

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu merupakan dasar yang sangat penting dalam proses penelitian, karena membantu memahami perkembangan topik yang sedang diteliti dan mengidentifikasi kesenjangan atau peluang untuk kontribusi baru. Penelitian terdahulu juga berfungsi sebagai sumber inspirasi yang nantinya membantu pelaksanaan penelitian. Berikut daftar penelitian terdahulu yang dijadikan bahan referensi dijelaskan pada Tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti (Tahun)	Judul	Hasil
1.	Rhini Fatmasaria, Riska Kurnia Septianib, Tuahta Hasiolan Pinemb, Dedik Fabiyantob, Windu Gatab (2023)	Implementasi Algoritma BERT Pada Komentar Layanan Akademik dan Non Akademik Universitas Terbuka di Media Sosial[3]	Pada penelitian ini dilakukan analisa terkait komentar sosial media twitter dan tiktok sebanyak 658 data dengan hasil 281 sentimen positif dan 404 data <i>sentimen</i> negatif menggunakan beberapa teknik yang diuji meliputi SVM dengan 88.34% akurasi, Naïve Bayes – MultinomialNB akurasi 87.37%, Naïve Bayes – GausiyanNB akurasi 87.37%, K-NN 85.43%, Decision tree 83% dan logistic regression 83% kemudian setelah dilakukan metode machine learning dilakukan klasifikasi dengan metode BERT yang

			menghasilkan akurasi sebesar 90%.
2.	Ardiansyah, Adika Sri Widagdo, Krisna Nuresa Qodri, Fachruddin Edi Nugroho Saputro, Nisrina Akbar Rizky P (2023)	Analisis Sentimen Terhadap Pelayanan Kesehatan Berdasarkan Ulasan Google Maps Menggunakan BERT[7]	Penelitian ini menggunakan model BERT indo-base-p1 untuk analisis sentimen dengan dataset 4228 data. Proses klasifikasi dilakukan dengan metode Lexicon yang mengkategorikan tweet menjadi tiga kategori: Negatif, Netral, dan Positif. Labeling menggunakan transformers Hugging Face dengan model RoBERTa berbahasa Indonesia menunjukkan bahwa sentimen

			positif adalah yang tertinggi dengan 2460 data (58.2%). Model menunjukkan akurasi tinggi pada data validasi (85%) dan testing (86%), menandakan kemampuan model dalam mengenali pola dengan baik. Namun, macro average yang lebih rendah (75% pada validasi dan 73% pada testing) menunjukkan beberapa kelas tidak diprediksi dengan baik. Weighted average yang tinggi dan konsisten (85% pada validasi dan 86% pada testing) menunjukkan bahwa kelas-kelas besar diprediksi dengan sangat baik.
3.	Bayu Kurniawan, Ahmad Ari Aldino, Auliya Rahman Isnain (2022)	Sentimen Analisis Terhadap Kebijakan Penyelenggara Sistem Elektronik (PSE) Menggunakan Algoritma Bidirectional Encoder Representations From Transformers (BERT)[8]	BERT diuji dengan parameter <i>batch size</i> 16 dan epoch 5. Hasil menunjukkan bahwa pada satu pengujian, BERT mencapai akurasi 69%, tetapi pada dua pengujian lainnya hanya mencapai 55%. Saat menggunakan BERT, akurasi dipengaruhi oleh keseimbangan data. Meskipun dataset yang seimbang lebih kecil, ia menghasilkan akurasi lebih tinggi (62%) dibandingkan dengan

			dataset yang tidak seimbang. Ketidakseimbangan data dapat mengurangi akurasi model karena model cenderung belajar lebih dari kelas mayoritas dan mengabaikan kelas minoritas.
4.	Muhammad Mahrus Zain, Rizky Nathamael Simbolon, Harlem Sulung dan Zaidan Anwar (2021)	Analisis Sentimen Pendapat Masyarakat Mengenai Vaksin Covid-19 Pada Media Sosial Twitter Dengan Robustly Optimized BERT Pretraining Approach[9]	Pada analisis sentimen, dilakukan pelabelan menggunakan model Indonesian RoBERTa Base <i>Sentimen Classifier</i> yang telah dilatih sebelumnya. Model Indonesian RoBERTa adalah varian dari RoBERTa yang telah melalui pelatihan dengan memanfaatkan 527MB teks dari Wikipedia berbahasa Indonesia menggunakan teknik Masked Language Modeling (MLM). Dari total 109.202 data yang digunakan, model Indonesian RoBERTa Base <i>Sentiment Classifier</i> memprediksi 6.250 sentimen positif, 75.883 sentimen netral, dan 26.934 sentimen negatif. Akurasi keseluruhan dari prediksi yang dilakukan adalah sebesar 95%. Secara rata-rata, hasil akurasi prediksi untuk masing-masing label adalah sebagai berikut: 84% untuk

			sentimen positif, 97% untuk sentimen netral, dan 93% untuk sentimen negatif.
5.	Alwi Jaya (2023)	Analisis Sentimen Pandangan Publik Terhadap Profesi PNS (Pegawai Negeri Sipil) Dari Twiter Menerapkan Indonesian Roberta Base Sentimen Classifier[10]	Dari 394 data tweet yang telah diberi label, 4.8% dianggap positif, 8.6% dianggap negatif, dan sisanya, yaitu 86.4%, dianggap netral. Selain itu, setelah dilakukan uji coba pelabelan pada penelitian ini, ternyata prediksi Indonesian Roberta Base Sentiment Classifier yang dilakukan memiliki akurasi yang baik, dengan rata-rata keseluruhan akurasi sebesar 90%.
6.	Nanang Husin (2023)	Komparasi Algoritma Random Forest, Naïve Bayes, Dan Bert Untuk Multi-Class Classification Pada Artikel Cable News Network (CNN)[2]	Model BERT menunjukkan kinerja terbaik dalam mengklasifikasikan dataset artikel berita CNN dari tahun 2011 hingga 2022, yang terdiri dari 37.904 baris artikel. Dataset ini tidak seimbang karena jumlah artikel dalam setiap kategori memiliki perbedaan yang signifikan. Untuk menyeimbangkan data, penelitian ini menggunakan library "imblearn" dengan metode RandomUnderSampler.

			Algoritma BERT mencapai akurasi <i>training</i> sebesar 93% dan akurasi testing sebesar 92%, dengan marco avg dari f1 score sebesar 92%. Dengan demikian, algoritma BERT terbukti efektif dalam mengklasifikasikan teks artikel berita, terutama dalam kasus dataset yang cukup besar dan tidak seimbang dibanding metode random forest yang memiliki akurasi <i>training</i> 81% dan Naïve Baiyes 78%.
7.	Yono Cahyono, Saprudin (2019)	Analisis <i>Sentimen</i> Tweets Berbahasa Sunda Menggunakan Naive Bayes Classifier dengan Seleksi Feature Chi Squared Statistic[4]	Dataset yang digunakan terdiri dari 316 tweet yang mencampurkan bahasa Sunda dan bahasa Indonesia yang diperoleh dari Twitter melalui proses pengambilan data menggunakan RapidMiner. Tahap pre-processing dilakukan dengan melakukan case folding, tokenize, dan penghapusan stopword. Seleksi fitur menggunakan statistik chi square untuk memilih kata-kata yang penting untuk analisis dokumen, dengan melakukan optimasi seleksi menggunakan forward selection. Hasil penelitian

			menunjukkan bahwa penggunaan seleksi fitur Chi Square Statistic dan algoritma Naïve Bayes Classifier menghasilkan akurasi sebesar 78.48%.
--	--	--	---

2.2 Landasan Teori

2.2.1 Parkir Elektronik Ponorogo (PARKIR-GO)

Parkir Elektronik Ponorogo (PARKIR-GO) menerapkan konsep retribusi parkir tepi jalan umum (PARKIR-GO), yang merupakan sebuah sistem digitalisasi perparkiran yang berbasis android. Tujuannya adalah untuk memfasilitasi transaksi perparkiran dan menyederhanakan proses pendataan, sehingga transparansi pengeluaran masyarakat terhadap pendapatan daerah dari hasil parkir dapat termonitor secara *real-time*. Metode yang digunakan dalam pengembangan ParkirGo mengadopsi tahapan metode pengembangan sistem *waterfall*, yang meliputi analisis, perancangan, implementasi, dan pengujian. Hasilnya adalah aplikasi sistem parkir online berbasis android yang responsif, yang mampu beradaptasi dengan baik pada layar smartphone yang berukuran besar atau kecil. Aplikasi ini juga memastikan bahwa setiap pengguna mendapatkan struk parkir setelah melakukan parkir di tepi jalan[1].

Langkah ini dimaksudkan untuk meningkatkan manajemen parkir, menaikkan tarif parkir, dan pada akhirnya berperan dalam penataan fasilitas parkir di daerah tersebut. Dengan memanfaatkan teknologi, terutama melalui aplikasi Parkir-Go, pemerintah berusaha beralih dari sistem parkir manual ke sistem elektronik yang lebih efisien dan transparan. Penerapan parkir elektronik tidak hanya dimaksudkan untuk meningkatkan efisiensi layanan parkir, tetapi juga untuk mengatasi masalah seperti parkir ilegal dan pelanggaran kendaraan. Dengan menerapkan tarif parkir progresif, sesuai dengan Peraturan Bupati Kabupaten Ponorogo No.27 Tahun 2022, pemerintah berupaya

memastikan pungutan yang adil dan standar berdasarkan jenis kendaraan. Dinas Perhubungan memainkan peran penting dalam mengelola dan mengumpulkan biaya parkir, dengan tujuan meningkatkan pengelolaan pendapatan dan mendukung pembangunan ekonomi. Langkah penerapan sistem parkir elektronik didorong oleh berbagai faktor, termasuk kurangnya kedisiplinan masyarakat dan petugas parkir, rendahnya pendapatan daerah dari biaya parkir, dan prevalensi pungutan liar. Inisiatif pemerintah untuk menerapkan parkir elektronik awalnya difokuskan pada pasar-pasar utama seperti Pasar Tonatan dan Relokasi Pasar Legi. Langkah ini diharapkan tidak hanya akan meningkatkan pendapatan, tetapi juga akan meningkatkan efisiensi dan integritas keseluruhan proses pengumpulan biaya parkir.[1]

2.2.2 Sentimen Analisis

Sentimen analysis secara luas merujuk pada bidang komputasi linguistik, pengolahan bahasa alami, dan text mining dengan tujuan untuk menentukan sikap dari pembicara atau penulis terkait topik tertentu. Teknik ini digunakan untuk mengekstraksi informasi mengenai opini dan sentimen. Sentimen sendiri adalah perasaan seseorang terhadap suatu hal. Pengelompokan polaritas teks dalam kalimat, dokumen, atau fitur entitas untuk menentukan apakah pendapat yang disampaikan bersifat positif, negatif, atau netral adalah tugas utama dalam analisis sentimen.[5]

2.2.3 Text Mining

Text mining, juga dikenal sebagai teks analitik, merupakan proses mengubah teks tidak terstruktur menjadi data terstruktur untuk analisis yang lebih mudah. Ini melibatkan teknik-teknik seperti pemrosesan bahasa alami (NLP), pengenalan entitas bernama, analisis sentimen, dan ekstraksi informasi. Tujuan utama dari *text mining* adalah untuk memperoleh wawasan berharga dari teks yang tidak terstruktur, yang dapat digunakan

untuk berbagai aplikasi seperti analisis tren pasar, manajemen hubungan pelanggan, dan deteksi penipuan.

Teknik *text mining* sering kali dimulai dengan tahap pra-pemrosesan, yang meliputi pembersihan data, tokenisasi, dan normalisasi. Pembersihan data mencakup penghapusan karakter khusus, tanda baca, dan kata-kata yang tidak memiliki makna penting, seperti kata sambung. Tokenisasi adalah proses memecah teks menjadi unit-unit yang lebih kecil, seperti kata atau frasa. Normalisasi melibatkan pengubahan bentuk kata menjadi bentuk dasar atau lematinya.

Dalam *text mining* terdapat proses yang bertujuan mengubah data teks ke dalam bentuk numerik agar dapat diolah dan dianalisis di proses selanjutnya yakni *pre-processing* dengan tahapan sebagai berikut:

- 1) Case Folding

Teknik yang digunakan untuk mengubah seluruh huruf dalam teks menjadi huruf kecil atau *lowercase* untuk memudahkan pemrosesan teks.

- 2) Filtering

Proses ini melakukan penghapusan atau pengecualian dokumen atau kata-kata tertentu dari teks berdasar aturan dan kriteria, tujuannya memperbaiki kualitas relevansi data serta mempermudah proses analisis.

- 3) Stemming

Tahap Dimana kata diubah ke akar katanya atau kembali ke kata dasar.

- 4) Tokenizing

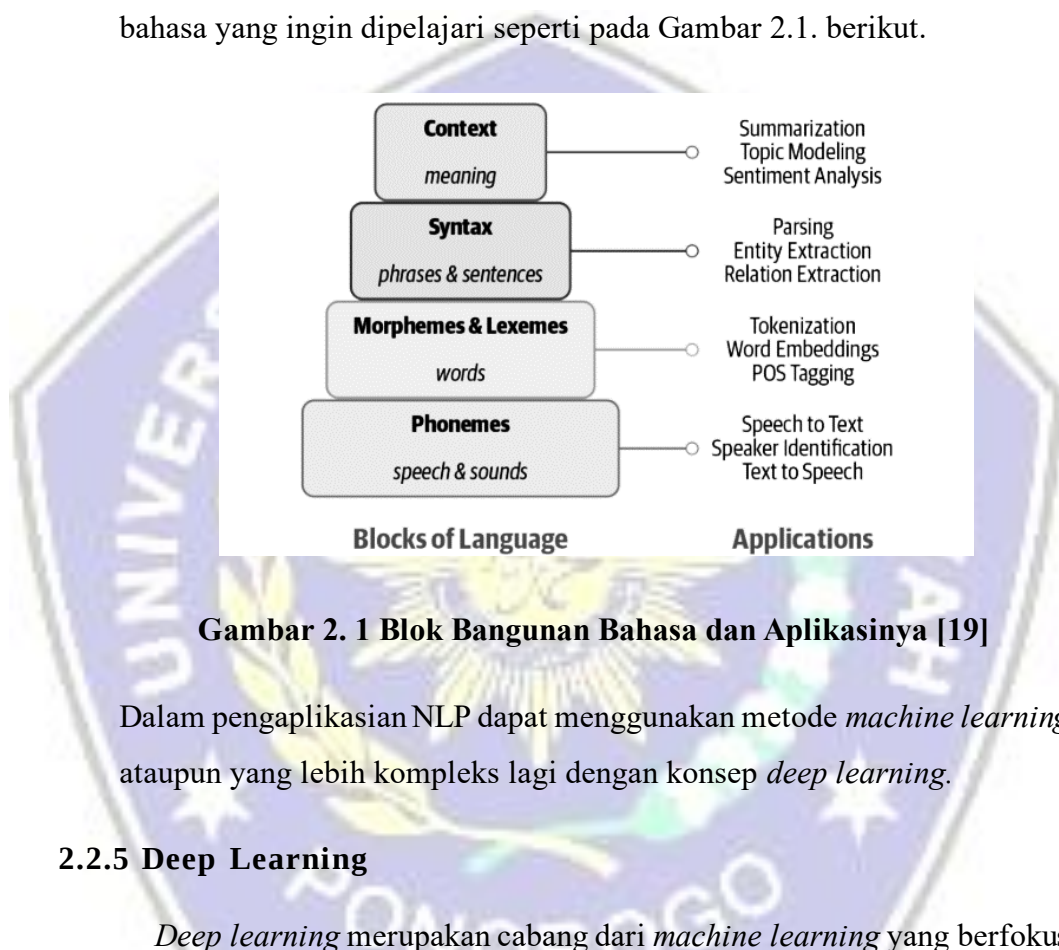
Proses ini membagi teks menjadi kata dan frasa, kalimat atau bagian bermakna.

2.2.4 Natural language Processing (NLP)

Natural language Processing atau NLP merupakan bidang yang berfokus pada pengembangan teknologi untuk memproses bahasa alami,

seperti bahasa inggris dan lainnya. Bidang ini adalah persimpangan dari *computer science*, *artificial intelligence* dan *linguistics*.

Dalam pengaplikasian NLP pada bahasa alami diperlukan pemahaman terkait struktur bahasa. Bahasa memiliki struktur sistem komunikasi yang kompleks dengan kombinasi komponen yang konstituen seperti karakter, kata, dan lainnya. Penerapan NLP pun bermacam-macam tergantung blok bahasa yang ingin dipelajari seperti pada Gambar 2.1. berikut.



Gambar 2. 1 Blok Bangunan Bahasa dan Aplikasinya [19]

Dalam pengaplikasian NLP dapat menggunakan metode *machine learning* ataupun yang lebih kompleks lagi dengan konsep *deep learning*.

2.2.5 Deep Learning

Deep learning merupakan cabang dari *machine learning* yang berfokus pada algoritma yang terinspirasi oleh struktur dan fungsi otak manusia, yang dikenal sebagai jaringan saraf tiruan. Dalam dekade terakhir, *Deep learning* telah menjadi salah satu teknologi paling inovatif dalam bidang kecerdasan buatan, memungkinkan terobosan signifikan dalam berbagai aplikasi seperti pengenalan gambar, pemrosesan bahasa alami, dan permainan komputer. *Deep learning* menggunakan berbagai arsitektur jaringan saraf, seperti *Multilayer Perceptrons* (MLP), *Convolutional*

Neural Networks (CNN), dan *Recurrent Neural Networks* (RNN). MLP merupakan jaringan dasar dengan lapisan *input*, satu atau lebih lapisan tersembunyi, dan lapisan *output*. CNN sangat efektif untuk pengenalan gambar dan video, menggunakan lapisan konvolusi untuk mendeteksi fitur visual, sementara RNN cocok untuk data deret waktu dan pemrosesan bahasa alami, menggunakan lapisan yang dapat mempertahankan informasi urutan[11].

Algoritma *Deep learning* menggunakan berbagai metode pembelajaran, termasuk pembelajaran terbimbing (*supervised learning*), pembelajaran tak terbimbing (*unsupervised learning*), dan pembelajaran penguatan (*reinforcement learning*). Dalam pembelajaran terbimbing, jaringan dilatih dengan dataset berlabel untuk memprediksi *output* yang benar. Pembelajaran tak terbimbing mencoba menemukan pola dalam data tanpa label, sedangkan pembelajaran penguatan mengharuskan jaringan belajar dengan mencoba-coba dan menerima umpan balik berupa *reward* atau *punishment*. [12]

Konsep dasar deep learning:

1. *Neural Network*

Konsep ini terdiri dari unit-unit pemrosesan(neuron) yang diorganisir dalam lapisan(layers). *Input* diberikan pada layer *input* dan *output* dihasilkan pada layer *output* setelah melalui serangkaian layer tersembunyi yang memungkinkan model untuk memahami fitur yang semakin kompleks pada data[6].

2. *Deep Neural Network*

Deep neural network mempunyai lebih dari satu layer tersembunyi. Semakin banyak layer menandakan kompleksnya model dan kemampuan yang baik dalam representasi yang abstrak dari data.

3. *Training*

Model dilatih dengan dataset besar untuk belajar menyesuaikan parameter agar memperoleh prediksi yang akurat.

4. *Backpropagation*

Algoritma ini menyesuaikan bobot(weight) dalam neural *network* berdasar perbedaan antara prediksi model dan nilai sebenarnya dari data *training*[12].

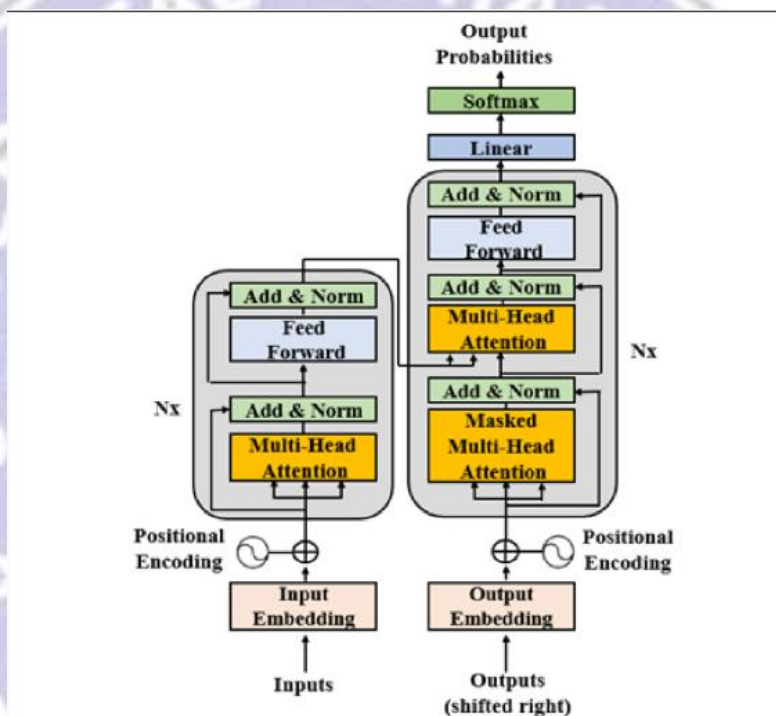
Deep learning kini telah banyak diterapkan dalam berbagai tugas pemrosesan salah satunya pemrosesan bahasa atau NLP. Kemampuan metode *Deep learning* untuk mempelajari berbagai masalah dengan konsep dan struktur mengikuti fungsi otak manusia memberikan kemampuan penyelesaian tugas yang baik. Bahasa adalah data yang kompleks dan tidak terstruktur. Pada tugas NLP membutuhkan model dengan representasi dan kemampuan belajar yang lebih baik untuk memahami dan menyelesaikan masalah bahasa[13].

2.2.6 Transformers

Vasvani,dkk pada 2017 memperkenalkan model *transformers* sebagai salah satu metode *Deep learning* dalam tugas pemrosesan bahasa alami atau NLP[14]. Ide inti dari *transformer* adalah mekanisme perhatian, yang memungkinkan model membaca kalimat dan memberikan perhatian lebih pada kata-kata tertentu. Saat memproses sebuah kata, *Transformer* memberikan "perhatian" pada kata-kata lain dalam kalimat tersebut. Untuk melakukan ini, Transformer menggunakan embedding yang mengubah kata-kata atau token ke dalam ruang vektor numerik. Dengan mekanisme *multihead attention*, model ini dapat memperhatikan hubungan antara kata-kata secara paralel, sehingga meningkatkan efisiensi. Mekanisme *self-attention* memungkinkan model memberikan bobot perhatian yang dinamis pada setiap kata dalam urutan *input*, tergantung pada konteksnya.

Model transformater didasarkan sepenuhnya pada mekanisme perhatian dan sepenuhnya menghilangkan reccurance. Metode ini menggunakan jenis mekanisme perhatian khusus yang disebut perhatian diri atau *self-attention*. Model transduksi urutan saraf yang bersaing umumnya mengadopsi struktur *encoder-decoder*. Model transduksi urutan saraf adalah sistem yang dapat mengubah urutan data saraf menjadi bentuk lain yaitu alat atau program

komputer yang dapat mengubah informasi dari satu bentuk urutan saraf ke bentuk urutan lainnya. Dalam sistem ini, bagian "*encoder*" bertugas mengubah urutan simbol-simbol *input* menjadi bentuk representasi kontinu yang disebut *z*. Setelah mendapatkan representasi *z*, "*decoder*" kemudian menghasilkan urutan *output* simbol-simbol satu per satu. Model ini bekerja secara *auto-regressive*, artinya pada setiap langkahnya, model menggunakan simbol yang dihasilkan sebelumnya sebagai masukan tambahan saat menghasilkan simbol berikutnya. Arsitektur Transformer mengadopsi pendekatan ini, dengan menggunakan perhatian diri bertumpuk dan lapisan-lapisan penuh yang terhubung pada *encoder* dan *decoder*.



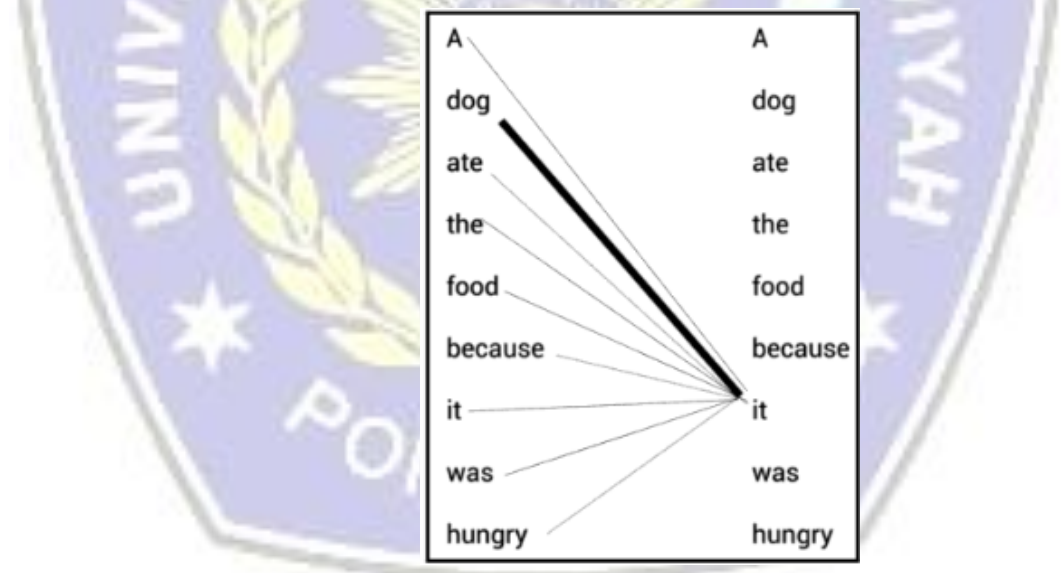
Gambar 2. 2 Arsitektur Transformers[14]

Berdasarkan Gambar 2.2 di sebelah kiri, *input* masuk ke sisi *encoder* Transformer melalui sublapisan perhatian dan sublapisan *FeedForward Network* (FFN). Di sebelah kanan, *output* target masuk ke sisi *decoder* Transformer melalui dua sublapisan perhatian dan sublapisan FFN. Dapat dilihat bahwa tidak ada RNN, LSTM, atau CNN.

Perhatian telah menggantikan pengulangan, yang memerlukan peningkatan jumlah operasi seiring bertambahnya jarak antara dua kata. Mekanisme perhatian adalah operasi "kata ke kata". Mekanisme perhatian akan menemukan bagaimana setiap kata terkait dengan semua kata lain dalam suatu urutan, termasuk kata yang sedang dianalisis itu sendiri.

- Mekanisme *Self-attention*

Manusia dapat dengan mudah menangkap makna dari setiap kata seperti kata ganti atau sebagainya mempresentasikan bagian yang mana. Namun model butuh mekanisme yang relevan agar dapat memahami maksud atau konteks dari kata. Dalam transformer mekanisme *self-attention* membantu model memahami kata melalui cara pengecekan relasional dari suatu kata ke setiap kata dalam kalimat[15]. Contoh kerja dari mekanisme ini seperti Gambar 2.3 Contoh Mekanisme *Self-attention* berikut.

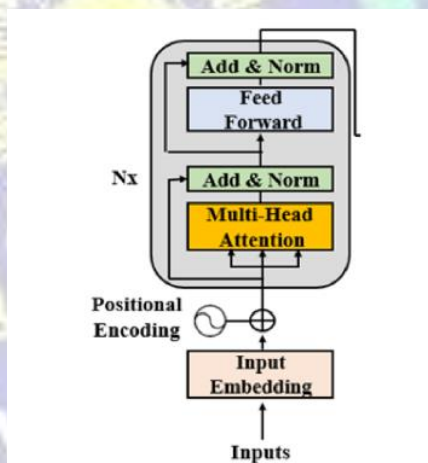


Gambar 2. 3 Contoh Mekanisme Self-Attention [18]

Mekanisme *self-attention* pada Gambar 2.3. melakukan perhatian terhadap kata "it" dengan mengecek ke seluruh kata dan menangkap maksud dari kata tersebut mengarah ke "dog".

2.2.6.1 EncoderStack

Encoder pada Gambar 2.4 merupakan bagian penting dalam model *transformers* Gambar 2. 2. *Encoder* adalah komponen dalam dunia komputasi dan pemrograman yang berfungsi untuk menerjemahkan data dari suatu bentuk ke bentuk lain. *Transformer* terdiri dari tumpukan sejumlah N *encoder*. *Output* dari satu *encoder* dikirim sebagai *input* ke *encoder* di atasnya. Seperti yang ditunjukkan pada gambar berikut, terdapat setumpuk N jumlah *encoder*. Setiap *encoder* mengirimkan *output* nya ke *encoder* di atasnya. Format baru hasil *encoder* dalam *Deep learning* memungkinkan data diproses dan dikompresi oleh lapisan hilir. *Encoder* dalam *transformer* memiliki $N=6$ lapisan identik dengan dua sub-lapisan di setiap lapisan *encoder*. Pada sub-lapisan pertama adalah mekanisme perhatian diri *multi-head*, kemudian sub lapisan kedua merupakan jaringan *feed-forward* yang sederhana. *Encoder* akhir mengembalikan representasi kalimat sumber yang diberikan sebagai *output*.



Gambar 2. 4 Blok *Encoder* [17]

- *Input embedding*

Sublapisan penyematan berfungsi seperti model transduksi standar lainnya. Sebuah *tokenizer* akan mengubah kalimat menjadi token. Setiap *tokenizer* memiliki metodenya sendiri, tetapi hasilnya serupa. Misalnya, sebuah *tokenizer* yang

diterapkan pada urutan kalimat “Kebijakan yang positif dan mampu mensejahterakan.” Akan menghasilkan token berikut: ['kebijakan', 'yang', 'positif', 'dan', 'mampu', 'mensejahterakan'] Pada layer ini teks akan diubah dengan memisahkan masing-masing suku kata dan juga menjadi huruf kecil. Selanjutnya untuk proses embending token teks akan di ubah menjadi token id numerik. Token IDs = [101, 17710, 5638, 18317, 2319, 8675, 13433, 28032, 10128, 4907, 5003, 8737, 2226, 2273, 3366, 18878, 14621, 9126, 102]. Teks yang ditokenisasi harus disematkan(*embedding*). Banyak metode penyematan yang dapat diterapkan pada masukan yang ditokenisasi.

- Positional embedding

Tahapan selanjutnya adalah *positional embedding* untuk menggambarkan posisi token dalam suatu urutan. Tujuannya agar proses pemahaman teks sesuai urutan seharusnya.

- Sub-layer 1 multi-head attention

Sub-layer multi-head attention terdiri dari delapan *heads* dan diikuti oleh normalisasi *post-layer*, yang akan menambahkan koneksi residual ke *output* dari *sub-layer* dan menormalkannya. *Input* dari *sub-layer multi-attention* pada layer pertama dari tumpukan *encoder* adalah sebuah vektor yang berisi *embedding* dan *encoding* posisi dari setiap kata. Layer-layer berikutnya dari tumpukan ini tidak mengulangi operasi ini. Setiap kata dipetakan ke semua kata lain untuk menentukan bagaimana kata tersebut cocok dalam suatu urutan. *Multi-head attention* memungkinkan model untuk secara bersamaan memperhatikan informasi dari subruang representasi yang berbeda pada posisi yang berbeda.

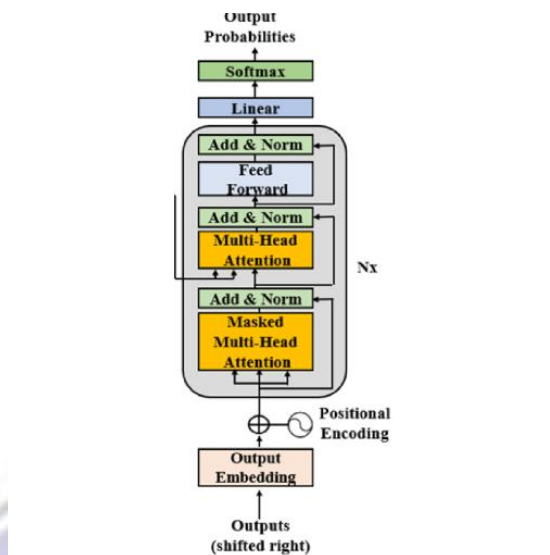
- Sub-layer 2 feed forward *network*

Setelah melewati sublayer multihead attention, *output* nya diteruskan ke feed forward sublayer. Feed forward sublayer

terdiri dari dua lapisan linear yang dipisahkan oleh fungsi aktivasi non-linear seperti ReLU (*Rectified Linear Unit*). Lapisan pertama memproyeksikan *input* dari dimensi model ke dimensi yang lebih besar, kemudian fungsi aktivasi diterapkan, dan lapisan kedua memproyeksikan kembali hasilnya ke dimensi asli. Proses ini membantu dalam meningkatkan kapasitas model untuk menangkap dan mempelajari representasi yang lebih kompleks dari data *input*. Selain itu, *feed forward sublayer* beroperasi secara independen pada setiap posisi dalam urutan *input*, sehingga memungkinkan pemrosesan paralel yang efisien. Kombinasi dari sublayer *multihead attention* dan *feed forward sublayer* memungkinkan *transformer* untuk menjadi model yang sangat efektif dalam berbagai tugas pemrosesan bahasa alami (NLP).

2.2.6.2 Decoderstack

Arsitektur Transformer, yang diperkenalkan oleh Vaswani et al. (2017) dalam makalah mereka "Attention is All You Need," telah menjadi landasan banyak model pemrosesan bahasa alami (NLP) modern[16]. Salah satu komponen utama dari arsitektur *Transformer* adalah *stack decoder*, yang berperan penting dalam menghasilkan *output* selama proses penerjemahan atau penguraian teks.



Gambar 2. 5 Blok *Decoder*[17]

1. Struktur Dasar *Decoder Stack*

Decoder stack pada transformer Gambar 2. 5 Blok *Decoder* terdiri dari beberapa lapisan identik (biasanya enam), masing-masing terdiri dari tiga sublayer utama:

- *Masked Multihead Self-Attention*
- *Multihead Attention* dengan *Encoder-Decoder Attention*
- *Feed Forward Neural Network*

Setiap sublayer ini diikuti oleh mekanisme normalisasi layer (*layer normalization*) dan koneksi residual untuk meningkatkan stabilitas dan efisiensi pelatihan.

2. *Masked Multihead Self-Attention*

Pada sublayer pertama, *masked multihead self-attention*, model memperhatikan semua posisi sebelumnya dalam urutan *output*. *Masking* dilakukan untuk memastikan bahwa prediksi pada posisi tertentu hanya bergantung pada posisi sebelumnya dan tidak pada posisi saat ini atau yang akan datang. Ini memungkinkan model untuk menghasilkan urutan *output* secara *autoregressive*, di mana setiap token dihasilkan satu per satu.

3. *Multihead Attention* dengan *Encoder-Decoder Attention*

Sublayer kedua adalah *multihead attention* yang menghubungkan informasi dari *encoder* ke *decoder*. Di sini, *queries* berasal dari lapisan *decoder* sebelumnya, sedangkan *keys* dan *values* berasal dari *output encoder*. Ini memungkinkan model untuk memperhatikan konteks *input* saat menghasilkan setiap token *output*, memastikan bahwa *output* selaras dengan *input*.

4. *Feed Forward Neural Network*

Sublayer ketiga adalah *feedforward neural network* (FFN), yang terdiri dari dua lapisan linear dengan fungsi aktivasi non-linear di antaranya (biasanya ReLU). FFN diterapkan secara independen pada setiap posisi, dan proses ini membantu dalam mempelajari representasi yang lebih kompleks dari data *input*.

5. Normalisasi Layer dan Koneksi Residual

Setiap sublayer diikuti oleh normalisasi layer (layer normalization) dan koneksi residual. Normalisasi layer membantu dalam mengatasi masalah pelatihan dengan menjaga skala aktivasi tetap stabil. Koneksi residual memungkinkan informasi asli mengalir dengan lebih mudah melalui jaringan, yang membantu dalam mengatasi masalah gradien menghilang (*vanishing gradients*) selama pelatihan.

2.2.7 Bidirectional Encoder Representations from Transformers (BERT)

BERT (*Bidirectional Encoder Representations from Transformers*) adalah algoritma *Natural Language Processing* (NLP) yang dikembangkan oleh Google dan diperkenalkan oleh para peneliti Google AI pada tahun 2018. BERT memberikan hasil yang optimal dalam berbagai tugas NLP seperti *question answering*, *natural language inference*, *classification*, dan *general language understanding evaluation*. BERT hadir dalam dua varian, yaitu BERT-base dan BERT-large. BERT-base memiliki parameter $L=12$, $H=768$, $A=12$, dengan total parameter 110 juta, sedangkan BERT-large

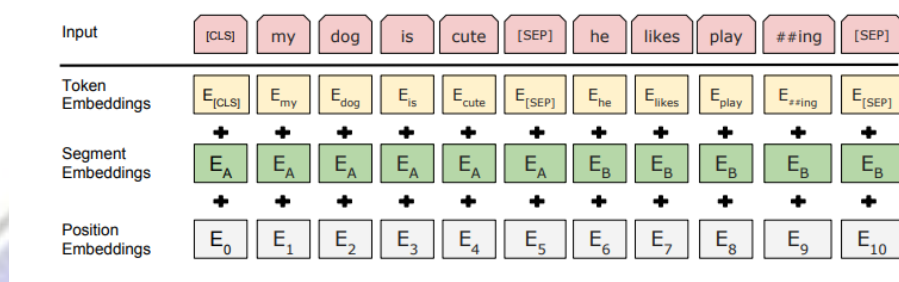
memiliki parameter $L=24$, $H=1024$, $A=16$, dengan total parameter 340 juta.[17]

Input yang diterima oleh BERT adalah vektor angka yang dihasilkan melalui teknik *word embedding* seperti Gambar 2.7. Proses *embedding* ini mengubah setiap token dalam urutan *input* menjadi representasi vektor. Karena arsitektur *Transformer* tidak memiliki koneksi berulang, posisi setiap token dalam urutan *input* harus direpresentasikan secara eksplisit dengan menambahkan vektor *positional encoding* ke *input embedding*. Urutan *input* beserta *positional encoding*-nya kemudian dimasukkan ke mekanisme *multi-head self-attention*. Mekanisme ini memungkinkan model untuk fokus pada bagian-bagian berbeda dalam urutan *input* pada setiap lapisan dan menangkap ketergantungan jarak jauh antar token. Setelah melalui mekanisme *self-attention*, *output* diproses melalui jaringan saraf *feed-forward* yang menerapkan transformasi non-linear pada setiap posisi secara independen. Untuk meningkatkan pelatihan model dan membantu konvergensi lebih cepat, digunakan *residual connections* dan lapisan *normalization*. *Residual connections* memungkinkan aliran gradien lebih mudah melalui jaringan, sedangkan lapisan *normalization* membantu menstabilkan distribusi nilai *output*.

Arsitektur *Transformer* terdiri dari *stack encoder* dan *decoder*. *Stack encoder* bertanggung jawab untuk meng-*encode* urutan *input*, sementara *stack decoder* bertugas menghasilkan urutan *output*. Pada *stack decoder*, mekanisme *self-attention* dimasking sehingga setiap posisi hanya bisa fokus pada posisi sebelumnya dan posisi saat ini. Ini mencegah model melihat token di masa depan selama proses prediksi. Selama proses decoding, *stack decoder* juga memperhatikan *output* dari *stack encoder*, memungkinkan model menggunakan informasi dari urutan *input* untuk menghasilkan urutan *output* yang sesuai [16].

BERT dilatih untuk memahami hubungan kontekstual antara kata-kata dalam sebuah kalimat menggunakan data teks dalam jumlah besar. Khususnya, BERT dilatih pada dua tugas *Masked Language Modeling*

(MLM) dan *Next Sentence Prediction* (NSP). Dalam MLM, beberapa kata dalam sebuah kalimat secara acak diganti dengan token [MASK], dan model harus memprediksi kata aslinya. Tugas ini membantu model memahami makna kata dalam konteks. Dalam NSP, model diberikan dua kalimat dan harus memprediksi apakah kalimat kedua kemungkinan mengikuti kalimat pertama. Tugas ini membantu model memahami hubungan antar kalimat [17]



Gambar 2. 6 Word Embending[20]

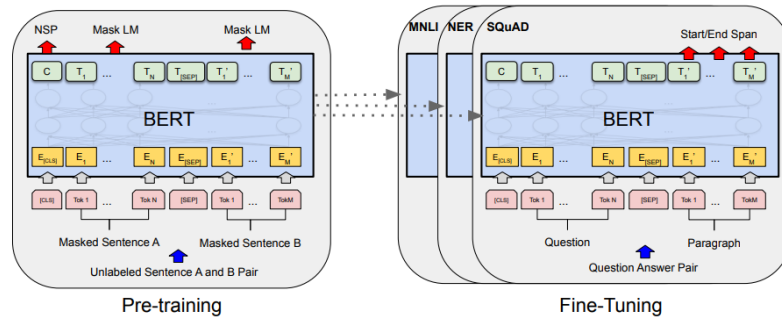
Representasi *input* BERT ditunjukkan pada Gambar 2.6. langkah-langkah tokenisasi dalam BERT adalah sebagai berikut:

1. Tokenisasi: Memecah teks menjadi token-token yang terdiri dari kata-kata. BERT menggunakan tokenisasi *WordPiece*, di mana beberapa token dapat dibagi lagi menjadi sub-token.
2. Token Embeddings: BERT menambahkan dua token khusus ke awal dan akhir setiap kalimat, yaitu [CLS] dan [SEP]. Token [CLS] digunakan untuk merepresentasikan keseluruhan kalimat di awal, sementara token [SEP] digunakan untuk memisahkan kalimat dalam urutan *input*.
3. Konversi Token menjadi ID: Setiap token dalam *input* dikonversi menjadi ID token yang sesuai menggunakan kamus token yang telah ditetapkan. Setiap ID token kemudian dikonversi menjadi vektor dengan mengambil nilai embedding dari matriks embedding kata yang telah dilatih sebelumnya.

4. *Segment Embeddings*: Jika *input* terdiri dari dua kalimat, setiap token dalam *input* diberi tanda sebagai milik kalimat pertama atau kedua dengan memberikan segmen ID 0 atau 1.

5. *Position Embedding*: BERT menggunakan *position embedding* untuk menambahkan informasi posisi absolut ke representasi token dengan menambahkan vektor posisional yang telah ditentukan sebelumnya ke setiap vektor token.

BERT menggunakan dua paradigma pelatihan, yaitu *pre-training* dan *fine-tuning*, yang ditunjukkan pada Gambar 2.7. *pre-training* adalah proses *unsupervised learning* di mana model dilatih pada dataset tanpa label untuk mengekstrak pola. Google melatih model ini pada BooksCorpus (800 juta kata) dan English Wikipedia (2,5 miliar kata). Proses *pre-training* melibatkan dua tugas *masked language modeling* (MLM) dan *next sentence prediction* (NSP). Selama *fine-tuning*, model dilatih kembali pada tugas *downstream* dengan data berlabel, menyesuaikan parameter model BERT yang telah dilatih sebelumnya untuk mengoptimalkan kinerja pada tugas tersebut. *Fine-tuning* melibatkan penambahan lapisan khusus untuk tugas di atas model BERT yang telah dilatih sebelumnya dan melatih seluruh model dari awal hingga akhir pada data tugas tersebut. Lapisan khusus ini memiliki jumlah parameter yang jauh lebih kecil dibandingkan model BERT utama. Selama *fine-tuning*, model dilatih dengan tingkat pembelajaran yang lebih rendah dibandingkan saat *pre-training* karena model sudah belajar fitur umum bahasa dan hanya perlu mempelajari fitur khusus dari tugas *downstream* [18].



Gambar 2. 7 Pre-Training dan Fine Tuning BERT [20]

Setelah melalui beberapa lapisan BERT, *output* terakhir adalah sekumpulan vektor yang mewakili setiap token dalam *input*. Untuk tugas klasifikasi, vektor yang sesuai dengan token khusus [CLS] sering digunakan sebagai representasi dari seluruh *input*. Representasi ini kemudian dikirim ke lapisan klasifikasi yang terdiri dari lapisan linear (*dense layer*) untuk menghasilkan logits.

Rumus perhitungan logits untuk setiap kelas iii diberikan oleh:

$$\text{Logits} = X \cdot W + b \quad (2.1)$$

- X adalah vektor representasi akhir dari token [CLS], yang direpresentasikan dengan nilai numerik hasil dari lapisan-lapisan Transformer dalam model seperti BERT.
- W adalah matriks bobot yang dipelajari selama proses pelatihan model.
- b adalah vektor bias yang juga dipelajari selama proses pelatihan.

Logits ini kemudian dilewatkan melalui fungsi aktivasi *softmax* untuk menghasilkan probabilitas prediksi untuk setiap kelas. Fungsi *softmax* mengubah logits menjadi probabilitas dengan rumus:

$$\text{Softmax}(z_j) = \frac{e^{z_j}}{\sum_{j=1}^k e^{z_j}} \quad (2.2)$$

Yang mana:

1. e^{z_j} = nilai eksponensial logits
2. $\sum_{j=1}^k e^{z_j}$ = jumlah keseluruhan eksponen logits
3. e = euler (2.718281828459045)

Fungsi ini memastikan bahwa semua probabilitas hasil prediksi berada dalam rentang $[0,1]$ dan jumlah totalnya adalah 1. Dengan kata lain, fungsi *softmax* menormalisasi nilai logits menjadi distribusi probabilitas yang dapat diinterpretasikan sebagai probabilitas prediksi untuk setiap kelas.

Penjelasan di atas memberikan gambaran tentang bagaimana BERT memproses *input*, menghitung logits, dan menggunakan fungsi *softmax* untuk menghasilkan probabilitas prediksi dalam tugas klasifikasi.

2.2.8 Sentimen Indonesia Lexicon

Sentimen Indonesia lexicon merupakan salah satu pustaka berbahasa Indonesia yang dibangun oleh Koto F dan Rahmaningtyas pada tahun 2017 dengan nama InSet(Indonesia Sentimen) lexicon[19]. Sumber data ini dibangun dengan memuat 3609 kata positif dan 6609 kata negatif dengan pembobotan dari -5 hingga 5. InSet lexicon dibangun dengan memproses data tweet berbahasa Indonesia sekitar 1000 data dengan memberi label setiap kata secara manual berdasarkan polaritasnya kemudian ditingkatkan dengan menambahkan stemming dan set sinonim. [19]. InSet dalam penelitian ini akan dijadikan sumber data dalam proses labelling data.

2.2.9 Confusion Matrix

Confusion matrix merupakan perhitungan untuk mengukur berbagai *performance metrics* terhadap kinerja model yang telah dibangun[2]. Dalam penelitian ini untuk mengukur akurasi dari model yang dibangun menggunakan *confusion matrix* yang merangkum kinerja model pembelajaran mesin pada sekumpulan data uji. Matriks ini digunakan untuk menampilkan jumlah kejadian yang diprediksi dengan benar dan salah oleh model. Matriks ini sangat berguna untuk menilai kinerja model klasifikasi, yang bertujuan untuk memprediksi label kategoris untuk setiap *input*. Matriks konfusi Tabel 2.2 menunjukkan jumlah kejadian yang diprediksi oleh model pada data uji.

- True Positive (TP): Model dengan benar memprediksi titik data sebagai positif.
- True Negative (TN): Model dengan benar memprediksi titik data sebagai negatif.
- False Positive (FP): Model salah memprediksi titik data sebagai positif.
- False Negative (FN): Model salah memprediksi titik data sebagai negatif.

Tabel 2. 2 Confusion Matrix

		Predicted Label	
		Negative	Positive
True Label	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

Berikut metrik evaluasi yang digunakan meliputi

- Akurasi (Accuracy)

Akurasi adalah rasio prediksi yang benar (positif dan negatif) terhadap total jumlah prediksi.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.3)$$

TP = True Positive,

TN = True Negative,

FP = False Positive,

FN = False Negative.

- Presisi (Precision)

Presisi adalah rasio true positive terhadap semua prediksi positif yang dilakukan oleh model.

$$\text{Precicion} = \frac{TP}{TP+FP} \quad (2.4)$$

- Recall (Sensitivitas atau Recall)

Recall adalah rasio true positive terhadap semua data aktual yang seharusnya positif.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (2.5)$$

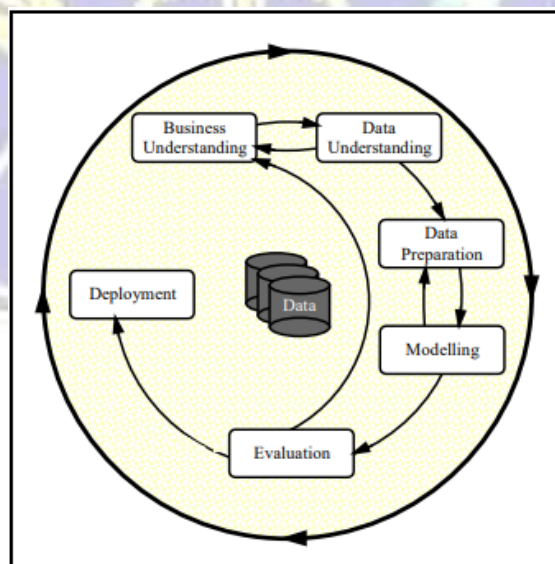
- F1-score

F1-score adalah rata-rata harmonik dari presisi dan recall, memberikan gambaran yang seimbang tentang performa model.

$$F1 - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.6)$$

2.2.10 CRISP-DM (*Cross-Industry Standard Process for Data Mining*)

Data mining membutuhkan standart yang dapat menerjemahkan masalah ke tugas *data mining*, mentransformasikan data dan teknik penambangan data yang tepat, serta menyediakan sarana untuk mengevaluasi efektivitas hasil dan dokumentasi. CRISP-DM atau *Cross-Industry Standard Process for Data Mining* merupakan metode dalam proses data mining dengan tujuan agar proyek penambangan data besar menjadi lebih murah, lebih andal, lebih iterative atau dapat diulang, mudah dikelola, dan lebih cepat[20].



Gambar 2. 8 Proses dalam CRISP-DM[20]

Berdasarkan Gambar 2.8, proses yang dilalui dalam pengerjaan data mining menggunakan metode CRISP-DM dijabarkan seperti berikut.

1. *Bussiness Understanding*

Tahap ini berfokus terhadap pemahaman tujuan dan persyaratan proyek dari perspektif bisnis, dan mengubah pengetahuan menjadi definisi masalah penambangan data, kemudian rencana proyek awal dirancang untuk mencapai tujuan.

2. *Data Understanding*

Tahapan pemahaman data ini dimulai dengan pengumpulan data dilanjutkan dengan eksplorasi data untuk mengidentifikasi masalah kualitas data, menemukan informasi atau untuk mendeteksi subset yang menarik untuk membentuk hipotesis untuk informasi tersembunyi. Ada hubungan erat antara Pemahaman Bisnis dan Pemahaman Data.

3. *Data Preparation*

Proses ini membangun kumpulan data akhir (data yang akan dimasukkan ke dalam alat pemodelan) dari data mentah sebelumnya. *Data preparation* kemungkinan akan dilakukan beberapa kali, dan tidak dalam urutan yang ditentukan. Tugas meliputi beberapa proses seperti pembersihan data, konstruksi atribut baru, serta transformasi data untuk pemodelan.

4. *Modelling*

Pada tahapan ini, berbagai teknik *modelling* dipilih untuk diterapkan, dan parameternya dikalibrasi ke nilai optimal. Biasanya, ada beberapa teknik untuk jenis masalah penambangan data yang sama. Beberapa teknik memerlukan format data tertentu. Tahap ini sangat berhubungan erat dengan data preparing karena mempertimbangkan kualitas data.

5. *Evaluation*

Mengevaluasi model secara lebih menyeluruh, dan meninjau langkah-langkah yang dijalankan untuk membangun model untuk

memastikan model mencapai tujuan bisnis yang direncanakan sudah sesuai.

6. Deployment

Pengetahuan yang diperoleh perlu diatur dan disajikan agar dapat menjadi insight bagi orang lain. Tahap ini dapat dengan membuat laporan atau membuat visualisasi hasil data mining.

2.2.11 Python

Python merupakan bahasa pemrograman populer yang sering digunakan dalam berbagai pengembangan seperti *data mining*, *machine learning* hingga *web development*. Python memiliki struktur data tingkat tinggi yang efisien dan pendekatan sederhana namun efektif terhadap pemrograman berorientasi objek. Sintaksnya yang elegan dan pengetikan dinamis, bersama dengan sifatnya yang terinterpretasi, menjadikannya bahasa yang ideal untuk *scripting* dan pengembangan aplikasi cepat di berbagai bidang pada sebagian besar *platform*[21].

2.2.12 Flask

Flask adalah kerangka kerja mikro berbasis Python yang dirancang agar mudah dipahami dan digunakan, menawarkan fleksibilitas melalui ekstensi tanpa menambah kompleksitas yang tidak perlu. Dengan tiga dependensi utama, yaitu Werkzeug, Jinja2, dan Click, Flask menyediakan inti yang kuat untuk pengembangan web[22]. Meski tidak mendukung fitur tingkat lanjut secara bawaan, seperti akses basis data atau autentikasi pengguna, fitur-fitur ini dapat ditambahkan melalui ekstensi, memberi pengembang kebebasan untuk memilih alat yang sesuai dengan kebutuhan proyek. Flask sangat cocok bagi mereka yang menginginkan kerangka kerja Python yang ringan namun tetap fungsional dan dapat diperluas.

2.2.13 HTML

HTML atau *Hypertext Markup Language* merupakan bahasa *markup* yang menjadi standarisasi pengembangan halaman website yang dapat ditampilkan pada *web browser*. HTML muncul pertama kali pada 1989 oleh Tim Berners Lee yang kemudian dikembangkan oleh W3C.[23]

2.2.14 CSS

Cascading style sheets atau dikenal dengan CSS merupakan bahasa untuk mengatur tampilan atau *layout website* bersanding dengan HTML dalam penggunaannya. Bahasa ini mendukung berbagai bahasa *markup* seperti HTML, XHTML, XML, SVG (*Scalable Vector Graphics*) dan Mozilla XUL (*XML User Interface Language*).[23]

2.2.15 Black Box Testing

Dalam pengembangan aplikasi ataupun website diperlukan tindakan untuk melihat hasil yang sudah dirancang dengan pengujian atau testing. *Black Box* merupakan salah satu teknik pengujian dengan fokus terkait spesifikasi fungsional dari perangkat lunak yang telah dibangun. [24]