untuk menilai relevansi antar elemen.
b. K (Key): Vektor untuk mencocokkan dengan *query*.
c. V (Value): Vektor berisi informasi input.
d. d_k: Dimensi untuk normalisasi skor.
e. *Softmax*: Mengubah skor jadi probabilitas.
f. T (Transpose): Membalik baris dan kolom matriks.

2.4 *Software* dan *Tools* yang digunakan

2.4.1 *Python*

Python, yang pertama kali diperkenalkan oleh Guido van Rossum pada tahun 1991, merupakan bahasa pemrograman tingkat tinggi yang bersifat interpreted dan dinamis dengan sintaks yang menekankan keterbacaan kode [120]. Python mendukung berbagai paradigma pemrograman seperti prosedural, berorientasi objek, dan fungsional, sehingga banyak digunakan dalam pengembangan aplikasi, analisis data, serta penelitian komputasional [121]. Ekosistem *Python* didukung oleh berbagai pustaka ilmiah serta sistem manajemen paket seperti pip dan virtual *environment* (*venv*) yang membantu menjaga konsistensi dependensi dan reprodusibilitas penelitian [122]. Dalam analisis data, pustaka *NumPy* digunakan untuk komputasi numerik berbasis array multidimensi, sedangkan pandas menyediakan struktur data seperti *Series* dan *DataFrame* yang memudahkan proses manipulasi, pembersihan, dan transformasi data [123]. Selain itu, Python juga banyak dimanfaatkan dalam pengembangan machine learning karena kompatibilitasnya dengan berbagai pustaka seperti *scikit-learn*, *TensorFlow*, dan *PyTorch* [124]. Dukungan komunitas yang besar serta dokumentasi yang lengkap menjadikan Python sebagai salah satu bahasa pemrograman yang populer dalam penelitian dan pengembangan teknologi berbasis data.

2.4.2 *PyCharm*

PyCharm merupakan *integrated development environment* (IDE) untuk Python yang dikembangkan oleh *JetBrains* dan dirancang untuk meningkatkan produktivitas pengembangan melalui fitur seperti *code completion* kontekstual,

analisis kode, *refactoring*, serta debugging terintegrasi [125]. PyCharm juga mendukung pengelolaan interpreter dan virtual environment untuk membantu isolasi dependensi proyek, sehingga memudahkan pengembangan aplikasi yang lebih terstruktur dan reprodusibel [126]. Dalam konteks analisis data dan komputasi ilmiah, *PyCharm* menyediakan dukungan untuk *Jupyter Notebook*, eksekusi dan *debugging* sel kode, serta visualisasi struktur data seperti *array*, *DataFrame*, dan *Series*, sehingga banyak digunakan sebagai lingkungan pengembangan yang mendukung pemrograman *Python*, *data science*, dan *machine learning* secara lebih efisien [127]. Selain itu, *PyCharm* menyediakan integrasi dengan sistem kontrol versi seperti Git yang memudahkan pengelolaan perubahan kode dalam proyek pengembangan perangkat lunak.

2.4.3 *Tensorflow*

TensorFlow adalah sebuah *framework open-source* yang dikembangkan oleh *Google* untuk memfasilitasi pengembangan dan implementasi model pembelajaran mesin (*machine learning*) dan pembelajaran mendalam (*deep learning*) [128]. *TensorFlow* menyediakan berbagai alat dan pustaka yang memungkinkan pengguna untuk merancang, melatih, dan mengevaluasi model dengan cara yang efisien dan skalabel, baik untuk aplikasi penelitian maupun industri [129]. Framework ini mendukung komputasi numerik menggunakan graf aliran data, di mana tensor (struktur data multidimensi) diproses dalam berbagai operasi matematis yang saling terhubung. *TensorFlow* juga dilengkapi dengan *TensorFlow Keras*, sebuah *API* tingkat tinggi yang menyederhanakan pembuatan dan pelatihan model jaringan saraf [130]. Keunggulan utama *TensorFlow* adalah kemampuannya untuk berjalan di berbagai platform, termasuk CPU, GPU, dan TPU, serta mendukung berbagai jenis perangkat seperti *desktop*, *server*, dan perangkat mobile [131].