BAB I

PENDAHULUAN

1.1 Latar Belakang

Esai adalah salah satu bentuk penilaian yang digunakan dalam proses pembelajaran untuk mengevaluasi pemahaman siswa terhadap materi yang telah diberikan. Dalam format ini, siswa diharapkan untuk menyampaikan ide dan argumen mereka secara tertulis, sehingga esai mampu mengukur kemampuan berpikir kritis dan analitis. Berbeda dengan soal pilihan ganda yang lebih efisien dalam hal pengolahan jawaban, esai menawarkan kesempatan untuk melihat kedalaman pemahaman siswa secara lebih komprehensif (Kinanti & Qoiriah, 2020). Esai juga dianggap penting dalam menilai keterampilan literasi dan kemampuan menyusun argumen yang baik. Namun, seiring meningkatnya jumlah siswa, penilaian esai secara manual memerlukan banyak tenaga dan waktu, terutama jika dilakukan dalam skala besar (*Model Evaluasi Pembelajaran Essay Grading Simple-O*, n.d.).

Proses penilaian manual terhadap esai sering kali dihadapkan pada kendala utama seperti memakan waktu lama dan bergantung pada subjektivitas penilai. Selain itu, beban kerja yang tinggi dalam penilaian esai dapat memengaruhi konsistensi hasil penilaian antar-penilai. Untuk mengatasi masalah ini, diperlukan solusi berbasis teknologi yang dapat melakukan penilaian esai secara otomatis dengan mempertimbangkan aspek kemiripan semantik antara jawaban siswa dan kunci jawaban (Yannakoudakis et al., 2011). Salah satu pendekatan yang menjanjikan adalah penggunaan model kemiripan semantik berbasis *word embeddings* dan algoritma berbasis deep learning, seperti BERT (*Bidirectional Encoder Representations from Transformers*), yang mampu menilai kedekatan makna antara jawaban siswa dan kunci jawaban.

Algoritma embedding statis seperti Word2Vec, GloVe, atau FastText memiliki keunggulan dalam memetakan kata ke dalam ruang vektor yang memperhitungkan konteks kata dalam kalimat, sehingga memungkinkan model untuk menangkap makna dari kata-kata tersebut (Mikolov et al., 2013). Pendekatan ini efektif untuk menangani teks dengan jumlah data yang lebih kecil dan cukup fleksibel dalam berbagai aplikasi NLP. Sementara itu, BERT, sebagai model transformator, memberikan pendekatan yang lebih canggih dengan memahami konteks bidirectional. Algoritma BERT mampu memahami konteks kalimat secara lebih mendalam, terutama dalam tugas-tugas yang membutuhkan pemahaman semantik secara menyeluruh (Devlin et al., 2019).

Beberapa penelitian terdahulu telah mengkaji berbagai metode dalam penilaian otomatis, khususnya dengan menggunakan embedding kata dan model berbasis deep learning. (Gaddipati et al., 2020) mengevaluasi model transfer learning seperti ELMo, BERT, GPT, dan GPT-2 dalam tugas penilaian jawaban pendek. Hasil penelitian mereka menunjukkan bahwa ELMo memiliki performa terbaik dengan nilai RMSE terendah (0,978) dan korelasi Pearson sebesar 0,485, melampaui BERT dan

GPT dalam menilai esai secara otomatis. Hal ini menunjukkan pentingnya pemahaman konteks lokal dalam jawaban esai. Namun, penelitian ini hanya menggunakan model pre-trained tanpa penambahan lapisan tambahan atau teknik lainnya, sehingga metrik error masih tinggi dan nilai korelasi Pearson masih rendah terhadap data Mohler.

Penelitian yang dilakukan oleh (Haidir & Purwarianti, 2020) berfokus pada embedding kontekstual dengan menggunakan BERT. Mereka membandingkan akurasi berbagai model BERT pre-trained dalam tugas penilaian esai dan menemukan bahwa BERT memberikan hasil yang baik pada data berbahasa Inggris. Namun, tantangan muncul ketika diterapkan pada dataset berbahasa Indonesia, di mana RMSE mencapai 1,893. Temuan ini mengindikasikan bahwa embedding kontekstual BERT efektif pada bahasa dengan sumber daya yang lebih banyak, tetapi menghadapi kendala dalam bahasa dengan data yang terbatas.

(Zhu et al., 2022) mengembangkan model berbasis BERT yang disempurnakan melalui jaringan Bi-LSTM dan Capsule Network, serta ditingkatkan dengan multi-head attention. Pendekatan ini menghasilkan peningkatan signifikan dengan nilai korelasi Pearson sebesar 0,897 dan RMSE 0,827. Hal ini menegaskan bahwa integrasi teknik deep learning yang lebih canggih dapat meningkatkan pemahaman semantik dalam teks.

Dalam konteks yang lebih spesifik, (Ripmiatin et al., n.d.) membandingkan berbagai pendekatan embedding, termasuk Word2Vec dan BERT, dalam penilaian jawaban pendek. Mereka menemukan bahwa Word2Vec yang dilatih pada domain spesifik memberikan hasil akurasi tertinggi, mengungguli BERT dalam metrik korelasi dan konsistensi penilaian. Hasil ini menunjukkan bahwa embedding statis tetap relevan, terutama dalam skenario di mana kontekstualisasi mendalam tidak terlalu diperlukan atau data yang tersedia terbatas.

(Sawatzki et al., 2022) menyoroti pentingnya kombinasi antara fitur embedding dan teknik *fine-tuning* model BERT. Mereka menggabungkan Ans2vec untuk pemetaan kata dengan BERT yang di fine-tune menggunakan regresi linier. Model ini mencapai koefisien korelasi Pearson sebesar 0,73 dan Mean Absolute Error (MAE) sebesar 0,4 pada dataset Short Answer Grading dari University of North Texas. Sementara itu, pada dataset berbahasa Jerman, model tersebut mencapai korelasi Pearson sebesar 0,78 dan MAE sebesar 1,2. Meskipun performa model dalam kedua bahasa cukup baik, masih terdapat ruang untuk peningkatan melalui fine-tuning atau penambahan teknik ekstraksi fitur pada embedding statis.

Berdasarkan kajian dari penelitian-penelitian tersebut, pendekatan berbasis embedding statis seperti Word2Vec dan embedding kontekstual seperti BERT memiliki kelebihan dan kelemahan masing-masing dalam tugas penilaian esai otomatis. Oleh karena itu, penelitian ini bertujuan untuk melakukan evaluasi komparatif terhadap kinerja kedua pendekatan tersebut pada data berbahasa Inggris dan Indonesia guna menentukan metode yang lebih efektif dalam meningkatkan akurasi dan konsistensi penilaian esai.

Penelitian ini secara khusus akan mengevaluasi algoritma embedding statis dan kontekstual seperti BERT dalam penilaian esai otomatis berbasis kemiripan

semantik. Hasil penelitian ini diharapkan dapat memberikan wawasan mengenai keunggulan dan kelemahan masing-masing pendekatan, meningkatkan konsistensi serta akurasi penilaian, dan berkontribusi dalam pengembangan sistem penilaian esai otomatis yang lebih efektif serta efisien dalam dunia pendidikan.

1.2 Landasan Teori

1.2.1 Penilaian Esai Otomatis

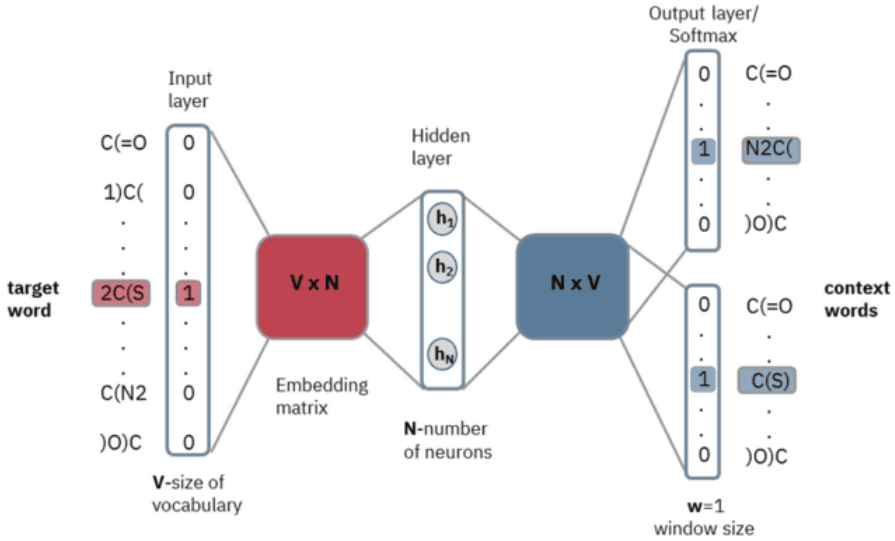
Penilaian esai otomatis adalah proses penggunaan teknologi untuk menilai jawaban siswa yang berbentuk esai tanpa intervensi manusia secara langsung. Dalam pendidikan, esai digunakan untuk mengevaluasi pemahaman siswa dengan lebih mendalam, menilai kemampuan berpikir kritis, analitis, serta keterampilan komunikasi tertulis. Penilaian esai manual menghadapi tantangan utama seperti membutuhkan waktu yang lama, kurang konsistensi antar-penilai, dan rentan terhadap bias subjektif.

Penilaian esai otomatis bertujuan untuk menggantikan atau mendukung proses penilaian manual yang sering kali memakan waktu lama dan rentan terhadap subjektivitas. Penilaian dirancang untuk mengevaluasi jawaban siswa berdasarkan aspek kemiripan semantik antara jawaban siswa dan kunci jawaban dengan menggunakan algoritma pemrosesan bahasa alami (NLP), seperti embedding statis (misalnya Word2Vec) dan embedding kontekstual (misalnya BERT - *Bidirectional Encoder Representations from Transformers*). Dengan pendekatan ini, penilaian esai otomatis mampu meningkatkan efisiensi, mengurangi beban kerja pengajar, dan memberikan hasil penilaian yang lebih objektif serta konsisten, terutama dalam situasi jumlah siswa yang besar.

1.2.2 *Word embedding* statis

Word embedding statis adalah metode untuk merepresentasikan kata-kata dalam bentuk vektor berdimensi tetap, di mana setiap kata memiliki representasi unik yang tidak berubah terlepas dari konteksnya dalam teks. Representasi ini dirancang untuk menangkap hubungan semantik dan sintaksis antara kata-kata. Teknik ini menjadi dasar dalam banyak aplikasi pemrosesan bahasa alami (NLP) sebelum munculnya metode kontekstual seperti Transformer. *Word embedding* statis mengasumsikan bahwa makna suatu kata dapat direpresentasikan secara konsisten tanpa memperhatikan lingkungan kalimat tempat kata tersebut berada.

Terdapat beberapa jenis algoritma yang paling banyak dipakai dalam *word embedding* statis, contohnya seperti Word2Vec dan FastText. Word2Vec adalah algoritma pembelajaran yang memetakan kata-kata ke dalam vektor berdimensi rendah, di mana hubungan semantik antara kata-kata direpresentasikan sebagai hubungan geometris dalam ruang vektor.



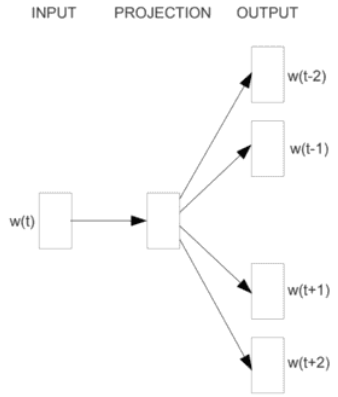
Gambar 1. Arsitektur *Continuous Bag-of-Words* (CBOW) untuk model *word embedding*
Sumber: (Oztrurk et al., 2024)

Model ini menggunakan arsitektur jaringan saraf sederhana (*shallow neural network*) yang terdiri dari tiga lapisan utama: *input layer, projection layer (hidden layer), dan output layer* (Firdaus, 2020). Proses pelatihan Word2Vec melibatkan jaringan saraf sederhana dengan tiga lapisan utama:
Pada tahap input, Word2Vec menerima representasi kata dalam bentuk *one-hot encoding*, yaitu vektor biner dengan panjang yang sama dengan jumlah kata unik (vocabulary) dalam data pelatihan. Satu elemen dalam vektor bernilai 1 (untuk kata yang sedang diproses), sementara sisanya bernilai 0.
Setelah melewati *input layer*, vektor kata diteruskan ke *hidden layer*, yang sering disebut *projection layer*. *Hidden layer* ini bertanggung jawab untuk memproyeksikan kata dari representasi *sparse* (seperti *one-hot encoding*) menjadi representasi *dense* (vektor kontinu). Proyeksi ini dilakukan menggunakan matriks bobot W berukuran $V \times N$, di mana V adalah ukuran *vocabulary*, dan N adalah dimensi embedding yang diinginkan (Girsang, 2020). Selanjutnya, *output layer* menghasilkan distribusi probabilitas atas kata-kata dalam vocabulary, tergantung pada arsitektur yang digunakan, yaitu kata target (CBOW) atau kata konteks (skip-gram).
Model ini menggunakan pendekatan probabilistik untuk memperkirakan hubungan antar kata. Untuk mengurangi kompleksitas komputasi, dua teknik sering digunakan:

- *Negative Sampling*: Memperbarui hanya pasangan kata positif dan beberapa pasangan kata negatif yang dipilih secara acak. (Goldberg & Levy, 2014)
- *Hierarchical Softmax*: Mempercepat perhitungan probabilitas kata target dengan menggunakan struktur pohon biner. (Mikolov et al., 2013)

Algoritma ini memiliki dua varian utama, yaitu *Continuous Bag of Words* (CBOW) dan Skip-Gram.

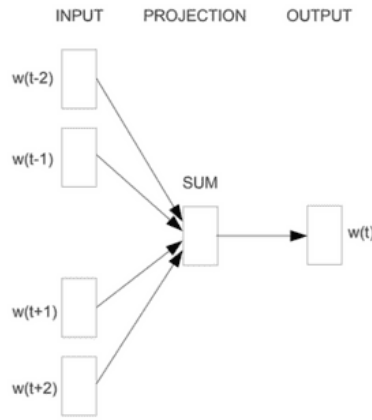
***Continuous Bag of Words* (CBOW)**. CBOW bertujuan untuk memprediksi kata target berdasarkan kata-kata dalam konteksnya.



Gambar 2. Arsitektur CBOW
Sumber: (Mikolov et al., 2013)

Model ini menerima input berupa kata-kata dalam jendela konteks yang ditentukan (misalnya, dua kata sebelum dan sesudah kata target). CBOW mengasumsikan bahwa makna kata dapat diturunkan dari kata-kata yang mengelilinginya. Misalnya, jika kata-kata dalam sebuah kalimat adalah "saya sedang makan nasi", model akan mempelajari untuk memprediksi kata "makan" dari konteks "saya", "sedang", dan "nasi".

Skip-Gram. Skip-Gram bekerja dengan cara yang berlawanan dengan CBOW, yaitu memprediksi kata-kata dalam konteks berdasarkan kata target.



Gambar 3. Arsitektur *Skip-gram*
Sumber: (Mikolov et al., 2013)

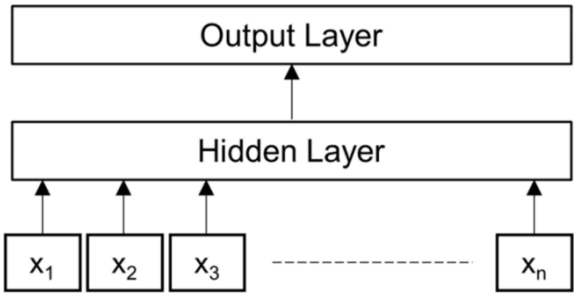
Model ini menerima input berupa kata target dan memprediksi kata-kata dalam konteks. Skip-Gram mencoba memaksimalkan probabilitas kata-kata dalam konteks yang muncul di sekitar kata target. Misalnya, jika kata

target adalah "makan", model akan mempelajari untuk memprediksi "saya", "sedang", dan "nasi".

Selain Word2Vec, FastText merupakan algoritma lain yang banyak digunakan dalam *word embedding* statis. Berbeda dengan Word2Vec, FastText dirancang untuk merepresentasikan kata-kata dalam bentuk vektor berdimensi rendah, dengan menekankan pada efisiensi dan fleksibilitas. Algoritma ini dikembangkan oleh tim *Facebook AI Research* sebagai pengembangan lebih lanjut dari Word2Vec (Bojanowski et al., 2017).

FastText memperluas kemampuan Word2Vec dengan memproses kata-kata menggunakan n-gram sebagai unit dasar, bukan kata utuh. Dengan pendekatan ini, setiap kata direpresentasikan sebagai kombinasi dari beberapa karakter n-gram. Misalnya, untuk kata “komputer,” trigramnya meliputi "kom," "omp," "mpu," "put," "ute," dan "ter." (Bojanowski et al., 2017).

Hal ini memungkinkan FastText untuk menangani kata-kata yang tidak terdapat dalam kamus (out-of-vocabulary words), yang sering menjadi kelemahan metode Word2Vec.

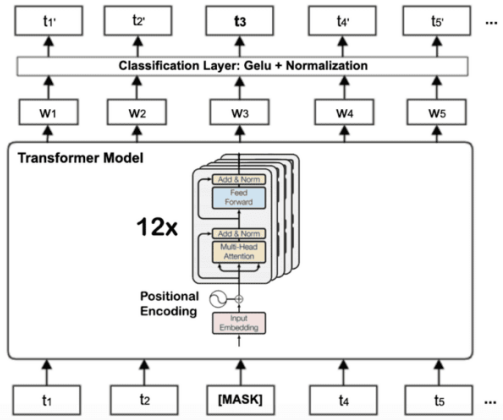


Gambar 4. Arsitektur FastText. Arsitektur ini menunjukkan hubungan antara input $(x_1, x_2, \dots, x_n)$ dengan hidden layer dan output layer.
Sumber: (“Text Classification Framework for Short Text Based on TFIDF-FastText | Request PDF,” 2024)

Model FastText menggunakan arsitektur jaringan saraf sederhana yang terdiri dari input layer yang menerima representasi kata dalam bentuk karakter n-gram, projection layer yang memetakan n-gram ke dalam ruang vektor berdimensi rendah, dan output layer yang digunakan untuk menghasilkan prediksi.

1.2.3 *Bidirectional Encoder Representations from Transformers* (BERT)

BERT adalah model berbasis Transformer yang dirancang untuk memahami konteks kata dalam dua arah (*bidirectional*). Dibangun dengan fokus pada representasi kontekstual, BERT tidak hanya memahami hubungan antar kata tetapi juga maknanya dalam kalimat secara keseluruhan, baik dari arah kiri maupun kanan. Model ini menggunakan *encoder stack* dari arsitektur Transformer (Vaswani et al., 2023), tanpa bagian *decoder*, untuk menangkap representasi konteks teks secara mendalam.



Gambar 5. Arsitektur model BERT. Arsitektur transformer dengan 12 lapisan, mekanisme *multi-head attention*, *positional encoding*, dan lapisan klasifikasi
Sumber: (Khalid et al., 2024)

Sebelum masuk ke model BERT, teks mentah harus diproses menjadi format yang dapat dimengerti oleh model. Pertama-tama, BERT akan menambahkan dua token khusus, yaitu [CLS] di awal teks, untuk tugas klasifikasi, dan [SEP] di akhir teks, sebagai penanda akhir kalimat. Selanjutnya, BERT menggunakan WordPiece Tokenization untuk menangani kata-kata langka atau kata baru. Sebuah kata dapat dipecah menjadi beberapa sub-token. Misalnya, "*unbelievable*" menjadi ["un", "##believe", "##able"]. Lalu, token diubah menjadi ID numerik sesuai dengan kosakata BERT.
Setiap token memiliki tiga jenis embedding yang dijumlahkan:

- *Token Embeddings*: Representasi token berbasis teknik WordPiece, atau disebut juga representasi numerik kata.
- *Segment Embeddings*: Informasi tentang segmen teks, terutama dalam tugas *Next Sentence Prediction* (NSP), digunakan untuk membedakan dua bagian teks.
- *Position Embeddings*: Informasi posisi token dalam urutan teks, memungkinkan model mempertimbangkan urutan kata.

Input embedding = Token embedding + Segment embedding + Position embedding

Komponen-komponen ini digabungkan menjadi vektor input embedding sebelum masuk ke lapisan *encoder.*Encoder terdiri dari beberapa lapisan mekanisme *multi-head self-attention* dan jaringan *feed-forward* yang sepenuhnya terhubung.

Pada tahap *multi-head self-attention*, setiap token berinteraksi dengan token lainnya dalam kalimat untuk memahami hubungan konteks menggunakan mekanisme perhatian. Hubungan antar token dihitung menggunakan formula:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Query (Q), *Key* (K), dan *Value* (V) adalah transformasi linear dari embedding token, sementara d_k adalah dimensi vektor. Q, K, V dihitung untuk setiap token.
Contohnya:
"*The bat flew over the stadium*."
Token:

Tabel 1. Gambaran tokenisasi kalimat pada model BERT

Token Index	0	1	2	3	4	5	6	7
Token	[CLS]	*The*	*bat*	*flew*	*over*	*the*	*stadium*	[SEP]

Pada contoh kalimat ini, makna kata “bat” ada dua, yaitu:
1. Kelelawar (hewan) jika dikaitkan dengan "*flew*" (terbang).
2. Tongkat pemukul (dalam olahraga) jika dikaitkan dengan konteks olahraga seperti "*baseball stadium*."

Tabel 2 menunjukkan bobot perhatian (*attention scores*) untuk token "*bat*":

Tabel 2. Gambaran pemberian nilai *attention scores* pada setiap token

Token	[CLS]	*The*	*bat*	*flew*	*over*	*the*	*stadium*	[SEP]
bat	0.01	0.05	0.50	**0.25**	0.06	0.04	**0.08**	0.00

Interpretasi Matriks

- Self-attention ke token "*bat*" (0.50):
  Token "*bat*" memberikan perhatian terbesar ke dirinya sendiri, sebagaimana normal dalam proses *self-attention*.
- Hubungan ke Token "*flew*" (0.25):
  Bobot perhatian yang signifikan antara "*bat*" dan "*flew*" karena kata *flew* memiliki keterkaitan langsung dengan makna "kelelawar" (binatang yang bisa terbang). BERT "menyadari" bahwa makna "*bat*" dalam konteks ini adalah hewan, bukan alat olahraga.
- Hubungan ke Token "*stadium*" (0.08):
  Ada sedikit perhatian ke "stadium", namun lebih kecil dibanding "*flew*" Ini menunjukkan bahwa "*stadium*" memberikan konteks sekunder, tapi tidak langsung mengarahkan makna "*bat*" ke tongkat pemukul.
- Token Lain Seperti "*over*" dan "*the*":
  Token seperti "*over*" (0.06) atau "*the*" (0.05) memiliki nilai perhatian kecil karena kontribusi semantiknya terhadap pemahaman makna "*bat*" tidak signifikan.

Setelah tahap *self-attention*, representasi token diproses lebih lanjut oleh jaringan saraf *feed-forward* dua lapis untuk memperkaya informasi kontekstual.
Setiap encoder layer memiliki koneksi residual di sekitar sub-lapisan, dengan layer *normalization* memastikan stabilitas selama pelatihan:

$$Output = LayerNorm(x + Sublayer(x)) \quad (2)$$

di mana Sublayer(x) adalah fungsi yang diimplementasikan oleh sub-lapisan tersebut. Pelatihan awal BERT dirancang dengan dua tugas utama.

a. *Masked Language Modeling* (MLM):
   Dalam pendekatan ini, beberapa token dalam input disembunyikan (*masked*) secara acak, dan model dilatih untuk memprediksi token yang hilang berdasarkan konteks. Pendekatan ini memungkinkan BERT memahami hubungan bidirectional antara kata-kata, baik sebelum maupun sesudah token yang disembunyikan.
   Contoh: Kalimat "*The* [MASK] *is on the table*" memberikan konteks kepada model untuk memprediksi "*book*".
b. *Next Sentence Prediction* (NSP):
   Selain MLM, pelatihan BERT juga mencakup tugas next sentence prediction untuk memahami hubungan antar kalimat. Dalam tugas ini, model diberikan dua segmen teks dan dilatih untuk memprediksi apakah segmen kedua mengikuti segmen pertama secara logis.
   Misalnya, model menerima dua kalimat berikut,
   - Kalimat 1: "*The bank is near the river.*"
   - Kalimat 2: "*People were fishing there.*"

   Kalimat yang benar-benar berhubungan akan diberi label 1 dan kalimat yang tidak berhubungan diberi label 0.
   Model dilatih untuk memprediksi apakah kedua kalimat saling berhubungan. Hal ini membantu BERT dalam menangkap hubungan semantik antar kalimat.

Output dari BERT adalah representasi vektor untuk setiap token. Terdapat token khusus [CLS] pada awal setiap input, yang digunakan untuk tugas klasifikasi teks. Token lainnya memberikan representasi kata kontekstual.
Setelah melalui lapisan encoder, output BERT dapat digunakan untuk berbagai tugas NLP dengan cara:

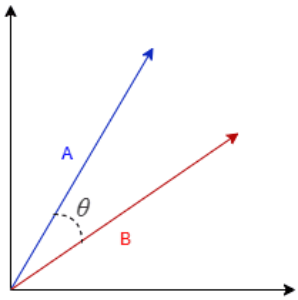
- Menggunakan representasi token spesifik (*token-level output*) untuk tugas seperti *Named Entity Recognition* (NER).
- Menggunakan representasi token khusus [CLS] untuk keseluruhan teks (*sentence-level output*), misalnya untuk tugas klasifikasi teks.
- Memanfaatkan representasi token untuk menemukan jawaban pada teks, pada *Question-Answering* (QA)

BERT telah merevolusi NLP dengan menghadirkan model pra-latihan yang fleksibel, di mana representasi bahasa yang diperoleh dapat disesuaikan lebih lanjut melalui *fine-tuning* untuk berbagai tugas.

1.2.4 *Cosine Similarity*

Cosine Similarity adalah metode untuk mengukur tingkat kesamaan antara dua vektor dalam ruang multidimensi berdasarkan sudut kosinus di antara keduanya. Pendekatan ini banyak digunakan dalam berbagai bidang, termasuk pemrosesan bahasa alami (NLP), *information retrieval*, dan sistem rekomendasi. Metode ini

mengabaikan panjang atau magnitudo dari vektor dan hanya mempertimbangkan orientasi atau arah vektor, yang membuatnya ideal untuk membandingkan teks atau dokumen.



Gambar 6. *Cosine Similarity*. Representasi vektor A dan B serta sudut θ dalam ruang vektor

Dalam konteks NLP, setiap teks atau dokumen dikonversi ke dalam vektor numerik, seperti dalam *word embeddings* atau *Term Frequency-Inverse Document Frequency* (TF-IDF), dan kesamaan antara dua vektor dihitung dengan rumus:

$$Cosine\ Similarity = \frac{A \cdot B}{\|A\|\|B\|} \tag{3}$$

di mana,
$A \cdot B$ adalah hasil perkalian *dot product* antara dua vektor A dan B
$\|A\|$ dan $\|B\|$ adalah panjang (magnitudo) dari masing-masing vektor.
Nilai cosine *Similarity* berkisar antara 0 hingga 1 jika sudutnya akut (positif) dan antara -1 hingga 0 jika vektor memiliki sudut yang besar (berlawanan arah). Nilai 1 menunjukkan kesamaan sempurna, sedangkan nilai 0 menunjukkan tidak ada kesamaan.

1.2.5 Ekstraksi Fitur

Dalam penelitian penilaian esai otomatis, ekstraksi fitur mengekstrak informasi yang relevan dari teks esai, seperti kesamaan semantik antara jawaban siswa dan jawaban kunci.

> ***Average Cosine Similarity***. *Average Cosine Similarity* adalah salah satu metode ekstraksi fitur di mana *embedding* dari setiap kata dalam kalimat diubah menjadi vektor rata-rata. Vektor ini kemudian dibandingkan menggunakan *cosine Similarity*, yang mengukur kesamaan arah antara dua vektor. Hasilnya memberikan skor kemiripan berdasarkan representasi semantik dari keseluruhan kalimat (Pennington et al., 2014).
> Jika v_i adalah embedding kata i dalam kalimat A, maka *embedding* rata-rata v_A didefinisikan sebagai:

$$v_A = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{4}$$

di mana n adalah jumlah kata dalam kalimat.

***Smooth Inverse Frequency* (SIF).** *Smooth Inverse Frequency* (SIF) adalah salah satu metode ekstraksi fitur yang menggunakan rata-rata embedding kata dengan bobot yang bergantung pada frekuensi kata, sehingga memberikan penekanan lebih pada kata-kata jarang yang sering kali memiliki informasi penting (Arora et al., 2017). Untuk meningkatkan kualitas embedding, komponen utama yang dominan dihilangkan menggunakan *Singular Value Decomposition* (SVD).
Bobot kata (w_i) adalah langkah awal dalam metode SIF untuk menekankan kontribusi kata tertentu berdasarkan frekuensinya dalam corpus. Kata yang sering muncul (kata umum seperti "dan", "adalah") cenderung memiliki bobot kecil, sedangkan kata yang jarang (biasanya lebih informatif) memiliki bobot yang lebih besar (Mikolov et al., 2013). Rumus bobot kata adalah:

$$w_i = \frac{a}{a + f_i} \tag{5}$$

di mana $a = 0.001$ adalah parameter *smoothing* (biasanya kecil, seperti 0.001) dan f_i adalah frekuensi kata i dalam *corpus*. Bobot ini digunakan untuk mengurangi dampak kata-kata umum yang tidak memiliki informasi semantik yang signifikan (Levy et al., 2015).

Setelah bobot kata dihitung, *embedding* setiap kata (v_i) dalam kalimat digabungkan menjadi satu representasi rata-rata berbobot (v_A) untuk keseluruhan kalimat:

$$v_A = \frac{\sum_{i=1}^{n} w_i \cdot v_i}{\sum_{i=1}^{n} w_i} \tag{6}$$

di mana w_i adalah bobot kata i, v_i adalah embedding kata i, dan n adalah jumlah kata dalam kalimat (Arora et al., 2017). Langkah ini menghasilkan satu representasi semantik untuk keseluruhan kalimat berdasarkan kombinasi *embedding* kata yang dihitung secara proporsional dengan bobotnya.

Representasi rata-rata berbobot (v_A) cenderung terpengaruh oleh komponen dominan di *embedding space*, yang sering kali merepresentasikan bias global (misalnya, panjang kalimat atau struktur bahasa umum). Untuk mengatasi ini, komponen utama pertama u

dihilangkan dari representasi rata-rata berbobot menggunakan *Singular Value Decomposition* (SVD) (Cer et al., 2018).
Langkah ini dilakukan dengan rumus:

$$v'_A = v_A - (v_A \cdot u)u \tag{7}$$

di mana:
u adalah komponen utama pertama (*principal component*) dan v'_A adalah representasi akhir setelah komponen utama dihilangkan. Langkah ini bertujuan untuk menghilangkan komponen utama memastikan bahwa representasi kalimat fokus pada informasi semantik yang lebih relevan, bukan pada pola umum atau bias dalam *embedding*.

Setelah representasi kalimat $A(v'_A)$ dan $B(v'_B)$ dihitung, *cosine Similarity* digunakan untuk mengukur kesamaan antara kedua kalimat (Arora et al., 2017).

***Word Mover's Distance* (WMD).** *Word Mover's Distance* adalah metode ekstraksi fitur dengan menghitung jarak minimum yang harus ditempuh untuk "memindahkan" distribusi *embedding* kata dalam satu kalimat ke kalimat lain. Metode ini mempertimbangkan urutan kata dan memberikan wawasan tentang sejauh mana dua kalimat berbeda secara semantik (Kusner et al., 2015). Jika *embedding* kata i dalam kalimat A adalah v_{A_i}, dan *embedding* kata j dalam kalimat B adalah v_{B_j}, maka jarak dihitung dengan menyelesaikan masalah dari persamaan 8:

$$minimize \sum_{i=1}^{n}\sum_{j=1}^{m} T_{ij} \cdot d(v_{A_i}, v_{B_j}) \tag{8}$$

dengan syarat:

$$\sum_{j=1}^{m} T_{ij} = p_{A_i}, \sum_{i=1}^{n} T_{ij} = p_{B_j}, T_{ij} \geq 0 \tag{9}$$

di mana T_{ij} adalah jumlah kata yang dipindahkan dari i ke j, $d(v_{A_i}, v_{B_j})$ adalah jarak *Euclidean* antara dua *embedding*, dan p_{A_i}, p_{B_j} adalah probabilitas kemunculan kata dalam distribusi kalimat A dan B.

1.2.6 Model Regresi

Model regresi digunakan untuk memodelkan hubungan antara fitur-fitur yang dihasilkan dan skor yang diinginkan. Misalnya, setelah mendapatkan fitur-fitur dari esai (AvgCos, WMD, SIF), model regresi digunakan untuk memprediksi skor esai berdasarkan fitur tersebut.

***Linear regression*.** *Linear Regression* atau regresi linear adalah salah satu metode dalam *machine learning* yang digunakan untuk memprediksi nilai numerik berdasarkan hubungan linear antara variabel independen (*predictor*) dan variabel dependen (*target*). Metode ini merupakan pendekatan statistika yang digunakan secara luas dalam berbagai bidang, seperti prediksi tren ekonomi, analisis ilmiah, dan pemodelan data kuantitatif (Montgomery et al., 2012). Secara matematis, regresi linear dinyatakan sebagai:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{10}$$

di mana Y adalah variabel dependen, X adalah variabel independen, β_0 adalah intercept (konstanta), β_1 adalah koefisien kemiringan garis, dan ϵ adalah *error term*. Koefisien β_0 dan β_1 diperoleh dengan meminimalkan *sum of squared errors* (SSE) menggunakan metode *Ordinary Least Squares* (OLS) (Kutner et al., 2005). Keunggulan dari regresi linear adalah kemudahan dalam interpretasi serta komputasinya yang cepat.

***Support Vector Regression* (SVR).** *Support Vector Regression* (SVR) adalah metode regresi yang didasarkan pada algoritma *Support Vector Machine* (SVM), yang bertujuan untuk memprediksi nilai numerik dengan menemukan hiperplane optimal dalam ruang fitur. Berbeda dengan metode regresi tradisional, SVR bekerja dengan menentukan margin toleransi (epsilon) di sekitar fungsi prediksi, sehingga model hanya berfokus pada error yang berada di luar margin tersebut.

SVR menggunakan konsep kernel untuk menangani data yang tidak linear dengan memetakan data ke dimensi yang lebih tinggi, di mana hubungan linear dapat ditemukan (Smola & Schölkopf, 2004). Kernel yang umum digunakan antara lain linear, polinomial, dan RBF (*Radial Basis Function*). Selama pelatihan, SVR meminimalkan fungsi biaya yang menggabungkan error prediksi dan kompleksitas model. Formula dasarnya dapat dinyatakan dengan persamaan 11:

$$min\frac{1}{2}\ \|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}(\xi i + \xi_i^*) \tag{11}$$

Dengan syarat:

$$y_i - (w \cdot x_i + b) \leq \in + \xi_{\boldsymbol{i}}\ dan$$
$$(w \cdot x_i + b) - y_i \leq \in + \xi_{\boldsymbol{i}}^* \tag{12}$$

Di mana:

- $\boldsymbol{w}$ adalah vektor bobot,
- b adalah bias,
- $\xi i, \xi_i^*$ adalah variabel *slack* untuk toleransi error,
- C adalah parameter regulasi yang mengontrol *trade-off* antara margin yang lebih lebar dan error pada data.

Hasil prediksi SVR diambil dari fungsi *hiperplane* yang ditemukan oleh model, yaitu:

$$\hat{\boldsymbol{y}} = \boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b} \quad (13)$$

SVR dikenal karena kemampuannya yang kuat dalam menangani data yang memiliki dimensi tinggi, serta fleksibilitasnya dalam memodelkan hubungan linier maupun nonlinier. Dengan parameter regulasi yang tepat, SVR dapat menghasilkan model yang tahan terhadap *overfitting* dan memberikan prediksi yang akurat, bahkan untuk dataset dengan ukuran kecil.

***Random Forest Regressor* (RFR).** *Random Forest Regressor* adalah metode regresi berbasis *ensemble learning* yang menggunakan kombinasi beberapa pohon keputusan (*decision trees*) untuk memprediksi nilai numerik. Metode ini dikenal karena keandalannya dalam mengatasi masalah *overfitting* yang sering muncul pada model pohon keputusan tunggal (Breiman, 2001). *Random Forest* bekerja dengan membangun banyak pohon keputusan (*forest*) selama proses pelatihan. Setiap pohon dalam hutan dilatih pada subset data yang berbeda melalui metode *bootstrap sampling* dan menggunakan subset fitur yang dipilih secara acak (Ho, 1998). Hasil prediksi dari setiap pohon kemudian dirata-rata untuk memberikan prediksi akhir dalam kasus regresi. Formula prediksi dapat dinyatakan dengan persamaan 14:

$$\hat{y} = \frac{1}{n}\sum_{i=1}^{n} T_i(x) \quad (14)$$

di mana (x) adalah prediksi dari pohon ke-i untuk input x, dan n adalah jumlah total pohon dalam hutan (Breiman, 2001). Proses rata-rata hasil prediksi membuat model lebih tahan terhadap data yang memiliki *noise* atau *outlier*.

1.2.7 *Pre-processing*

Pre-processing merupakan tahap awal dalam sistem penilaian esai otomatis yang bertujuan untuk mempersiapkan data teks agar lebih optimal saat diproses oleh model *embedding* statis atau kontekstual. Tahapan *pre-processing* ini membantu mengurangi *noise*, menyederhanakan teks, dan meningkatkan kualitas representasi data.

Case folding. *Case folding* adalah suatu tahapan proses mengubah semua karakter dalam dokumen menjadi huruf kecil. Tujuannya adalah untuk menyamakan format penulisan kata sehingga tidak sensitif terhadap huruf kapital. Misalnya, kata "BERT" dan "bert" akan dianggap sama setelah *case folding*. Langkah ini membantu meningkatkan konsistensi

data dan mengurangi kompleksitas model karena kata-kata dengan variasi huruf besar/kecil akan dihitung sebagai satu entitas yang sama.

Filtering. *Filtering* adalah proses menyaring atau menghapus komponen-komponen yang tidak relevan dalam teks, seperti angka, tanda baca, simbol khusus, atau karakter non-alfabet yang tidak memiliki kontribusi signifikan terhadap pemahaman semantik teks. Tahap ini dilakukan untuk membersihkan data dan memastikan bahwa hanya kata-kata yang memiliki makna penting yang diproses lebih lanjut oleh model *embedding*.

Stopword removal. *Stopword removal* adalah proses menghapus kata-kata umum yang tidak memberikan kontribusi signifikan terhadap makna teks, seperti "dan", "ke", "yang", dan sebagainya. Proses ini membantu mengurangi dimensi data dan fokus pada kata-kata yang memiliki makna lebih spesifik. Tahapan ini akan menghilangkan *stopword* berdasarkan yang tertera di dalam daftar *stopword*. Contoh:
Kalimat asli: "Saya suka belajar NLP di universitas."
Setelah *stopword removal*: "suka belajar NLP universitas."

Normalisasi skor. Normalisasi skor adalah proses transformasi nilai numerik dari rentang tertentu ke dalam skala yang konsisten. Pada penelitian ini, skor awal dari jawaban siswa berada dalam rentang nilai 0 hingga 5, yang mewakili tingkat kesesuaian jawaban dengan kunci jawaban. Skor ini dinormalisasi ke rentang [0, 1] untuk mempermudah proses pelatihan model regresi dan meningkatkan stabilitas komputasi. Normalisasi skor dirumuskan dengan,

$$normalized\ score = \frac{original\ score}{5.0} \tag{15}$$

di mana $original\ score$ adalah nilai skor awal pada skala 0-5, dan $normalized\ score$ adalah nilai skor setelah dinormalisasi ke skala 0-1.

Tokenisasi. Tokenisasi adalah proses memecah teks menjadi unit-unit kecil yang disebut token, seperti kata, frasa, atau karakter. Proses ini penting agar teks dapat diproses secara lebih efisien oleh model *embedding*.
Pada model *embedding* statis seperti Word2Vec, tokenisasi dilakukan pada tingkat kata, di mana teks dipecah menjadi daftar kata-kata. Contohnya kalimat "Saya suka belajar NLP" akan diubah menjadi ["saya", "suka", "belajar", "nlp"]. Sedangkan pada model embedding kontekstual seperti BERT, Tokenisasi menggunakan teknik *WordPiece*, yang memecah kata menjadi sub-kata untuk menangani kata-kata *out-of-vocabulary* (OOV). Contohnya kata "*playing*" akan diubah menjadi token ["*play*", "*##ing*"] (Devlin et al., 2019).

1.2.8 Evaluasi model

Evaluasi kualitas model penilaian esai otomatis sangat bergantung pada metrik yang digunakan untuk mengukur performa prediksi. Tiga metrik evaluasi yang umum digunakan adalah *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), dan Korelasi Pearson.

Mean Absolute Error **(MAE).** MAE mengukur rata-rata perbedaan absolut antara nilai prediksi dan nilai sebenarnya. Metrik ini memberikan gambaran seberapa dekat prediksi model dengan nilai referensi tanpa memperhitungkan arah kesalahan. Rumusnya dinyatakan dengan persamaan 16.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{16}$$

di mana y_i adalah nilai sebenarnya, $\hat{y}_i$ adalah nilai prediksi, dan n adalah jumlah data. MAE lebih mudah diinterpretasikan karena satuannya sama dengan data aslinya. Namun, MAE kurang sensitif terhadap *outlier* dibandingkan dengan MSE (Botchkarev, 2019). Rentang nilai MAE berkisar dari 0 sampai tak terhingga, dengan nilai MAE yang lebih rendah menunjukkan prediksi yang lebih akurat. MAE = 0 berarti tidak ada kesalahan prediksi sama sekali. Dalam penelitian ini, nilai MAE ≤ 0,2 dapat dikategorikan sebagai sangat baik, sedangkan jika berada dalam rentang 0,2 < MAE ≤ 0,4, maka model masih dapat dikatakan baik. Namun, jika MAE lebih besar dari 0,4, maka model memerlukan perbaikan.

Mean Squared Error **(MSE).** MSE adalah rata-rata kuadrat dari selisih antara nilai prediksi dan nilai sebenarnya. Rumus MSE dinyatakan oleh persamaan 17.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{17}$$

Dengan mengkuadratkan selisih, MSE memberikan bobot lebih besar pada kesalahan yang lebih besar, sehingga lebih sensitif terhadap outlier. Meskipun demikian, nilai MSE sulit diinterpretasikan secara langsung karena satuannya berbeda dari data asli (Chai & Draxler, 2014). Penggunaan MSE sering dikombinasikan dengan metrik lain untuk evaluasi yang lebih komprehensif. Dalam penelitian ini, MSE dengan nilai ≤ 0,05 menunjukkan performa yang sangat baik, sedangkan jika berada pada rentang 0,05 < MSE ≤ 0,1, maka model masih dianggap cukup baik. Namun, jika MSE lebih besar dari 0,1, maka model masih perlu diperbaiki.

***Root Mean Squared Error* (RMSE).** RMSE adalah akar kuadrat dari MSE dan memiliki satuan yang sama dengan skor asli.

$$RMSE = \sqrt{MSE} \tag{18}$$

Metrik ini lebih mudah diinterpretasikan karena memberikan ukuran kesalahan dalam skala yang sama dengan nilai target. RMSE sering digunakan bersamaan dengan MSE untuk memberikan gambaran intuitif mengenai besar rata-rata kesalahan model. Dalam penelitian ini, model dengan RMSE ≤ 0,3 dikategorikan sebagai sangat baik, sedangkan jika berada dalam rentang 0,3 < RMSE ≤ 0,5, model masih dapat dikatakan baik. Namun, jika RMSE lebih besar dari 0,5, maka model masih memiliki tingkat kesalahan yang cukup tinggi dan perlu dilakukan perbaikan.

Korelasi Pearson. Korelasi Pearson mengukur kekuatan dan arah hubungan linier antara nilai prediksi dan nilai sebenarnya. Nilai koefisien korelasi (r) berkisar antara -1 hingga 1, di mana di mana $r = 1$ menunjukkan korelasi positif sempurna, $r = -1$ menunjukkan korelasi negatif sempurna, dan $r = 0$ menunjukkan tidak ada hubungan linier.

$$r = \frac{\sum(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum(y_i - \bar{y})^2 \sum(\hat{y}_i - \bar{\hat{y}})^2}} \tag{19}$$

Korelasi Pearson sering digunakan dalam penilaian esai otomatis untuk mengukur sejauh mana prediksi model sesuai dengan penilaian manusia. Metrik ini penting karena mencerminkan kualitas hubungan antar variabel, yang berguna dalam sistem evaluasi berbasis prediksi. Dalam penelitian ini, model dengan korelasi Pearson (r) $\geqslant$ 0,8 dikategorikan sebagai sangat baik, karena menunjukkan hubungan yang sangat kuat antara prediksi model dan penilaian manusia. Jika nilai korelasi berada dalam rentang 0,6 $\leqslant$ r < 0,8, model masih dapat dikatakan baik, meskipun terdapat sedikit perbedaan dalam pola penilaian. Sementara itu, jika korelasi berada dalam rentang 0,4 $\leqslant$ r < 0,6, model masih cukup dapat diterima, tetapi variasi terhadap penilaian manusia cukup besar. Jika nilai korelasi kurang dari 0,4, maka model dikategorikan kurang baik dan masih perlu dilakukan perbaikan agar dapat menghasilkan prediksi yang lebih akurat.

1.3 Tujuan dan Manfaat

1.3.1 Tujuan

Adapun tujuan penelitian ini adalah sebagai berikut:

1. Menerapkan algoritma embedding statis dan BERT dalam penilaian jawaban esai secara otomatis.
2. Menganalisis dan membandingkan kinerja dan akurasi antara pendekatan statis seperti embedding statis dan pendekatan kontekstual dalam mengukur kemiripan semantik pada jawaban esai

1.3.2 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Meningkatkan efisiensi dan akurasi dalam penilaian esai secara otomatis, mengurangi waktu dan usaha yang dibutuhkan dibandingkan dengan penilaian manual oleh tenaga pengajar, terutama pada skala besar.
2. Memberikan analisis komparatif antara algoritma embedding statis dan kontekstual, yang dapat membantu memahami kelebihan dan kekurangan masing-masing pendekatan dalam mengukur kemiripan semantik, sehingga dapat digunakan dalam pengembangan sistem penilaian esai yang lebih canggih dan sesuai dengan kebutuhan pendidikan.

BAB II

METODE PENELITIAN

2.1 Tempat dan Waktu

Penelitian ini dilakukan sejak Penelitian ini dilakukan sejak disetujuinya proposal penelitian ini, yaitu pada tanggal 21 Oktober 2024 sampai pada Januari 2025. Penelitian ini dilakukan di Laboratorium *Artificial Intelligence* (AI), Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.

2.2 Benda Uji dan Alat

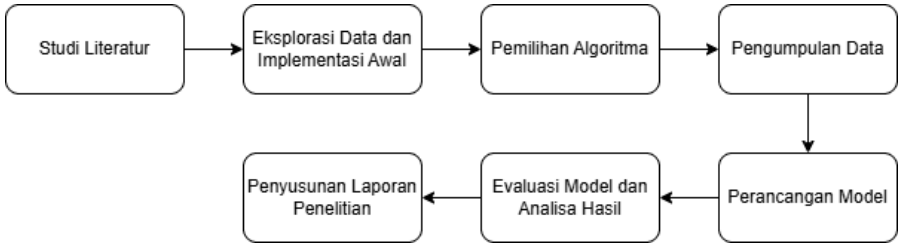
Benda dan alat yang digunakan dalam penelitian ini meliputi perangkat lunak dan perangkat keras berikut:

1. Perangkat lunak
   a. Windows 11 x64
   b. Visual Studio Code
   c. Google Colab
   d. Microsoft Word
   e. Zotero
2. Perangkat keras
   a. Laptop ASUS TUF FX506LI-I75TB6T-O dengan processor Intel(R) Core(TM) i7-10870H CPU @ 2.20GHz hingga 5.00 GHz, RAM 8 GB, SSD 512 GB
3. Bahasa pemrograman
   a. Python 3.10.14
4. Library
   a. os vposix oleh tim Python Software Foundation berguna untuk menyediakan antarmuka untuk berinteraksi dengan sistem operasi, seperti membaca atau menulis file, dan mengelola proses.
   b. Gensim v4.3.3 oleh tim Gensim Development berguna untuk pemodelan topik, pemrosesan teks, dan analisis semantik, seperti pembentukan embedding kata dan pengaturan parameter pada model.
   c. NLTK v3.2.4 oleh tim *Natural Language Toolkit* berguna untuk pemrosesan bahasa alami (NLP), termasuk tokenisasi, dan penghilangan *stopwords* berbahasa Inggris.
   d. Sastrawi v1.0.1 oleh Andreas Tjandra digunakan untuk penghilangan *stopwords* pada pemrosesan bahasa alami (NLP) dalam bahasa Indonesia.
   e. NumPy v1.26.4 oleh Travis E. Oliphant et al. berguna untuk menyediakan dukungan array dan matriks besar, serta fungsi matematis untuk operasi pada array tersebut.

f. Pandas v2.2.2 oleh tim The Pandas Development berguna untuk manipulasi dan analisis data, termasuk pengelolaan data dalam bentuk tabel atau DataFrame.
g. Scikit-learn v1.2.2 oleh tim Scikit-learn berguna untuk pembelajaran mesin, termasuk klasifikasi, regresi, klasterisasi, dan pemrosesan data.
h. PyTorch v2.5.1+cu121 oleh tim PyTorch Development berguna untuk komputasi numerik berbasis tensor dengan akselerasi GPU serta pembangunan dan pelatihan model pembelajaran mendalam (*deep learning*).
i. CSV v1.0 oleh tim Python Software Foundation berguna untuk membaca dan menulis data dalam format file CSV (*Comma-Separated Values*), yang sering digunakan untuk menyimpan data tabular.

2.3 Tahapan penelitian

Penelitian dilakukan dengan beberapa tahapan seperti yang ditunjukkan pada Gambar 7.



Gambar 7. Tahapan penelitian. Diagram ini menggambarkan langkah-langkah dalam penelitian yang mencakup studi literatur, eksplorasi data dan implementasi awal model, pemilihan algoritma, pengumpulan data, perancangan model, evaluasi model dan analisa hasil, dan penyusunan laporan akhir penelitian. Seluruh tahapan penting untuk menyelesaikan penelitian ini.

Studi Literatur

Pada tahap ini, peneliti melakukan studi literatur atau pra-penelitian untuk mengenali dan memahami teori, konsep dasar, jenis data, dan algoritma yang relevan dengan sistem penilaian esai otomatis berbasis kemiripan semantik. Peneliti mengeksplorasi dan mempelajari algoritma embedding statis seperti Word2Vec, FastText, dan lainnya, serta embedding kontekstual seperti BERT, teknik regresi, serta metode pengukuran similaritas semantik. Selain itu, peneliti juga mempelajari metrik evaluasi yang umum dan efektif untuk penilaian esai otomatis. Studi literatur ini bertujuan untuk mengidentifikasi metode yang paling sesuai dan menemukan solusi penelitian yang belum terjawab dari studi sebelumnya.

Eksplorasi Data dan Implementasi Awal

Tahap ini diawali dengan eksplorasi berbagai jenis dataset yang digunakan dalam penelitian sejenis, baik untuk esai berbahasa Inggris maupun bahasa Indonesia. Peneliti kemudian mencoba mengimplementasikan kode dasar sistem penilaian esai otomatis dari penelitian terdahulu tanpa modifikasi. Tujuan dari tahap ini adalah untuk memahami cara kerja sistem tersebut dan memastikan kelayakan penelitian yang akan dilakukan.

Pemilihan Algoritma

Setelah memahami dasar implementasi sistem, peneliti melakukan analisis untuk menentukan algoritma yang paling sesuai untuk penelitian ini. Algoritma yang dipilih meliputi embedding statis (Word2Vec) dengan berbagai teknik regresi dan perhitungan similaritas, serta embedding kontekstual menggunakan model BERT dengan teknik *fine-tuning*. Pemilihan ini dilakukan untuk membandingkan efektivitas kedua pendekatan dalam menilai kemiripan semantik jawaban esai.

Pengumpulan Dataset

Pada tahap ini, peneliti melakukan pengumpulan dataset dari dua sumber sesuai dengan bahasa yang digunakan, yaitu dataset berbahasa Inggris yang tersedia secara publik, dan dataset berbahasa Indonesia yang dikumpulkan langsung melalui pelaksanaan ujian mata pelajaran tertentu di sekolah dengan memberikan soal esai dengan bentuk esai terbatas kepada siswa. Jawaban dari siswa dinilai oleh tenaga pengajar sebagai acuan penilaian. Selanjutnya, dilakukan proses pembersihan dan transformasi dataset agar sesuai dengan format yang dibutuhkan untuk implementasi model. Tahap pra-proses data yang dilakukan mencakup normalisasi teks, seperti penghapusan karakter khusus dan tanda baca, tokenisasi teks menjadi unit kata atau sub-kata, dan konversi data ke dalam format yang sesuai untuk model *embedding* statis dan kontekstual (BERT). Tahap ini bertujuan untuk memastikan data siap digunakan dalam proses pemodelan. Dataset ini kemudian akan digunakan untuk melatih dan mengevaluasi model penilaian esai otomatis.

Perancangan Model

Pada tahap ini, peneliti mengimplementasikan model *embedding* statis seperti Word2Vec, lalu dilakukan penilaian dilakukan dengan teknik regresi dan perhitungan similaritas semantik menggunakan metode seperti cosine *Similarity*, *Word Mover ' s Distance*, dan lainnya. Selain itu, model embedding kontekstual (BERT) juga diimplementasikan dengan teknik *fine-tuning* pada model BERT menggunakan dataset yang telah diproses. Model dilatih untuk mengevaluasi kemiripan semantik antara jawaban

siswa dan kunci jawaban. Proses implementasi ini bertujuan untuk menguji performa kedua algoritma pada dataset yang tersedia.

Evaluasi Model dan Analisa Hasil

Setelah implementasi selesai, model dievaluasi menggunakan metrik *Mean Absolute Error* (MAE) yang mengukur rata-rata selisih absolut antara prediksi model dan nilai sebenarnya, *Mean Squared Error* (MSE) untuk mengukur rata-rata kuadrat selisih antara prediksi dan nilai sebenarnya, dan korelasi pearson untuk mengukur tingkat korelasi antara hasil prediksi model dan penilaian manual dari tenaga pengajar. Evaluasi dilakukan pada kedua dataset (berbahasa Inggris dan Indonesia) untuk menilai keakuratan dan efektivitas model dalam menilai jawaban esai. Tahap ini dilakukan untuk mengidentifikasi kelebihan dan kekurangan masing-masing model berdasarkan jenis dataset yang digunakan, dan juga perbandingan performa antara dataset berbahasa Inggris dan Indonesia untuk melihat bagaimana model beradaptasi dengan bahasa yang berbeda.

Penyusunan Laporan Penelitian

Tahap terakhir adalah penyusunan laporan penelitian yang mencakup seluruh tahapan, mulai dari pendahuluan, tinjauan pustaka, metode penelitian, implementasi model, evaluasi kinerja model, hingga kesimpulan dan saran untuk penelitian ini. Laporan disusun sebagai bentuk pertanggungjawaban ilmiah atas penelitian yang dilakukan.

2.4 Teknik Pengumpulan Data

Data yang digunakan pada penelitian ini adalah data berupa jawaban siswa, kunci jawaban, jawaban siswa, dan penilaian manual oleh tenaga pengajar. Pengumpulan data ini dilakukan dalam dua bahasa, yaitu bahasa Inggris dan bahasa Indonesia, dengan teknik pengumpulan data yang berbeda sesuai dengan karakteristik dan ketersediaan dataset yang digunakan.

Dataset berbahasa Inggris yang digunakan dalam penelitian ini merupakan dataset publik dalam bidang *computer science* yang diperoleh dari salah satu kelas di *University of North Texas*. Dataset ini dibuat oleh Mohler *et al.* (Mohler et al., 2011) dan berasal dari sepuluh tugas dan dua ujian pada mata kuliah pengantar *computer science*. Dataset ini terdiri dari 80 soal dengan 2273 jawaban siswa. Setiap jawaban siswa dinilai oleh dua orang tenaga pengajar menggunakan skala integer 0 hingga 5. Untuk memperoleh nilai akhir dari jawaban siswa, diambil rata-rata dari dua nilai yang diberikan oleh tenaga pengajar tersebut.

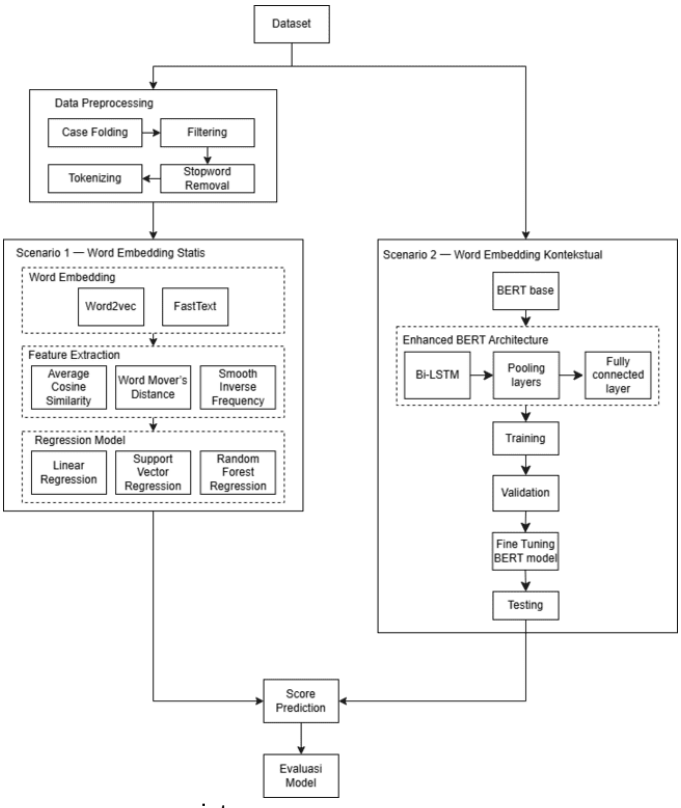
Untuk menghindari overfitting, ditambahkan metode untuk memilih satu jawaban siswa yang benar dari setiap soal sebagai jawaban referensi tambahan, seperti yang dilakukan penulis dalam penelitian "*Automatic Short-Answer Grading via BERT-Based Deep Neural Networks*" (Zhu et al., 2022). Metode ini berhasil menambah jumlah data latih dari 2083 pasangan menjadi sekitar 3300 pasang

jawaban dan kunci jawaban. Namun, pada beberapa soal tertentu, tidak ditemukan jawaban siswa yang benar, sehingga jawaban dari soal tersebut tidak ditambahkan dalam perluasan data latih.

Berbeda dengan dataset berbahasa Inggris, pengumpulan dataset berbahasa Indonesia dilakukan secara langsung melalui kegiatan ujian yang diselenggarakan di salah satu SMA. Penulis melaksanakan ujian pada tiga kelas berbeda dengan tiga mata pelajaran yang berbeda, yaitu Bahasa Indonesia, Seni Tari, dan Fisika. Setiap kelas diberikan beberapa soal esai yang terdiri dari 15-20 soal, dengan jumlah responden sekitar 30 siswa. Data kemudian diolah dan dibersihkan menjadi 1.847 butir soal dan jawaban yang dikumpulkan dari siswa digunakan sebagai bagian dari dataset dalam penelitian ini. Selain itu, setiap jawaban siswa dinilai secara manual oleh satu orang tenaga pengajar untuk menghasilkan skor sebagai label atau data acuan dalam proses analisis lebih lanjut. Dengan demikian, kedua dataset ini (bahasa Inggris dan bahasa Indonesia) menjadi komponen penting dalam penelitian untuk mengevaluasi model penilaian otomatis yang diusulkan.

2.5 Perancangan dan Implementasi Sistem

Perancangan sistem memberikan gambaran terhadap sistem yang akan dibuat. Terdapat beberapa skenario dalam penelitian ini, yang ditunjukkan pada gambar .



Gambar 8. Alur perancangan sistem

Proses dimulai dengan memuat dataset mentah yang berisi pasangan kalimat, yaitu *response* dan *answer*, serta label kesamaan dalam rentang 0 hingga 5. Dataset ini kemudian dinormalisasi agar nilainya berada pada rentang [0, 1] dengan membagi labelnya dengan nilai 5. Dataset ini dimuat melalui fungsi load_custom_dataset yang membaca setiap baris dari file dataset, memisahkannya menjadi komponen-komponen yang sesuai, lalu menambahkan data ke dalam list.

Setelah dataset berhasil dimuat, data tersebut dibagi menjadi tiga bagian: *training set*, *validation set*, dan *test set*. Parameter yang dimasukkan dalam penelitian ini adalah 80% *training*, 10% *validation*, dan 10% *testing*. Sebagai perbandingan, pada data berbahasa Indonesia dilakukan pula beberapa skenario lain, yaitu pembagian 70% *training*, 15% *validation*, dan 15% *testing*, serta 60% *training*, 20% *validation*, dan 20% *testing*. Perbandingan ini bertujuan untuk menentukan konfigurasi yang menghasilkan kinerja model paling optimal pada data uji. Proses pembagian dilakukan menggunakan fungsi split_dataset, di mana proporsi data ditentukan berdasarkan parameter validasi dan pengujian yang diberikan. Data training digunakan untuk melatih model, sementara data validasi digunakan untuk mengevaluasi kinerja model selama proses pelatihan dan mencegah *overfitting*. Sementara itu, data uji digunakan untuk pengujian akhir setelah model dilatih. Selanjutnya, pasangan kalimat diproses lebih lanjut menurut masing-masing skenario.

2.5.1 Skenario 1 (*Word embedding* statis)

Pada skenario ini, dataset diproses melalui beberapa tahapan, yaitu *Preprocessing*, *word embedding*, *feature extraction*, *regression model*.

1. *Preprocessing*

   Tahap *Preprocessing* sangat penting dilakukan pada skenario pelatihan *word embedding* statis, karena model yang digunakan skenario ini bergantung sepenuhnya pada kualitas data input untuk menghasilkan representasi vektor kata yang akurat. Model Word2Vec dan FastText bersifat statis, di mana setiap kata akan memiliki satu vektor tetap tanpa memahami konteks penggunaannya. Oleh karena itu, jika data mengandung noise seperti variasi huruf besar dan kecil, tanda baca, atau simbol yang tidak relevan, representasi kata akan menjadi tidak konsisten dan mengganggu kualitas hasil *embedding*. Selain itu, stopword yang sering muncul dalam teks akan memberikan bias pada distribusi vektor, karena Word2Vec memperlakukan setiap kata secara setara tanpa memperhitungkan kepentingan semantik kata tersebut.

   Proses seperti *case folding*, *filtering*, *stopword removal*, dan *tokenizing* membantu memastikan bahwa data yang digunakan lebih bersih, seragam, dan hanya berisi kata-kata yang bermakna, sehingga model dapat mempelajari hubungan antar kata dengan lebih efektif. Dengan demikian, *Preprocessing* tidak hanya meningkatkan efisiensi komputasi tetapi juga memastikan bahwa vektor kata yang dihasilkan lebih representatif dan

berkualitas dalam mendukung sistem penilaian esai otomatis berbasis kemiripan semantik. Berikut adalah penjelasan detail mengenai beberapa tahapan pada *Preprocessing data*:

a. *Case folding*

   Case folding adalah proses mengubah semua huruf dalam teks menjadi huruf kecil (*lowercase*). Proses ini bertujuan untuk memastikan bahwa perbedaan huruf besar dan kecil tidak memengaruhi analisis teks. Tahapan ini berfungsi untuk mengurangi redundansi data karena kata dengan huruf besar dan kecil dianggap berbeda oleh sistem, dan juga menstandarkan teks agar semua kata dalam format yang sama sehingga mudah diproses lebih lanjut. Contohnya, kalimat sebelum *case folding*, “Indonesia adalah Negara Kepulauan TERBESAR di Dunia.”, setelah *case folding* berubah menjadi “Indonesia adalah negara kepulauan terbesar di dunia.”

b. *Filtering*

   Filtering adalah proses membuang karakter atau simbol-simbol yang tidak relevan dalam teks, seperti tanda baca, angka, atau simbol tertentu. Proses ini membantu fokus hanya pada kata-kata yang bermakna dalam teks. *Filtering* menghapus elemen teks yang tidak memberikan kontribusi signifikan dalam proses analisis semantik, dan membantu menghindari kebisingan (*noise*) dalam data. Contohnya, kalimat “Hari ini cuaca sangat cerah, 100% menyenangkan! :) #liburan” akan berubah menjadi “Hari ini cuaca sangat cerah menyenangkan liburan” setelah *filtering*.

c. *Stopword removal*

   Stopword removal adalah proses menghapus kata-kata yang sering muncul dalam teks tetapi memiliki bobot informasi yang rendah, seperti kata sambung, kata depan, atau kata kerja bantu. Proses ini menggunakan daftar *stopword* yang umum digunakan dalam bahasa Indonesia maupun bahasa inggris. Tahap ini mengurangi jumlah kata yang harus diproses sehingga efisiensi model meningkat, dan memfokuskan analisis hanya pada kata-kata yang memiliki makna penting. Contohnya, kalimat “Dia pergi ke sekolah dan bertemu temannya di kelas.” akan berubah menjadi “pergi sekolah bertemu temannya kelas” setelah penghapusan *stopword*.

d. *Tokenizing*

   Tokenizing adalah proses memecah teks menjadi unit-unit yang lebih kecil yang disebut token. Token bisa berupa kata, frasa, atau bahkan karakter tergantung pada metode tokenisasi yang digunakan. Pada skenario ini, tokenisasi dilakukan berbasis kata (*word-level tokenization*). Tokenisasi berfungsi untuk mengubah teks menjadi format yang dapat dipahami oleh model dan mempermudah pemrosesan teks dalam tahapan selanjutnya, seperti penerapan *word embedding*. Contohnya, kalimat “Saya belajar pemrosesan bahasa

alami” akan berubah menjadi ["Saya", "belajar", "pemrosesan", "bahasa", "alami"] setelah *tokenizing*.

2. *Word embedding*

Pada model Word2Vec, proses pelatihan dilakukan menggunakan library Gensim, yang menyediakan metode untuk membangun embedding dari data teks. Embedding dilatih menggunakan data domain spesifik yang terdiri atas kunci jawaban, jawaban siswa, dan label nilai yang diberikan oleh tenaga pengajar. Setiap kata direpresentasikan dalam bentuk vektor berdimensi 200. Sebagai contoh, kata-kata dalam kalimat "ruang riil atau ruang nyata mencakup tempat-tempat nyata" akan dikonversi menjadi vektor, seperti "ruang" : [0.12, 0.34, 0.25, ..., 0.07] (dimensi 200), "riil" : [0.15, 0.38, 0.21, ..., 0.06] (dimensi 200), dan "nyata" : [0.09, 0.30, 0.22, ..., 0.10] (dimensi 200).

Jendela konteks (window size) ditetapkan sebesar 4, sehingga model dapat menangkap hubungan semantik antar kata dalam rentang empat kata di sekitarnya. Sebagai contoh, dalam kalimat "Ruang imajiner adalah ruang pergerakan penari", kata "ruang" memiliki kata-kata konteks ["imajiner", "adalah", "pergerakan", "penari"].

Batas minimum kemunculan kata dalam data ditetapkan sebesar 1 (min_count=1), sehingga seluruh kata yang jarang muncul tetap terwakili dalam embedding. Contohnya, dari kalimat "Ruang imajiner adalah ruang pergerakan penari", diperoleh vocabulary = ["ruang", "imajiner", "adalah", "pergerakan", "penari"].

Proses pelatihan Word2Vec menggunakan metode Skip-gram, yang lebih efektif dalam menangkap hubungan semantik kata-kata dengan frekuensi kemunculan rendah. Untuk mengoptimalkan embedding, diterapkan teknik negative sampling dengan jumlah sampel negatif sebanyak 15. Teknik ini mengurangi beban komputasi pada dataset besar dengan memilih kata-kata acak dari vocabulary yang tidak berhubungan dengan kata target. Misalnya, untuk kata "ruang", model akan memilih 15 kata negatif secara acak, seperti ["tampil", "nyata", "benar", dll.]. Model kemudian mempelajari distribusi kata dengan menggunakan tingkat pembelajaran (learning rate) awal sebesar 0.03, yang dikurangi secara bertahap guna mencegah overfitting. Model ini dilatih selama 30 epoch, yang cukup untuk mencapai konvergensi tanpa menyebabkan overfitting. Setelah proses pelatihan selesai, representasi vektor untuk kata "ruang" berubah menjadi [0.25, -0.18, 0.42, ..., -0.12] (dimensi 200), yang telah mengandung informasi semantik berdasarkan konteks penggunaannya dalam teks.

Selain Word2Vec, model embedding juga dikembangkan menggunakan FastText untuk menangkap representasi kata yang lebih kaya dengan mempertimbangkan informasi subword. Model FastText dilatih dengan dimensi vektor sebesar 300, jendela konteks sepanjang 3 kata, dan frekuensi minimum kata sebesar 1. Proses pelatihan dilakukan selama 30

epoch. Tidak seperti Word2Vec yang merepresentasikan kata sebagai satu unit tunggal, FastText memecah kata menjadi subword berdasarkan rentang panjang tertentu. Sebagai contoh, kata "ruang" diproses dengan min_n = 3, menghasilkan subword seperti ['<ru', 'rua', 'uan', 'ang', 'ng>'], serta max_n = 6, menghasilkan subword seperti ['<ruang', 'ruang>', '<ruan', 'uang>', ...].

Pada kedua pendekatan embedding tersebut, selain pelatihan menggunakan data domain spesifik melalui library Gensim, penelitian ini juga menguji skenario penggunaan model pretrained. Model pretrained Word2Vec dan FastText yang digunakan telah dilatih pada corpus skala besar, seperti Google News untuk bahasa Inggris dan Wikipedia (versi terbaru) untuk bahasa Indonesia. Pendekatan ini bertujuan untuk membandingkan performa antara model yang dilatih secara domain spesifik dan model pretrained, sehingga diperoleh gambaran yang lebih komprehensif mengenai efektivitas representasi kata dalam konteks penilaian esai berbasis semantik.

3. *Feature extraction*

Proses *feature extraction* menggunakan tiga pendekatan utama, yaitu *Average Cosine Similarity* (AvgCos), *Word Mover's Distance* (WMD), dan *Smooth Inverse Frequency* (SIF). Representasi vektor dari setiap kata yang terdapat dalam kalimat, misalnya "ruang" dengan vektor [0.23, -0.16, 0.41, ..., -0.10] #(Dimensi 300), "riil" dengan vektor [0.07, 0.40, 0.28, ..., 0.23] #(Dimensi 300), …, n kata dengan v1, v2, ..., vn, akan diproses menurut masing-masing jenis ekstraksi fitur. Untuk AvgCos, nilai cosine similarity dari setiap pasangan kata di dalam kalimat jawaban dan kunci jawaban, seperti nilai *cosine similarity* antara "ruang" (Kalimat 1) dan "riil" (Kalimat 2), akan dihitung sebagai berikut.

$$\frac{(0.23, -0.16, \ldots, -0.10) \cdot (0.07, 0.40, \ldots, 0.23)}{||0.23, -0.16, \ldots, -0.10|| \; ||0.07, 0.40, \ldots, 0.23||}$$

Maka nilai *average cosine similarity* akan dihitung sebagai berikut.

$$\frac{[\,1 + 0.90 + \ldots + \mathrm{cs(n)}\,]}{n\ pasangan\ vektor} = 0{,}92\ (misalkan)$$

Dalam ekstraksi fitur *Smooth Inverse Frequency* (SIF), setiap kata dalam kalimat akan direpresentasikan sebagai vektor dan diberikan bobot sesuai dengan frekuensinya dalam korpus. Kata-kata yang jarang muncul akan mendapatkan bobot lebih tinggi, sedangkan kata-kata yang sering muncul akan memiliki bobot lebih rendah. Setelah itu, embedding kalimat dihitung dengan mengambil rata-rata tertimbang dari vektor kata. Untuk menghilangkan bias distribusi vektor, dilakukan penghapusan komponen utama menggunakan *Principal Component Analysis* (PCA) guna menemukan arah dominan dalam embedding kalimat dan menguranginya. Langkah terakhir adalah menghitung *cosine similarity* antara embedding dua

kalimat yang dibandingkan, yang kemudian menghasilkan vektor representasi akhir.

Sementara itu, dalam ekstraksi fitur *Word Mover's Distance* (WMD), jarak optimal antara dua kalimat dihitung dengan menentukan biaya minimum untuk "memindahkan" kata-kata dari satu kalimat ke kalimat lainnya berdasarkan representasi vektornya. Metode ini mempertimbangkan hubungan semantik antar kata dan mampu mengukur kesamaan teks dengan tingkat akurasi yang lebih tinggi dibandingkan pendekatan berbasis frekuensi sederhana.

4. Model regresi

   Fitur yang dihasilkan dari tahap ekstraksi kemudian digunakan sebagai input ke model regresi. Dalam penelitian ini, digunakan beberapa jenis model regresi untuk memprediksi skor kemiripan, yaitu Regresi Linear, Support Vector Regression (SVR), dan Random Forest Regressor (RFR). Pemilihan model ini bertujuan untuk mengevaluasi kemampuan pendekatan berbasis regresi dalam menangkap hubungan linear maupun non-linear antara fitur yang diekstraksi dan skor kemiripan.

   Sebagai ilustrasi, misalkan model belajar dengan parameter

   $$w_0 = 0.1 \;\; w_1 = 1.2$$

   w_0 sebagai bias dan w_1 sebagai bobot fitur. Prediksi skor kemiripan $\hat{y}$ dapat dihitung dengan persamaan regresi linear sebagai berikut:

   $$\hat{y} = w_0 + w_1 * hasil\ ekstraksi\ fitur$$

   Jika hasil ekstraksi fitur bernilai 0.75, maka prediksi skor kemiripan yang dihasilkan adalah:

   $$\hat{y} = 0.1 + (1.2 \times 0.75) = 1.0$$

   Dalam model SVR, digunakan kernel linear untuk menjaga efisiensi komputasi serta mempermudah interpretasi hubungan antara fitur dan skor dengan nilai bias($\in$) yang bervariasi. Sementara itu, dalam RFR, parameter seperti jumlah estimator = 100 dan max_depth = None diatur untuk mengurangi risiko *overfitting* dan meningkatkan kemampuan model dalam menangkap pola data yang lebih kompleks. Dengan pendekatan ini, diharapkan model regresi dapat belajar dari fitur yang telah diekstraksi dan menghasilkan prediksi skor kemiripan semantik yang akurat antara dua kalimat yang dibandingkan.

2.5.2 Skenario 2 (*Word embedding* kontekstual)

Pada skenario pelatihan ini, BERT digunakan sebagai model dasar dalam menangani teks mentah secara langsung tanpa memerlukan tahapan *Preprocessing* seperti *stopword removal*, *stemming*, atau *filtering*. Hal ini dikarenakan tokenizer khusus yang digunakan dalam model BERT, yang secara otomatis memproses teks menjadi token sub-kata menggunakan pendekatan *WordPiece*. Tokenizer ini mampu

memecah kata menjadi unit-unit yang lebih kecil, sehingga dapat menangani kata-kata yang tidak dikenal atau *out-of-vocabulary* (OOV) dengan efektif.

Selain itu, BERT menggunakan representasi kata berbasis konteks yang mempertimbangkan posisi dan makna kata dalam kalimat. Hal ini berbeda dengan pendekatan tradisional seperti *bag-of-words* atau *TF-IDF* yang hanya bergantung pada kata secara individual. Dengan mempertimbangkan konteks secara keseluruhan, BERT menghilangkan kebutuhan untuk teknik seperti *stemming* atau *stopword removal*, karena setiap kata, termasuk *stopword*, memiliki nilai informasi yang dapat memengaruhi makna kalimat secara keseluruhan.

Pendekatan *WordPiece* yang digunakan oleh BERT juga memungkinkan representasi sub-kata untuk kata yang tidak dikenal, memastikan bahwa semua elemen teks tetap direpresentasikan dengan baik di dalam model. Dengan demikian, tahapan *Preprocessing* tradisional tidak diperlukan dalam pipeline BERT karena tokenizer dan arsitektur model ini sudah secara otomatis menangani berbagai tantangan pemrosesan teks mentah dengan lebih efisien.

Perancangan sistem pada skenario kedua terdiri atas beberapa tahapan proses, yaitu pemrosesan dataset, tokenisasi dengan BERT base, pengembangan arsitektur model BERT, proses training, validasi, *fine-tuning* BERT model, implementasi pada data testing, dan evaluasi model.

a. Pemrosesan dataset

Dataset yang telah diproses dikonversi ke dalam format DataLoader menggunakan class CustomDataset. Class ini memproses setiap pasangan kalimat melalui tokenizer BERT dengan panjang maksimum 128 token. Proses tokenisasi menghasilkan dua komponen penting, yaitu input_ids dan attention_mask, yang digunakan sebagai input ke model. Setiap batch data yang dihasilkan oleh DataLoader akan memuat pasangan tokenized input serta label kesamaan dalam bentuk tensor.

Data training dimuat ke dalam train_dataloader dengan batch size sebesar 8 dan opsi shuffle=True untuk memastikan urutan data yang berbeda pada setiap epoch. Sedangkan data validasi dimuat ke dalam validation_dataloader dengan shuffle=False untuk menjaga konsistensi data selama evaluasi.

b. BERT-*base tokenizing*

Pada tahap ini, teks diproses menggunakan *tokenizer* dari model *bert-base-uncased*. Tokenizer ini bertugas untuk mengubah teks input menjadi token-token yang dapat dipahami oleh model BERT. Tokenizer menghasilkan dua komponen utama: *input_ids* (ID token) dan *attention_mask* (masking perhatian). Input yang sudah di-*tokenize* ini memiliki panjang maksimum yang diatur hingga 128 token. Proses ini memastikan setiap pasangan kalimat memiliki representasi numerik yang seragam untuk diproses oleh model.

c. *Enhanced BERT Architecture*

Pengembangan arsitektur model BERT pada penelitian ini terdiri dari tiga komponen utama, yaitu Bi-LSTM, *pooling layers*, dan *fully connected layer*, yang

bekerja secara terintegrasi untuk menyelesaikan tugas penilaian kesamaan semantik antara dua teks.

- *Bidirectional*-LSTM (Bi-LSTM)
  Setelah representasi token *embeddings* diperoleh dari model BERT yang telah dilatih sebelumnya, *embeddings* tersebut dilewatkan ke lapisan *Bidirectional*-LSTM. Lapisan ini terdiri dari dua arah pemrosesan, maju (*forward*) dan mundur (*backward*), yang memungkinkan model menangkap konteks dari kedua arah dalam urutan token. Bi-LSTM menggunakan 64 *hidden units* dengan satu lapisan (*layer*), yang menghasilkan keluaran dengan dimensi dua kali lipat karena sifat *bidirectional*-nya. Fungsinya adalah untuk memodelkan *dependencies* jangka panjang antara token dalam satu kalimat, sehingga konteks semantik dapat dipahami secara lebih mendalam.
- *Pooling Layers*
  Keluaran dari lapisan Bi-LSTM diproses melalui dua jenis *pooling layers* untuk mereduksi dimensi dan menangkap informasi penting dari token *embeddings*:
  - *Average Pooling*: Menghitung rata-rata nilai embeddings dari semua token dalam satu kalimat. *Pooling* ini membantu menangkap representasi umum dari kalimat secara keseluruhan.
  - *Max Pooling*: Mengambil nilai maksimum dari setiap dimensi embeddings. Pooling ini bertujuan menangkap fitur-fitur paling menonjol dalam urutan token.

  Hasil dari kedua metode *pooling* ini kemudian digabungkan (*concatenated*) menjadi satu vektor fitur yang lebih informatif dan komprehensif.
- *Fully Connected Layer*
  Vektor hasil dari proses *pooling* kemudian dilewatkan ke lapisan *Fully Connected* (FC). Lapisan ini terdiri dari unit linear dengan tambahan *dropout* sebesar 0.3 untuk mencegah *overfitting*. *Fully connected layer* berfungsi untuk memetakan vektor fitur yang telah diolah menjadi skor prediksi akhir, yang menunjukkan tingkat kesamaan antara dua kalimat. Proses ini memastikan bahwa informasi yang diperoleh dari Bi-LSTM dan pooling dapat dikonversi menjadi hasil numerik yang dapat diinterpretasikan.

  Dengan demikian, integrasi antara Bi-LSTM, *pooling layers*, dan *fully connected layer* pada arsitektur ini memungkinkan model menangkap konteks semantik yang kompleks, mereduksi dimensi data, dan menghasilkan skor prediksi yang akurat untuk tugas penilaian kesamaan teks.

d. *Training phase*

Proses pelatihan model dilakukan selama 8 *epoch*. Pada setiap *epoch*, model melakukan *forward pass*, di mana input data berupa *input_ids* dan

attention_mask dilewatkan melalui model untuk menghasilkan skor prediksi. *Loss* dihitung menggunakan fungsi *Mean Squared Error* (MSE) dengan membandingkan skor prediksi dengan label target yang telah dinormalisasi. Proses *backward pass* kemudian dijalankan untuk memperbarui parameter model dengan teknik *gradient descent*, dan *gradient clipping* diterapkan untuk mencegah gradien meledak.

Pada akhir setiap *epoch*, model dievaluasi menggunakan data validasi. Nilai *validation loss* dihitung untuk memantau kinerja model. Setelah semua *epoch* selesai, lapisan BERT yang sebelumnya dibekukan kemudian diaktifkan kembali (*unfreeze*), dan proses *fine-tuning* dijalankan agar model dapat menyesuaikan representasi *embedding*nya secara lebih spesifik untuk tugas ini.

Proses pelatihan model menggunakan Adam *optimizer* dengan *learning rate* sebesar 1e-5. Pemilihan Adam dilakukan karena algoritma ini merupakan kombinasi antara *adaptive gradient algorithm* (AdaGrad) dan *Root Mean Square Propagation* (RMSProp), yang mampu menyesuaikan *learning rate* secara otomatis berdasarkan gradien dari parameter. Adam sangat efektif untuk model kompleks seperti BERT karena mempercepat konvergensi dan stabilitas dalam pembaruan bobot (Kingma & Ba, 2017).

Setelah proses pelatihan pada setiap skenario sudah selesai, model dievaluasi menggunakan data validasi dan data uji. Evaluasi dilakukan dengan menghitung metrik seperti *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), dan *Pearson Correlation*. Karena label skor awal memiliki rentang 0 hingga 5 tetapi telah dinormalisasi menjadi [0, 1] selama proses pelatihan. Hasil prediksi yang dihasilkan dari setiap skenario pelatihan akan dikalikan kembali dengan 5 untuk mendapatkan nilai skor dalam skala aslinya. Proses ini memastikan bahwa metrik evaluasi yang dihitung, seperti RMSE dan MAE, mencerminkan kinerja model dalam rentang skor asli dataset.