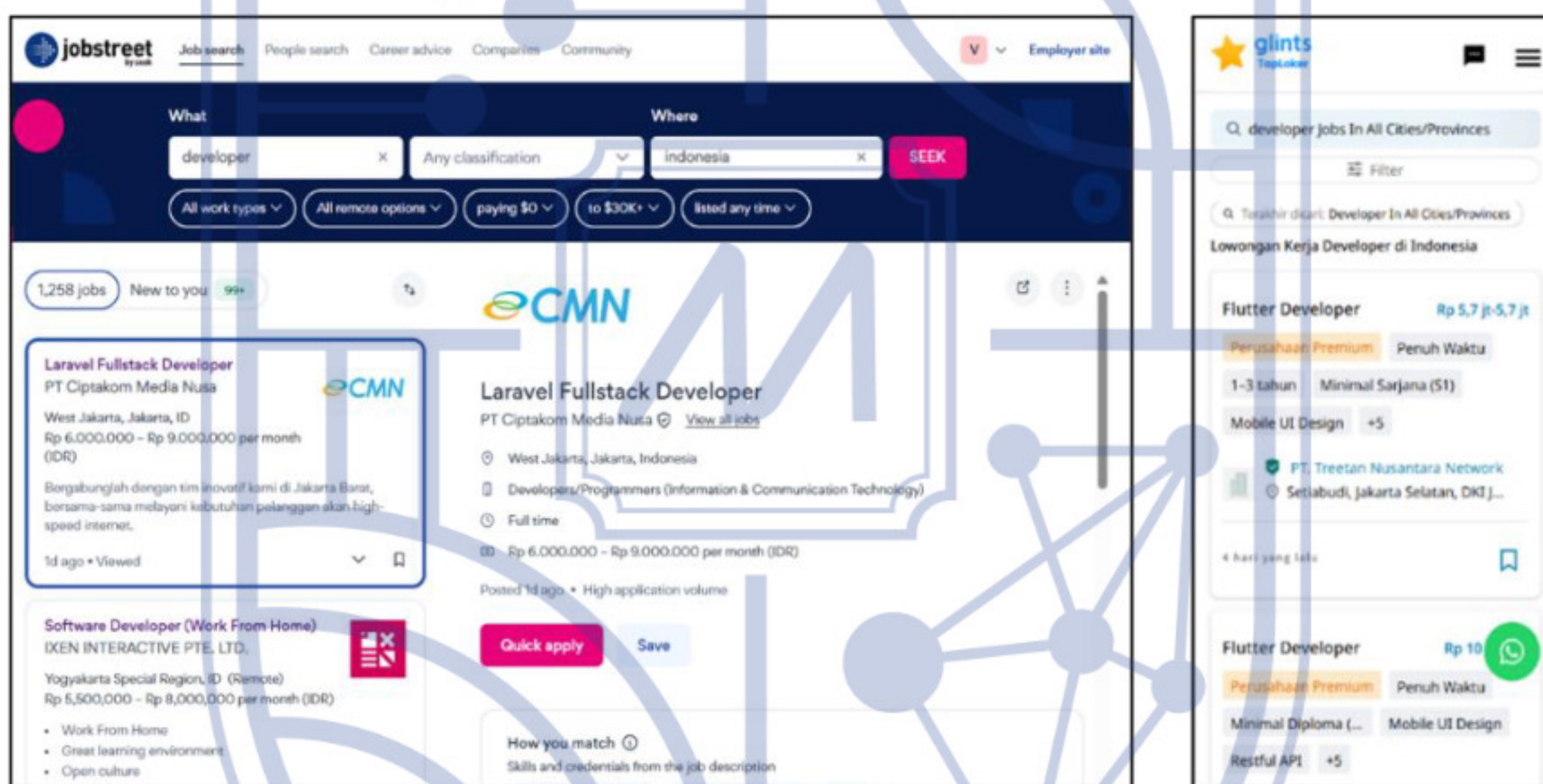


BAB II

KAJIAN LITERATUR

2.1 Platform Pencari Lowongan Kerja

Platform pencari lowongan kerja merupakan sebuah sistem digital yang berfungsi sebagai perantara antara pencari kerja dan penyedia lapangan kerja. Platform ini memungkinkan terjadinya interaksi antara kedua pihak dalam ruang digital, baik melalui media berbasis *website* maupun aplikasi *mobile*. Contoh platform lowongan kerja berbasis *website* dan aplikasi *mobile* ditunjukkan pada Gambar 2.1. Platform pencari lowongan kerja ini dirancang untuk membantu pengguna menemukan informasi pekerjaan yang sesuai dengan kebutuhan, kualifikasi, dan preferensi pengguna [16].



Gambar 1.1 Tampilan platform pencari kerja berbasis *website* dan *mobile* [17], [18]

Seiring dengan perkembangan teknologi, proses penyebaran dan pencarian informasi lowongan kerja telah mengalami transformasi yang signifikan. Sebelum adanya pemanfaatan teknologi digital, informasi lowongan kerja umumnya disebarluaskan melalui media cetak, seperti surat kabar, papan pengumuman, ataupun melalui komunikasi lisan. Metode tersebut memiliki keterbatasan dalam hal jangkauan geografis, kecepatan penyebaran dan pembaruan informasi. Dengan hadirnya internet, proses penyebaran dan pencarian lowongan kerja mulai beralih ke platform daring berbasis *website* yang memungkinkan akses informasi secara lebih luas dan cepat [16], [19]. Namun, dengan adanya perkembangan di bidang kecerdasan buatan, pemanfaatan teknologi digital dalam proses pencarian lowongan pekerjaan tidak lagi hanya berfungsi sebagai penyedia informasi,

tetapi juga dimanfaatkan untuk memberikan informasi yang lebih terpersonalisasi bagi pencari kerja [20].

Platform pencari lowongan kerja yang ada saat ini memungkinkan pengguna untuk memasukkan kriteria atau kata kunci tertentu, seperti posisi pekerjaan, lokasi, pengalaman, pendidikan, maupun kisaran gaji yang diinginkan [16], [19]. Pengguna juga dapat mengunggah dokumen pendukung, seperti *curriculum vitae* (CV) dan surat lamaran, serta mengirimkan lamaran secara digital ke beberapa lowongan yang tersedia [16]. Sistem kemudian memproses masukan pengguna melalui mekanisme pencarian yang telah ditentukan untuk menghasilkan daftar lowongan kerja yang dianggap relevan. Hasil pencarian tersebut selanjutnya akan disajikan kepada pengguna dalam bentuk daftar lowongan yang ditampilkan oleh sistem [16], [19]. Namun, pendekatan pencarian yang berbasis pada pencocokan kata kunci sering kali menghasilkan hasil yang kurang optimal apabila terdapat perbedaan istilah atau pilihan kata antara masukan pengguna dan deskripsi lowongan yang disediakan oleh pemberi kerja. Hal ini dikarenakan sistem masih melakukan pencocokan kata secara literal tanpa memperhatikan hubungan semantik antar kata, sehingga makna yang serupa tidak dapat dipahami dengan baik oleh sistem [9].

2.2 Sistem Rekomendasi

Sistem rekomendasi merupakan sistem yang dirancang untuk memberikan saran, informasi, serta objek yang relevan berdasarkan preferensi dan kebutuhan pengguna. Dalam perkembangannya, sistem rekomendasi banyak menerapkan kecerdasan buatan dan *machine learning* untuk meningkatkan akurasi dan personalisasi rekomendasi. Sistem ini berperan sebagai mekanisme penyaring informasi yang mendukung pengguna dalam menavigasi kumpulan data yang banyak agar proses pencarian dan pengambilan keputusan dapat dilakukan secara lebih efisien [21]. Dalam implementasinya, sistem rekomendasi telah banyak diterapkan pada berbagai platform digital, seperti platform *e-commerce*, layanan *streaming* musik dan film, hingga pada platform pencari lowongan kerja yang bertujuan mempertemukan kandidat dengan posisi yang paling sesuai [21], [22].

Efektivitas sistem rekomendasi sangat bergantung pada kualitas representasi data dan ketersediaan histori interaksi di dalam sistem. Namun, proses ini sering kali dihadapkan pada tantangan teknis seperti *cold-start problem*, yaitu kesulitan dalam memberikan rekomendasi yang akurat ketika data pengguna baru atau objek baru masih sangat terbatas. Selain itu, sistem rekomendasi juga sering dihadapkan pada permasalahan *data sparsity*, di mana

jumlah interaksi yang tersedia relatif sedikit dibandingkan dengan keseluruhan data, sehingga menyulitkan sistem dalam mengenali pola preferensi pengguna secara optimal. Tantangan-tantangan tersebut merupakan karakteristik umum yang perlu diperhatikan dalam perancangan dan analisis sistem rekomendasi [21].

2.2.1 Cara Kerja Sistem Rekomendasi

Secara umum, sistem rekomendasi bekerja melalui serangkaian tahapan yang memodelkan hubungan antara pengguna dan objek yang akan direkomendasikan, dengan tujuan untuk memprediksi tingkat kecocokan atau preferensi pengguna. Proses ini memungkinkan sistem mengolah berbagai informasi yang tersedia dan memberikan rekomendasi yang bersifat personal dan relevan bagi setiap pengguna [21].

Pada tahap awal, sistem rekomendasi memerlukan pengumpulan data atau sumber informasi untuk memahami karakteristik pengguna. Data tersebut dapat bersasal dari masukan yang diberikan secara langsung oleh pengguna ataupun melalui riwayat interaksi antara pengguna dan sistem. Interaksi ini umumnya dibedakan menjadi *implicit feedback*, seperti klik, riwayat pencarian, atau durasi interaksi, serta *explicit feedback*, seperti penilaian atau ulasan yang diberikan secara langsung oleh pengguna. Data dan interaksi tersebut menjadi dasar dalam pemodelan preferensi pengguna di dalam sistem [21], [22].

Tahap selanjutnya adalah tahap pemrosesan, yaitu ketika data masukan yang telah dikumpulkan akan dianalisis menggunakan algoritma tertentu untuk mengekstraksi preferensi pengguna. Pada tahap ini, sistem mengolah data pengguna dan informasi untuk menemukan pola hubungan yang dapat digunakan dalam proses rekomendasi. Dalam banyak pendekatan, data tersebut ditransformasikan ke dalam representasi vektor yang memudahkan sistem dalam menghitung tingkat kemiripan dan kecocokan antara preferensi pengguna dengan konten atau item yang tersedia. Pendekatan *machine learning* dan pemodelan statistik sering digunakan untuk membantu sistem mengenali pola hubungan antara pengguna dan informasi yang tersedia [23].

Pada tahap akhir, sistem menghasilkan keluaran berupa daftar rekomendasi yang telah disusun berdasarkan urutan skor relevansi atau tingkat kecocokan tertinggi. Skor tersebut umumnya diperoleh melalui perhitungan metrik kemiripan atau perhitungan jarak antar vektor untuk menentukan sejauh mana suatu item sesuai dengan kebutuhan pengguna. Melalui mekanisme pengurutan tersebut, sistem rekomendasi mampu menyajikan informasi yang paling relevan berdasarkan prioritas, sehingga membantu pengguna dalam mengambil keputusan secara lebih efisien [23].

2.2.2 Klasifikasi Sistem Rekomendasi

Sistem rekomendasi dapat diklasifikasikan ke dalam beberapa kategori utama berdasarkan logika pemrosesan dan jenis data yang digunakan untuk menghasilkan rekomendasi. Secara umum, terdapat tiga pendekatan utama yang banyak digunakan dalam pengembangan sistem rekomendasi, yaitu:

1. *Content-Based Filtering*

Metode ini memberikan rekomendasi kepada pengguna dengan mencocokkan profil minat pengguna dengan karakteristik atau atribut yang melekat pada suatu objek [24]. Dalam konteks data teks, sistem akan menganalisis deskripsi, kata kunci, atau kategori dari item untuk mencari kemiripan semantik dengan preferensi yang dinyatakan oleh pengguna. Keunggulan utamanya adalah kemampuan untuk bekerja secara mandiri tanpa bergantung pada data pengguna lain, sehingga sangat efektif dalam menangani masalah *cold-start* bagi item-item baru [21], [24].

2. *Collaborative Filtering*

Pendekatan ini mengandalkan riwayat interaksi dan kesamaan pola perilaku antar pengguna untuk memprediksi ketertarikan terhadap item baru. Logika utamanya adalah berasumsi bahwa pengguna yang memiliki selera serupa di masa lalu kemungkinan besar akan menyukai item yang sama di masa depan. Meskipun sangat efektif dalam memberikan rekomendasi yang tidak terduga, metode ini sering menghadapi kendala kelangkaan data pada sistem dengan basis pengguna yang masih kecil [21].

3. *Hybrid Recommendation System*

Merupakan pendekatan yang menggabungkan dua atau lebih metode rekomendasi (misalnya menggabungkan *content-based* dan *collaborative*) guna menutupi keterbatasan masing-masing teknik. Dengan mengintegrasikan berbagai logika pemrosesan, sistem *hybrid* mampu memberikan hasil yang lebih akurat, meningkatkan keragaman rekomendasi, serta mengatasi masalah teknis seperti *cold-start problem* dan *data sparsity* [21].

Tabel 1.1 Perbandingan Pendekatan Sistem Rekomendasi

Aspek Perbandingan	Content-Based	Collaborative	Hybrid
Dasar penentuan rekomendasi	Fitur dan deskripsi item	Perilaku dan interaksi user	Gabungan keduanya
Penggunaan data pengguna lain	Tidak	Ya	Ya

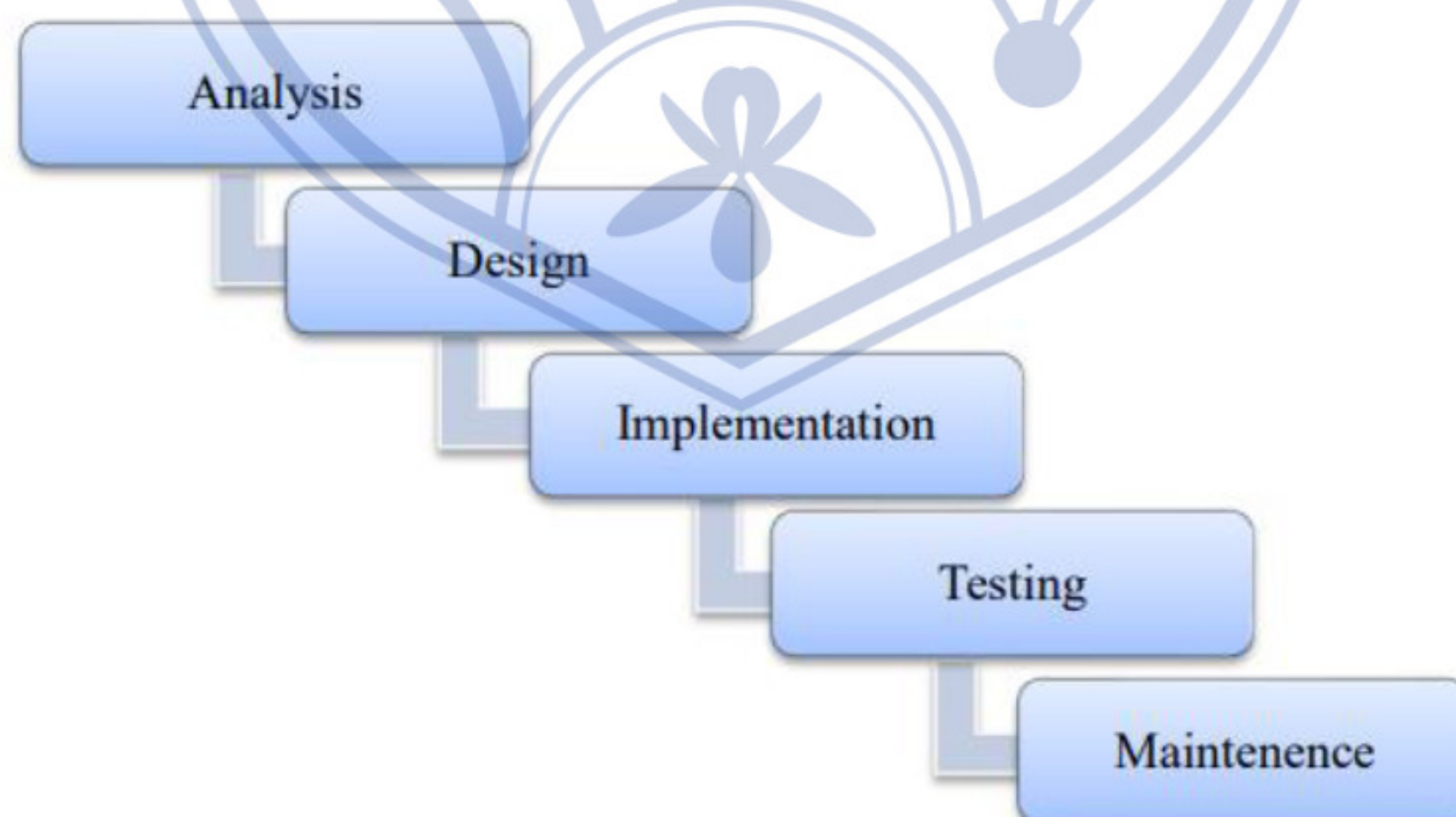
Kemampuan memproses teks deskriptif	Ya	Tidak	Ya
Kebutuhan riwayat interaksi	Tidak	Ya	Ya
Kemampuan menangani masalah <i>cold-start</i>	Ya	Tidak	Ya

Berdasarkan perbandingan pada Tabel 2.1, dapat dilihat bahwa pendekatan *Content-Based Filtering* memiliki keunggulan dalam memproses deskripsi teks dan tidak bergantung pada riwayat interaksi pengguna maupun data pengguna lain. Pendekatan ini efektif dalam mengekstraksi fitur semantik langsung dari teks tanpa memerlukan adanya histori interaksi sebelumnya, sehingga mampu memberikan rekomendasi yang relevan hanya berdasarkan informasi deskriptif yang diberikan secara eksplisit oleh pengguna [21], [24].

2.3 Model *Waterfall*

Model *Waterfall* merupakan salah satu metode manajemen proyek tertua yang pertama kali diperkenalkan oleh Dr. Winston Royce pada tahun 1970 untuk pengelolaan pengembangan perangkat lunak. Metode ini banyak digunakan dalam bidang teknologi informasi karena memiliki tahapan yang terstruktur dan sistematis. Dalam model *Waterfall*, setiap proses dilakukan secara berurutan sehingga setiap tahap harus diselesaikan terlebih dahulu sebelum melanjutkan ke tahap berikutnya [25].

Pada model *Waterfall*, proses pengembangan sistem dilakukan melalui beberapa tahapan yang saling berurutan. Tahapan model *Waterfall* dapat dilihat pada Gambar 2.2 [26].



Gambar 1.2 Tahapan Model *Waterfall*

Setiap tahapan pada model *Waterfall* memiliki fungsi dan proses yang berbeda, yaitu sebagai berikut [26]:

1. *Analysis Phase* (Tahap Analisis)

Tahap ini merupakan proses untuk menganalisis kebutuhan sistem secara lengkap, yang mencakup kebutuhan fungsional dan non-fungsional berdasarkan interaksi pengguna serta karakteristik dan kualitas sistem yang diharapkan.

2. *Design Phase* (Tahap Perancangan Sistem)

Tahap ini merupakan proses perancangan solusi berdasarkan hasil analisis kebutuhan. Proses ini mencakup perancangan algoritma, perancangan arsitektur sistem, perancangan basis data, serta perancangan antarmuka pengguna.

3. *Implementation Phase* (Tahap Implementasi)

Tahap ini merupakan proses penerapan rancangan sistem ke dalam bentuk program yang dapat dieksekusi. Pada tahap ini dilakukan penulisan kode program untuk menghasilkan komponen sistem yang fungsional berdasarkan desain yang telah dibuat. Pada tahap ini sistem mulai direalisasikan menjadi aplikasi yang berjalan.

4. *Testing Phase* (Tahap Pengujian Sistem)

Tahap ini merupakan proses verifikasi dan validasi untuk memastikan sistem yang dibangun sesuai dengan kebutuhan yang telah ditetapkan. Selain itu, tahap ini juga mencakup pengujian untuk menemukan dan memperbaiki kesalahan (*bug*) melalui proses *debugging*.

5. *Maintenance Phase* (Tahap Pemeliharaan Sistem)

Tahap ini merupakan proses pemeliharaan dan pengembangan sistem. Tujuannya adalah untuk memperbaiki kesalahan yang tidak terdeteksi sebelumnya, meningkatkan kinerja dan kualitas sistem, serta menyesuaikan sistem dengan kebutuhan atau lingkungan yang berubah.

2.4 Web Scraping

Web scraping merupakan teknik yang digunakan untuk mengambil atau mengekstraksi data dalam jumlah besar dari sebuah situs web secara otomatis. Teknik ini dilakukan dengan cara membaca struktur halaman web, kemudian mengambil informasi tertentu sesuai kebutuhan. Data hasil ekstraksi dapat disimpan dalam berbagai format seperti *Comma-Separated Values* (CSV), *spreadsheet*, maupun basis data sehingga dapat digunakan untuk proses pengolahan dan analisis data lebih lanjut [27].

Penggunaan teknik *web scraping* lebih efektif dibandingkan pengambilan data secara manual, terutama ketika data yang dibutuhkan memiliki jumlah yang besar dan terus diperbarui. Salah satu metode yang sering digunakan dalam proses *web scraping* adalah *HTML Parsing* dan *CSS Selector* untuk memilih elemen HTML berdasarkan *class*, *id*, maupun atribut tertentu. Implementasi *web scraping* umumnya menggunakan bahasa pemrograman Python karena memiliki berbagai *library* yang mendukung proses ekstraksi data. Beberapa *library* yang sering digunakan antara lain [27]:

1. BeautifulSoup

BeautifulSoup merupakan *library* Python yang digunakan untuk melakukan *parsing* dokumen HTML dan mengekstraksi informasi dari halaman web seperti teks, atribut, gambar, maupun tautan.

2. Selenium

Selenium digunakan untuk melakukan otomatisasi browser dan berinteraksi dengan elemen *website*, seperti membuka halaman, menekan tombol, melakukan navigasi halaman, maupun mengisi *form* secara otomatis.

3. Pandas

Pandas digunakan untuk mengolah, memanipulasi, dan menyimpan data hasil *scraping* ke dalam bentuk tabel atau *dataframe* sehingga memudahkan proses analisis data.

4. Time

Time digunakan untuk mengatur jeda waktu selama proses *scraping* berlangsung agar halaman *website* memiliki waktu untuk dimuat secara sempurna sebelum proses ekstraksi data dilakukan.

2.5 Data Preprocessing

Data preprocessing merupakan salah satu tahapan penting dalam pengolahan data sebelum data digunakan pada proses analisis maupun pengembangan sistem. Tahap ini dilakukan untuk mempersiapkan data agar memiliki kualitas yang lebih baik dan sesuai dengan kebutuhan pengolahan data. Proses *preprocessing* meliputi pembersihan data, transformasi data, integrasi data, serta pengurangan data yang tidak diperlukan sehingga data menjadi lebih terstruktur dan mudah diproses. Beberapa tahapan yang umum digunakan pada *data preprocessing* antara lain [28]:

1. Data Cleaning

Data cleaning merupakan teknik untuk mendeteksi dan memperbaiki kesalahan maupun ketidaksesuaian pada data yang telah dikumpulkan. Proses ini bertujuan agar data

menjadi lebih konsisten dan sesuai dengan kebutuhan sistem. Beberapa metode yang umum digunakan pada *data cleaning* antara lain penghapusan data duplikat, penanganan data yang hilang, penghapusan *noisy* data, dan pembersihan format data.

2. *Data Transformation*

Data transformation merupakan proses mengubah format, struktur, maupun nilai data agar sesuai dengan kebutuhan pengolahan dan analisis data. Proses transformasi dilakukan untuk menyesuaikan data dengan kebutuhan sistem maupun karakteristik *dataset* yang digunakan. Beberapa metode yang umum digunakan pada *data transformation* antara lain konversi tipe data, perubahan format data, dan normalisasi data.

3. *Data Reduction*

Data reduction merupakan proses mengurangi volume data tanpa menghilangkan informasi penting sehingga proses penyimpanan dan analisis data menjadi lebih efisien. Teknik ini digunakan untuk mengurangi kompleksitas data serta meningkatkan efisiensi penyimpanan dan pengolahan data. Beberapa metode yang umum digunakan pada *data reduction* antara lain pengurangan atribut data dan pengelompokan data.

2.6 *Natural Language Processing*

Natural Language Processing (NLP) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) dan linguistik yang berfokus pada pengembangan metode dan algoritma untuk memungkinkan komputer memproses bahasa alami manusia secara komputasional [29], [30]. Bahasa alami yang digunakan manusia bersifat kaya makna, kontekstual, dan sering kali ambigu, sehingga menimbulkan tantangan ketika direpresentasikan dalam sistem komputasi yang pada dasarnya bekerja secara formal dan terstruktur [30]. Oleh karena itu, NLP bertujuan untuk menjembatani kesenjangan antara cara manusia memahami bahasa dengan cara komputer memproses data berbasis simbol dan aturan formal [29].

Melalui pemanfaatan teknik komputasional, statistik, dan *machine learning*, NLP memungkinkan sistem komputer untuk melakukan berbagai tugas pengolahan bahasa, seperti memahami isi teks, mengekstraksi informasi, mengklasifikasikan dokumen, menerjemahkan bahasa, hingga menghasilkan teks baru yang relevan dan bermakna. Kemampuan tersebut menjadikan NLP sebagai komponen penting dalam berbagai aplikasi modern, termasuk sistem pencarian informasi, sistem rekomendasi, asisten virtual, serta analisis teks dalam skala besar [30], [31].

Lingkup NLP umumnya dibagi ke dalam dua ranah utama, yaitu *Natural Language Understanding* (NLU) dan *Natural Language Generation* (NLG). NLU berfokus pada kemampuan mesin untuk menganalisis dan memahami makna, struktur, serta konteks dari bahasa alami yang diberikan sebagai masukan (*input*). Proses ini mencakup identifikasi informasi linguistik dan semantik yang terkandung dalam teks, sehingga sistem dapat merepresentasikan makna bahasa dalam bentuk yang dapat diproses secara komputasional. Sebaliknya, NLG berkaitan dengan kemampuan mesin untuk menghasilkan bahasa alami yang koheren dan dapat dipahami manusia. NLG memungkinkan sistem untuk menyampaikan informasi, kesimpulan, atau respons dalam bentuk teks yang menyerupai bahasa manusia, sehingga interaksi antara pengguna dan sistem menjadi lebih mudah dipahami [31].

2.6.1 Tingkatan Analisis dalam NLP

Pemrosesan bahasa alami bukanlah sebuah proses tunggal, melainkan terdiri atas serangkaian tingkatan analisis linguistik yang terstruktur. Setiap tingkatan mencerminkan kedalaman pemahaman sistem terhadap bahasa, mulai dari unit linguistik paling dasar hingga pemaknaan yang bergantung pada konteks penggunaan bahasa secara lebih luas.

Secara umum, tingkatan analisis dalam NLP dapat dibagi menjadi tujuh tingkatan utama sebagai berikut [31]:

1. Fonologi (*Phonology*)

Tingkatan ini berkaitan dengan interpretasi bunyi bahasa (*speech*) sebagai unit dasar komunikasi. Dalam pemrosesan berbasis teks, analisis ini umumnya direpresentasikan melalui pengenalan karakter atau fonem yang membentuk fondasi input sebelum diproses ke tahap lanjut.

2. Morfologi (*Morphology*)

Tingkatan ini berfokus pada struktur internal kata dan pembentukannya melalui unit terkecil bermakna (*morpheme*), yang berperan penting dalam mengidentifikasi kata dasar (*lemma*) serta pengaruh imbuhan terhadap perubahan fungsi dan makna kata.

3. Leksikal (*Lexical*)

Tingkatan ini memandang kata sebagai unit tunggal yang mandiri secara makna. Proses ini mencakup pembagian teks menjadi unit kecil (*tokenization*) dan penentuan kelas kata (*part-of-speech tagging*).

4. Sintaksis (*Syntactic*)

Tingkatan ini mengatur aturan penggabungan kata untuk membentuk kalimat yang valid secara tata bahasa, yang melibatkan proses parsing untuk menentukan hubungan struktural antar kata, seperti identifikasi subjek, predikat, dan objek, guna memahami logika dasar sebuah pernyataan.

5. Semantik (*Semantic*)

Tingkatan ini mengeksplorasi makna literal dari kata dan kalimat. Proses ini memetakan hubungan makna antar kata agar sistem mampu mengenali sinonim dan keterkaitan konsep, sehingga mesin tidak hanya melakukan pencocokan karakter, melainkan pencocokan makna.

6. Wacana (*Discourse*)

Tingkatan ini berfokus pada hubungan konteks antar kalimat dalam satu kesatuan teks utuh. Fokus utamanya adalah menjaga kesinambungan informasi, mengidentifikasi referensi kata ganti agar interpretasi makna antar paragraf tetap konsisten.

7. Pragmatik (*Pragmatic*)

Tingkatan pragmatik merupakan tingkat analisis tertinggi yang menafsirkan makna bahasa berdasarkan konteks penggunaan dan intensi pengguna. Tahap ini memungkinkan sistem menangkap makna implisit, tujuan komunikasi, serta nuansa bahasa yang tidak dinyatakan secara eksplisit.

Pembagian tingkatan analisis ini menunjukkan bahwa pemahaman bahasa oleh mesin memerlukan proses yang kompleks dan bertingkat. Dalam perkembangannya, berbagai pendekatan NLP berupaya mengintegrasikan beberapa tingkatan analisis ini untuk menghasilkan sistem yang mampu melakukan pemrosesan dan penalaran bahasa dengan tingkat akurasi yang lebih baik.

2.6.2 Sejarah dan Pendekatan dalam NLP

Evolusi metode dalam pemrosesan bahasa alami mencerminkan transisi dari sistem kaku berbasis logika manusia menuju arsitektur saraf yang fleksibel. Secara umum, evolusi metode NLP dapat dikelompokkan ke dalam tiga pendekatan utama, yaitu [30]:

1. Sistem Berbasis Aturan (*Rule-based Systems*)

Pendekatan awal ini mengandalkan kumpulan aturan linguistik yang dirancang secara manual oleh pakar bahasa. Mekanisme kerjanya menggunakan algoritma deterministik untuk memproses teks berdasarkan pola tata bahasa dan kamus leksikal yang kaku. Meskipun memberikan kontrol penuh terhadap logika sistem, pendekatan ini memiliki

keterbatasan dalam menangani ambiguitas bahasa dan memerlukan upaya manual yang besar untuk memperbarui basis pengetahuan sistem.

2. NLP Statistik (*Statistical NLP*)

Pendekatan statistik mulai mendominasi seiring dengan meningkatnya ketersediaan korpus data yang besar. Metode ini menggunakan teori probabilitas untuk mempelajari pola bahasa secara otomatis dari data, di mana model seperti *Hidden Markov Models* (HMM) atau representasi vektor seperti TF-IDF digunakan untuk menganalisis teks. Keunggulannya terletak pada kemampuan menangani variasi bahasa yang lebih luas dibandingkan sistem berbasis aturan, namun masih memiliki kelemahan dalam menangkap hubungan makna jangka panjang (*long-term dependencies*) dalam sebuah kalimat.

3. Pembelajaran Mendalam (*Deep Learning*)

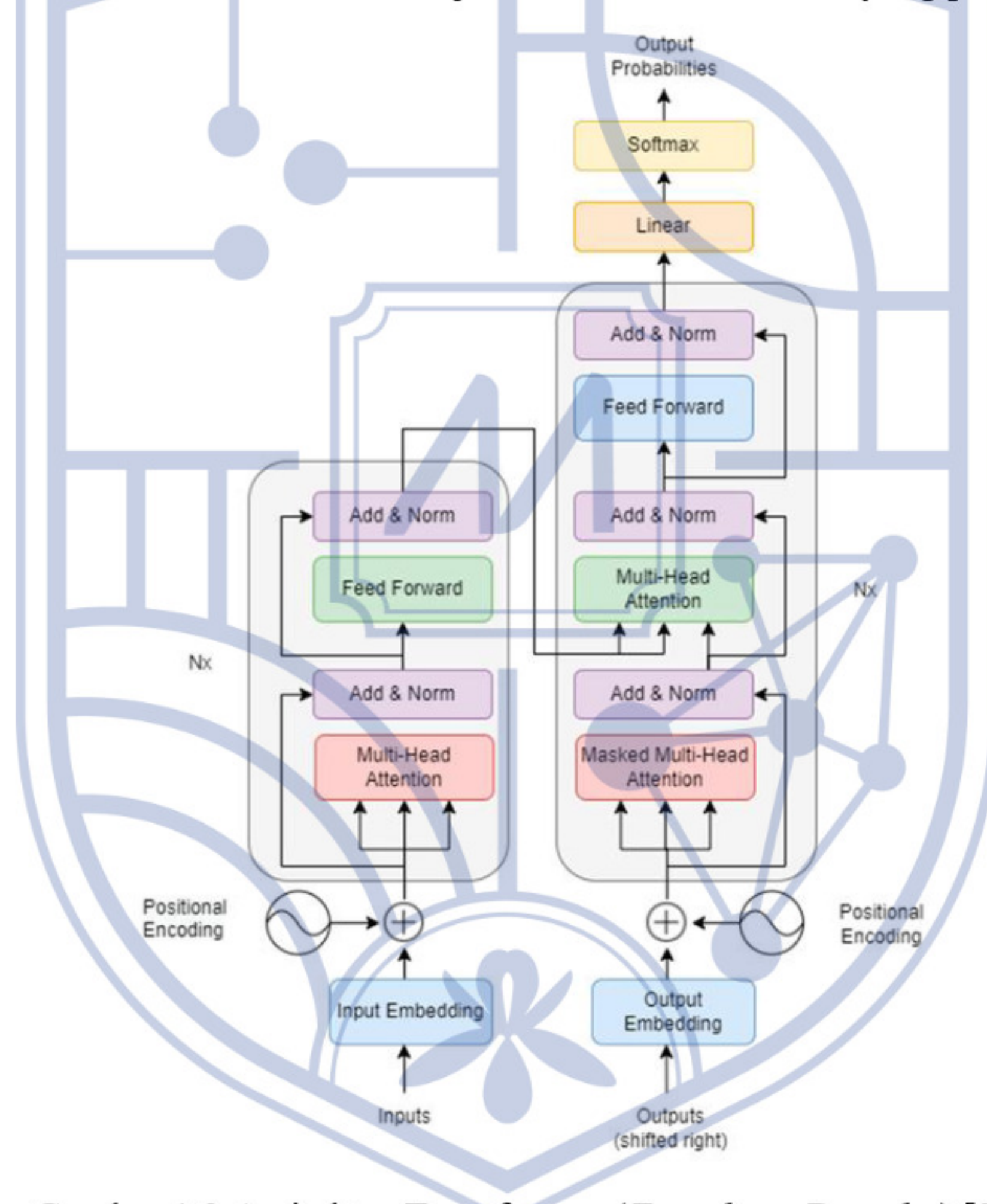
Fase ini merupakan standar terkini yang menggunakan jaringan saraf tiruan (*neural networks*) untuk mempelajari representasi bahasa tanpa ekstraksi fitur manual. Pendekatan ini memungkinkan model untuk memahami konteks yang kompleks dan hubungan semantik antar kata melalui lapisan-lapisan saraf. Perkembangan paling krusial dalam fase ini adalah munculnya arsitektur Transformer, yang mendasari model seperti BERT. Arsitektur ini menggunakan mekanisme *attention* untuk memproses informasi secara dua arah (*bidirectional*), sehingga menghasilkan pemahaman teks yang jauh lebih akurat dibandingkan metode statistik.

Pergeseran menuju pendekatan *Deep Learning* memungkinkan sistem untuk melakukan pemrosesan pada tingkat semantik. Hal ini menjadi alasan utama pemilihan metode berbasis arsitektur saraf dalam menangani tugas-tugas NLP modern.

2.7 Arsitektur Transformer

Transformer merupakan inovasi dalam bidang NLP yang mengubah paradigma pemodelan bahasa dari pendekatan sekuensial menjadi pendekatan berbasis *self-attention* [30]. Sebelum diperkenalkannya arsitektur ini, pemrosesan data teks umumnya mengandalkan model *Recurrent Neural Networks* (RNN) dan *Long Short-Term Memory* (LSTM). Meskipun kedua model tersebut mampu memproses data sekuensial, pendekatan RNN dan LSTM masih memiliki keterbatasan. Model RNN mengalami kesulitan dalam mempertahankan informasi jangka panjang (*long-term dependencies*) akibat masalah *vanishing gradient*. Sementara itu, model LSTM memiliki beban komputasi yang tinggi karena proses analisis kata yang dilakukan satu per satu secara berurutan [31].

Sebagai solusi atas kendala tersebut, Transformer dirancang untuk memproses keseluruhan urutan kata dalam sebuah kalimat secara paralel, sehingga memungkinkan model menangkap dependensi antar kata tanpa terhambat oleh jarak posisinya dalam kalimat [32]. Keunggulan utama arsitektur ini terletak pada mekanisme *self-attention*, yang berfungsi mengevaluasi tingkat relevansi setiap kata terhadap kata lainnya dalam konteks yang sama guna membangun representasi makna yang lebih presisi [30]. Arsitektur Transformer terdiri dari dua bagian pembentuk, yaitu *Encoder* yang bertugas menyandikan teks masukan menjadi representasi vektor, serta *Decoder* yang bertugas mengonversi representasi vektor tersebut kembali menjadi urutan teks keluaran yang padu [32].



Gambar 1.3 Arsitektur Transformer (*Encoder – Decoder*) [33]

Berdasarkan struktur pada Gambar 2.3, berikut adalah komponen-komponen krusial yang menyusun arsitektur *Transformer* [32]:

1. *Multi-Head Attention*

Merupakan komponen inti yang bekerja dengan memproyeksikan input ke dalam tiga vektor representasi, yaitu *Query* (Q), *Key* (K), dan *Value* (V). Mekanisme ini memungkinkan model untuk memusatkan perhatian pada bagian-bagian kalimat yang

berbeda secara simultan guna menangkap nuansa makna yang kompleks, seperti hubungan teknis antar kata dalam deskripsi pekerjaan.

2. *Positional Encoding*

Karena Transformer memproses seluruh kata sekaligus secara paralel, model memerlukan mekanisme ini untuk menyisipkan informasi mengenai urutan posisi setiap kata. Hal ini memastikan model tetap mampu memahami struktur tata bahasa dan urutan kata yang benar meskipun tidak diproses secara sekuensial.

3. *Feed-Forward Networks*

Setelah mekanisme atensi selesai memetakan hubungan antar kata, lapisan ini bertugas memproses setiap representasi kata secara independen untuk memperdalam fitur semantiknya sebelum diteruskan ke lapisan berikutnya.

4. *Add & Norm*

Komponen ini berfungsi untuk menjaga stabilitas aliran informasi melalui jaringan yang dalam. Residual connection memastikan informasi asli tidak terdegradasi, sementara layer normalization menstabilkan nilai numerik vektor untuk mempercepat proses pelatihan model.

2.8 *Bidirectional Encoder Representations from Transformers (BERT)*

BERT merupakan model bahasa yang diperkenalkan oleh Google pada tahun 2018 untuk meningkatkan kemampuan komputer dalam memahami konteks dan makna bahasa alami. Sebelum munculnya BERT, representasi teks umumnya menggunakan model *embedding* statis seperti Word2Vec, GloVe, dan FastText yang menghasilkan representasi vektor tetap (*static embeddings*) untuk setiap kata tanpa mempertimbangkan konteks kemunculannya dalam kalimat. Untuk mengatasi permasalahan tersebut, BERT dikembangkan untuk menghasilkan representasi vektor kontekstual (*contextualized embeddings*), sehingga representasi suatu kata dapat berubah sesuai dengan konteks kemunculannya dalam kalimat [34], [35].

2.8.1 **Arsitektur BERT**

BERT dikembangkan di atas arsitektur Transformer, namun hanya memanfaatkan lapisan *encoder* sebagai komponen utamanya. BERT hanya memanfaatkan lapisan *encoder* karena dirancang untuk menangani tugas-tugas NLU yang berfokus pada pemahaman

makna, konteks, dan relasi semantik dalam teks, bukan pada NLG yang berfokus pada pembentukan teks secara otomatis [36].

Salah satu karakteristik utama BERT adalah penerapan mekanisme *bidirectional self-attention* yang memungkinkan model memahami konteks kata dari sisi kiri maupun kanan suatu kalimat secara bersamaan. Pendekatan ini membuat BERT mampu memahami hubungan antar kata dengan lebih akurat, termasuk pada kalimat yang kompleks [35]. Selain itu, BERT juga menggunakan tokenisasi berbasis subkata yang dikenal sebagai WordPiece untuk menangani kosakata yang tidak dikenal (*out-of-vocabulary*) serta memahami variasi bentuk kata dengan lebih baik [34].

2.8.2 Proses Pelatihan BERT

Pelatihan model BERT dilakukan melalui pendekatan *transfer learning*, yaitu suatu paradigma pelatihan yang memungkinkan pengetahuan linguistik yang telah dipelajari dari data berskala besar digunakan kembali untuk menyelesaikan tugas lain yang lebih spesifik. Pendekatan ini membuat BERT tidak perlu dilatih dari awal untuk setiap tugas baru [34], [36]. Proses pelatihan BERT secara umum terdiri dari dua tahap utama, yaitu:

1. Tahap *Pre-training*

Pada tahap ini, model BERT dilatih menggunakan korpus teks berskala besar untuk mempelajari pola bahasa, struktur kalimat, serta hubungan antar kata melalui pendekatan *Masked Language Modeling* (MLM) dan *Next Sentence Prediction* (NSP). Tahap ini menghasilkan model dasar (*pre-trained model*) yang telah memiliki pemahaman linguistik umum dan dapat dimodifikasi untuk tugas yang lebih spesifik [37].

2. Tahap *Fine-tuning*

Setelah tahap *pre-training* selesai, model BERT dapat diadaptasi untuk berbagai tugas yang lebih spesifik. Tahap *fine-tuning* lebih efisien dibandingkan tahap *pre-training* karena hanya memerlukan penyesuaian parameter menggunakan *dataset* yang lebih kecil. Tahap ini memungkinkan BERT mencapai performa yang baik tanpa harus melakukan pelatihan model dari awal [36].

2.8.3 Keterbatasan BERT

Meskipun BERT mampu menghasilkan representasi kontekstual yang baik, model ini dirancang untuk memahami konteks kata melalui representasi token, bukan untuk menghasilkan representasi *embedding* pada tingkat kalimat (*sentence embedding*).

Akibatnya, representasi yang dihasilkan lebih optimal untuk memahami hubungan antar kata dibandingkan untuk merepresentasikan keseluruhan makna kalimat dalam satu representasi vektor [38].

Pada tugas perbandingan antar kalimat, BERT standar umumnya menggunakan pendekatan *cross-encoder*, yaitu dua kalimat dimasukkan secara bersamaan ke dalam model dalam satu proses *encoding*. Meskipun pendekatan ini memungkinkan model memahami hubungan antar kalimat secara mendalam, namun memiliki kelemahan utama pada aspek efisiensi komputasi. Setiap pasangan kalimat harus diproses ulang secara bersamaan setiap kali dilakukan perbandingan, sehingga kebutuhan komputasi meningkat secara signifikan seiring bertambahnya jumlah data. Kondisi ini menyebabkan waktu pemrosesan menjadi lebih lambat dan penggunaan sumber daya komputasi menjadi lebih besar, terutama untuk tugas *semantic similarity*, *semantic search*, dan *clustering* dalam skala besar [38].

2.8.4 Sentence-Bert (SBERT)

Untuk mengatasi keterbatasan komputasi penggunaan BERT dalam tugas perbandingan kalimat, Reimers dan Gurevych memperkenalkan Sentence-Bert (SBERT) pada tahun 2019. SBERT dirancang untuk menghasilkan representasi *embedding* pada tingkat kalimat (*sentence embedding*), sehingga lebih efektif untuk menyelesaikan tugas *Semantic Text Similarity (STS)*, *semantic search*, *clustering*, dan *classification* [38].

SBERT tidak mengubah kemampuan dasar BERT dalam memahami bahasa, tetapi mengubah cara BERT digunakan untuk menghasilkan dan membandingkan representasi kalimat. Secara umum, pengembangan SBERT dilakukan melalui beberapa perubahan utama sebagai berikut:

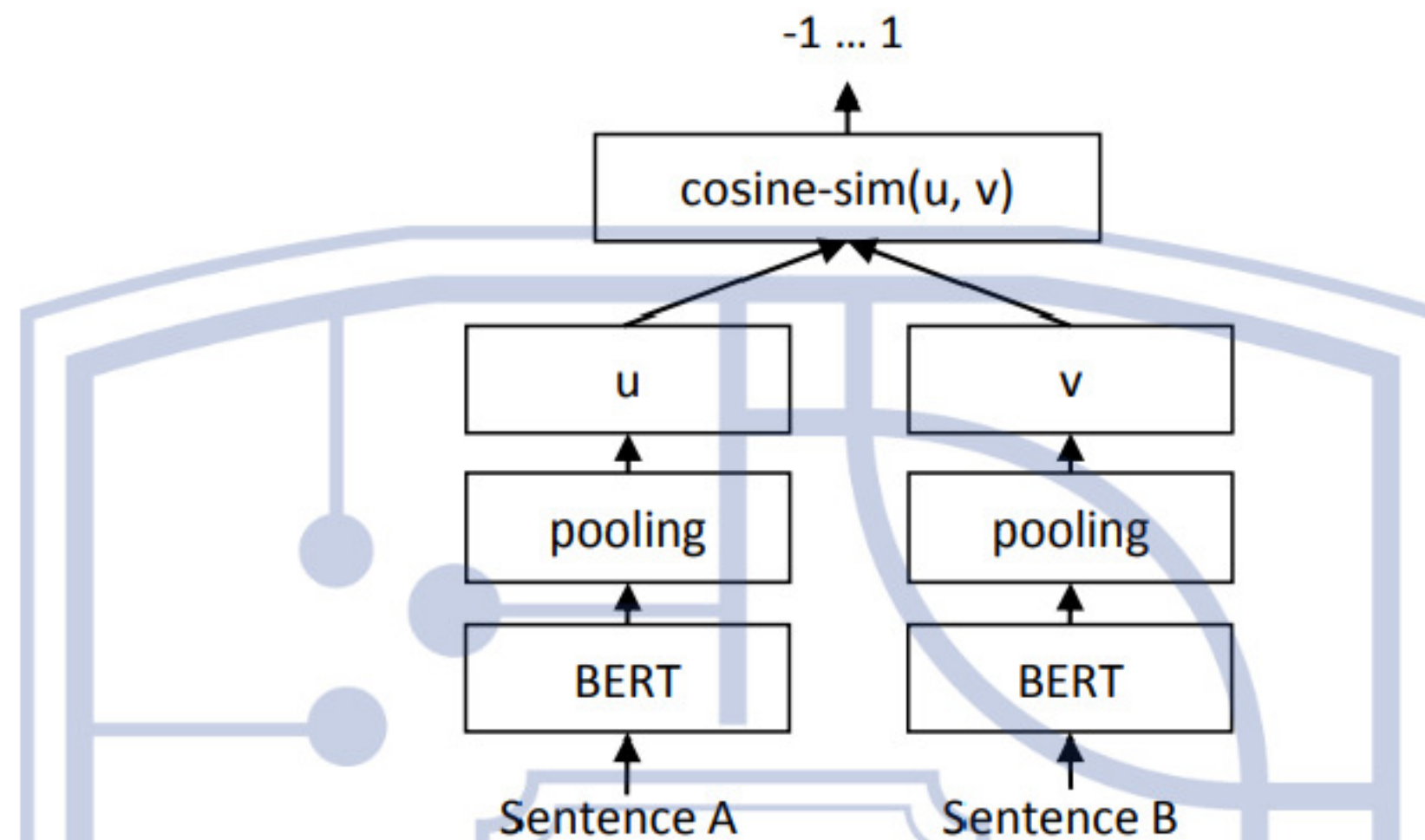
1. Penerapan *Siamese Network* dan *Triplet Network*

SBERT memodifikasi pendekatan *cross-encoder* pada BERT menjadi arsitektur *Siamese Network* dan *Triplet Network* [38]. Pada pendekatan ini, setiap kalimat diproses secara terpisah menggunakan *encoder* dengan bobot yang sama (*shared weights*), sehingga masing-masing kalimat menghasilkan representasi *embedding* secara independen [39]. Hal ini yang memungkinkan *embedding* kalimat untuk disimpan terlebih dahulu dan dapat digunakan kembali pada proses perbandingan teks lainnya, tanpa perlu melakukan proses *encoding* ulang terhadap seluruh pasangan kalimat [38].

2. Penambahan *Pooling layer*

Selain perubahan pada struktur pemrosesan, SBERT juga menambahkan *pooling layer* pada keluaran Transformer untuk mengubah *token embedding* menjadi representasi

embedding pada tingkat kalimat (*sentence embedding*). Proses ini berfungsi untuk merangkum seluruh informasi token dalam suatu kalimat menjadi satu vektor representasi tetap. Melalui standarisasi dimensi vektor tersebut, tingkat kemiripan antar-kalimat dapat langsung dihitung secara efisien menggunakan metrik seperti *cosine similarity*, seperti yang ditunjukkan pada Gambar 2.4 [38].



Gambar 1.4 Arsitektur SBERT untuk menghitung tingkat kemiripan antar kalimat

Meskipun pada penelitian awalnya arsitektur SBERT ini diujicobakan secara spesifik pada model BERT dan RoBERTa [38], implementasi SBERT saat ini telah mencakup lebih banyak model. Berdasarkan dokumentasi Sentence-Transformers, arsitektur SBERT kini dapat diimplementasikan menggunakan berbagai jenis model *pre-trained* berbasis Transformer lainnya, seperti MiniLM atau MPNet. Variasi model siap pakai (*pre-trained models*) ini memiliki karakteristik dimensi vektor, performa *semantic search*, serta kecepatan pemrosesan *query* yang berbeda-beda seperti yang ditunjukkan pada Tabel 2.2 [40].

Tabel 1.2 Perbandingan Spesifikasi Model SBERT

Nama Model	Dimensi Vektor	Performa <i>Semantic Search</i>	Kecepatan (<i>Queries</i> / detik)
<i>all-MiniLM-L12-v2</i>	384	50.82	7500
<i>all-mpnet-base-v2</i>	768	57.02	2800
<i>paraphrase-multilingual-MiniLM-L12-v2</i>	384	39.19	7500
<i>paraphrase-multilingual-mpnet-base-v2</i>	768	41.68	2500

Berdasarkan data perbandingan model SBERT pada Tabel 2.2, model seri *all*- memiliki skor performa *semantic search* yang lebih tinggi dibandingkan model *multilingual*. Hal ini dikarenakan model *all*- dirancang sebagai model serbaguna (*general purpose*) yang dilatih menggunakan lebih dari 1 miliar pasang data teks Bahasa Inggris. Sementara itu, model *multilingual* dikembangkan untuk menangani pemrosesan teks lintas bahasa (*cross-lingual*), sehingga lebih sesuai digunakan pada data teks multibahasa atau bilingual. Selain itu, perbedaan ukuran dimensi vektor juga memengaruhi kecepatan pemrosesan *query*. Model SBERT berbasis MPNet dengan dimensi vektor 768 memiliki kemampuan pemahaman semantik yang lebih baik, namun dengan waktu pemrosesan yang lebih lambat. Sebaliknya, model MiniLM yang memiliki dimensi vektor lebih kecil memiliki kecepatan pemrosesan *query* yang lebih tinggi [40].

2.9 MiniLM

MiniLM merupakan model berbasis Transformer yang pertama kali dikembangkan oleh Microsoft Research untuk mengatasi tingginya kebutuhan komputasi pada model bahasa besar seperti BERT. Model Transformer umumnya terdiri dari banyak layer dan parameter, sehingga membutuhkan sumber daya komputasi yang besar serta memiliki waktu inferensi yang relatif lambat. Tujuan pengembangan model MiniLM adalah untuk menghasilkan model bahasa yang lebih ringan dan efisien dengan tetap mempertahankan kualitas representasi dari model Transformer berukuran besar [41].

Model ini dikembangkan melalui teknik *knowledge distillation*, yaitu proses transfer pengetahuan dari model besar (*teacher model*) ke model yang lebih kecil (*student model*). Pada pendekatan ini, *student model* tidak hanya mempelajari keluaran akhir dari *teacher model*, tetapi juga mempelajari mekanisme *self-attention* dari *teacher model*, sehingga kemampuan representasi bahasa dapat tetap dipertahankan meskipun dengan jumlah parameter yang lebih sedikit [41].

Berdasarkan hasil penelitian pada tugas *semantic search*, model *all-mpnet-base-v2* menunjukkan performa tertinggi dalam pencocokan dokumen berdasarkan kesamaan makna. Namun, model berbasis MiniLM, khususnya mode *all-MiniLM-L6-v2* dan *all-MiniLM-L12-v2*, menunjukkan performa yang mendekati MPNet. Selain itu, kedua varian MiniLM tersebut memiliki keunggulan dalam hal efisiensi komputasi dan waktu pemrosesan [42]. Oleh karena itu, MiniLM cocok digunakan pada sistem yang membutuhkan pemrosesan cepat dan skalabilitas tinggi, dengan tetap mempertahankan kualitas representasi.

2.10 Cosine Similarity

Cosine Similarity merupakan teknik yang digunakan untuk mengukur tingkat kemiripan antara dua vektor dengan menghitung nilai kosinus sudut yang terbentuk di antara keduanya. Berbeda dengan perhitungan jarak biasa, *cosine similarity* menilai kemiripan berdasarkan arah atau orientasi vektor dalam ruang multidimensi. Semakin kecil sudut yang terbentuk antara dua vektor, semakin tinggi nilai kemiripannya. Sebaliknya, semakin besar sudut, semakin rendah Tingkat kemiripan kedua vektor tersebut [43].

Secara matematis, nilai *cosine similarity* didapatkan dengan membagi hasil perkalian (*dot product*) kedua vektor dengan hasil perkalian panjang (*Euclidean norm*) masing-masing vektor. Perhitungan ini dirumuskan pada Persamaan 1 sebagai berikut.

$$\text{cosine similarity} = \frac{A \cdot B}{\|A\| \|B\|} \quad (1)$$

Keterangan:

A dan B = dua vektor yang akan dibandingkan.

$A \cdot B$ = hasil *dot product* antara vektor A dan B .

$\|A\|$ dan $\|B\|$ = panjang (norma Euclidean) dari masing-masing vektor.

Nilai *cosine similarity* berada pada rentang -1 hingga 1. Nilai 1 menunjukkan bahwa kedua vektor memiliki arah yang sama atau identik. Nilai 0 menunjukkan bahwa kedua vektor saling tegak lurus (*orthogonal*), yang berarti tidak ada kemiripan di antara keduanya. Nilai -1 menunjukkan bahwa kedua vektor memiliki arah yang berlawanan, sehingga dianggap sangat tidak mirip [44].

Dalam konteks dokumen, setiap dokumen direpresentasikan sebagai vektor dalam ruang berdimensi banyak (*multidimensional space*). Dengan demikian, dokumen yang memiliki arah vektor hampir sama dianggap sangat mirip, sedangkan dokumen dengan arah vektor berbeda dianggap kurang mirip. Pendekatan ini memungkinkan perhitungan kemiripan antar dokumen tanpa memperhatikan ukuran dokumen, sehingga dokumen pendek dan panjang tetap bisa dibandingkan secara adil [43].

2.11 Metrik Evaluasi

Evaluasi merupakan tahapan penting dalam pengembangan sistem yang bertujuan untuk menilai efektivitas dan performa sistem secara terukur. Proses evaluasi umumnya dilakukan dalam lingkungan yang terkontrol agar pengujian dapat dilakukan secara konsisten dan dapat direplikasi. Penerapan evaluasi menjadi sangat krusial karena digunakan

untuk menilai sejauh mana hasil yang diberikan sistem sesuai dengan kebutuhan atau preferensi pengguna sehingga performa sistem dapat diukur secara lebih objektif [45].

2.11.1 *Precision@k*

Precision@k merupakan salah satu metrik evaluasi yang digunakan untuk mengukur kualitas sistem rekomendasi. Metrik ini menghitung proporsi item yang relevan dari sejumlah rekomendasi teratas yang dihasilkan oleh sistem. Nilai *Precision@k* menunjukkan tingkat ketepatan sistem dalam merekomendasikan item yang relevan. Semakin tinggi nilai *Precision@k*, semakin baik pula kemampuan sistem dalam menghasilkan rekomendasi yang relevan. Rumus *Precision@k* didefinisikan pada Persamaan 2 sebagai berikut [46].

$$Precision@k = \frac{|Rel(u) \cap R@k|}{|R@k|} \quad (2)$$

Keterangan:

Precision@k = nilai *precision* pada *k* rekomendasi teratas.

Rel(u) = item yang relevan bagi pengguna.

R@k = jumlah *k* item yang direkomendasikan oleh sistem.

2.11.2 *Average Precision (AP@k)*

Average Precision merupakan metrik evaluasi yang digunakan untuk mengukur kualitas peringkat hasil rekomendasi dengan memperhatikan mempertimbangkan tidak hanya jumlah item yang relevan, tetapi juga posisi kemunculan item relevan dalam daftar rekomendasi. *AP@k* dihitung dengan merata-ratakan nilai *precision* pada setiap posisi ketika ditemukan item yang relevan. Dengan demikian, item relevan yang muncul pada posisi lebih awal akan memberikan kontribusi yang lebih besar terhadap nilai *AP@k*. Semakin tinggi nilai , semakin baik kualitas urutan rekomendasi yang dihasilkan. Rumus *AP@k* didefinisikan pada Persamaan 3 sebagai berikut [46].

$$AP@k = \frac{1}{k} \sum_{i=1}^k P@i \times rel(i) \quad (3)$$

Keterangan:

AP@k = nilai *Average Precision* pada *k* rekomendasi teratas.

k = jumlah item pada daftar rekomendasi.

P@i = nilai *Precision* pada posisi ke-*i*.

$rel(i)$ = indikator relevansi pada posisi ke- i , bernilai 1 apabila item relevan dan 0 apabila tidak relevan.

2.11.3 Mean Average Precision (MAP@k)

Mean Average Precision merupakan pengembangan dari $AP@k$ yang digunakan untuk mengevaluasi performa sistem secara keseluruhan. $MAP@k$ diperoleh dengan menghitung rata-rata nilai $AP@k$ dari seluruh data yang digunakan dalam proses evaluasi. Metrik ini memberikan gambaran mengenai kemampuan sistem dalam menghasilkan daftar rekomendasi yang menempatkan item relevan pada posisi yang lebih tinggi secara konsisten. Semakin tinggi nilai $MAP@k$, semakin baik performa sistem rekomendasi dalam menghasilkan urutan rekomendasi yang relevan. Rumus $MAP@k$ didefinisikan pada Persamaan 4 sebagai berikut [47].

$$MAP@k = \frac{1}{|U|} \sum_{u \in U} AP@k_u \quad (3)$$

Keterangan:

$MAP@k$ = nilai *Mean Average Precision* pada k rekomendasi teratas.

U = jumlah data yang digunakan dalam proses evaluasi.

$AP@k_u$ = nilai *Average Precision* untuk setiap data u .

2.12 Black-Box Testing

Black-box testing atau sering disebut *behavioral testing* merupakan metode pengujian perangkat lunak yang berfokus pada pemenuhan kebutuhan fungsional sistem. Pengujian ini memungkinkan penyusunan serangkaian skenario kondisi masukan (*input*) yang bertujuan untuk mengevaluasi seluruh fungsi program secara menyeluruh dengan cara mengamati keluaran (*output*) yang dihasilkan. Dalam penerapannya, *black-box testing* digunakan untuk mengidentifikasi berbagai kategori kesalahan pada sistem, seperti fungsi yang tidak sesuai, kesalahan pada antarmuka, kesalahan dalam pengolahan atau akses data, hingga kesalahan pada proses awal maupun akhir eksekusi sistem [47], [48].

Salah satu keunggulan *black-box testing* adalah pengujian dapat dilakukan tanpa memerlukan pengetahuan mengenai kode program atau arsitektur internal perangkat lunak, sehingga pengujian sistem sepenuhnya diambil dari sudut pandang pengguna. Salah satu teknik yang digunakan dalam *black-box testing* adalah *use case testing*, yaitu teknik pengujian yang memastikan kesesuaian antara kebutuhan sistem dan kebutuhan pengguna

melalui skenario yang menggambarkan bagaimana pengguna berinteraksi dengan sistem untuk mencapai tujuan tertentu. Dalam *use case testing*, langkah awalnya adalah menentukan *use case* utama yang menggambarkan kebutuhan sistem. Setelah itu, setiap *use case* dijelaskan alurnya, mulai dari *input* yang diberikan, proses yang terjadi di sistem, sampai hasil yang diharapkan. Dari alur ini kemudian dibuat *test case* untuk mengecek apakah sistem berjalan sesuai dalam berbagai kondisi penggunaan. Pengujian ini bisa dilakukan secara manual atau dengan alat bantu, dan juga bisa membantu menemukan kesalahan yang tidak terlihat dari pengujian biasa [48].

2.13 Penelitian Terkait

Bagian ini menyajikan penelitian-penelitian terdahulu yang relevan dengan pengembangan sistem pencarian dan rekomendasi berbasis semantik serta pemanfaatan metode kecerdasan buatan. Penelitian yang ditinjau tidak hanya terbatas pada topik terkait pencarian lowongan kerja, tetapi juga mencakup penelitian dengan topik berbeda yang menerapkan metode serupa dalam menghasilkan rekomendasi. Tinjauan ini bertujuan untuk memberikan gambaran mengenai metode yang telah digunakan serta konteks penerapan sistem rekomendasi pada berbagai permasalahan. Ringkasan penelitian terkait disajikan dalam Tabel 2.3.

Tabel 1.3 Perbandingan Penelitian Terdahulu

Peneliti (Tahun)	Judul Penelitian	Metode / Algoritma	Intisari dan Hasil Penelitian
Singla & Verma (2024) [49]	An Intelligent Job Recommendation System based on Semantic Embeddings and Machine Learning	Sentence-BERT (SBERT), Random Forest	Penelitian ini melakukan pengembangan sistem rekomendasi lowongan kerja yang memanfaatkan representasi <i>embedding</i> dari data CV kandidat. Tujuannya untuk meningkatkan kecocokan antara profil kandidat dan persyaratan pekerjaan. Hasil penelitian menunjukkan bahwa pendekatan semantik meningkatkan relevansi rekomendasi pekerjaan.

Almalki dkk. (2024) [50]	BERT-based Job Recommendation System Using LinkedIn Dataset	BERT	Penelitian ini melakukan pengembangan sistem rekomendasi pekerjaan berbasis Transformer dengan memanfaatkan BERT untuk memahami hubungan semantik antara deskripsi pekerjaan dan profil pengguna. Tujuannya untuk menghasilkan rekomendasi pekerjaan yang lebih personal dan relevan. Hasil penelitian menunjukkan bahwa pemanfaatan BERT meningkatkan akurasi rekomendasi dibandingkan metode <i>content-based</i> tradisional.
Alsaif dkk. (2022) [51]	NLP-Based Bi-Directional Recommendation System: Towards Recommending Jobs to Job Seekers and Resumes to Recruiters	Named Entity Recognition (NER), Word2Vec, Cosine Similarity	Penelitian ini melakukan pengembangan sistem rekomendasi dua arah (<i>bi-directional</i>) berbasis NLP untuk mencocokkan <i>resume</i> kandidat dan deskripsi pekerjaan. Tujuannya untuk mendukung pencari kerja dalam menemukan lowongan yang sesuai serta membantu rekruter dalam mengidentifikasi kandidat yang relevan. Hasil penelitian menunjukkan bahwa pendekatan NLP berbasis <i>content-based filtering</i> mampu meningkatkan akurasi pencocokan dibandingkan teknik <i>information retrieval</i> tradisional.

Fathuddin dkk. (2025) [11]	Penerapan Sentence-BERT dan Cosine Similarity untuk Pencarian Semantik Dokumen Skripsi dalam Format PDF	Sentence-BERT (SBERT), Cosine Similarity	Penelitian ini melakukan penerapan pencarian semantik pada dokumen skripsi menggunakan S-BERT dan <i>cosine similarity</i> . Tujuannya untuk meningkatkan relevansi hasil pencarian dokumen. Hasil penelitian menunjukkan bahwa pencarian berbasis semantik mampu menghasilkan hasil yang lebih relevan dibandingkan pencarian berbasis kata kunci.
----------------------------	---	--	---

