BAB II

TINJAUAN PUSTAKA

2.1. Teori Dasar

Pada Bab dua ini, penulis akan memfokuskan pembahasan pada beberapa teori dasar yang menjadi landasan dalam penelitian ini. Penulis akan menjelaskan tentang konsep *deep learning* yang merupakan cabang dari *machine learning* yang memiliki kemampuan untuk menganalisis data dalam bentuk yang lebih kompleks. Selanjutnya, penulis akan membahas teori mengenai *machine learning*, termasuk tiga pendekatan utama yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Berikutnya penulis akan membahas konsep *Natural Language Processing* (NLP), salah satu teknologi untuk memahami dan memproses bahasa alami manusia serta algoritma transformers yang menjadi dasar dari model *Indonesian-RoBERTa* yang penulis gunakan untuk melakukan analisis sentimen dalam penelitian ini. Penulis juga akan memberi penjelasan terhadap model BERT, Sejarah singkat terkait model BERT sampai pengembangannya yang menjadi awal mula terciptanya model *RoBERTa* dan *Indonesian-RoBERTa*. Setelah itu penulis juga akan membedah arsitektur dari algoritma transformers serta model *RoBERTa* untuk mengetahui bagaimana model tersebut bekerja dalam melakukan tugas analisis sentimen.

Selain itu, beberapa *software* pendukung yang penulis gunakan dalam penelitian ini seperti *Visual Studio Code*, *Google Colab* dan *Postman* akan penulis jelaskan, mulai dari sejarah singkat terkait *software* tersebut, hingga fungsi dan alasan penulis menggunakan *software* tersebut. Setelah melakukan penjelasan

terkait teknologi dan *software* yang penulis gunakan, dalam bab ini penulis juga akan melanjutkan dengan memberi penjelasan mengenai penelitian-penelitian terdahulu yang berkaitan dengan penelitian ini, untuk memberikan gambaran tentang perkembangan penelitian yang telah dilakukan. Terakhir, penulis akan memaparkan kerangka pemikiran yang menggambarkan alur dari penelitian ini, sehingga seluruh pembahasan di bab ini dapat memberikan dasar yang kuat bagi penelitian yang sedang dilakukan.

2.1.1. *Deep learning*

Deep learning berkembang sebagai cabang *machine learning* yang menggunakan arsitektur jaringan saraf tiruan (*artificial neural networks*), teknologi ini dirancang untuk menirukan cara kerja jaringan saraf biologis manusia melalui struktur jaringan saraf tiruan. Pada jaringan saraf tiruan tersebut terdapat banyak lapisan (*layers*) yang disebut *multilayered* untuk melakukan ekstrasksi fitur dari data yang di *Input* seperti gambar, teks ataupun suara (Yudistira 2021). Teknologi ini memungkinkan komputer untuk belajar dari sejumlah data yang cukup besar secara mendalam dan bertahap. *Deep learning* telah banyak digunakan dalam berbagai bidang, terutama dalam pemrosesan bahasa seperti pengenalan teks, terjemahan bahasa, dan analisis sentimen. Kemampuan *Deep learning* untuk memahami konteks dan makna dalam teks membuat teknologi ini menjadi pilihan yang tepat untuk menganalisis sentimen dalam ulasan pengguna aplikasi instagram di *Google Play Store*.

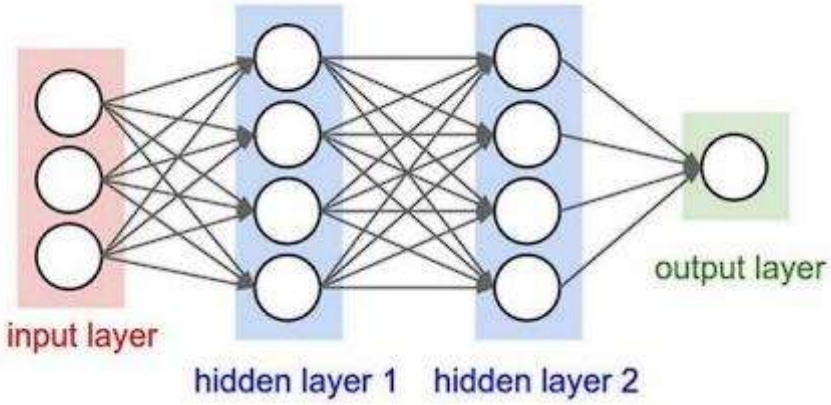
Perbedaan mendasar mengenai *deep learning* dan *machine learning* terdapat dari cara kerja kedua teknologi ini dalam memproses dan menganalisis data. Dalam *machine learning* tradisional, proses pemilihan fitur dilakukan secara manual, fitur-fitur penting dari data harus dipilih dan disiapkan sebelum algoritma dilatih, setiap informasi yang dianggap relevan, seperti panjang kalimat atau kata kunci, harus diidentifikasi dan diekstrak dengan teliti. Proses ini memerlukan keahlian khusus dan waktu yang cukup panjang, sebaliknya *deep learning* dapat melakukan ekstrak fitur dengan otomatis melalui arsitektur jaringan saraf yang dimilikinya.

Arsitektur *deep learning* terdiri dari beberapa lapisan yang saling berhubungan, mirip seperti cara kerja neuron dalam otak manusia, yang dimana setiap lapisan mampu menerima *Input*, melakukan transformasi, dan menghasilkan *output* yang selanjutnya menjadi *Input* untuk lapisan berikutnya. Proses ini memungkinkan *deep learning* untuk membangun representasi data yang semakin abstrak dan kompleks pada setiap lapisan, semakin dalam lapisan jaringan saraf semakin kompleks pula fitur yang dapat diidentifikasi.

2.1.2. Arsitektur dasar *Deep learning*

Arsitektur *deep learning* dibangun berdasarkan konsep *neural networks* atau jaringan saraf tiruan yang terinspirasi dari cara kerja otak manusia. *Neural networks* terdiri dari sekumpulan unit pemrosesan sederhana yang saling terhubung satu sama lain dan bekerja sama untuk memproses informasi. Setiap unit dalam jaringan menerima *Input*, melakukan perhitungan sederhana, dan menghasilkan *output* yang akan diteruskan ke unit berikutnya (Taye 2023).

Struktur dasar *neural networks* terdiri dari tiga jenis lapisan utama. Lapisan pertama yang disebut *Input layer* bertugas untuk menerima data mentah, misalnya dalam konsep analisis sentimen, *Input layer* menerima teks yang akan dianalisis. Lapisan kedua yaitu lapisan *hidden layer* yang memiliki funsi untuk memproses dan mentransformasi data. Lapisan terakhir yang disebut sebagai *output layer* merupakan lapisan yang akan menghasilkan *output* dari data mentah yang di *Input* sesuai dengan tugas yang diberikan, seperti klasifikasi sentimen yang memiliki nilai positif, netral, atau negatif, stuktur *neural network* dari pada *deep learning* dapat dilihat pada gambar di bawah (Anton et al. 2021).



Gambar 2.1 Struktur *Neural Network Deep learning*

Setiap koneksi antar unit dalam *neural networks* memiliki nilai *weight* dan *bias*, yang dimana *weight* merupakan parameter yang digunakan untuk mengukur seberapa besar pengaruh suatu unit neuron terhadap neuron berikutnya. Setiap koneksi antara neuron memiliki bobot yang dapat diubah selama proses pelatihan, bobot ini berfungsi untuk menentukan seberapa kuat sinyal yang diterima oleh neuron, sedangkan *bias* adalah suatu nilai tambahan yang ditambahkan ke hasil keluaran neuron sebelum diterapkan fungsi aktivasi.

Bias memungkinkan model untuk lebih fleksibel dalam menyesuaikan hasil keluarannya, tanpa *bias*, model mungkin tidak dapat mempelajari pola yang kompleks dalam data. Nilai *weight* dan *bias* ini terus diperbarui selama proses pelatihan untuk meningkatkan akurasi model.

Arsitektur *neural nerwork* pada *deep learning* jugan memiliki *activation function* yang berperan penting dalam *neural networks* dengan mengubah *Input* menjadi *output* melalui fungsi matematika tertentu. Fungsi aktivasi memungkinkan model untuk mempelajari pola *non-linear* dalam data. Beberapa *activation function* yang umum digunakan adalah ReLU (*Rectified Linear Unit*) yang mengubah nilai negatif menjadi nol dan mempertahankan nilai positif. Dalam fungsi ReLU, jika *Input x* lebih besar dari nol, maka *output*nya adalah *x* itu sendiri. Namun, jika *Input x* kurang dari atau sama dengan nol, *output*nya akan menjadi nol. Berikutnya fungsi aktivasi sigmoid yang menghasilkan rentang nilai antara 0 dan 1. Hal ini yang membuat fungsi sigmoid sangat berguna untuk menentukan probalitias suatu kelas dalam klasifikasi biner, misalnya jika suatu kelas memiliki nilai yang mendekati 1 maka kemungkinan besar kelas tersebut termasuk kedalam kategori nilai 1, sebaliknya jika kelas tersebut mendekati nilai 0 maka kemungkinan besar nilai tersebut termasuk kedalam kategori nilai 0.

2.1.3. *Training Pipeline* dalam *Deep Learning*

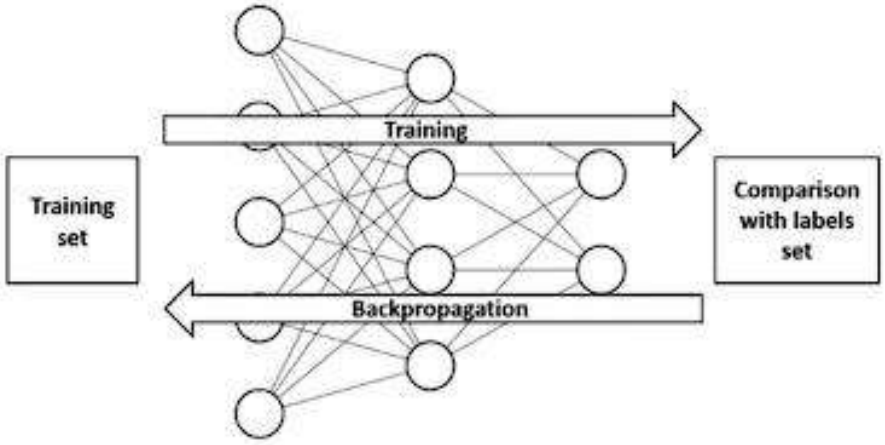
Training pipeline mengatur seluruh proses pelatihan model *deep learning* dari awal hingga akhir. Proses pelatihan dimulai saat data masuk ke dalam model sampai model dapat membuat prediksi dengan tepat. *Pipeline* bekerja seperti rangkaian rantai, di mana setiap bagian saling terhubung dan bergantung satu sama lain.

Forward propagation membuat prediksi, *backpropagation* menghitung kesalahan, *loss function* mengukur kesalahan, *optimizer* memperbaiki kesalahan, dan *learning rate* mengatur kecepatan perbaikan. Semua bagian ini bekerja sama untuk menciptakan model *deep learning* yang dapat belajar dan memprediksi dengan akurat. Berikut adalah penjelasan *training pipeline* dalam *deep learning.*

1. *Forward Propogation*

Forward propagation merupakan proses meneruskan data dari awal hingga akhir melalui jaringan saraf tiruan untuk menghasilkan prediksi. Proses ini dimulai saat data masuk ke lapisan *Input*, kemudian bergerak maju melewati setiap lapisan tersembunyi (*hidden layer*), hingga akhirnya menghasilkan nilai prediksi pada lapisan *output*. Cara kerja *forward propagation* dapat dijelaskan dalam beberapa tahapan, pertama setiap angka atau nilai pada data masuk ke lapisan *Input*, setiap nilai tersebut dikalikan dengan bobot (*weight*) masing-masing yang telah ditentukan. Hasil perkalian tersebut kemudian dijumlahkan dengan nilai bias, selanjutnya hasil penjumlahan akan diproses menggunakan fungsi aktivasi untuk menghasilkan nilai baru.

Nilai baru yang dihasilkan dari lapisan pertama akan diteruskan ke lapisan berikutnya, proses yang sama akan berulang di setiap *hidden layer*, setiap nilai akan dikalikan dengan bobot ditambahkan dengan bias dan diproses menggunakan fungsi aktivasi. Proses perkalian, penjumlahan, dan transformasi ini terus berlanjut hingga mencapai lapisan *output*, gambar *learning process neural network* dapat dilihat pada gambar dibawah (Drewek-Ossowicka, Pietrołaj, and Rumiński 2021).



Gambar 2.2 *Learning process neural network*

Pada lapisan *output*, nilai akhir yang dihasilkan merupakan prediksi dari model *deep learning*. Misalnya dalam analisis sentimen, nilai akhir dapat berupa angka yang menunjukkan probabilitas suatu teks memiliki sentimen positif atau negatif. *forward propagation* berperan penting dalam proses prediksi karena menentukan bagaimana data diproses dan ditransformasi melalui setiap lapisan dalam jaringan saraf tiruan.

Forward propagation dapat dianalogikan seperti proses memasak makanan yang melewati beberapa tahap pengolahan. Bahan mentah (data *Input*) melewati serangkaian proses pengolahan (lapisan tersembunyi) seperti pemotongan, pencampuran, dan pemanasan. Setiap proses mengubah bentuk dan karakteristik bahan hingga akhirnya menjadi makanan jadi (*output* prediksi), setiap tahap pengolahan mempengaruhi hasil akhir, sama seperti setiap lapisan dalam *forward propagation* mempengaruhi hasil prediksi akhir.

2. *Backpropogation*

Backpropagation merupakan proses belajar dalam jaringan saraf tiruan untuk memperbaiki kesalahan prediksi, proses ini bekerja dengan cara menghitung kesalahan prediksi pada lapisan *output*, kemudian bergerak mundur ke setiap lapisan untuk memperbaiki nilai bobot dan bias. Proses perbaikan nilai bobot dan bias bertujuan untuk mengurangi kesalahan prediksi pada iterasi berikutnya, proses *backpropagation* dimulai setelah jaringan saraf tiruan selesai melakukan prediksi melalui *forward propagation*. Langkah pertama dalam *backpropagation* adalah menghitung perbedaan antara nilai prediksi dengan nilai target sebenarnya, perbedaan nilai ini disebut sebagai kesalahan prediksi, kesalahan prediksi menjadi dasar untuk menentukan seberapa besar perbaikan yang perlu dilakukan pada nilai bobot dan bias dalam jaringan (Zaida Muflih 2021).

Setelah mendapatkan nilai kesalahan prediksi, *backpropagation* melakukan perhitungan untuk mengetahui kontribusi setiap bobot dan bias terhadap kesalahan tersebut. Perhitungan dimulai dari lapisan *output* dan bergerak mundur ke setiap lapisan tersembunyi. Pada setiap lapisan, *backpropagation* menghitung seberapa besar pengaruh setiap bobot dan bias terhadap kesalahan prediksi. Informasi ini digunakan untuk menentukan seberapa besar perubahan yang perlu dilakukan pada setiap bobot dan bias. Proses perubahan nilai bobot dan bias dalam *backpropagation* dipengaruhi oleh laju pembelajaran atau *learning rate* yang akan menentukan seberapa besar perubahan yang dilakukan pada setiap nilai bobot dan bias (Sekhar and Meghana 2020).

Proses pembelajaran menggunakan *backpropagation* akan berhenti ketika mencapai kondisi tertentu, seperti jumlah pengulangan maksimal yang telah ditentukan, nilai kesalahan prediksi yang sudah cukup kecil, atau waktu pelatihan yang sudah mencapai batas maksimal, setelah proses pembelajaran selesai, jaringan saraf tiruan dapat digunakan untuk melakukan prediksi pada data baru. *Backpropagation* memiliki peran sangat penting dalam pembelajaran jaringan saraf tiruan karena memungkinkan model untuk memperbaiki kesalahan secara otomatis, tanpa *backpropagation*, tidak mungkin melakukan perbaikan nilai bobot dan bias secara sistematis untuk mengurangi kesalahan prediksi. Kemampuan *backpropagation* untuk memperbaiki parameter model secara otomatis menjadikan *deep learning* sangat efektif dalam menyelesaikan berbagai tugas kompleks seperti analisis sentimen, pengenalan gambar, atau pemrosesan bahasa alami.

Dalam konteks analisis sentimen, *backpropagation* membantu model untuk mempelajari pola-pola dalam teks yang menunjukkan sentimen positif atau negative, setiap kali model melakukan kesalahan dalam memprediksi sentimen, *backpropagation* akan memperbaiki nilai bobot dan bias sehingga model menjadi lebih baik dalam memahami nuansa bahasa dan konteks kalimat. Proses pembelajaran berkelanjutan ini memungkinkan model untuk mencapai tingkat akurasi yang tinggi dalam menganalisis sentimen teks.

3. *Loss Function*

Loss function adalah komponen penting dalam pelatihan model pembelajaran mendalam, karena mereka mengukur seberapa baik model dapat memprediksi hasil yang diinginkan. Dalam jurnal yang berjudul "*loss functions and metrics in deep learning*" (Terven et al. 2023) menjelaskan berbagai *loss function* yang digunakan untuk tugas-tugas seperti regresi dan klasifikasi.

Untuk regresi, terdapat fungsi seperti *Mean Squared Error* (MSE), yang menghitung rata-rata kuadrat perbedaan antara nilai yang diprediksi dan nilai aktual, serta *Mean Absolute Error* (MAE), yang mengukur rata-rata perbedaan absolut. *loss function* untuk klasifikasi meliputi *binary cross entropy* (BCE) untuk klasifikasi biner dan *categorical cross entropy* (CCE) untuk klasifikasi multi-kelas.

Jurnal ini juga membahas sifat-sifat penting dari *loss function*, seperti diferensiasi, dan ketahanan terhadap *outlier*, yang mempengaruhi pemilihan fungsi yang tepat untuk setiap tugas. Jurnal ini memberikan contoh aplikasi *loss function* dalam berbagai bidang, seperti *computer vision* dan *natural language processing*.

Selama pelatihan, teks dimasukkan ke dalam model, kemudian model memberikan prediksi, setelah itu fungsi kehilangan atau *loss function* menghitung seberapa besar kesalahan dari prediksi tersebut. Setelah proses pelatihan selesai, metrik akurasi digunakan untuk melakukan penilaian seberapa baik model dapat bekerja dengan data baru. Memilih fungsi kehilangan yang tepat sangat penting agar model dapat memahami sentimen dengan lebih baik dan memberikan hasil yang akurat dalam analisis teks.

2.1.4. *Machine learning*

Machine learning (ML) adalah cabang dari kecerdasan buatan (*Artificial Intelligence*/AI) yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. Secara umum, *machine learning* dapat diklasifikasikan menjadi tiga kategori utama: *supervised learning*, *Unsupervised learning*, dan *reinforcement learning*.

Menurut (Mitchell 1997) dalam bukunya yang berjudul *"Machine learning," machine learning* dapat didefinisikan sebagai *"a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."* Artinya, suatu program komputer dikatakan belajar dari pengalaman E terhadap beberapa kelas tugas T dan ukuran kinerja P, jika kinerjanya pada tugas-tugas dalam T, yang diukur dengan P, meningkat dengan pengalaman E.

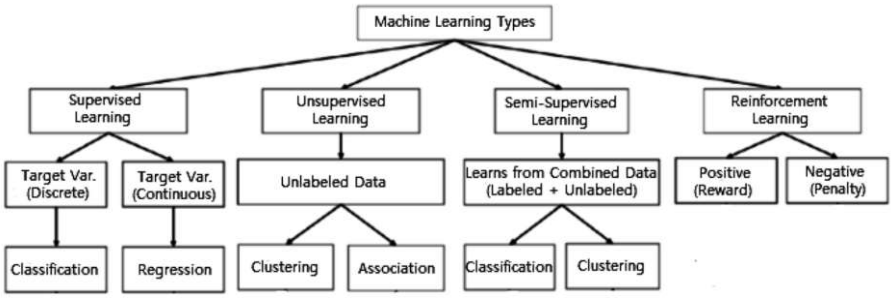
Dari pengertian *machine learning* menurut Mitchell, adalah proses di mana sebuah program computer dikatakan “belajar” jika program tersebut bisa meingkatkan kinerjanya dalam menyelesaikan tugas tugas tertentu berdasarkan pengalaman. Jadi, misalnya ada tiga komponen utama,

a) Pengalaman (E): Data atau informasi yang diperoleh program dari lingkungan atau pelatihan sebelumnya.
b) Tugas(T): Pekerjaan atau masalah spesifik yang harus diselesaikan oleh program.

c) Ukuran Kinerja(P): Cara untuk mengukur seberapa baik program menyelesaikan tugas tersebut.

Jika program komputer menjadi lebih baik dalam menyelesaikan tugas-tugas ini setelah mendapatkan lebih banyak data atau pengalaman, maka program tersebut dianggap telah belajar, contoh sederhananya jika kita mengajari program komputer untuk mengenali gambar kucing (tugas T), dan setiap kali program melihat gambar kucing baru (pengalaman E), program tersebut menjadi lebih baik dalam mengenali gambar kucing dengan akurasi yang lebih tinggi (ukuran kinerja P), maka program itu dikatakan telah belajar. Dengan kata lain, *machine learning* adalah tentang program komputer yang meningkatkan kemampuannya untuk melakukan tugas tertentu dengan lebih baik setelah mempelajari data atau pengalaman sebelumnya.

Dalam *machine learning* algoritma digunakan untuk menangani data histrois dalam jumlah besar serta mengidentifikasi pola dari sebuah data. Komputer dapat belajar dari data dan membuat keputusan atau prediksi tanpa perlu diprogram dengan menginstruksikan langkah langkah detail untuk setiap tugas yang akan dikerjakan (Sitohang 2021). *Machine learning* dapat membuat komputer belajar dan bekembang dengan sedirinya melalui data yang diberikan, dan oleh karena itu *machine learning* digunakan menjadi inovasi untuk perkembangan teknologi di zaman modern. Gambar 2.5 (Sarker 2021) merupakan gambar Berbagai jenis teknik *machine learning.*



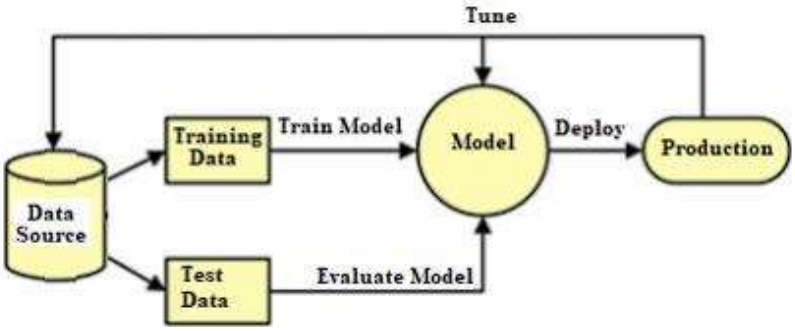
Gambar 2.3 Berbagai jenis teknik *machine learning*

Machine learning banyak digunakan dalam berbagai aplikasi, mulai dari pengenalan gambar hingga rekomendasi produk. Ada tiga cara bagaimana model *machine learning* belajar melalui data, cara belajar yang pertama yaitu pembelajran yang diawasi (*supervised learning*), pembelajaran tanpa pengawasan (*unsupervised learning*) dan pembelajaran penguatan (*reinforcement learning*), ketiga cara itu penting untuk diketahui agar dapat lebih memahami tentang *machine learning* dengan lebih baik,

2.1.5. *Supervised learning*

Dalam *supervised learning*, model dilatih dengan menggunakan data yang telah diberi label dan bertujuan untuk mempelajari hubungan antara *Input* dan *output* untuk membuat prediksi pada data baru yang tidak berlabel. Algoritma *supervised learning* membutuhkan bantuan eksternal dari manusia untuk menyediakan data pembelajaran atau pelatihan yang akan dibagi menjadi set data latih dan set data uji. Data yang digunakan untuk pelatihan memiliki variabel keluaran yang pola nya akan di pelajari oleh algoritma *supervised learning* dan menerapkannya ke data uji untuk prediksi dan klasifikasi (Mahesh 2024).

Algoritma yang umum digunakan dalam *supervised learning* meliputi regresi linier, regresi logistik, pohon keputusan, *random forest*, dan *support vector machine* (SVM). Alur kerja algoritma pembelajaran mesin yang diawasi dapat dilihat pada gambar di bawah ini (Mahesh 2024)



Gambar 2.4 *Supervied Learning Workflow*

Supervised learning bekerja dengan menggunakan data latih yang memiliki label di mana setiap data memiliki *Input* dan *output* yang sudah diketahui. Misalnya dalam konteks spam email, setiap data sudah diberi label "spam" dan "tidak spam," yang akan dipelajari oleh model agar nantinya dapat malakukan klasifikasi pada data baru yang tidak memiliki label. Terdapat beberapa contoh algoritma dalam *supervised learning* sebagai berikut:

a) *Linear Regression*: Digunakan untuk memprediksi nilai kontinu berdasarkan hubungan *linear* antara variabel *Input* dan *output*.

b) *Logistic Regression: Digunakan untuk klasifikasi biner, seperti memprediksi apakah email adalah spam atau tidak.*

c) *Support Vector Machine (SVM): Algoritma yang mencari hyperplane optimal untuk memisahkan kelas data.*

2.1.6. *Unsupervised learning*

Dalam *Unsupervised learning* model dilatih dengan data yang tidak berlabel, dan tujuan dari jenis pembelajran ini adalah untuk menemukan pola atau struktur tersembunyi dalam data. Pembelajaran tanpa pengawasan atau *unupervised learning* sama halnya dengan menyuruh seseorang belajar sendiri tentang suatu hal tanpa adanya arahan atau contoh yang jelas. *Unsupervised learning* umumnya digunakan untuk melakukan tugas seperti *cluster*ing, estimasi kepadatan, pembelajaran fitur, pengurangan dimensi, analisis asosiasi, deteksi anomali, dll (Sarker 2021).

Unsupervised learning bekerja dengan data yang tidak diberi label dan bertujuan untuk menemukan struktur tersembunyi dalam data, seperti pengelompokan atau pengurangan dimensi. Algoritma ini mengidentifikasi pola atau kelompok berdasarkan kemiripan antara data tanpa panduan yang jelas, berikut ini merupakan beberapa contoh algoritma dalam *unsupervised learning.*

a) *K-means Clustering*: Algoritma yang membagi data ke dalam kelompok (*cluster*) berdasarkan kemiripan fitur.
b) *Hierarchical Clustering:* Algoritma yang membangun hierarki *cluster* berdasarkan kesamaan data.
c) *Principal Component Analysis (PCA):* Teknik untuk mengurangi dimensi data dengan menemukan variabel utama yang menjelaskan sebagian besar variabilitas dalam data.
d) *Association Rules:* Algoritma yang menemukan hubungan antara variabel dalam dataset besar, seperti dalam analisis keranjang belanja.

2.1.7. *Reinforcement learning*

Reinforcement learning adalah jenis *machine learning* di mana sistem belajar untuk membuat keputusan dengan mencoba berbagai tindakan dan menerima umpan balik dalam bentuk *reward* atau *punishment*. Tujuan agen adalah memaksimalkan *reward* jangka panjang melalui serangkaian Tindakan (Kommey et al. 2024).

Reinforcement learning (RL) adalah teknik *Machine learning* (ML) yang berada di bawah payung *Artificial Intelligence* (AI). *Reinforcement learning* memungkinkan mesin dan perangkat lunak untuk belajar sendiri dalam menentukan perilaku terbaik dalam situasi tertentu, dengan tujuan memaksimalkan kinerja. Agen RL belajar melalui umpan balik yang disebut sinyal penguatan, yang memberi petunjuk apakah tindakan yang diambil itu baik atau buruk.

Contohnya, *platform* media sosial yang terintegrasi dengan perangkat IoT menggunakan RL untuk secara otomatis mengenali wajah orang atau mengidentifikasi objek umum, seperti *landmark* dalam foto yang diunggah. Ada banyak algoritma RL yang bisa digunakan untuk berbagai aplikasi ini, dan algoritma tersebut juga terus berkembang seiring dengan waktu pembelajaran yang makin efektif (Shafik et al. 2020), berikut ini adalah beberapa contoh algoritma dalam *reinforcement learning.*

a) *Q-Learning*: Algoritma yang menggunakan tabel Q untuk menyimpan nilai *reward* yang diharapkan dari setiap tindakan dalam setiap keadaan.

b) *Deep Q-Network (DQN)*: Algoritma yang menggunakan jaringan saraf dalam untuk memprediksi nilai Q, sangat efektif dalam lingkungan yang kompleks seperti permainan video.

2.1.8. *Natural Language Processing*, *Transformers*, *BERT*, dan *RoBERTa*

Natural Language Processing (NLP) menjadi cabang dari kecerdasan buatan yang berfokus pada interaksi antara komputer dan manusia melalui bahasa alami manusia (Nur Oktavia et al. 2024). Seiring dengan kemajuan teknologi, model-model berbasis NLP telah mengalami perkembangan yang pesat, sehingga analisis dan pemrosesan teks dapat dilakukan dengan lebih akurat dan efisien. *Transformers* menjadi salah satu inovasi yang besar dalam perkembangan NLP yang memperkenalkan mekanisme perhatian (*attention*) untuk memahami konteks secara lebih baik.

Dari algoritma *Transformers* ini, lahirlah berbagai jenis model NLP yang canggih seperti *BERT* (*Bidirectional Encoder Representations from Transformers*) yang mampu memahami bahasa dengan melihat konteks dati kata kata sebelum dan sesudahnya secara bersamaan dengan menggunakan teknik bidirectional attention. *RoBERTa* (*Robustly Optimized BERT Pretraining Approach*) adalah versi optimasi dari *BERT* yang memberikan kinerja lebih baik dalam berbagai tugas NLP. Bagian ini akan membahas secara mendalam tentang NLP, *Transformers*, *BERT*, dan *RoBERTa*, menjelaskan konsep, arsitektur, dan aplikasi mereka dalam analisis sentimen yang menjadi fokus penelitian ini.

2.1.9. *Natural Language Processing* (NLP)

NLP memungkinkan komputer untuk membaca, memahami, dan menghasilkan teks dengan bahasa manusia dan biasa digunakan pada alat penerjemah seperti *Google Translate* dan *DeepL*. NLP merupakan jembatan antara komputer dan bahasa manusia, memungkinkan mesin untuk menganalisis, menafsirkan, dan mengekstrak informasi yang berguna dari teks atau ucapan.

NLP mencakup berbagai bidang, seperti *chatbot*, asisten virtual, terjemahan mesin, dan analisis sentimen. Dengan meningkatnya permintaan untuk teknologi yang dapat memproses bahasa manusia, banyak startup yang berfokus pada pengembangan solusi NLP. Berikut adalah beberapa contoh penerapan NLP dalam kehidupan sehari-hari:

a) *Chatbot* dan Asisten Virtual: *Chatbot* seperti siri, alexa, dan *google assistant* menggunakan NLP untuk memahami dan merespons pertanyaan atau perintah pengguna.

b) Terjemahan Mesin: Layanan seperti *google translate* menggunakan NLP untuk menerjemahkan teks dari satu bahasa ke bahasa lain.

c) Analisis Sentimen: Perusahaan menggunakan NLP untuk menganalisis opini pengguna di media sosial, ulasan produk, dan survei untuk memahami sentimen publik terhadap produk atau layanan mereka.

d) Pencarian Informasi: Mesin pencari seperti Google menggunakan NLP untuk memahami dan menjawab pertanyaan pengguna dengan memberikan hasil pencarian yang relevan.

NLP bekerja dengan memproses bahasa manusia dalam beberapa langkah, seperti *preprocessin* yang mengubah teks mentah menjadi format yang dapat diproses oleh komputer. Ini melibatkan beberapa langkah seperti tokenisasi, penghilangan *stop words*, dan *stemming* atau *lemmatization*.

Langkah berikutnya sintaksis dan parsing, pada tahap ini terdapat proses untuk menganalisis struktur gramatikal teks untuk memahami hubungan antar kata dan frasa. Ini dapat melibatkan pengenalan bagian-bagian dari kalimat *(Part-of-Speech Tagging*) dan analisis dependensi.

Yang terakhir langkah semantik, disini mesin mencoba untuk memahami makna teks. Ini mencakup tugas-tugas seperti pengenalan entitas bernama (*Named Entity Recognition*) dan analisis sentimen. *Natural Language Processing* (NLP) terdiri dari berbagai teknik dasar yang memungkinkan komputer untuk memahami dan memproses bahasa manusia. Teknik-teknik ini adalah dasar dari berbagai aplikasi NLP yang lebih kompleks. Berikut adalah beberapa teknik dasar dalam NLP yang penting untuk dipahami:

1. Tokenisasi

Tokenisasi adalah proses memecah teks menjadi unit-unit kecil yang disebut "token." Token ini biasanya berupa kata-kata, frasa, atau karakter individual. Misalnya, kalimat "Saya suka belajar NLP" dapat dipecah menjadi token-token ["Saya", "suka", "belajar", "NLP"]. Tokenisasi adalah langkah awal yang sangat penting dalam *preprocessing* teks karena memungkinkan analisis lebih lanjut pada tingkat kata atau frasa.

2. *Stop Words Removal*

Stop words adalah kata-kata umum yang biasanya tidak memiliki makna penting dalam analisis teks, seperti "dan," "atau," "yang," dan "di." Menghilangkan *stop words* dari teks dapat meningkatkan efisiensi dan akurasi model NLP dengan mengurangi jumlah data yang perlu diproses. Misalnya, dari kalimat "Saya suka belajar NLP di malam hari," kita bisa menghilangkan "di" dan "hari."

3. *Stemming* dan *Lemmatization*

Stemming dan *lemmatization* adalah teknik untuk mengurangi kata-kata ke bentuk dasarnya. *Stemming* menghapus akhiran dari kata untuk mendapatkan bentuk dasar (stem), meskipun hasilnya mungkin tidak selalu merupakan kata yang benar secara gramatikal. *Lemmatization*, di sisi lain, mengubah kata ke bentuk dasar (lemma) yang benar secara gramatikal dengan mempertimbangkan konteks, misalnya kata-kata "berlari," "berlari-lari," dan "lari-lari" dapat dikembalikan ke bentuk dasar "lari."

4. *Part-of-Speech* (POS) *Tagging*

POS *tagging* adalah proses memberi label pada setiap kata dalam teks dengan kategori tata bahasa yang sesuai, seperti kata benda, kata kerja, kata sifat, dll. POS *tagging* membantu dalam memahami struktur kalimat dan hubungan antar kata misalnya dalam kalimat "Saya belajar NLP," kata "Saya" diberi label sebagai kata ganti, "belajar" sebagai kata kerja, dan "NLP" sebagai kata benda.

5. Parsing

Parsing adalah proses menganalisis struktur gramatikal teks dan membangun pohon sintaksis yang merepresentasikan hubungan antar kata dalam kalimat. Parsing membantu dalam memahami tata bahasa dan makna kalimat yang bisa berupa parsing sintaksis, yang fokus pada struktur kalimat, atau parsing dependensi, yang fokus pada hubungan dependensi antar kata.

6. TF-IDF (*Term Frequency-Inverse Document Frequency*)

TF-IDF adalah teknik yang meningkatkan representasi *Bag of Words* dengan memperhitungkan pentingnya sebuah kata dalam dokumen tertentu dan di seluruh korpus. TF (*Term Frequency*) mengukur seberapa sering sebuah kata muncul dalam dokumen, sedangkan IDF (*Inverse Document Frequency*) mengukur seberapa jarang kata tersebut muncul di korpus.

7. *Word Embeddings*

Word Embeddings adalah representasi kata dalam bentuk vektor yang menangkap makna semantik kata-kata berdasarkan konteksnya. Teknik seperti *Word2Vec, GloVe*, dan *FastText* menghasilkan vektor kata yang memungkinkan model untuk memahami kesamaan semantik antar kata. Misalnya, kata-kata "raja" dan "ratu" mungkin memiliki vektor yang dekat satu sama lain dalam ruang vektor, menunjukkan kesamaan semantik mereka.

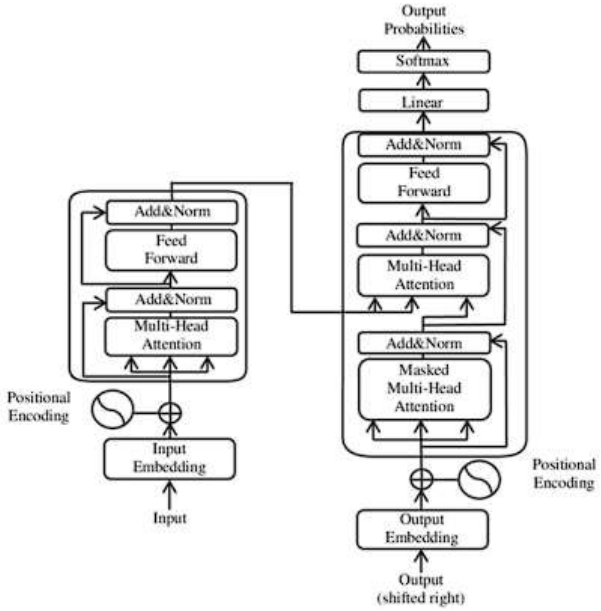
8. *N-grams*

N-grams adalah serangkaian n kata yang berurutan dalam teks. Mereka digunakan untuk menangkap hubungan dan pola antar kata yang berdekatan. Misalnya, dalam *bigram* (2-gram), kalimat "Saya suka belajar NLP" akan dipecah menjadi ["Saya suka", "suka belajar", "belajar NLP"].

2.1.10. Transformers

Transformers, arsitektur jaringan saraf yang digunakan dalam pembelajaran mesin yang biasa digunakan pada tugas tugas pemrosesan bahasa alami manusia, terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas untuk menganalisis dan mengubah data *Input* menjadi representasi yang lebih mudah dipahami, sementara *decoder* menghasilkan *output* berdasarkan representasi tersebut. Transformers memiliki kemampuan untuk memahami hubungan konteks antar kata, sehingga sering digunakan dalam pemrosesan bahasa alami seperti penerjemahan, analisis sentimen, dan pengenalan gambar. (Thoyyibah T, Wasis Haryono, Achmad Udin Zailani, Yan Mitha Djaksana, Neny Rosmawarni 2023).

Dalam jurnal *"Attention is All You Need"* yang diterbitkan pada tahun 2017, *Transformers* telah merevolusi berbagai tugas NLP karena kemampuannya untuk memproses dan memahami konteks dalam sebuah kata lebih efektif dibandingkan algoritma lainnya seperti *Recurrent Neural Networks* (RNN) dan *Long Short-Term Memory* (LSTM). Arsitektur transformers dapat dilihat pada Gambar 2.10 (Firmanto, Aziz, and Sesoca 2024) di bawah.



Gambar 2.5 Arsitektur Transformers

Transformers menggunakan mekanisme yang disebut "*Self-attention*" yang bekerja seperti pengamat yang memeriksa setiap kata pada kalimat lalu menentukan kata mana yang saling berhubungan dan seberapa besar hubungan setiap kata tersebut dengan tujuan untuk membantu model dalam memahami sebuah konteks dalam sebuah kalimat dengan lebih baik (Bahari and Dewi 2024). Berikut adalah beberapa konsep utama yang membentuk arsitektur *Transformers*:

1) *Self-attention Mechanism*

Self-attention menjadi inti dari arsitektur *transformers* yang dimana mekanisme ini akan menghubungkan posisi pada setiap kata yang berbeda dari satu urutan kata untuk menghitung representasi dari urutan tersebut. *Self-attention* menghitung *attention scores* yang menunjukkan seberapa relevan satu kata terhadap kata lainnya, ini memungkinkan model untuk menangkap hubungan jangka panjang dan konteks secara keseluruan dalam teks atau kalimat (Roy et al. 2023).

Pada Proses *Self-attention,* setiap kata yang masuk akan menghasilkan 3 vektor yang berbeda *query, key*, dan *value* dan setiap kata yang masuk akan dihitung kecocokan antara *query* dan *key* nya dengan cara mengalikan nilainya. Hasil perkalian tersebut akan diubah menjadi angka-angka yang mewakili tingkat perhatian menggunakan fungsi *softmax* dan angka-angka ini digunakan untuk memberikan bobot pada value sehingga menghasilkan keluaran akhir yang lebih bermakna.

2) *Positional Encodings*

Di dalam algoritma *transformers, positional encodings* digunakan untuk memberi urutan kepada setiap kata yang di *Input*, hal ini penting agar model dapat memahami makna dari sebuah kata dan memahami konteks dari kalimat yang di *Input*. Transformers pada dasarnya bersifat *permutation equivariant*, yang berarti algoritma ini tidak dirancang secara alami untuk memperhatikan urutan dari token *Input* (Chen et al. 2021).

3) *Multi-head Attention*

Multi-head attention yang menjadi pengembangan dari mekanisme *Self-attention* dapat membatnu model dalam memahami hubungan antar kata dalam sebuah kalimat yang di*Input*. Sama seperti *selft-attention*, kata yang masuk akan di bagi menjadi tiga vector yang berbeda, *query, key*, dan *value* yang dimana ketiga bagian itu akan dihitung secara bersamaan didalam beberapa bagian yang disebut sebagai *heads*, dan hasil dari setiap *heads* akan digabungkan untuk melihat representasi akhir sehingga model dapat memahami berbagai pola hubungan, baik sintaksis maupun *semantic* secara lebih efektif.

4) *Feed-Forward Neural Networks*

Setelah proses *multi-head attention*, representasi yang dihasilkan diproses oleh jaringan saraf *feed-forward* yang terdiri dari lapisan *dense* dengan aktivasi *non-linear*. Proses ini dilakukan secara independen pada setiap posisi dalam urutan dan membantu dalam transformasi representasi yang lebih kompleks.

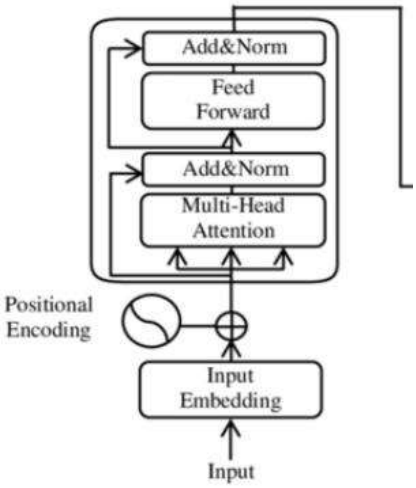
Arsitektur *transformers* terdiri dari dua komponen utama *encoder* dan *decoder*. *Encoder* terdiri dari beberapa lapisan (*layers*) dan setiap lapsan memiliki dua sub lapisan yang pertama adalah mekanisme *multi-head Self-attention* dan yang kedua adalah jaringan *feed-forward* yang bertugas untuk mengubah urutan *Input* (seperti kalimat) menjadi representasi yang dapat dipahami oleh model.

Decoder juga terdiri dari beberapa lapisan dan juga memiliki sub lapisan pada setiap lapisannya, *decoder* juga menyisipkan sub lapisan ketiga berfungsi sebagi *multi-head attention* pada tumpukan *encoder*. *Decoder* bertugas untuk mengubah representasi yang dihasilkan oleh *encoder* menjadi urutan *output* (misalnya, terjemahan)

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

BERT, atau *Bidirectional Encoder Representations from Transformers*, adalah model berbasis transformers yang dirancang untuk memahami konteks dari kedua arah dalam sebuah teks, baik dari kiri ke kanan maupun dari kanan ke kiri. Model ini diperkenalkan oleh Jacob Devlin dan timnya dari Google AI pada tahun 2019 dan dengan cepat menjadi salah satu model paling berpengaruh di bidang *Natural Language Processing* (NLP).

Sebelum munculnya *BERT*, model NLP masih mengandalkan pendekatan sekuensial seperti *Recurrent Neural Network*s (RNN) dan *Long Short-Term Memory* (LSTM) yang memproses teks secara bertahap dari kiri ke kanan atau sebaliknya. Meskipun efektif, pendekatan ini memiliki keterbatasan dalam menangkap relasi jangka panjang dalam teks dan proses nya juga lebih lambat karena sifat sekuensialnya. Arsitektur BERT dapat di lihat pada gambar 2.11 (Firmanto et al. 2024) dibawah.



Gambar 2.6 Arsitektur BERT

BERT dilatih menggunakan teks yang tidak berlabel, dan model belajar untuk memprediksi kata-kata yang hilang dalam sebuah kalimat, ini dikenal sebagai *masked language model* (Sayeed, Mohan, and Muthu 2023). Misalnya, jika dalam kalimat "Saya pergi ke [MASK] untuk membeli buah," BERT akan mencoba menebak kata yang hilang berdasarkan konteks kalimat tersebut. Teknik ini memungkinkan BERT untuk memahami konteks yang lebih luas dan hubungan antar kata dalam kalimat dengan lebih baik.

Dalam proses pelatihan BERT terdapat dua tahapan utama yang harus di lewati, yaitu tahap *pre-training* dan *fine-tuning*. Tahap pertama pada proses *pre-training*, melibatkan tugas-tugas tanpa pengawasan untuk membantu model memahami bahasa secara umum. Setelah itu, pada proses kedua di tahap *fine-tuning*, BERT disesuaikan atau di latih ulang agar dapat mengerjakan tugas-tugas spesifik seperti analisis sentimen atau menjawab pertanyaan. Berikut ini penjelasan lebih detailnya.

1) *Pre-Training Tasks*

Proses *pre-training* merupakan tahapan awal dalam proses pelatihan model BERT dengan menggunakan dua teknik pelatihan unsupervised seperti *Masked Language Modeling* dan *Next Sentence Prediction.* Selama proses pelatihan menggunakan teknik *Masked Language Modeling*, sebagian kata yang di*Input* akan di *mask* atau disamarkan dengan tujuan agar model dapat belajar dengan memprediksi kata-kata yang disamarkan. Dalam teknik MLM ada sebanyak 15% token dalam kalimat akan di mask, dan model akan mencoba memprediksi token asli berdasarkan konteks yang terdapat pada kalimat tersebut.

Dalam proses *next sentence prediction* model memiliki sepasang kalimat sebagai *Input* data dan harus menentukan apakah kedua kalimat tersebut saling memliliki hubungan dalam konteks yang sama. Dalam proses ini, 50% pasangan kalimat merupakan pasangan yang benar-benar saling berhubungan dan berurutan, sedangkan 50% lainnya merupakan pasangan acak yang tidak berurutan.

2) *Fine-tuning*

Setelah *pre-training*, BERT dapat di *fine-tune* untuk berbagai tugas NLP spesifik. Proses *fine-tuning* melibatkan beberapa tahapan seperti tahap pergantian lapisan *output*, pada tahap ini lapisan *output* dari model BERT diganti dengan lapisan baru yang sesuai dengan tugas spesifik yang akan dikerjakan, seperti tugas klasifikasi sentimen, penjawaban pertanyaan, atau pemrosesan teks lainnya.

Pada tahap *fine-tuning* selanjutnya, model kemudian dilatih pada dataset spesifik untuk tugas tersebut dengan menggunakan label yang tersedia. Karena parameter inti model sudah dipelajari selama *pre-training*, *fine-tuning* hanya memerlukan penyesuaian kecil pada parameter ini, sehingga proses ini relatif cepat.

BERT menggunakan representasi *Input* yang kaya dan mendetail untuk menangkap konteks kata dalam teks. Representasi *Input* ini terdiri dari tiga jenis *Embeddings* yang digabungkan, *Embeddings* pertama merupakan *token Embeddings*, ini adalah representasi vektor dari kata-kata individu dalam teks. BERT menggunakan *word piece Embeddings*, yang berarti setiap kata dipecah menjadi sub-kata atau token. Misalnya, kata "*playing*" mungkin dipecah menjadi "*play*" dan "##ing". Ini memungkinkan model untuk memahami kata-kata yang tidak umum atau tidak dikenal dengan lebih baik.

Embeddings yang kedua adalah *segment Embeddings*, *segment Embeddings* memungkinkan BERT untuk dapat memhami konteks, dan hubungan dalam teks, misalnya semua token dalam kalimat pertama akan memiliki *segment Embedding* yang sama (misalnya 0), dan semua token dalam kalimat kedua akan memiliki *segment Embedding* yang lain (misalnya 1), ini memungkinkan BERT untuk

memahami konteks dalam kalimat dengan lebih baik, dan yang terakhir adalah *Positional Embeddings*, Transformer tidak memiliki cara bawaan untuk memahami urutan kata dalam teks. Oleh karena itu, *positional Embeddings* digunakan untuk menunjukkan posisi setiap kata dalam kalimat. Ini adalah vektor yang memberi tahu model posisi setiap kata dari setiap token dalam *Input*.

Ketiga *Embeddings* ini, token, *segment*, dan *positional Embedding*, digabungkan dengan cara dijumlahkan untuk membentuk representasi akhir dari setiap token sebagai *Input*, sehingga mencakup informasi lengkap tentang identitas kata, konteks kalimat, dan urutan posisinya dalam teks. Selanjutnya, representasi *Input* ini diproses oleh tumpukan *encoder* dalam arsitektur transformer, yang dirancang untuk memahami hubungan antar token dalam sebuah kalimat maupun antar kalimat. Dalam penerapannya, BERT tersedia dalam dua versi utama, yaitu BERT-Base, yang memiliki 12 lapisan *encoder* masing-masing dengan 768unit *hidden layers* dan 12 *heads* dengan 110 juta parameter, dan BERT-Large, yang lebih kompleks dengan 24 lapisan *encoder* masing-masing dengan 1024unit *hidden layers* dan 16 *heads* serta 340 juta parameter, sehingga dapat menangani tugas-tugas pemrosesan bahasa alami dengan lebih presisi.

2.1.12. *Indonesian-RoBERTa*

Setelah tim Googel AI menciptakan BERT pada tahun 2018 satu tahun setelah nya pada tahun 2019, tim Facebook AI kemudian mengembangkan model BERT yang diberi nama *RoBERTa*. *RoBERTa* menunjukkan bahwa dengan pelatihan yang lebih lama dan pengoptimalan tertentu, dapat mencapai performa yang lebih baik

pada berbagai tugas NLP (Liu et al. 2019). *Indonesian-RoBERTa* menggunakan arsitektur yang sama dengan *RoBERTa*, yang merupakan varian yang lebih baik dari arsitektur BERT. Berikut adalah komponen utama dari arsitektur *Indonesian-RoBERTa*:

1) *Tokenization*

Indonesian-RoBERTa menggunakan tokenisasi berbasis *WordPiece* atau *Byte-Pair Encoding* (BPE) yang telah disesuaikan dengan bahasa Indonesia. BPE atau *Byte-Pair Encoding* bekerja dengan memecah kata-kata menjadi sub-kata atau token yang lebih kecil, memungkinkan model untuk mengenali kosa kata yang besar dan beragam dengan lebih efisien.

2) *Embeddings*

Modeel *Indonesian-RoBERTa* hanya memiliki 2 *Embeddings* yaitu *Token Embeddings* dan *positional Embeddings,* tidak ada *segment Embeddings* karena model ini tidak nemiliki NSP. Model *Indonesian-RoBERTa* memiliki *Masked Language Model* yang bekerja dengan melatih atau mengajari model untuk memprediksi kata kata yang hilang atau "*termask*" dalam sebuah kalimat (Sinha et al. 2021). MLM di *Indonesian-RoBERTa* dan *RoBERTa* memiliki perbedaan dengan MLM yang ada di BERT, MLM pada model *Indonesian-RoBERTa* menggunakan *dynamic masking* yang artinya setiap kalimat bisa memliliki pola msaking yang berbeda di setiap *epoch,* yang membuat pelatihan model menjadi lebih efisien (Naseer et al. 2022).

3) *Encoder stack*

Indonesian-RoBERTa menggunakan tumpukan *encoder* yang terdiri dari beberapa lapisan *Self-attention* dan *Feed-Forward Neural Networks*. *Encoder* ini dirancang untuk menangkap hubungan antar token dalam teks secara mendalam dari dua arah. *RoBERTa*, menerapkan beberapa optimasi dalam proses pelatihan yang lebih baik apabila dibandingkan dengan BERT, seperti jumlah data pelatihan yang lebih besar.

RoBERTa dilatih lebih lama dengan jumlah iterasi yang lebih banyak, untuk memastikan bahwa model mengenali lebih banyak variasi dalam data dan mampu menangkap pola yang lebih kompleks dengan lebih baik, begitu juga dengan model *Indonesian-RoBERTa* yang memiliki arsitektur dan pendekatan yang sama dengan *RoBERTa* yang dilatih menggunakan dataset SmSA yang merupakan kumpulan komentar dan ulasan online berbahasa Indonesia yang telah diberi label sentimen (positif, negatif, atau netral)

Model *Indonesian-RoBERTa* memiliki performa yang luar biasa pada berbagai tugas NLP dalam bahasa Indonesia seperti klasifikasi teks, *named entity recognition*, dan *machine translation*. *Indonesian-RoBERTa* digunakan untuk mengklasifikasikan teks dalam beberapa tugas seperti sentimen analisis, topik dokumen, dan pengenalan entitas, model ini menunjukkan akurasi yang lebih tinggi dibandingkan model sebelumnya yang tidak dioptimalkan untuk bahasa Indonesia.

Pada tugas *named entity recognition* (NER) *Indonesian-RoBERTa* mampu mengenali dan mengklasifikasikan entitas seperti nama orang, tempat, dan organisasi dalam teks bahasa Indonesia dengan tingkat akurasi yang tinggi.

Meskipun *Indonesian-RoBERTa* bukan model penerjemah, representasi yang dihasilkan oleh *Indonesian-RoBERTa* dapat digunakan sebagai bagian dari pipeline penerjemahan untuk meningkatkan kualitas terjemahan dari dan ke bahasa Indonesia

2.1.13. *Software* pendukung

Dalam penelitian ini, beberapa *software* pendukung digunakan untuk membantu proses pengembangan, pelatihan model, pengujian, dan visualisasi. *Software-software* ini dipilih karena fitur-fitur mereka yang mendukung berbagai tahap penelitian dan pengembangan proyek *machine learning*, khususnya dalam konteks *Natural Language Processing* (NLP) dan web *scraping*. Penggunaan *software-software* ini memungkinkan penelitian dilakukan dengan efisien, akurat, dan sesuai dengan standar industri. Berikut adalah penjelasan detail mengenai *software-software* tersebut:

2.1.14. *Visual Studio Code*

Visual Studio Code (VS *Code*) adalah editor kode sumber yang dikembangkan oleh Microsoft. Ini adalah salah satu editor kode yang paling populer di kalangan pengembang karena kemampuannya yang luas, fleksibilitas, dan dukungan untuk berbagai bahasa pemrograman. VS *Code* sangat cocok digunakan untuk pengembangan proyek *machine learning* karena beberapa alasan:

1. Kemudahan Penggunaan

Antarmuka pengguna yang intuitif dan kemudahan dalam pengaturan membuat VS *Code* menjadi pilihan yang nyaman bahkan bagi pengembang pemula.

Dengan fitur seperti *auto-completion*, *syntax highlighting*, dan *error checking*, proses penulisan kode menjadi lebih cepat dan minim kesalahan.

2. Ekstensi dan Integrasi

VS *Code* memiliki marketplace ekstensi yang sangat luas, memungkinkan pengguna untuk menambahkan fungsionalitas tambahan sesuai kebutuhan. Contohnya, ekstensi untuk *Python, Jupyter Notebooks, linting, debugging*, dan integrasi dengan GitHub sangat berguna dalam pengembangan dan pengelolaan proyek *machine learning*. Ekstensi-ekstensi ini tidak hanya menambah kapabilitas editor tetapi juga memungkinkan pengguna untuk menyesuaikan lingkungan pengembangan sesuai dengan kebutuhan spesifik mereka.

3. Fitur *Debugging*

VS *Code* menyediakan alat *debugging* yang kuat dan mudah digunakan. Pengembang dapat mengatur *breakpoint*, menginspeksi variabel, dan menjalankan kode langkah demi langkah untuk menemukan dan memperbaiki *bug*. *Debugging* menjadi lebih efisien dengan fitur-fitur seperti *inline debugging* dan *interactive debugging* yang memungkinkan pengembang untuk melihat dan mengubah nilai variabel secara langsung dalam editor.

4. Remote *Development*

Dengan kemampuan *remote development*, pengembang dapat bekerja pada kode yang berada di server dari jarak jauh, yang sangat berguna dalam skenario pelatihan model yang memerlukan sumber daya komputasi besar. Fitur ini memungkinkan pengembang untuk memanfaatkan infrastruktur komputasi yang lebih kuat.

5. Terminal Terintegrasi

VS *Code* memiliki terminal bawaan yang memungkinkan pengguna menjalankan perintah langsung dari dalam editor, mengurangi kebutuhan untuk beralih antara editor dan terminal terpisah. Ini membuat alur kerja menjadi lebih efisien dan menghemat waktu.

2.1.15. *Google Colab*

Google Colab, atau *Colaboratory*, adalah *platform* berbasis *cloud* yang memungkinkan pengguna untuk menulis dan menjalankan kode *Python* di browser dengan kemudahan akses ke GPU dan TPU tanpa biaya tambahan. *Google Colab* sangat bermanfaat dalam proyek ini karena beberapa alasan:

1. Akses ke GPU dan TPU

Salah satu keunggulan utama *Colab* adalah akses gratis ke GPU dan TPU, yang sangat mempercepat proses pelatihan model *machine learning* dibandingkan dengan CPU biasa. Penggunaan GPU dan TPU memungkinkan eksperimen dengan model yang lebih kompleks dan data yang lebih besar, yang akan memakan waktu sangat lama jika dilakukan dengan CPU.

2. Integrasi dengan *Google Drive*

Pengguna dapat menyimpan dan mengelola notebook langsung di *Google Drive*, memudahkan penyimpanan dan berbagi dokumen. Penyimpanan *cloud* ini memastikan bahwa pekerjaan tidak akan hilang dan dapat diakses dari mana saja.

3. *Jupyter Notebook Environment*

Colab menggunakan antarmuka *Jupyter Notebook*, yang familiar bagi banyak peneliti dan praktisi *machine learning*. *Notebook* ini mendukung kombinasi kode, teks, dan visualisasi dalam satu dokumen, sehingga memudahkan eksplorasi data dan pelaporan hasil. Dengan *Jupyter Notebook*, pengembang dapat menggabungkan dokumentasi dan eksekusi kode dalam satu tempat, memudahkan pemahaman dan replikasi hasil.

4. *Pre-installed Libraries*

Colab sudah dilengkapi dengan berbagai *library machine learning* seperti *TensorFlow, PyTorch, scikit-learn*, dan lainnya, menghemat waktu dan usaha dalam instalasi dan konfigurasi lingkungan kerja. Ini memungkinkan pengguna untuk fokus pada pengembangan dan eksperimen tanpa perlu khawatir tentang setup awal.

2.1.16. *Postman*

Postman adalah alat yang digunakan untuk pengembangan dan pengujian API. Dalam konteks proyek ini, *Postman* digunakan untuk menguji *endpoint* API yang mungkin dibuat untuk keperluan pengumpulan data atau pengujian model. Beberapa fitur utama dari *Postman* adalah:

1. Antarmuka Pengguna yang Intuitif

Postman menyediakan antarmuka grafis yang memudahkan pengguna untuk membuat, mengelola, dan menguji permintaan API tanpa perlu menulis kode tambahan. Ini memudahkan proses pengujian API, terutama bagi mereka yang tidak memiliki latar belakang teknis yang mendalam.

2. Pengelolaan Koleksi

Pengguna dapat menyimpan dan mengelola permintaan API dalam koleksi yang terorganisir, memungkinkan pengujian ulang dengan mudah dan berbagi koleksi dengan tim lain. Koleksi ini dapat berisi berbagai jenis permintaan (*GET, POST, PUT, DELETE*, dll.) yang terstruktur dengan baik, memudahkan pengujian batch dan dokumentasi.

3. Automasi Pengujian

Postman mendukung automasi pengujian melalui skrip *pre-request* dan test yang dapat dijalankan sebelum dan sesudah permintaan API, membantu dalam pengujian otomatis berbagai skenario. Fitur ini memungkinkan pembuatan skenario pengujian yang kompleks dan pengujian regresi yang konsisten.

4. Mock Servers

Postman memungkinkan pengguna untuk membuat *mock server* yang dapat digunakan untuk menguji aplikasi bahkan ketika API backend belum sepenuhnya dikembangkan. Ini sangat berguna dalam tahap awal pengembangan aplikasi, memungkinkan frontend dan backend berkembang secara paralel.

5. Monitoring

Fitur monitoring di Postman membantu memeriksa kesehatan dan kinerja API secara periodik, memastikan bahwa layanan API berfungsi dengan baik sepanjang waktu. Monitoring ini dapat diatur untuk memberikan laporan berkala dan notifikasi jika terjadi masalah.

2.2. Penelitian terdahulu

Pada sub bab ini, penulis akan membahas beberapa penelitian terdahulu yang relevan dengan topik *sentiment analysis*. Melalui tinjauan terhadap penelitian-penelitian ini, penulis dapat memahami berbagai pendekatan yang telah digunakan sebelumnya, hasil yang telah dicapai, serta keunggulan dan kelemahan dari metode-metode tersebut. Dengan demikian, penulis dapat mengidentifikasi persamaan dan perbedaan antara penelitian penelitian terdahulu dengan penelitian yang sedang dilakukan saat ini.

Tujuan dari peninjauan ini adalah untuk menempatkan penelitian yang penulis lakukan dalam konteks yang lebih luas, mengidentifikasi celah-celah yang ada dalam literatur, serta menunjukkan bagaimana penelitian ini berkontribusi terhadap pengembangan ilmu pengetahuan di bidang *sentiment analysis* dan *Natural Language Processing* (NLP). Dengan membandingkan metode dan hasil dari penelitian-penelitian sebelumnya, penulis dapat lebih memahami inovasi dan kontribusi penelitian penulis, serta memastikan bahwa pendekatan yang penulis gunakan adalah tepat dan efektif untuk mencapai tujuan penelitian. Berikut ini adalah beberapa jurnal terdahulu yang akan di bahas oleh penulis.

Tabel 2.1 Penelitian Terdahulu

No	Author	Judul Penelitian	Deskripsi	Relevansi
1	(Jain, Pamula, and Srivastava 2021)	*A Systematic Literature Review on Machine learning Applications for Consumer Sentiment Analysis Using Online Reviews*	*ScienceDirect*	Penelitian ini membahas penerapan *machine learning* untuk analisis sentimen konsumen dari ulasan online di sektor perhotelan dan pariwisata. Dengan pendekatan s*ystematic literature review* dan *web scraping*, studi ini mengumpulkan dan memproses data ulasan untuk melatih model ML. Hasilnya menunjukkan bahwa ML, terutama teknik seperti regresi, *decision tree*, dan *random forest*, sangat bermanfaat untuk memahami kepuasan konsumen dan membantu penyedia layanan meningkatkan kualitas serta strategi bisnis mereka.

2	(Wardianto, Farikhin, and Kusumo Nugraheni 2023)	Analisis Sentimen Berbasis Aspek Ulasan Pelanggan Restoran Menggunakan LSTM Dengan Adam Optimizer	JOINTECS *(Journal of Information Technology and Computer Science)*	Penelitian ini menganalisis sentimen berbasis aspek pada ulasan pelanggan restoran berbahasa Indonesia dari *TripAdvisor*, khususnya restoran Bandar Djakarta Ancol. Tujuannya adalah mengklasifikasikan opini berdasarkan aspek seperti makanan, tempat, pelayanan, dan harga. Data sebanyak 1700 ulasan dikumpulkan dengan *web scraping* dari Juni hingga Juli 2022. LSTM, didukung oleh *global max pooling* dan *Adam Optimizer*, digunakan untuk klasifikasi. Hasilnya menunjukkan akurasi 78,7% untuk analisis sentimen berbasis aspek dan 78% untuk kategori aspek, yang membantu pemahaman opini pelanggan.

3	(Maulida et al. 2024)	*Sentiment Analysis on TikTok Shop Reviews Using Long Short-Term Memory Method to Find Business Opportunity*	Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi	Penelitian berfokus untuk menganalisis sentimen dari ulasan pengguna *TikTok Shop* yang diambil dari *Google Play Store,* dengan menggunakan *Long Short-Term Memory* (LSTM). Tujuannya adalah mengevaluasi potensi bisnis *TikTok Shop* dengan memahami pandangan pengguna. Data ulasan dikumpulkan melalui *web scraping* menggunakan Python, lalu diproses dan diberi label sentimen (positif dan negatif). Algoritma LSTM kemudian menganalisis data untuk menangkap ketergantungan teks jangka panjang. Hasil analisis dievaluasi menggunakan *confusion matrix* untuk akurasi, presisi, *recall*, dan *F1-score*

4	(Ipmawati, Saifulloh, and Kusnawi 2024)	Analisis Sentimen Tempat Wisata Berdasarkan Ulasan pada Google Maps Menggunakan Algoritma Support Vector Machine	MALCOM: *Indonesian Journal of Machine Learning and Computer Science*	Penelitian ini menganalisis sentimen ulasan tempat wisata untuk mengidentifikasi sentimen positif atau negatif dari pengunjung. Tujuannya adalah mengembangkan model klasifikasi akurat dan mengeksplorasi faktor-faktor yang memengaruhi sentimen. Penelitian ini menggunakan metode SVM. Ulasan dikumpulkan dari *Google Maps*, lalu diproses melalui penghapusan *stopwords*, *stemming*, dan tokenisasi. Hasilnya menunjukkan bahwa model SVM dengan TF-IDF memiliki akurasi tinggi, dan sentimen pengunjung sangat dipengaruhi oleh kebersihan, daya tarik, fasilitas, dan pelayanan.

5	(Nardilasari et al. 2023)	Analisis Sentimen Calon Presiden 2024 Menggunakan Algoritma SVM Pada Media Sosial Twitter	JOINTECS *(Journal of Information Technology and Computer Science)*	Penelitian ini berfokus pada analisis sentimen calon presiden 2024 di Twitter, bertujuan meningkatkan performa analisis menggunakan algoritma *Support Vector Machine* dibandingkan *Naïve Bayes* yang memiliki tingkat akurasi yang rendah. Data sebanyak 8.959 *tweet* dikumpulkan dengan *web scraping* dengan kata kunci Anies, Ganjar, dan Prabowo pada 17-25 Oktober 2022. Tahap pra-pemrosesan dilakukan untuk membersihkan data dari elemen tidak relevan dan mengubah teks menjadi format yang sesuai. Data yang sudah diproses kemudian diklasifikasikan menggunakan SVM dengan aplikasi RapidMiner dan metode *10-fold cross-validation*

6	(Munandar, Farikhin, and Widodo 2023)	Sentimen Analisis Aplikasi Belajar Online Menggunakan Klasifikasi SVM	JOINTECS *(Journal of Information Technology and Computer Science)*	Penelitian ini menganalisis sentimen ulasan pengguna aplikasi belajar *online* (Ruang Guru, Zenius, Quipper) dari *Google Play Store* menggunakan algoritma *Support Vector Machine* (SVM). Data sebanyak 30.000 ulasan (masing-masing 10.000) dikumpulkan melalui web *scraping*. Ulasan diproses melalui normalisasi, *case folding*, *cleaning*, *tokenizing*, dan *stopwords removal*, lalu dibagi menjadi 90% data latih dan 10% data uji, dengan pelabelan positif (1), netral (0), atau negatif (-1). Hasil analisis menggunakan SVM menunjukkan Ruang Guru memiliki sentimen positif tertinggi, dengan akurasi 99%, Zenius 96%, dan Quipper 82%.

7	(Laksono et al. 2025)	*SENTIMENT ANALYSIS OF ONLINE DATING APPS USING SUPPORT VECTOR MACHINE AND NAÏVE BAYES ALGORITHMS*	Jurnal Teknik Informatika (JUTIF)	Penelitian berjudul "Sentiment *Analysis* Of Online Dating Apps Using Support Vector Machine And Naïve Bayes Algorithms" menganalisis sentimen pengguna aplikasi kencan online, khususnya Tinder, menggunakan algoritma Support Vector Machine (SVM) dan Naïve Bayes. Tujuannya adalah mengevaluasi performa dan efektivitas kedua algoritma ini dalam klasifikasi teks untuk sentimen analisis, serta mendapatkan pemahaman lebih dalam tentang pandangan pengguna Tinder. Penelitian ini menggunakan pendekatan metodologi mengadopsi kerangka kerja CRISP-DM, meliputi pengumpulan data, pra-pemrosesan data, optimasi

				SMOTE (untuk mengatasi ketidakseimbangan data), ekstraksi fitur, klasifikasi, dan evaluasi model. Dari pengujian terhadap 2000 data, SVM menunjukkan akurasi 85%, sedangkan Naïve Bayes 84%. Hasil ini menunjukkan bahwa SVM lebih unggul dan memiliki performa yang lebih stabil dalam mengenali sentimen positif dan negatif

2.3. Objek Penelitian

instagram banyak digunakan untuk berbagai aktivitas digital seperti membangun jejaring sosial. berbagi gambar, hingga melakukan personal branding yang menjadikannya sebagai salah satu aplikasi dengan pengguna terbanyak di berbagai kalangan pengguna perangkat mobile mulai dari anak anak higga orang dewasa, tidak sedikit juga para pengguna aplikasi instagram memberikan kritik dan saran dengan menulis ulasan mereka di *Google Play Store*. Pada penelitian ini penting untuk memberikan penjelasan lebih lanjut mengenai instagram dan *Google Play Store* yang menjadi objek dalam penelitian ini.

2.3.1. Instagram

Instagram mulai dikembangkan oleh Kevin Systrom dan Mike Krieger sebagai sebuah aplikasi berbasis *mobile* yang memungkinkan pengguna untuk mengambil gambar, menerapkan filter digital, dan membagikannya secara instan ke berbagai *platform* sosial lainnya, dengan peluncuran awal pada 6 Oktober 2010 di iOS dan kemudian pada tahun 2012 instagram hadir di Android karena meningkatnya jumlah pengguna. Instagram mulai di akuisisi oleh Facebook Inc. (sekarang *Meta Platforms Inc.*) pada tahun 2012 dengan nilai satu miliar dolar Amerika Serikat yang menjadi titik penting dalam perkembangan Instagram, karena sejak saat itu fitur-fitur utama seperti *Stories*, *IGTV*, *Reels*, serta kemampuan *live streaming* mulai ditambahkan untuk memperluas fungsi dan menarik lebih banyak interaksi dari pengguna global.

Dengan desain yang menekankan pada konten visual dan pengalaman pengguna yang sederhana namun menarik, Instagram digunakan secara luas untuk keperluan pribadi, komunitas, hingga bisnis seperti promosi produk, kolaborasi merek, dan pengembangan identitas digital, baik oleh individu maupun organisasi. Tingginya tingkat keterlibatan melalui fitur interaktif seperti *likes*, komentar, *direct messages*, hingga *explore page* menjadikan *platform* ini tidak hanya sebagai media sosial, tetapi juga sebagai ruang ekspresi dan komunikasi yang dinamis, sehingga menjadikannya objek yang relevan untuk dianalisis dari sisi pengalaman dan sentimen pengguna dalam konteks penelitian berbasis ulasan aplikasi.

2.3.2. *Google Play Store*

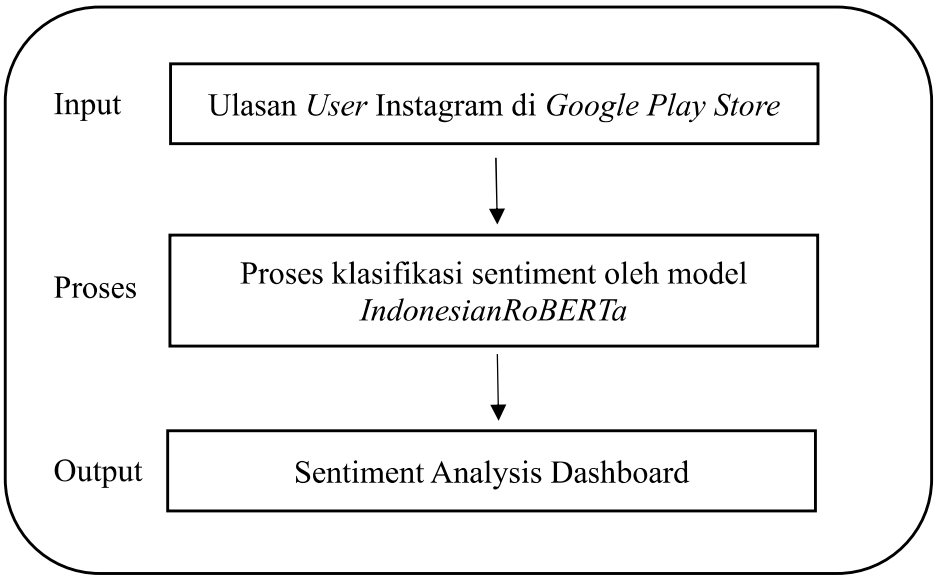
Google Play Store adalah *platform* digital yang dikembangkan oleh Google untuk sistem operasi Android, yang mulai diperkenalkan pada 6 Maret 2012 sebagai hasil penggabungan dari beberapa layanan sebelumnya, seperti *Android Market*, *Google Music*, dan *Google eBookstore*. *Platform* ini menyediakan berbagai konten digital seperti aplikasi, permainan, buku, film, dan musik, serta terintegrasi langsung dengan sistem Android sehingga memudahkan pengguna dalam mencari, mengunduh, dan memperbarui aplikasi secara aman dan efisien.

Selain menyediakan berbagai konten, *Google Play Store* juga memiliki fitur yang memungkinkan pengguna memberikan ulasan dan penilaian terhadap aplikasi yang telah digunakan, yang mencakup teks ulasan, peringkat bintang, serta waktu penggunaan. Informasi yang terkandung dalam ulasan tersebut tidak hanya berguna bagi pengembang untuk mengevaluasi kualitas aplikasinya, tetapi juga dapat dimanfaatkan dalam penelitian untuk menganalisis persepsi dan sentimen pengguna terhadap suatu aplikasi secara lebih terstruktur dan objektif.

2.4. Kerangka pemikiran

Kerangka pemikiran dari penelitian ini terbagi menjadi 3 tahap yaitu *Input*, proses, dan *output* untuk menganalisis sentimen ulasan pengguna Instagram di *Google Play Store*. *Input* berpusat pada permasalahan tentang volume data ulasan yang terus meningkat dan melakukan pengumpulan data ulasan Instagram di *Google Play Store* yang akan di analisis.

Proses melibatkan pengumpulan data dengan teknik *web scraping*, melakukan *finetuning* model *Indonesian-RoBERTa* untuk klasifikasi sentimen, pra-pemrosesan data, serta implementasi dan integrasi model ke dalam sebuah *dashboard* analisis sentimen dengan yang dibangun menggunakan Laravel, yang kemudian divalidasi dan diuji. *Output* dari penelitian ini akan menghasilkan sebuah *dashboard* analisis sentimen yang menyajikan informasi bermanfaat dari ulasan, serta kontribusi pada bidang sentimen analisis dan NLP melalui penerapan *Indonesian-RoBERTa* dalam bahasa Indonesia.



Gambar 2.7 Kerangka Pemikiran

(**Sumber**: Penelitian 2025)