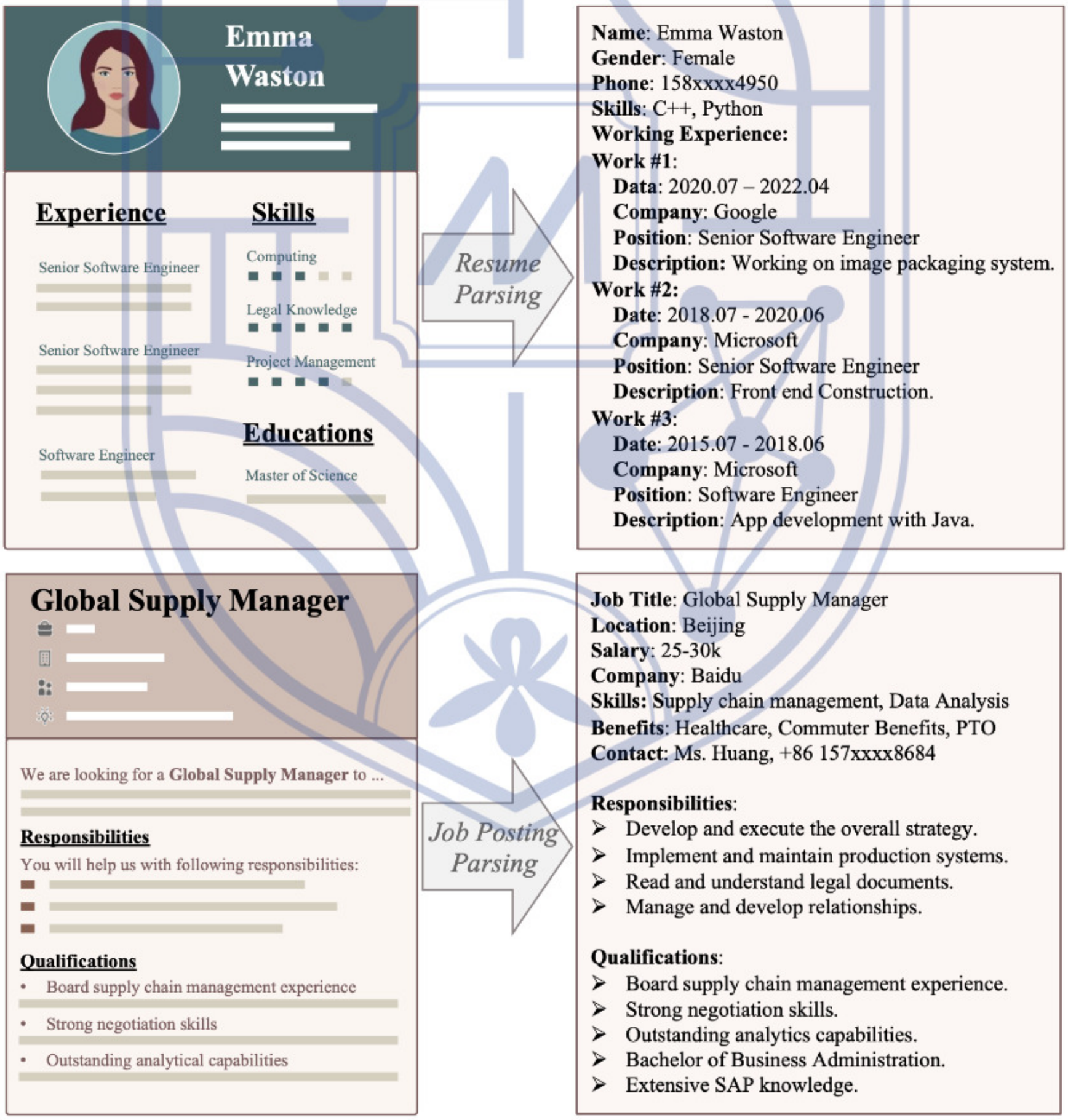
BAB II
KAJIAN LITERATUR

2.1 Resume & *Job Description*

Resume atau *Curriculum Vitae* (CV) adalah dokumen yang menguraikan latar belakang, keterampilan, dan prestasi seseorang, yang memainkan peran penting dalam proses rekrutmen. Contoh format CV tersebut dapat dilihat pada Gambar 2.1. CV berfungsi sebagai alat penting bagi para pencari kerja untuk menunjukkan kualifikasi dan kesesuaian mereka untuk posisi pekerjaan. Sebaliknya, *job description* menjelaskan tanggung jawab dan tugas posisi tersebut kepada kandidat, dan persyaratan pekerjaan, yang menguraikan pengalaman profesional, keterampilan, dan pengetahuan domain yang diharapkan oleh pemberi kerja dari kandidat ideal untuk menjalankan posisi tersebut [11]



Gambar 2.1 *Curriculum Vitae* (CV) [11]

Pencocokan resume terhadap *job description* merupakan proses krusial untuk menentukan sejauh mana kandidat memenuhi kualifikasi pekerjaan secara menyeluruh. Sistem modern tidak hanya mengandalkan pencocokan kata kunci, tetapi juga mempertimbangkan konteks dan semantik dari isi dokumen dengan bantuan model *transformer* [12].

Kesesuaian antara resume dan deskripsi pekerjaan mencerminkan tingkat kecocokan antara karakteristik individu dengan tuntutan pekerjaan. Konsep ini menekankan bahwa kesesuaian orang dengan pekerjaan telah mengidentifikasi dua bentuk kesesuaian yang berbeda, yaitu kesesuaian dengan nilai-nilai (keinginan, tujuan, minat, dan preferensi) dan kesesuaian dengan kemampuan (pengetahuan, keterampilan, dan kemampuan karyawan untuk memenuhi tuntutan pekerjaan) . Dengan demikian, kesesuaian resume terhadap *job description* dapat dipahami sebagai gambaran praktis dari keseimbangan antara kemampuan yang dimiliki pelamar dengan persyaratan dan ekspektasi yang ditetapkan dalam suatu posisi pekerjaan [13].

2.2 Proses Rekrutmen dan Evaluasi Resume

Proses rekrutmen merupakan salah satu fungsi manajemen sumber daya manusia yang strategis untuk menarik dan menjaring calon tenaga kerja berkualifikasi sesuai tuntutan kebutuhan organisasi [14]. Secara umum, alur rekrutmen dimulai dari identifikasi kebutuhan posisi, penyusunan kriteria jabatan (*job description*), publikasi lowongan, hingga pengumpulan berkas administrasi pelamar berupa dokumen resume atau *Curriculum Vitae* (CV). Tahapan seleksi berkas administrasi ini bertindak sebagai gerbang utama yang sangat krusial, sebab efektivitas dan efisiensi waktu pada tahapan wawancara serta pengujian berikutnya sangat bergantung pada kualitas penyaringan awal tersebut [15]

Dalam mengevaluasi dokumen resume, tim perekrut (*Human Resources*) maupun sistem otomatis tidak sekadar melakukan pemeriksaan format, melainkan berfokus pada informasi untuk menghitung skor keterampilan (*skills*), pendidikan (*education*), serta pengalaman [16].

1. Kualifikasi Kompetensi (*Skills*): Komponen ini terbagi menjadi kompetensi teknis (*hard skills*) seperti bahasa pemrograman atau sertifikasi keahlian, dan kompetensi non-teknis (*soft skills*) seperti kemampuan komunikasi dan kepemimpinan. Penguasaan jenis kompetensi ini harus selaras dengan persyaratan minimum lowongan kerja
2. Pengalaman Kerja (*Work Experience*): Rekam jejak profesional kandidat yang umumnya disusun secara kronologis terbalik. Bagian ini ditinjau secara mendalam berdasarkan

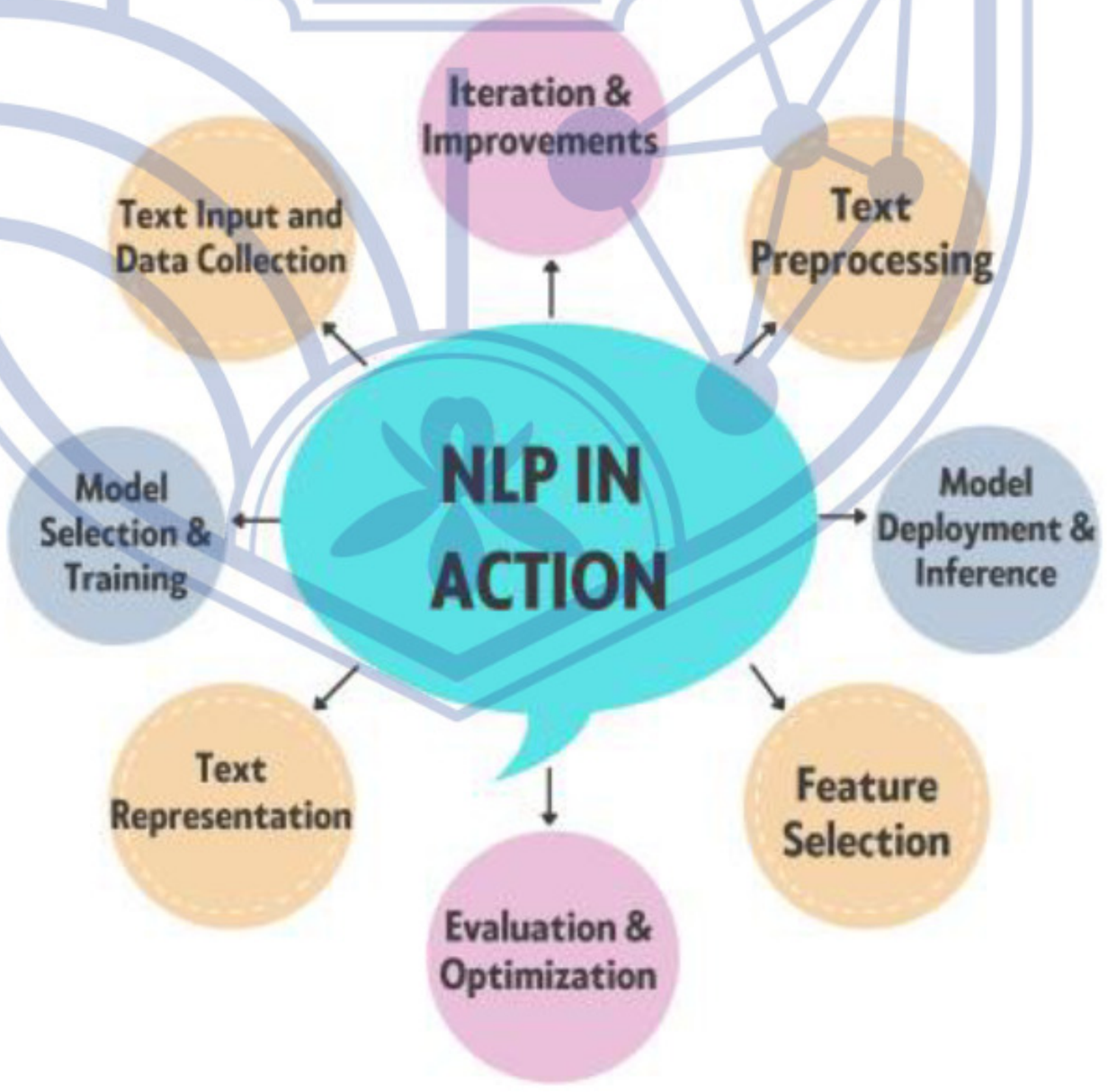
relevansi tanggung jawab, ruang lingkup proyek terdahulu, serta dampak atau metrik pencapaian yang berhasil diraih pada perusahaan sebelumnya

3. Latar Belakang Pendidikan dan Pelatihan (*Education&Training*): Riwayat akademis formal serta pelatihan profesional kedinasan yang memvalidasi bahwa kandidat memiliki fondasi teori teoretis yang kuat pada domain posisi pekerjaan yang dituju [11]

Skema pembobotan evaluasi berdasarkan HR (Human Resource) dari perusahaan Sinar Jasa Powerindo yang menetapkan prioritas bobot 50% untuk keahlian teknis, 30% untuk pengalaman dan 20% untuk pendidikan sangat relevan untuk merepresentasikan standar pemfilteran otomatis dunia kerja saat ini, di mana kualifikasi siap kerja (*plug-and-play*) menjadi penentu kelayakan utama.

2.3 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah cabang dari kecerdasan buatan (AI) yang memungkinkan komputer untuk memahami, menginterpretasikan, dan menghasilkan bahasa manusia secara alami. NLP menggabungkan linguistik komputasi dengan teknik statistik, pembelajaran mesin, dan pembelajaran mendalam untuk memproses teks dan ucapan dalam berbagai bahasa. Tujuan utamanya adalah menjembatani komunikasi antara manusia dan mesin melalui bahasa alami [17].



Gambar 2.2 NLP *Working* [17].

Berdasarkan gambar 2.2, menjelaskan alur kerja umum dalam *Natural Language Processing* (NLP) yang terdiri atas serangkaian tahapan terstruktur untuk mengubah data teks mentah menjadi keluaran yang dapat dianalisis atau diprediksi oleh model kecerdasan buatan. Setiap tahap memiliki fungsi spesifik dan saling berkaitan, sehingga membentuk proses komputasi bahasa yang sistematis dan terukur. Pertama, data teks dikumpulkan dari berbagai sumber seperti dokumen, situs web, dan database, sebagai dasar untuk proses selanjutnya. Setelah itu, teks yang dikumpulkan dibersihkan dan dinormalisasi dengan menghapus elemen yang tidak perlu, serta melakukan tokenisasi, *stemming*, dan *lemmatization* agar siap diolah. Selanjutnya, teks diubah menjadi bentuk numerik menggunakan metode seperti TF-IDF atau *embedding* kata yang lebih modern, sehingga komputer dapat memprosesnya. Pada tahap fitur, hanya elemen penting yang dipilih untuk meningkatkan efisiensi dan mengurangi gangguan pada model. Lalu, model yang sesuai dipilih dan dilatih dengan data yang sudah disiapkan. Setelah model siap, ia diterapkan untuk memproses data baru dan menghasilkan prediksi atau *output*. Kinerja model kemudian dievaluasi menggunakan metrik seperti akurasi dan *F1-score*, kemudian dioptimalkan jika diperlukan. Karena proses ini bersifat berulang, pengembangan dilakukan secara terus-menerus dengan memperbaiki data, teknik, dan model agar hasilnya semakin baik.

Dalam konteks rekrutmen, *Natural Language Processing* (NLP) digunakan untuk menganalisis dan mengekstraksi informasi penting dari dokumen seperti resume dan deskripsi pekerjaan. NLP memungkinkan sistem untuk memahami konten secara kontekstual dan menilai kesesuaian antara resume kandidat dengan persyaratan yang tercantum dalam deskripsi pekerjaan. Penerapan NLP dalam rekrutmen bertujuan untuk mengotomatisasi proses seleksi, meningkatkan efisiensi, dan meminimalkan bias yang mungkin timbul dari evaluasi manual [18].

2.4 *Information Extraction* (IE)

Information Extraction (IE) merupakan domain krusial dalam *Natural Language Processing* (NLP) yang mengubah teks biasa menjadi pengetahuan terstruktur (misalnya, entitas, relasi, dan peristiwa), dan berfungsi sebagai persyaratan dasar untuk berbagai tugas hilir, seperti konstruksi grafik pengetahuan, penalaran pengetahuan, dan menjawab pertanyaan [19]. *Information Extraction* bertujuan untuk mengekstrak informasi terstruktur dari teks tak terstruktur ke dalam format data terstruktur, termasuk tugas-tugas seperti ekstraksi rangkap tiga relasi entitas (RE), pengenalan entitas bernama (NER), ekstraksi

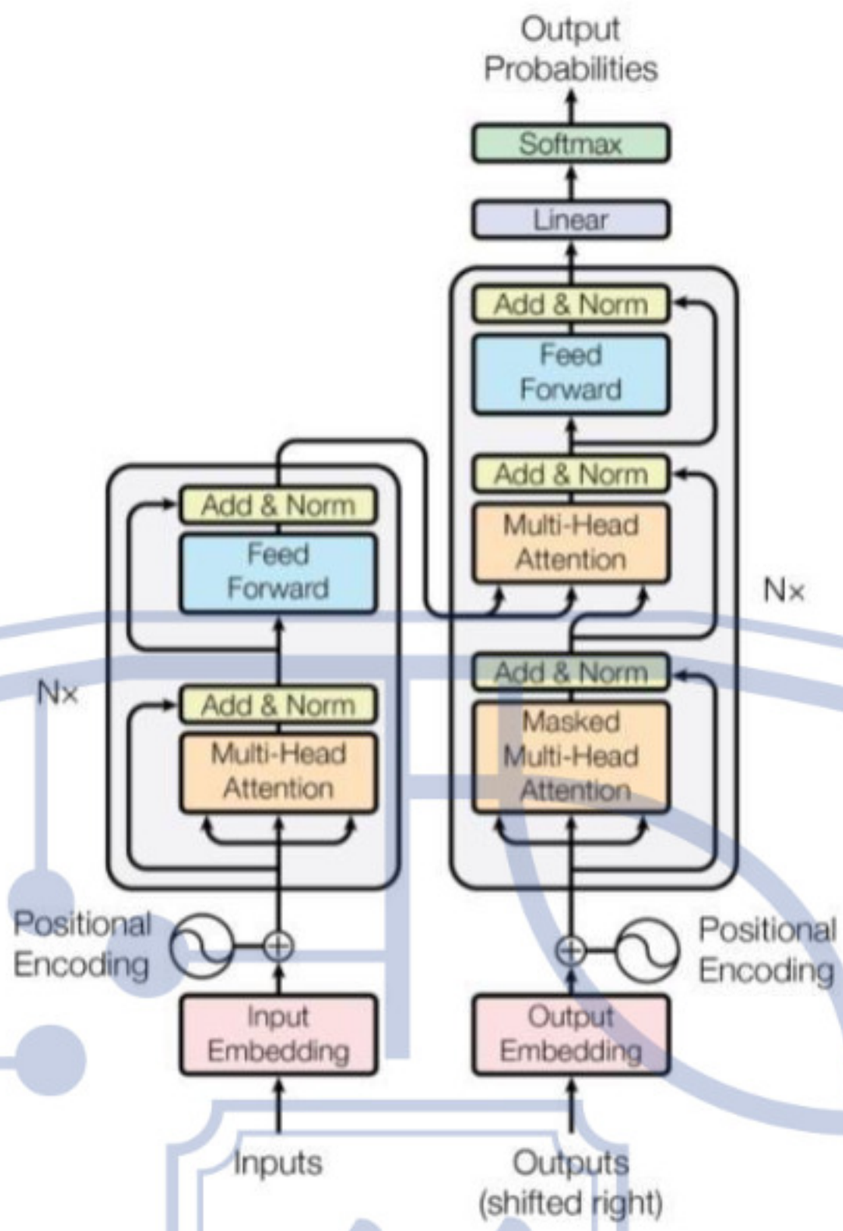
peristiwa (EE). Ini merupakan tugas fundamental dan krusial dalam *Natural Language Processing* (NLP) [20].

Peran IE adalah menemukan kumpulan definisi yang telah ditentukan sebelumnya dalam domain tertentu, mengabaikan informasi lain yang tidak terkait. Domain tersebut terdiri dari korpus teks beserta kebutuhan informasi yang spesifik. Dengan kata lain, dari teks yang tidak terstruktur, IE bertujuan untuk memperoleh pengetahuan faktual yang t*eror*ganisir. Karena tingkat pertumbuhan informasi tidak terstruktur sangat tinggi dan terus meningkat belakangan ini, tantangan utama dalam IE, penambangan, dan analisis harus dipahami [21].

Tugas IE sangat beragam karena targetnya yang bervariasi (entitas, relasi, peristiwa, sentimen, dll.), struktur yang heterogen (rentang, triplet, rekaman, dll.), dan skema spesifik permintaan. Saat ini, sebagian besar pendekatan IE bersifat terspesialisasi tugas, yang mengarah pada arsitektur khusus, model terisolasi, dan sumber pengetahuan khusus untuk berbagai tugas IE [22].

2.5 Arsitektur Transformer

Arsitektur Transformer merupakan terobosan fundamental yang secara drastis mengubah bidang *Natural Language Processing* (NLP). Model ini hadir menggantikan arsitektur sekuensial sebelumnya, seperti *Recurrent Neural Networks* (RNN) dan *Long Short-Term Memory* (LSTM). Arsitektur lama tersebut dominan namun memiliki dua kelemahan utama: kesulitan dalam menangani dependensi jarak jauh dan proses komputasi yang lambat karena sifatnya yang berurutan. Transformer sepenuhnya meninggalkan metode *recursive* dan beralih menggunakan mekanisme *self-attention*. Mekanisme ini memungkinkan model untuk mengevaluasi dan menimbang tingkat kepentingan setiap kata dalam sebuah kalimat dalam hubungannya dengan semua kata lain di kalimat tersebut. Dalam konteks analisis kesesuaian resume dengan deskripsi pekerjaan, misalnya, model dapat memahami kaitan semantik antara *engineer* di resume dan pengembangan perangkat lunak, meskipun kedua frasa itu tidak terletak berdekatan.



Gambar 2.3 Diagram Arsitektur *Transformer* [23]

Gambar 2.3 adalah gambar diagram arsitektur Transformer fundamental dari makalah "*Attention Is All You Need*" yang menjadi fondasi model bahasa modern seperti BERT dan GPT. Arsitektur ini berbasis *sequence-to-sequence* dengan dua komponen utama yaitu Encoder dan Decoder [23]. Keduanya bekerja secara paralel menggunakan mekanisme *self-attention* untuk memproses urutan teks. Penjelasan rinci tiap komponen adalah sebagai berikut:

1. *Input Embedding* dan *Positional Encoding*

   Input teks diubah menjadi vektor numerik melalui *embedding*, kemudian ditambahkan positional encoding untuk memberikan informasi posisi karena Transformer tidak memiliki mekanisme urutan bawaan seperti *Recurrent Neural Networks* (RNN).

2. *Encoder*

   Setiap lapisan *encoder* terdiri atas dua modul utama: *Multi-Head Self-Attention* dan *Feed Forward Network*. Mekanisme *self-attention* memungkinkan setiap token memperhatikan token lain dalam satu kalimat, sedangkan jaringan *feed-forward* memperkaya representasi informasi. Kedua modul ini dilengkapi *residual connection* dan *layer normalization* untuk stabilitas pelatihan.

3. *Decoder Layer*

Decoder juga tersusun atas beberapa lapisan, tetapi memiliki tambahan *Masked Multi-Head Self-Attention*, yang membatasi perhatian hanya pada token sebelumnya sehingga prediksi dilakukan secara *autoregressive*. Selain itu, *decoder* memiliki *Multi-Head Attention* yang terhubung dengan keluaran *encoder* sehingga dapat memanfaatkan konteks input secara langsung.

4. *Output Embedding*

   Representasi token *output* diberi *positional encoding* dan diproses sebagai masukan *decoder* pada setiap langkah prediksi. *Output* ini digeser satu posisi (*shifted right*) untuk memastikan model memprediksi token berikutnya, bukan token saat ini.

5. *Linear Layer dan Softmax*

   Keluaran terakhir dari *decoder* diteruskan ke lapisan linear untuk memetakan vektor ke ruang kosakata, kemudian *softmax* digunakan untuk menghasilkan probabilitas setiap token. Token dengan probabilitas tertinggi dipilih sebagai keluaran model.

Riset dalam lima tahun terakhir telah berfokus pada penyempurnaan, efisiensi, dan skalabilitas arsitektur ini. Sebuah tinjauan komprehensif telah memetakan lanskap varian *transformer*, mengklasifikasikan lebih dari 100 model berbeda yang dikembangkan untuk meningkatkan efisiensi komputasi dan penggunaan memori. Tren optimasi ini sangat krusial untuk penerapan praktis *Large Language Model* (LLM) [24].

Kemampuan memproses dokumen yang lebih panjang membuka jalan untuk penerapan praktis di bidang-bidang khusus, seperti rekrutmen. Pada penelitian Deshmukh & Raut model *Bidirectional Encoder Representations from Transformers* (BERT) telah terbukti efektif untuk proses penyaringan resume otomatis dan pemeringkatan kandidat, memahami konteks bahasa secara mendalam dan menggantikan metode manual yang memakan waktu. Model BERT juga secara efisien mengekstraksi kata kunci, mengidentifikasi keahlian, dan menghitung kesamaan antara resume dan deskripsi pekerjaan, sehingga secara signifikan meningkatkan presisi dan efisiensi dalam proses seleksi kandidat [25].

2.6 *Large Language Model* (LLM)

Large Language Models (LLM) adalah evolusi langsung dari arsitektur *transformer*, yang ditingkatkan secara utuh dalam hal jumlah parameter, volume data pelatihan, dan sumber daya komputasi. LLM pada dasarnya adalah model *transformer* yang dilatih pada dataset berukuran *terabyte* untuk memprediksi token berikutnya dalam sebuah urutan. Hal

ini menandai pergeseran paradigma dari model *pre-trained* generasi awal seperti BERT, yang sebelumnya wajib melakukan *fine-tuning* pada dataset spesifik [26].

Era LLM modern, yang dipicu oleh GPT-3, mengubah pendekatan dari yang sebelumnya wajib melakukan *fine-tuning* pada dataset spesifik. GPT-3 memperkenalkan *in-context learning* (ICL), sebuah kemampuan yang di mana model dapat memahami dan melakukan tugas baru hanya dengan petunjuk atau contoh yang diberikan dalam prompt (*few-shot atau zero-shot learning*). Ini memungkinkan satu model yang sama digunakan untuk beragam tugas tanpa perlu dilatih kembali secara spesifik untuk masing-masing tugas [27].

Untuk mengatasi ini proses penyelarasan (*alignment*) menggunakan *Reinforcement Learning from Human Feedback* (RLHF) melalui InstructGPT [28]. Proses ini sangat penting dan kemudian diadopsi serta disempurnakan secara ekstensif oleh model-model seperti LLaMA 2 untuk mengajari model agar menghasilkan *output* yang membantu, jujur, dan tidak berbahaya.

2.7 LLaMA

LLaMA adalah kumpulan model bahasa dasar dengan rentang parameter dari 7B hingga 65B, dilatih menggunakan triliunan token untuk mencapai kinerja terbaik pada berbagai anggaran *inferensi*. Model canggih dapat dibuat hanya dengan menggunakan kumpulan data yang tersedia untuk umum secara eksklusif, tanpa bergantung pada data privat yang tidak dapat diakses. Hasilnya menunjukkan performa kompetitif: LLaMA-13B unggul dibanding GPT-3 (175B) dibanyak tolok ukur, meskipun berukuran 10 kali lebih kecil, sementara LLaMA-65B sejajar dengan model besar seperti Chinchilla-70B dan PaLM-540B. Semua model ini dirilis ke komunitas riset dengan keyakinan akan membantu mendemokratisasi akses dan studi LLM, terutama karena dapat dijalankan pada satu GPU [6].

Dalam perkembangan LLaMA diperkenalkan LLaMA 2 sebagai versi terbaru yang mencakup koleksi model *pre-trained* dan *fine-tuned*. Model ini dilatih menggunakan campuran data publik baru dengan beberapa pembaruan teknis: peningkatan 40% pada ukuran korpus *pre-training*, penggandaan panjang konteks, dan adopsi arsitektur *grouped-query attention*. LLaMA 2 dirilis ke publik dalam varian 7B, 13B, dan 70B (sementara varian 34B yang juga dikembangkan, tidak dirilis namun dilaporkan dalam penelitian). Secara khusus, varian yang telah disesuaikan bernama LLaMA 2-Chat, dioptimalkan untuk kasus penggunaan dialog. LLaMA 2-Chat terbukti melampaui kinerja model obrolan *open-source*

lainnya di sebagian besar tolok ukur. Evaluasi manusia terhadap kegunaan dan keamanan menunjukkan bahwa model ini dapat menjadi substitusi yang layak untuk model *closed-source* [29].

Secara struktural, LLaMA -3.1-8B mengadopsi arsitektur *Transformer decoder-only* autoregresif standar yang dioptimalkan untuk memproses konteks panjang secara efisien hingga 128K token. Cara kerja arsitektur ini bertumpu pada interaksi empat komponen utama dalam mengekstrak dan memproses informasi teks.

1. Tokenisasi & RoPE 128K: Teks masukan dikonversi menjadi token melalui *tokenizer* berskala 128.000 kosakata. Vektor token diberi informasi posisi menggunakan RoPE dengan frekuensi dasar ditingkatkan menjadi 500.000 guna mencegah degradasi informasi spasial pada konteks panjang hingga 128K token.
2. Normalisasi & Atensi GQA: *Input* diproses secara sekuensial melalui 32 lapisan *Transformer decoder*. Di setiap lapisan, data dinormalisasi oleh RMSNorm, lalu dihitung dependensi semantiknya menggunakan *Grouped-Query Attention* (GQA) yang memetakan 32 *Query heads* ke 8 *Key-Value heads* demi memangkas memori *KV-cache*.
3. Aktivasi SwiGLU: *Output* atensi diteruskan ke lapisan *Feed-Forward Network* (FFN) berdimensi 14.336 yang menggunakan fungsi aktivasi SwiGLU untuk mengekstrak fitur representasi bahasa tingkat tinggi melalui metode gerbang linier.
4. Prediksi *Autoregresif*: Lapisan terakhir menerapkan operasi *softmax* pada ruang kosakata untuk menghitung distribusi probabilitas kata berikutnya (*next-token prediction*). Token terpilih dikeluarkan sebagai *output* dan diumpankan kembali sebagai *input* baru secara berulang hingga sekuens selesai [30]

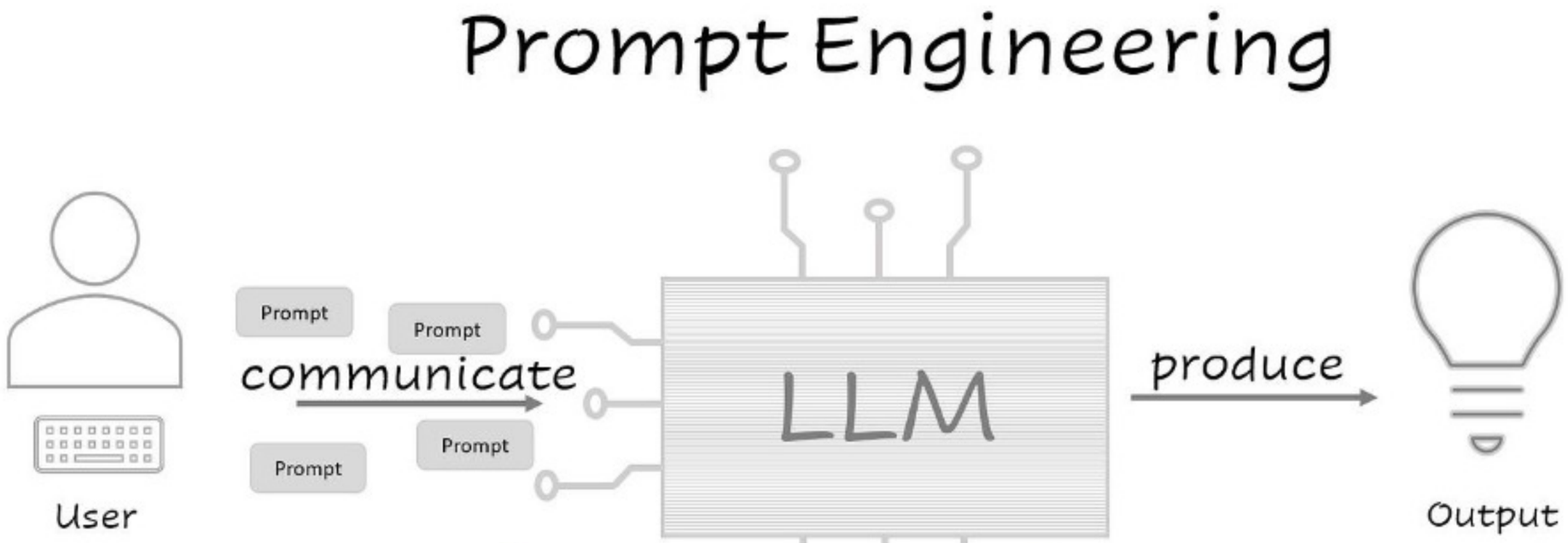
LLaMA-3.1 405B menunjukkan performa kompetitif terhadap model - model proprietary pada berbagai benchmark evaluasi dalam berbagai tugas, dengan keseimbangan antara kegunaan dan keamanan yang lebih baik. Model ini dirilis di bawah lisensi komunitas LLaMA 3 yang diperbarui untuk mendorong inovasi dan kemajuan kecerdasan buatan umum. Rilis ini mencakup versi *pre-trained* dan pasca-latihan dari model 405B serta model baru LLaMA Guard untuk keamanan *input* dan *output*. Pengembangan ekstensi multimodel yang mendukung pengenalan gambar, video, dan pemahaman ucapan sedang berlangsung untuk memperluas kemampuan LLaMA 3. Dengan peningkatan kinerja, efisiensi, dan skalabilitas, LLaMA 3 menjadi alat yang sangat baik untuk berbagai aplikasi AI [31].

2.8 *Prompt Engineering*

Prompt engineering adalah disiplin ilmu baru yang berfokus pada pengembangan dan optimasi *prompt*, sehingga membantu pengguna menerapkan LLM ke berbagai skenario dan bidang penelitian. Dalam ilmu komputer, LLM dapat memperoleh jawaban yang ideal dan stabil melalui rekayasa kata kunci, dan penggunaan kata kunci yang berbeda akan memengaruhi kinerja LLM, yang sedikit banyak tercermin dalam masalah matematika. Desain kata kunci yang baru digunakan saat ini mencakup kata kunci *chain of thoughts* (COT) dan kata kunci *tree of thoughts* (TOT). Dengan adanya usulan teori COT dan TOT di bidang LLM ilmu komputer, kata kunci yang sesuai telah dikembangkan dan menunjukkan peningkatan kinerja dalam masalah matematika [32].

Prompt engineering menyediakan kerangka kerja untuk mendokumentasikan pola-pola penyusunan prompt guna menyelesaikan berbagai masalah, memungkinkan pencari informasi dan pengembang mengadaptasinya ke domain berbeda, mengombinasikan pola untuk meningkatkan *output* LLM, serta memfasilitasi transfer pengetahuan antar pengguna LLM. Ini mendukung pengembangan aplikasi LLM yang lebih canggih dan efektif, membantu memahami perilaku model, mengarahkan ke *output* yang benar dan informatif, serta meningkatkan kinerja *few-shot learning* melalui *prompt in-context* yang dioptimalkan, sekaligus memfasilitasi chatbot, asisten virtual, alat khusus domain, dan sistem AI percakapan lainnya akhirnya mendorong kemajuan tugas NLP.

Pada masa yang akan datang *prompt engineering* akan semakin krusial untuk membuka potensi penuh LLM dengan memfasilitasi pengguna untuk menghasilkan *output* bahasa cepat dan akurat, *prompt engineering* menjadi elemen yang layak dijual untuk efisiensi bisnis lintas domain, menciptakan peluang kerja baru, serta mendorong antarmuka pengguna secara langsung untuk mengontrol *output* LLM, sehingga memungkinkan penyempurnaan konten dan aplikasi inovatif yang sebelumnya mustahil [33].



Gambar 2.4 *Prompt Engineering Process*

2.9 Metrik Evaluasi

Metrik evaluasi berperan penting sebagai alat untuk memverifikasi dan membandingkan kinerja model dalam berbagai bidang dan aplikasi. Metrik ini menjadi acuan bagi peneliti dan membantu mengevaluasi nilai praktis suatu model. Tanpa standar metrik yang jelas, sulit membandingkan hasil model yang sejenis. Selain itu, metrik evaluasi memungkinkan pengembang mengukur kekuatan dan kelemahan model secara akurat, memastikan keandalan dan akurasi dalam penggunaan nyata. Dengan metrik ini, penilaian terhadap kemampuan generalisasi, ketahanan terhadap *overfitting*, dan performa pada data yang beragam dapat dilakukan untuk mendukung pengambilan keputusan dalam pengembangan dan pemilihan model terbaik [34].

Banyak metrik evaluasi digunakan untuk mengukur performa model prediksi, terutama pada klasifikasi, seperti akurasi, *precision*, *recall*, dan *F1-score*. Akurasi adalah perbandingan antara total prediksi yang benar dengan seluruh prediksi yang dibuat (1). *Precision* mengukur seberapa banyak prediksi positif yang benar dari semua prediksi positif model (2). *Recall* menunjukkan seberapa banyak prediksi positif yang benar dari seluruh data positif sebenarnya (3). *F1-score* adalah rata-rata yang sesuai dari *precision* dan *recall*, yang memberikan keseimbangan antara keduanya dalam satu nilai [35] (4).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F1 - score = 2 \text{ x} \frac{precision \text{ x } recall}{precision + recall} \quad (4)$$

Keterangan:

Accuracy = Perbandingan antara total prediksi dengan seluruh prediksi.

Precision = Mengukur seberapa akurat prediksi positif model

Recall = Mengukur berapa banyak kasus positif yang berhasil ditemukan

F1-score = Rata-rata harmonik antara *Precision* dan *Recall*.

TP (*True Positive*) = Jumlah *Instance* positif dikategorikan positif oleh model

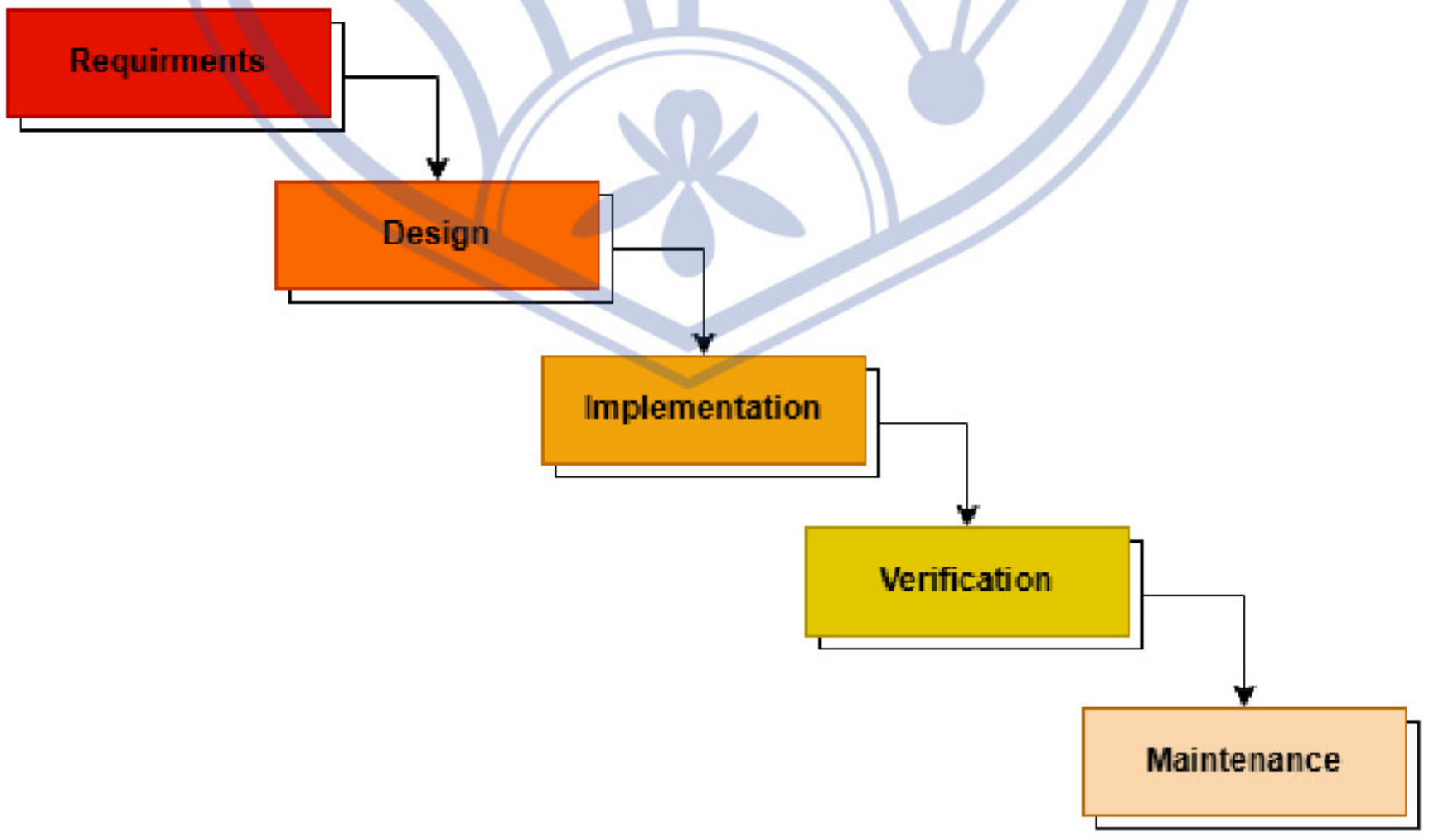
TN (*True Negative*) = Jumlah *Instance* negatif dikategorikan negatif oleh model

FP (*False Positive*) = Jumlah *Instance* negatif dikategorikan positif oleh model

FN (*False Negative*) = Jumlah *Instance* positif dikategorikan negatif oleh model

2.10 Metode Waterfall

Metode *Waterfall*, yang juga dikenal dengan sebutan *classic life cycle* atau *Linear Sequential Model*, merupakan salah satu model *System Development Life Cycle* (SDLC) yang populer dalam pengembangan perangkat lunak dan sistem informasi. Metode ini mengedepankan pendekatan yang sangat sistematis, terstruktur, dan berurutan. Proses pengembangannya dilakukan selangkah demi selangkah, diawali dengan analisis kebutuhan pengguna, perencanaan, pemodelan, konstruksi, penyerahan sistem (*deployment*), hingga tahap pemeliharaan. Karena sifatnya yang berurutan, para pengembang diwajibkan untuk memahami secara mendalam karakteristik dan cara kerja model ini agar proses pengembangan berjalan lancar [36]



Gambar 2.5 Tahapan Metodologi *Waterfall* [37]

Metode *Waterfall* terbagi menjadi lima fase:

1. *Requirement Analysis*: Tahap ini menyusun dokumen spesifikasi persyaratan perangkat lunak yang lengkap. Analis sistem dan analis bisnis bekerja sama untuk merinci persyaratan fungsional, menggambarkan bagaimana pengguna akan berinteraksi dengan sistem, dan menetapkan persyaratan nonfungsional seperti keandalan, skalabilitas, kemampuan pengujian, ketersediaan, pemeliharaan, kinerja, serta standar kualitas.
2. *Design*: Tahap perencanaan teknis untuk solusi perangkat lunak. Pengembang dan perancang menyusun rencana implementasi yang mencakup algoritma, arsitektur perangkat lunak, skema basis data konseptual, diagram logis, desain konsep, antarmuka pengguna grafis, dan struktur data yang akan digunakan.
3. *Implementation*: Tahap penulisan kode sumber dan penggabungan komponen menjadi aplikasi yang dapat dijalankan, termasuk pembuatan basis data dan file pendukung.
4. *Verification*: Tahap verifikasi dan validasi untuk memastikan aplikasi memenuhi spesifikasi yang ditetapkan dan mencapai tujuan fungsional serta nonfungsional yang diharapkan.
5. *Maintenance*: Tahap akhir yang mencakup perbaikan bug yang terdeteksi setelah peluncuran, serta pembaruan dan penyesuaian agar sistem tetap berfungsi sesuai kebutuhan [37].