

BAB I

PENDAHULUAN

A. Latar Belakang

Data statistik yang dirilis Standish Group tahun 2020 menyebutkan bahwa 66% proyek aplikasi perangkat lunak mengalami kegagalan [1]. Tidak hanya itu, akumulasi kerugian akibat kegagalan perusahaan Amerika Serikat (AS) lainnya mencapai \$260 miliar. Tingginya angka kegagalan dan kerugian ini menunjukkan bahwa rekayasa perangkat lunak membutuhkan pendekatan inovatif, terutama dalam proses analisis kebutuhan. Selain itu, permasalahan mengenai kebutuhan dapat berubah baik dari segi kompleksitas maupun seiring berjalannya waktu [2]. Komunikasi antar kedua belah pihak, pengguna dan pengembang yang tidak efektif dapat menjadi penyebab kesalahpahaman. Akhirnya, miskomunikasi berdampak pada kualitas produk perangkat lunak yang dihasilkan [3].

Kebutuhan perangkat lunak disajikan dalam bentuk kalimat terstruktur dengan bahasa alami, mengenai layanan dan batasan yang harus dipenuhi sistem [4]. Pemenuhan ini untuk menunjang kesesuaian dengan harapan pengguna ataupun pemangku kepentingan. Tidaklah mudah untuk mendefinisikan kebutuhan ini, mengingat pengguna berasal dari latar belakang yang berbeda-beda. Pengguna mungkin cukup kesulitan mengartikulasikan keinginannya atau kemampuan dari sistem, sehingga menggunakan kata-kata yang dapat menimbulkan kerancuan [5].

Berbagai penelitian menunjukkan bahwa eksperimen dengan pendekatan berbasis *Deep Learning* (DL) telah merevolusi pemrosesan bahasa alami, dengan model representasi bahasa berbasis *Transformer* memberikan performa yang lebih baik dalam tugas klasifikasi teks [6]. Model *deep learning* yang digunakan dalam eksperimen penelitian ini terdiri dari dua kelompok: (1) model berbasis arsitektur BERT dengan tiga varian berbeda, yaitu *Distilled Bidirectional Encoder Representations from Transformers* (Distil BERT) sebagai model ringan, *Bidirectional Encoder Representations from*

Transformers Base (BERT *Base*) sebagai model kelas menengah, dan *Bidirectional Encoder Representations from Transformers Large* (BERT *Large*) sebagai model berskala besar, serta dilakukan perbandingan dengan (2) model Large Language Model (LLM) berupa GPT-4o, yang digunakan dengan teknik *prompting*.

Teknik NLP yang sesuai dapat menangani permasalahan proses analisis kebutuhan perangkat lunak. Permasalahan tersebut sebagaimana telah dijelaskan terdapat lama waktu, *human error*, dan kompleksitas bahasa alami. Proses tradisional dalam analisis kebutuhan dilakukan secara manual dan memakan waktu lama serta berisiko menghasilkan kesalahan akibat interpretasi manusia. Sebagai makhluk sosial, manusia juga memiliki keterbatasan fisik jika mengalami kelelahan kerja yang berdampak pada produktivitas maupun kualitas kerja [7]. Pengguna menggunakan bahasa alami dalam menyampaikan kebutuhan juga kerap kali mengandung sifat ambigu, bervariasi, dan sulit dipahami dikarenakan tidak memiliki latar belakang teknis [5]. Permasalahan inilah yang menjadi kerentanan kesalahan dalam proses analisis kebutuhan perangkat lunak.

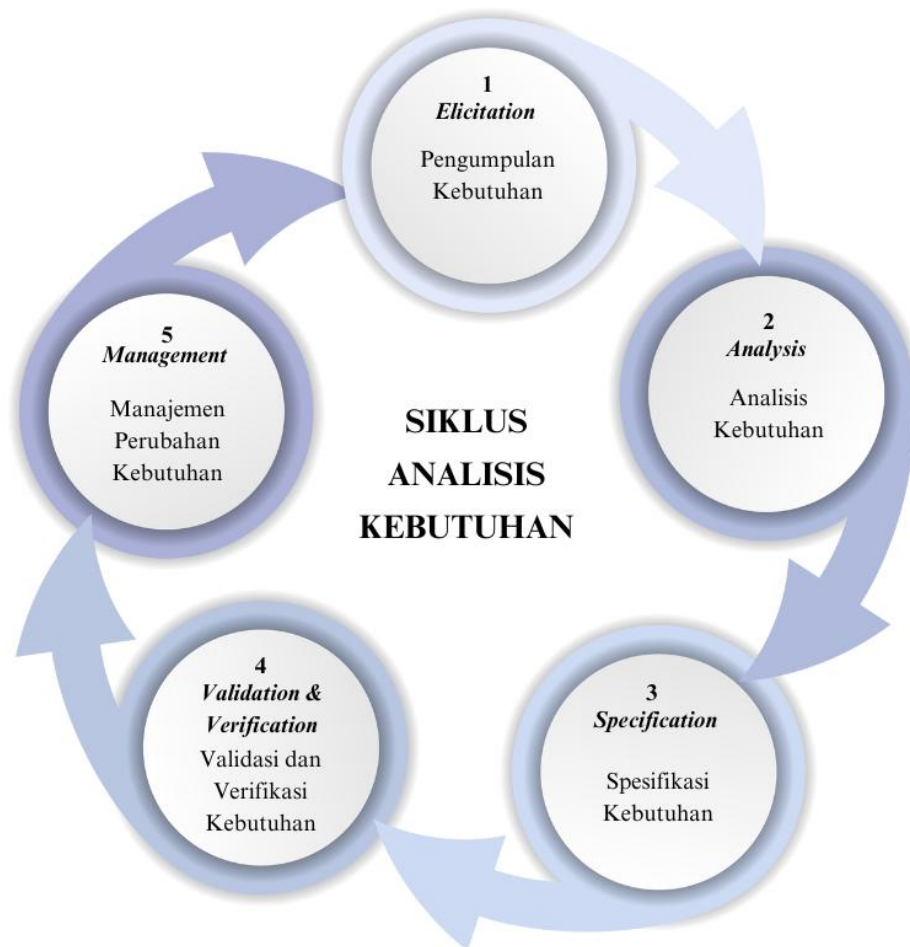
Pendekatan berbasis *deep learning* dalam penelitian terbaru menunjukkan adanya peningkatan akurasi secara signifikan dalam klasifikasi teks dan ekstraksi informasi. Studi literatur oleh Alfando & R. Hayami (2023) menyimpulkan teknik *deep learning* dalam klasifikasi banyak label pada teks, dapat memberikan hasil akurat dengan proses yang tepat, performa di beberapa literatur berhasil mencapai akurasi 95% [8]. Hal ini sejalan dengan penelitian Abdul Raup, dkk. (2022), yang menyoroti perkembangan pesat kecerdasan buatan, termasuk *deep learning*, dalam beberapa tahun terakhir. Kecerdasan buatan kini telah digunakan untuk menyelesaikan masalah kompleks yang sebelumnya sulit dipecahkan oleh manusia, dan teknologi ini telah diterapkan di berbagai sektor [9].

B. Rumusan Masalah

Proses analisis kebutuhan perangkat lunak masih dilakukan secara tradisional, terutama melalui wawancara [10]. Metode ini memerlukan waktu lama karena melibatkan berbagai pihak yang harus berkoordinasi untuk menangkap dan mendefinisikan kebutuhan sistem secara akurat. Terlebih, kebutuhan yang ditangkap dapat bersifat dinamis menjadi suatu tantangan besar dalam pengembangan [11]. Baik itu suatu perusahaan yang mengembangkan sistem informasi harus mengumpulkan input dari pengguna akhir, pengembang, dan manajer proyek melalui wawancara atau *Focus Group Discussion* (FGD). Namun, hasilnya yang ditulis dalam bahasa alami sering kali menyebabkan subjektivitas dan ambiguitas dalam pemahaman [12].

Analisis menghasilkan pernyataan kebutuhan perangkat lunak dalam bentuk kalimat terstruktur, pernyataan tersebut disusun dalam sebuah dokumen [13]. Kebutuhan dalam suatu sistem dapat diklasifikasikan secara manual melalui proses analisis, di mana manusia berperan dalam menginterpretasikan informasi. Namun, proses ini tidak selalu mudah, terutama ketika kebutuhan dinyatakan dalam bahasa alami yang kompleks dan bervariasi. Kesulitan dalam menggali dan mendokumentasikan kebutuhan yang tidak terungkap secara eksplisit menjadi tantangan utama dalam analisis kebutuhan sistem [11]. Hal ini berdampak pada keakuratan analisis dan efektivitas sistem yang dikembangkan.

Gambar 1.1 menunjukkan siklus analisis kebutuhan perangkat lunak yang terdiri dari lima tahap utama dan bersifat iteratif [14]. Tahapan ini dimulai dari pengumpulan kebutuhan (*elicitation*), diikuti oleh analisis (*analysis*), di mana informasi dievaluasi untuk memahami kebutuhan bisnis dan teknis [15]. Pada tahap ini, ambiguitas dalam bahasa alami dan perubahan kebutuhan sering menyebabkan potensi perbaikan berulang [11].



Gambar 1.1 Siklus Analisis Kebutuhan Secara Keseluruhan ([link](#))

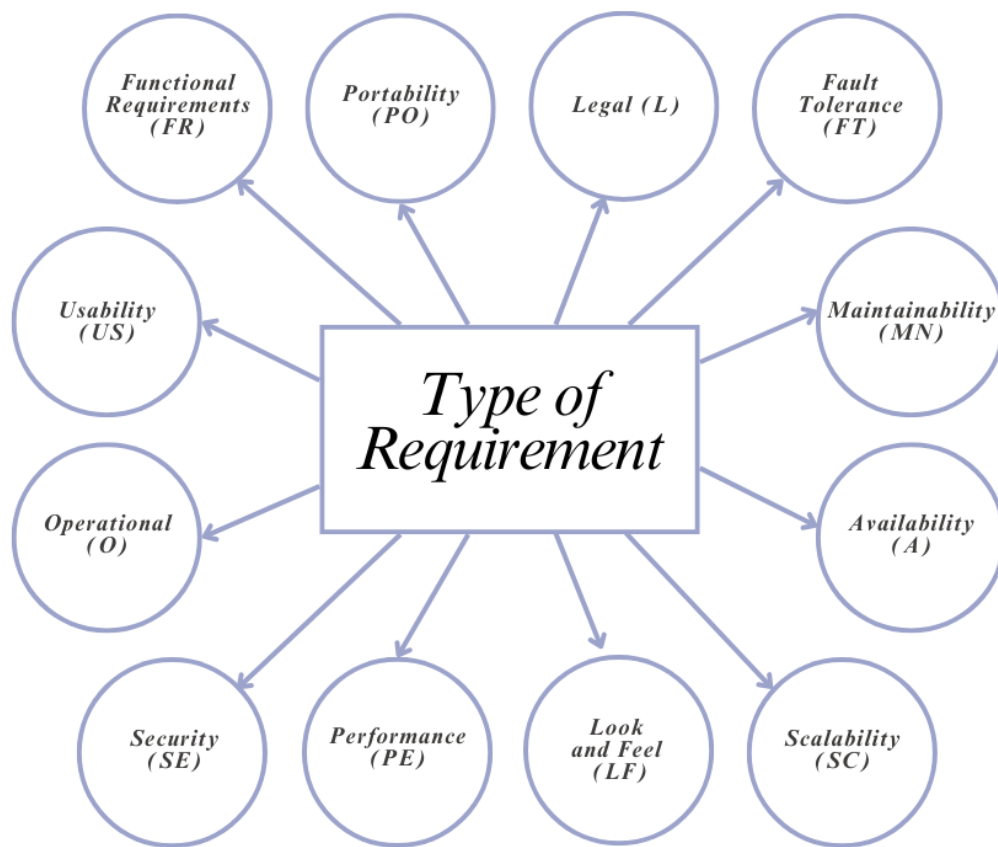
Dalam klasifikasi kebutuhan perangkat lunak, kebutuhan (*requirement*) umumnya dibagi menjadi dua kategori utama, *Functional Requirements* (FR) dan *Non-Functional Requirements* (NFR) [16]. *Functional Requirements* (FR) berkaitan dengan fitur dan fungsi spesifik yang harus dimiliki oleh sistem, seperti interaksi pengguna dengan sistem atau bagaimana sistem merespons input tertentu. Sementara itu, *Non-Functional Requirements* (NFR) mencakup aspek kualitas dan kendala sistem.

Tabel 1.1 menunjukkan contoh kalimat *requirement* hasil pengumpulan data yang masih perlu diklasifikasikan ke dalam kategori spesifik seperti Gambar 1.2 [17], terdapat 12 kategori diantaranya *Functional Requirements* (FR), *Portability* (PO), *Legal* (L), *Fault Tolerance* (FT), *Maintainability* (MN),

Availability (A), Scalability (SC), Look and Feel (LF), Performance (PE), Security (SE), Operability (O), dan Usability (US).

Tabel 1.1 Kalimat Terstruktur Mengenai Kebutuhan Perangkat Lunak

No.	<i>Requirement (English)</i>	Persyaratan (Bahasa Indonesia)
1.	<i>The system shall refresh the display every 60 seconds.</i>	Sistem harus menyegarkan tampilan setiap 60 detik.
2.	<i>The Disputes System must be accessible by both internal and external users.</i>	Sistem Sengketa harus dapat diakses oleh pengguna internal dan eksternal.
3.	<i>The product must support Internet Explorer 5.5 and above.</i>	Produk harus mendukung Internet Explorer 5.5 dan yang lebih baru.
4.	<i>The system shall prevent malicious attacks including denial of service.</i>	Sistem harus mencegah serangan jahat termasuk penolakan layanan.
5.	<i>The product shall have a conservative and professional appearance.</i>	Produk harus memiliki tampilan yang konservatif dan profesional.



Gambar 1.2 Kategori Kebutuhan Perangkat Lunak ([link](#))

Permasalahan utama dalam klasifikasi ini adalah prosesnya yang masih manual, berisiko menimbulkan inkonsistensi, serta memperlambat siklus pengembangan [18]. Hasil dari klasifikasi ini digunakan untuk menyusun dokumen spesifikasi kebutuhan (*specification*), yang menjadi acuan utama dalam pengembangan perangkat lunak. Agar perangkat lunak yang dikembangkan sesuai dengan kebutuhan pengguna, dokumen ini harus melalui proses validasi dan verifikasi (*validation & verification*). Selain itu, karena kebutuhan perangkat lunak sering mengalami perubahan, manajemen (*management*) perubahan yang baik sangat diperlukan untuk memastikan sistem dapat beradaptasi tanpa mengorbankan konsistensi dan keberhasilan proyek [15].

Berdasarkan permasalahan dan potensi solusi yang telah dijelaskan, pertanyaan penelitian dalam Tugas Akhir ini adalah:

1. Bagaimana cara membuat dataset untuk klasifikasi kategori kebutuhan perangkat lunak dari kalimat terstruktur menggunakan berbagai sumber?
2. Bagaimana membangun model klasifikasi kategori kebutuhan perangkat lunak dari kalimat terstruktur menggunakan *deep learning* secara akurat?
3. Bagaimana menentukan model terbaik dengan membandingkan kinerja model *deep learning* dengan Large Language Model (LLM) GPT-4o?

C. Batasan Masalah

Penelitian ini akan memfokuskan pada beberapa aspek tertentu untuk menjaga fokus dan kelengkapan hasil penelitian, dengan batasan-batasan sebagai berikut:

1. Penelitian ini terbatas pada 12 kategori kebutuhan perangkat lunak, yaitu:
 - a. *Functional Requirements (FR)*
 - b. *Portability (PO)*
 - c. *Legal (L)*
 - d. *Fault Tolerance (FT)*
 - e. *Maintainability (MN)*
 - f. *Availability (A)*
 - g. *Scalability (SC)*
 - h. *Look and Feel (LF)*
 - i. *Performance (PE)*
 - j. *Security (SE)*
 - k. *Operability (O)*
 - l. *Usability (US)*
2. Penelitian ini akan membandingkan beberapa metode *deep learning* dalam tugas pemrosesan klasifikasi kategori kebutuhan perangkat lunak dari kalimat terstruktur dengan menggunakan:
 - a. *Distilled Bidirectional Encoder Representations from Transformers (Distil BERT)*
 - b. *Bidirectional Encoder Representations from Transformers Base (BERT Base)*

- c. *Bidirectional Encoder Representations from Transformers Large* (BERT *Large*)
3. Model dalam penelitian ini melakukan klasifikasi kategori kebutuhan perangkat lunak secara langsung dalam satu fase klasifikasi, tanpa memisahkan proses terlebih dahulu antara pengenalan kebutuhan *Functional Requirements* (FR) dan *Non-Functional Requirements* (NFR) sebelum mengklasifikasikan sub kategori spesifik.
4. Perbandingan tiga model *deep learning* menggunakan hasil keluaran model LLM GPT-4o dengan metode *prompting* sebagai *baseline*, untuk menentukan model terbaik.
5. Sumber pengumpulan dataset penelitian ini menggunakan tiga repositori ilmiah berikut:
 - a. [kaggle.com](https://www.kaggle.com)
 - b. zenodo.org
 - c. promisedata.org

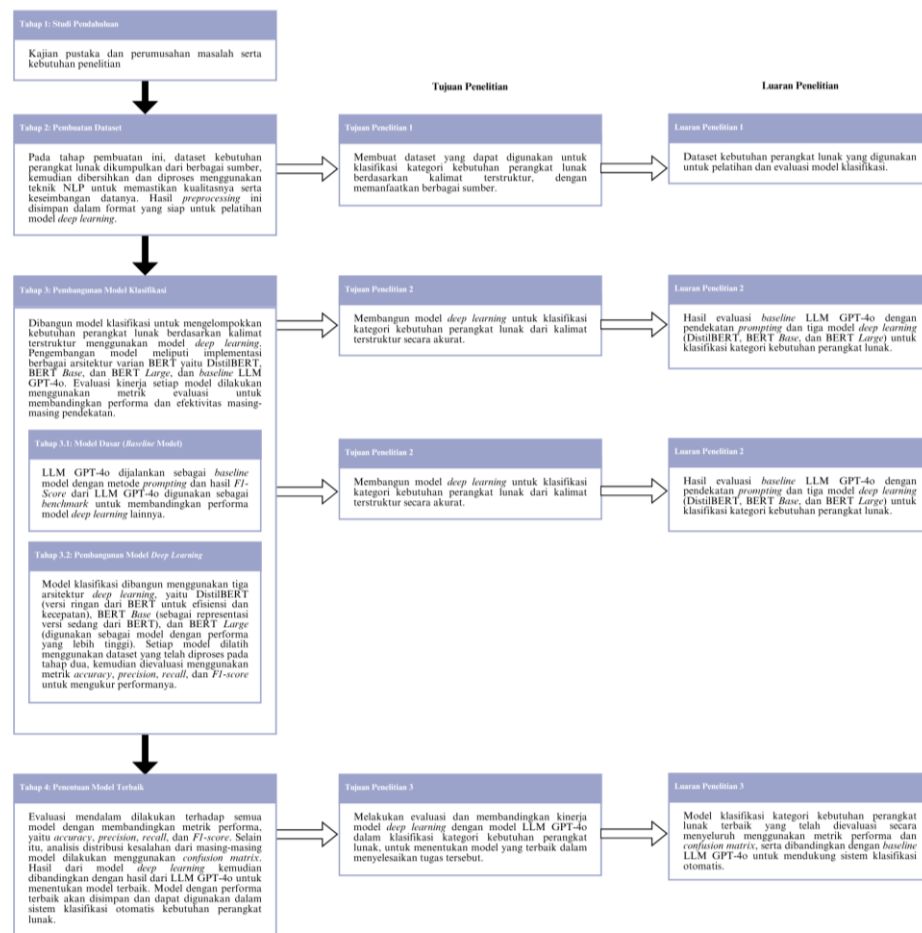
D. Tujuan Penelitian

Penelitian ini memiliki tujuan yang ingin dicapai dalam mengatasi permasalahan terkait, yaitu:

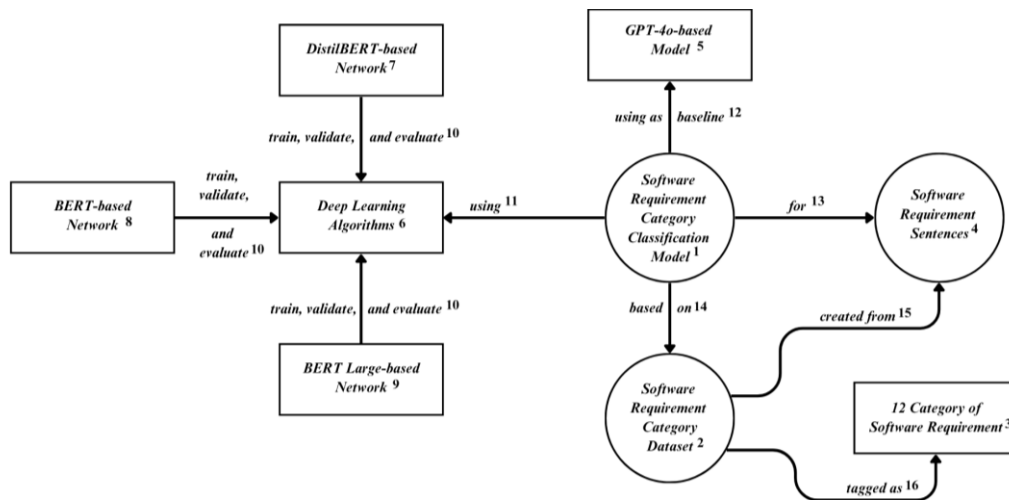
1. Membuat dataset yang dapat digunakan untuk klasifikasi kategori kebutuhan perangkat lunak berdasarkan kalimat terstruktur, dengan memanfaatkan berbagai sumber.
2. Membangun model *deep learning* untuk klasifikasi kategori kebutuhan perangkat lunak dari kalimat terstruktur secara akurat.
3. Melakukan evaluasi dan membandingkan kinerja model *deep learning* dengan model LLM GPT-4o (*baseline*) dalam klasifikasi kategori kebutuhan perangkat lunak, untuk menentukan model yang terbaik dalam menyelesaikan tugas tersebut.

E. Metode Penelitian

Penelitian ini bertujuan untuk membangun model kecerdasan buatan menggunakan pendekatan berbasis *deep learning* dengan arsitektur varian BERT yaitu Distil BERT, BERT *Base*, dan BERT *Large*. Penentuan model terbaik dilakukan berdasarkan hasil evaluasi dan perbandingannya dengan *baseline* LLM GPT-4o. Gambar 1.3 menunjukkan tahapan metode penelitian ini secara keseluruhan, yang terdiri dari empat tahap untuk mencapai tiga tujuan penelitian. Secara umum, gambaran keseluruhan dapat dilihat pada Gambar 1.4.



Gambar 1.3 Kerangka Operasional Penelitian ([link](#))



Terjemahan Bahasa Indonesia:

1. Model Klasifikasi Kategori Kebutuhan Perangkat Lunak
2. Dataset Kategori Kebutuhan Perangkat Lunak
3. 12 Kategori Kebutuhan Perangkat Lunak
4. Kalimat Kebutuhan Perangkat Lunak
5. Model Berbasis GPT-4o
6. Algoritma Pembelajaran Mendalam

7. Jaringan Berbasis DistilBERT
8. Jaringan Berbasis BERT Base
9. Jaringan Berbasis BERT Large
10. melatih, memvalidasi, dan mengevaluasi
11. menggunakan
12. digunakan sebagai acuan dasar

13. untuk
14. berdasarkan
15. dibuat dari
16. diberi label sebagai

Gambar 1.4 Gambaran Umum Penelitian ([link](#))

1. Studi Pendahuluan

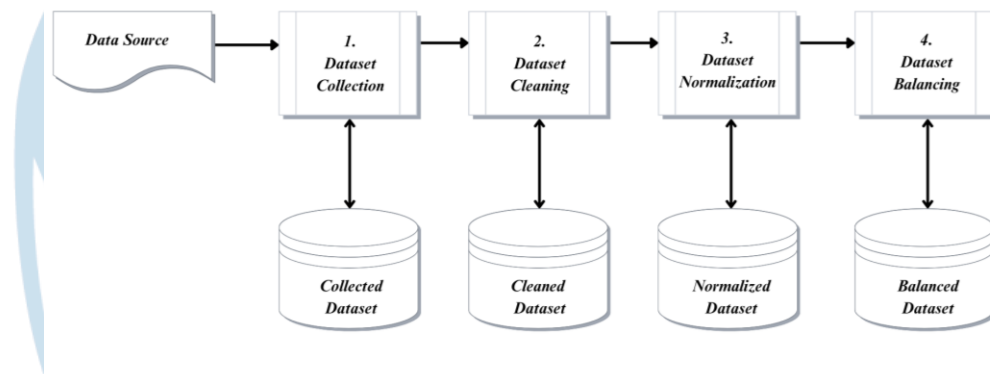
Tahap pertama dalam penelitian ini adalah Studi Pendahuluan mengenai kebutuhan perangkat lunak dalam pengembangan model *deep learning*. Kajian pustaka dilakukan untuk memahami metode, tantangan, dan solusi yang dapat diterapkan. Selanjutnya, penelitian difokuskan pada pembuatan dataset, klasifikasi kebutuhan perangkat lunak, serta perbandingan performa model *deep learning* dengan LLM GPT-4o sebagai *baseline*.

Tantangan yang dihadapi dalam penelitian ini meliputi pembuatan dataset kategori kebutuhan perangkat lunak yang baik, dengan ukuran yang memadai, seimbang (*balanced*), serta lebih variatif dibandingkan dengan dataset awal, yaitu PROMISE_exp [19]. Selain itu, tantangan lainnya terletak pada proses komputasi yang langsung difokuskan pada satu fase klasifikasi tanpa melalui fase klasifikasi kebutuhan menjadi NFR atau FR terlebih dahulu. Platform dan teknologi yang digunakan dalam pengolahan

dan analisis data meliputi Lovelace UAJY¹, Google Colaboratory², Hugging Face Model Hub³, HTML (HyperText Markup Language) dan Flask.

2. Pembuatan Dataset

Setelah studi pendahuluan dilakukan, tahap selanjutnya adalah Pembuatan Dataset yang digunakan dalam pelatihan model klasifikasi. Pada tahap ini terdiri dari empat tugas sebagaimana ditunjukkan Gambar 1.5. Hasil dari setiap tugas Pembuatan Dataset disimpan, sehingga terdapat empat dataset utama yang merupakan luaran setiap prosesnya.



Terjemahan Bahasa Indonesia:

- | | |
|------------------------|--------------------------|
| 1. Sumber Data | 6. Normalisasi Dataset |
| 2. Pengumpulan Dataset | 7. Dataset Normalisasi |
| 3. Dataset Terkumpul | 8. Penyeimbangan Dataset |
| 4. Pembersihan Dataset | 9. Dataset Seimbang |
| 5. Dataset Bersih | |

Gambar 1.5 Kerangka Pembuatan Dataset ([link](#))

Pertama, *Dataset Collection* terdiri dari kalimat terstruktur yang menggambarkan kebutuhan perangkat lunak. Kategori kebutuhan perangkat lunak terdiri atas *Functional Requirements* (FR) serta 11 kategori *Non-Functional Requirements* (NFR), yang diperoleh dari tiga repositori, yaitu kaggle.com, zenodo.org, dan promisedata.org. Dataset awal berjumlah

¹ <https://lovelace.uajy.ac.id/>

² <https://colab.research.google.com/>

³ <https://huggingface.co/>

6.704 entri dalam format .xlsx dengan dua kolom utama *Type* dan *Requirement*. Luaran hasil dari proses pertama ini disimpan dalam file bernama Dataset_Tahap_1A.xlsx.

Selanjutnya, tugas kedua *Dataset Cleaning* dilakukan sebelum data digunakan untuk pelatihan model, data dibersihkan melalui beberapa proses. Mulai dari kolom relevan dipilih menggunakan `pandas.read_excel()`, dan dipastikan tidak ada nilai kosong [20]. Pembersihan dimulai dari penghapusan kategori yang tidak diperlukan, yaitu NFR sebagai kategori general [17]. Jumlah data setelah tahap ini dicatat dan disimpan dalam file terpisah. Kategori *Functional* (F) kemudian diseragamkan menjadi FR untuk konsistensi, dan jumlah data kembali disimpan. Data duplikat juga dihapus guna menghindari pengulangan informasi menggunakan `drop_duplicates()`, dan jumlah akhir baris kembali dicatat [21]. Data duplikasi tersebut membutuhkan penyeragaman format dengan agresif pada kolom *Requirement* agar mampu bersih dari duplikat. Rangkaian pembersihan ini menghasilkan dataset bersih yang siap untuk tugas normalisasi. Luaran hasil dari proses kedua ini disimpan dalam file bernama Dataset_Tahap_2D.xlsx.

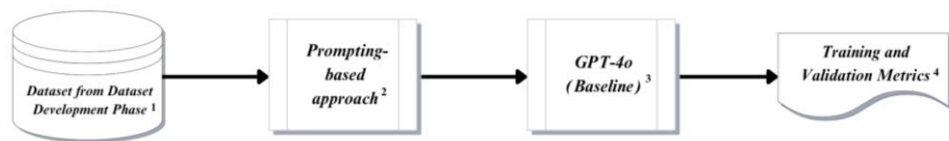
Ketiga, *Dataset Normalization* dilakukan untuk menyiapkan data agar lebih bersih dan seragam sebelum digunakan dalam proses klasifikasi. Proses ini meliputi penghapusan spasi berlebih, konversi seluruh huruf menjadi huruf kecil, serta penghapusan atau penggantian karakter-karakter asing dan simbol yang tidak relevan. Selain itu, dilakukan penyesuaian akhir kalimat agar memiliki struktur yang lebih konsisten. Hasil dari proses normalisasi ini kemudian disimpan kembali dalam format .xlsx. Luaran hasil dari proses ketiga ini disimpan dalam file bernama Dataset_Tahap_3A.xlsx.

Pada tahap terakhir, dilakukan tugas *Dataset Balancing* untuk memastikan setiap kategori memiliki jumlah data yang seimbang. Proses ini dimulai dengan membaca data yang telah dinormalisasi sebelumnya, kemudian menganalisis distribusi jumlah data pada setiap kategori. Untuk

kategori dengan jumlah data yang melebihi target, dipilih sejumlah data unik yang memiliki tingkat kemiripan rendah satu sama lain agar menghindari duplikasi informasi. Sebaliknya, untuk kategori yang memiliki jumlah data di bawah target, dilakukan augmentasi data menggunakan teknik *back-translation*, yaitu menerjemahkan kalimat ke bahasa lain dan kemudian kembali ke bahasa aslinya guna menghasilkan variasi teks yang tetap relevan secara semantik [22]. Hasil dari proses penyeimbangan ini kemudian disimpan dalam file *Dataset_Tahap_4A.xlsx* sebagai keluaran dari proses keempat.

3. Pembangunan Model Klasifikasi

Setelah tahap Pembuatan Dataset, dilakukan Pembangunan Model Klasifikasi. Terbagi menjadi dua proses pertama pembuatan *baseline* menggunakan model LLM GPT-4o dengan kerangka pada Gambar 1.6. *Baseline* ini dibangun dengan pendekatan *prompting based approach*, di mana setiap kalimat masukan diberikan dalam bentuk instruksi (*prompt*) untuk mendapatkan prediksi klasifikasi dari model LLM GPT-4o [23]. Pendekatan ini dilakukan tanpa proses pelatihan model tambahan, sehingga dapat memberikan gambaran awal mengenai performa klasifikasi dari model LLM tanpa *fine-tuning*. Hasil prediksi dari LLM GPT-4o kemudian dievaluasi menggunakan metrik pelatihan dan validasi, seperti *accuracy*, *precision*, *recall*, dan *f1-score*, untuk digunakan sebagai acuan pembandingan terhadap model-model lainnya.

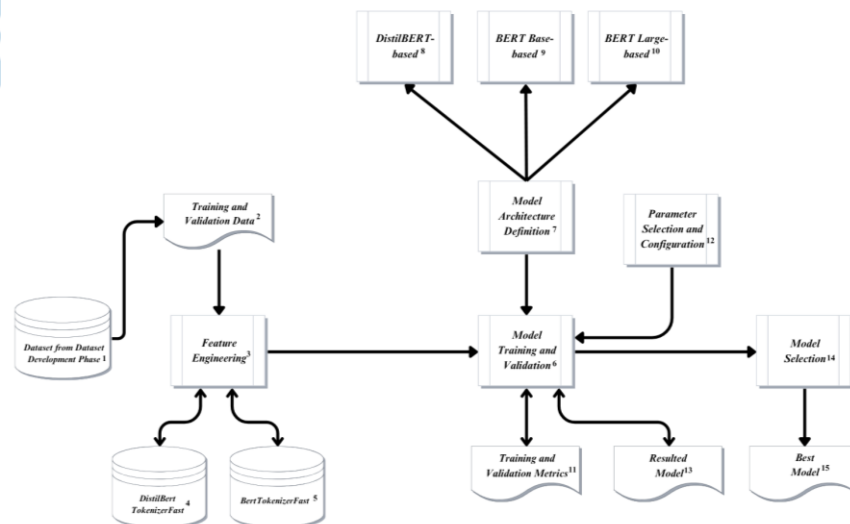


Terjemahan Bahasa Indonesia:

1. Dataset dari Tahap Pembuatan Dataset
2. Pendekatan Berbasis Prompt
3. GPT-4o (Tolok Ukur)
4. Metrik Pelatihan dan Validasi

Gambar 1.6 Kerangka Pembangunan Baseline LLM GPT-4o ([link](#))

Selanjutnya, proses kedua untuk meningkatkan akurasi klasifikasi kebutuhan perangkat lunak, dibangun beberapa model *deep learning* sesuai kerangka Gambar 1.7. Dataset yang telah dipersiapkan pada tahap Pembuatan Dataset terdapat dua, yaitu Dataset_Tahap_3A.xlsx (*imbalanced*) dan Dataset_Tahap_4A.xlsx (*balanced*). Dataset tersebut kemudian dibagi menjadi data pelatihan dan validasi untuk mendukung proses pembelajaran model secara optimal. Data akan dikonversi menjadi representasi token agar dapat diproses model melalui *feature engineering*.



Terjemahan Bahasa Indonesia:

- | | | |
|---|---|---------------------|
| 1. Dataset dari Tahap Pembuatan Dataset | 7. Definisi Arsitektur Model | 13. Hasil Model |
| 2. Data Pelatihan dan Validasi | 8. Berbasis DistilBERT | 14. Pemilihan Model |
| 3. Rekayasa Fitur | 9. Berbasis BERT Base | 15. Model Terbaik |
| 4. DistilBertTokenizerFast | 10. Berbasis BERT Large | |
| 5. BertTokenizerFast | 11. Metrik Pelatihan dan Validasi | |
| 6. Pelatihan dan Validasi Model | 12. Pemilihan dan Konfigurasi Parameter | |

Gambar 1.7 Kerangka Pembangunan Model Klasifikasi ([link](#))

Tiga arsitektur model dikembangkan dan diuji, yakni *DistilBERT-based*, *BERT Base-based*, dan *BERT Large-based*. Masing-masing arsitektur model dikonfigurasi dengan karakteristik dan parameter yang sesuai, serta dilakukan proses *tuning hyperparameter* untuk mencapai performa terbaik. Selama proses pelatihan dan validasi, digunakan metrik evaluasi seperti *accuracy*, *precision*, *recall*, dan *f1-score* untuk menilai efektivitas masing-masing model. Hasil dari model-model tersebut kemudian dibandingkan dengan *baseline* LLM GPT-4o untuk menentukan model terbaik dalam klasifikasi kebutuhan perangkat lunak

4. Penentuan Model Terbaik

Tahap terakhir dalam penelitian ini adalah melakukan perbandingan dan penentuan model terbaik berdasarkan performa klasifikasi. Model yang telah dilatih dievaluasi menggunakan metrik *accuracy*, *precision*, *recall*, *f1-score*, serta *confusion matrix* untuk memahami distribusi kesalahan dalam klasifikasi. *Confusion matrix* pada Gambar 1.8 menunjukkan bagaimana model mengklasifikasikan data ke dalam kategori *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN) [24]. TP terjadi ketika model berhasil mengklasifikasikan data positif dengan benar, sedangkan FP menunjukkan kesalahan model dalam mengklasifikasikan data negatif sebagai positif. FN terjadi ketika model salah mengklasifikasikan data positif sebagai negatif, sementara TN menunjukkan keberhasilan model dalam mengenali data negatif dengan benar. Metrik *precision* mengukur ketepatan prediksi positif dibandingkan dengan semua prediksi positif yang dibuat, sedangkan *recall* menunjukkan seberapa baik model dalam menangkap semua sampel positif sebenarnya. Sementara itu, *accuracy* menghitung persentase keseluruhan prediksi yang benar, dan *f1-score* digunakan sebagai ukuran keseimbangan antara *precision* dan *recall*, terutama dalam kondisi ketidakseimbangan kelas.

		<i>ACTUAL</i>		
		<i>Positive</i>	<i>Negative</i>	
<i>PREDICT</i>	<i>Positive</i>	<i>True Positive</i>	<i>False Positive</i>	<i>Precision</i>
	<i>Negative</i>	<i>False Negative</i>	<i>True Negative</i>	
		<i>Recall</i>		<i>Accuracy</i>

$$Recall = \frac{TP}{TP + FN} \quad Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad Precision = \frac{TP}{TP + FP}$$

Gambar 1.8 Confusion Matrix ([link](#))

Selain itu, hasil evaluasi model dibandingkan dengan keluaran LLM GPT-4o, yang digunakan sebagai *benchmark* dalam tugas klasifikasi kebutuhan perangkat lunak. Perbandingan ini bertujuan untuk menilai sejauh mana model yang dikembangkan mampu mendekati atau bahkan melampaui performa model bahasa canggih saat ini. Dengan melakukan analisis terhadap *confusion matrix* dan metrik evaluasi lainnya, kelebihan dan kelemahan masing-masing model dapat diidentifikasi secara lebih mendalam. Setelah evaluasi selesai, model terbaik disimpan dan antarmuka pengguna berupa *website* dikembangkan menggunakan HTML dan Flask (sebuah *micro web framework* berbasis *Python*) untuk memungkinkan interaksi dengan model klasifikasi kebutuhan perangkat lunak.

F. Sistematika Penulisan

Sistematika laporan untuk penulisan Tugas Akhir ini disusun sebagaimana struktur berikut:

BAB I PENDAHULUAN

Menguraikan mengenai latar belakang penelitian yang menjelaskan pentingnya otomatisasi dalam analisis kebutuhan perangkat lunak. Bab ini terdiri dari sub bab rumusan masalah, pertanyaan penelitian, batasan penelitian, tujuan, metode yang digunakan, serta sistematika penulisan untuk memberikan gambaran umum tentang struktur tugas akhir ini.

BAB II TINJAUAN PUSTAKA

Membahas dan melakukan peninjauan dari penelitian terdahulu yang relevan, termasuk teknik *deep learning* dan *Natural Language Processing* (NLP) untuk klasifikasi teks, khususnya kategori kebutuhan perangkat lunak. Menyajikan analisis metode yang telah digunakan serta tabel perbandingan guna menunjukkan kontribusi penelitian ini.

BAB III LANDASAN TEORI

Menjelaskan teori dasar yang mendukung penelitian. Teori tersebut seperti konsep *deep learning*, metode klasifikasi teks, dan teknik NLP. Teori-teori ini menjadi dasar dalam pembangunan baseline dan model yang digunakan dalam penelitian ini.

BAB IV PEMBUATAN DATASET

Membahas dataset yang digunakan, termasuk proses pembuatan mulai dari pengumpulan dataset, pembersihan dataset, normalisasi dataset, dan penyeimbangan dataset. Hal ini untuk mempersiapkan dataset agar dapat diterima dalam pelatihan model.

BAB V PEMBUATAN MODEL

Pada bab ini terbagi menjadi tiga proses, pembangunan *baseline*, pembangunan model dan pembuatan antarmuka. Pembangunan *baseline* menggunakan model LLM GPT-4o dengan metode *prompting*. Setelah itu pada proses pembuatan model merupakan tahapan dimana model yang dibangun

dan dievaluasi masing-masing performa dari arsitektur yang digunakan. Selain itu, dilakukan evaluasi performa model dalam mengklasifikasikan kebutuhan perangkat lunak. Model tersebut dapat digunakan pengguna melalui antarmuka berupa *website*.

BAB VI PENUTUP

Menyimpulkan hasil penelitian terkait akurasi dan penentuan model dalam klasifikasi kebutuhan perangkat lunak. Selain itu, bab ini memberikan saran untuk pengembangan lebih lanjut, seperti peningkatan akurasi model atau penerapan dalam skenario nyata.

