BAB I
PENDAHULUAN

1.1 Latar Belakang Penelitian

Pemilihan bahasa pemrograman merupakan salah satu aspek kunci dalam membangun dasar kode yang kuat dalam mengembangkan perangkat lunak. Perkembangan perangkat lunak saat ini, bahasa pemrograman JavaScript masih merupakan bahasa pemrograman yang sering digunakan (Wang, Ji, dan Feng, 2020). Karena fleksibilitas dan kemampuannya dalam pengembangan aplikasi diberbagai platform, JavaScript tetap menjadi salah satu bahasa pemrograman yang paling populer dan relevan hingga saat ini (Ferguson, 2019). Meskipun begitu, berdasarkan buku (Tomar dan Dangi, 2021), JavaScript tidak memiliki aturan ketat dalam penulisan sintaks kode secara bawaan. Hal Ini dapat mengarah pada potensi munculnya sebuah *bug* yang sulit dideteksi saat pengembangan. *Bug* pada perangkat lunak didefinisikan sebagai cacat atau kesalahan dalam pembuatan perangkat lunak yang menyebabkan suatu fungsi tidak berjalan sebagaimana mestinya (Rodríguez-Pérez dkk., 2020). Dengan begitu kualitas perangkat lunak menurun karena tidak menerapkan aspek *Reliability* pada pengembangan perangkat lunak. Selain itu, secara bawaan JavaScript tidak memiliki alat untuk *debugging* sehingga membuat pengembang kesulitan dalam memeriksa kode JavaScript.

Untuk mengatasi permasalahan adanya kecacatan atau bugs pada kode yang diakibatkan kesalahan penulisan sintaks pada kode JavaScript, diperlukan penggunaan metode dan *tools* yang dapat membantu pengembang dalam menjaga penulisan sintaks kode agar kode JavaScript bebas dari adanya kesalahan penulisan logika. Metode umum yang sering digunakan adalah penggunaan alat analisis statis. Penggunaan alat analisis statis dalam pencegahan cacat perangkat lunak biasanya melibatkan *automation tools* untuk memeriksa cakupan kode yang dibuat oleh pengembang. Alat yang biasanya digunakan untuk mencegah kesalahan penulisan sintaks kode adalah *Linter*. Survey yang dilakukan oleh penelitian (Tómasdóttir,

Iman Nurohman, 2024
PERBANDINGAN KINERJA LARGE LANGUAGE MODEL (LLM) BERBASIS LLAMA2 DALAM MEMPERBAIKI KESALAHAN PENULISAN KODE JAVASCRIPT
Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

Aniche, dan Van Deursen, 2020a) 93% partisipan yang merupakan pengambang perangkat lunak setuju bahwa penggunaan sebuah *linter* pada kode JavaScript dapat mengurangi kesalahan logika dan kecacatan pada aplikasi yang mereka buat dengan memberikan aturan penulisan sintaks kode secara ketat. *Linter* memiliki banyak variasi jenis dan bentuknya, disesuaikan dengan bahasa pemrograman yang digunakan. Pada bahasa pemrograman JavaScript, *tools linter* yang biasanya digunakan oleh banyak pengembang adalah *ESlint*. Survey yang dilakukan (Tómasdóttir, Aniche, dan Van Deursen, 2020b), dari 337 pengembang perangkat lunak dan 15 orang expert di bidangnya, menyatakan bahwa saat mereka menggunakan *tools linter* terutama *ESLint*, dapat menjaga konsistensi penulisan sintaks kode dan dapat secara awal mendeteksi kesalahan dalam menulisakan sebuah sintaks kode.

Pada praktiknya, beberapa kesalahan kode masih memerlukan perbaikan secara manual pada kode tersebut, meskipun telah menggunakan analisis statis seperti *linter* yang dapat mendeteksi secara langsung kesalahan pada penulisan sintaks kode, pengembang masih sering kali melakukan pencarian terkait perbaikan yang harus dilakukan pada kode program pada website pencarian. Menurut (Berabi, He, Raychev, dan Vechev, 2021a), Hal yang menyebabkan permasalahan dimana sulitnya alat analisis statis mengenal jenis kesalahan penulisan kode yang berbeda, seperti contohnya kesalahan penulisan sebuah variabel dan kesalahan penulisan dari penempatan variabel bertipe angka bilangan bulat. Sebuah *tools static analytic* dapat dipercaya tetapi kurang fleksibel dalam melakukan perbaikan dan menghasilkan kode dengan gaya penulisan yang sulit dipahami antar pengembang. Berdasarkan dua penelitian yang dilakukan (Tómasdóttir dkk., 2020b) dan (Rafnsson Willardand Giustolisi, 2020) pada kesimpulannya, alat analisis statis seperti *linter* masih perlu pengembangan lebih lanjut agar dapat lebih efektif membantu pengembang dalam menulis sintaks kode yang efisien dan terhindar dari adanya bug. Salah satu penelitian yang melakukan pengembangan dalam melakukan perbaikan penulisan kode program dengan pendekatan *Deep Learning* adalah TFix (Berabi, He, Raychev, dan Vechev, 2021b). TFix dapat memperbaiki kesalahan penulisan kode JavaScript yang bersumber dari laporan

perbaikan ESLint. Dan untuk saat ini, dengan berkembangnya sebuah model *Machine Learning* dan *Deep Learning*, pendekatan yang lebih canggih telah muncul dalam mengatasi permasalahan tersebut salah satunya adalah *Large Language Model* (LLMs). *Large Language Model* (LLMs) dapat digunakan secara efektif untuk melakukan tugas spesifik pada sebuah sintaks kode seperti contohnya menghasilkan sebuah kode program yang diminta sesuai dengan instruksi yang diberikan (Fried dkk., 2022a).

*Large Language Model*s (LLMs) atau model bahasa besar, seperti ChatGPT, merupakan model *Deep Learning* yang popular dan sudah banyak digunakan oleh banyak orang. LLM menggunakan teknik *transformers* yang dilatih pada sebuah dataset yang besar. LLM merupakan sebuah model *Deep Learning* dengan arsitektur *transformers* yang dapat menyelesaikan *task Natural Language Processing* (NLP) seperti *Text Classification*, *Text Generation*, *Summarization* dan *Translation* (Cerf, 2023; Hadi dkk., 2023; Naveed dkk., 2023). Salah satu contoh model bahasa besar yang bersifat *open source* dan popular untuk saat ini adalah *Llama2*. *Llama2* ini telah mengalami proses pelatihan dan penyesuaian pada informasi data yang besar. Model *Llama2* memiliki rentang parameter antara 7 miliar hingga 70 miliar parameter (Touvron dkk., 2023). Selain itu model *Llama 2* juga terbukti dapat memahami sebuah sintaks kode dengan dilakukan *fine-tuning* atau penyesuaian terhadap sebuah dataset *source code* program python yang dinamakan *Code Llama* (Roziere dkk., 2023). Hasilnya mendapatkan kinerja terbaik di antara model bahasa besar lainnya, dengan kemampuan untuk melakukan prediksi pengisian kode program, dukungan untuk konteks khusus pada sebuah input besar, dan kemampuan untuk memahami instruksi secara nol untuk tugas pembuatan kode program. *Code Llama* mencapai kinerja terbaik di antara model *open source* lainnya pada beberapa *benchmark* atau pengujian kode program, dengan skor hingga 67% dan 65% pada HumanEval dan MBPP.

Dengan nilai benchmark dan pengujian yang baik, model yang berbasis *Llama2* terbukti dapat dilatih pada dataset sebuah program dan mengerjakan tugas yang lebih spesifik. Berdasarkan uraian diatas mengenai penulis tertarik untuk melakukan penelitian terkait "PERBANDINGAN KINERJA *LARGE LANGUAGE*

MODEL (LLM) BERBASIS *LLAMA2* DALAM MEMPERBAIKI KESALAHAN PENULISAN KODE JAVASCRIPT*"* untuk mengetahui sejauh mana model berbasis *Llama2* mengerjakan tugas yang lebih spesifik yakni melakukan perbaikan kesalahan pada penulisan sintaks kode program secara otomatis. Berdasarkan penelitian (Fried dkk., 2022a; Lu, Yu, Li, Yang, dan Zuo, 2023) dengan menggunakan model dengan variasi dasar 7B parameter, model dengan variasi parameter lebih banyak seperti variasi 13B dan 70B, dapat dipastikan model tersebut dapat juga dilatih pada tugas yang sama dengan akurasi yang lebih baik. Model-model berbasis *Llama2* dilatih pada dataset yang telah diformat sebelumnya dengan mencoba untuk menambahkan optimasi pada pelatihan model seperti *Parameter Efficient Fine-Tuning* (PEFT), QLoRa dan *Supervised Fine-Tuning Trainer* (SFT) pada model *Llama2*. Dalam hal ini kode JavaScript dan data *error* ataupun warning yang dilaporkan oleh *ESLint* dijadikan sebagai subjek pada penelitian ini dengan dataset yang telah digunakan pada penelitian sebelumnya. Pengujian model ini akan melakukan perbandingan dengan model *TFix*, yang juga meneliti model pemrosesan bahasa alami dalam perbaikan kesalahan sintaks JavaScript berdasarkan laporan *ESLint*.

1.2 Rumusan Masalah Penelitian

Berdasarkan latar belakang yang telah dipaparkan mengenai permasalahan tersebut, menghasilkan rumusan masalah sebagai berikut :

1) Apakah penerapan (*fined-tuning)* pada *Large Language Model* (LLM) berbasis *Llama2* dapat mengerjakan tugas spesifik untuk memperbaiki kesalahan pada kode JavaScript ?
2) Bagaimana perbedaan hasil skor akurasi perbaikan *Large Language Model* (LLM) berbasis *Llama2* dalam memperbaiki kesalahan penulisan sintaks kode JavaScript dibanding dengan model *TFix* ?
3) Bagaimana hasil skor uji evaluasi dengan menggunakan metrik BLEU dan CodeBLEU dalam mengukur kualitas kode perbaikan yang dihasilkan *Large Language Model* (LLM) berbasis *Llama2* ?

1.3 Tujuan Penelitian

Dalam penelitian ini menghasilkan tujuan penelitian, yaitu:

1) Melakukan penerapan (*fined-tuning)* pada *Large Language Model* (LLM) berbasis *Llama2* sehingga dapat memperbaiki kesalahan pada kode JavaScript.
2) Mengetahui perbedaan hasil skor akurasi perbaikan *Large Language Model* (LLM) berbasis *Llama2* dalam memperbaiki kesalahan penulisan sintaks kode JavaScript dibanding dengan model *TFix*.
3) Mengetahui hasil skor uji evaluasi dengan menggunakan metrik BLEU dan CodeBLEU dalam mengukur kualitas kode perbaikan yang dihasilkan *Large Language Model* (LLM) berbasis *Llama2*.

1.4 Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut :

1) Mengetahui perbedaan hasil kinerja akurasi pada *Large Language Model* (LLM) yang berbasis *Llama2* dengan model sebelumnya dalam melakukan perbaikan kesalahan pada JavaScript.
2) Memberikan informasi terkait hasil kode yang dihasilkan *Large Language Model* (LLM) berbasis *Llama2* dalam memperbaiki kode JavaScript.
3) Kemampuan untuk melakukan perbaikan kesalahan program pada kode JavaScript yang dapat membantu tim pengembang untuk meningkatkan kualitas kode JavaScript yang efektif dan efisien dalam mengurangi kesalahan yang mungkin terjadi dalam pengembangan perangkat lunak.
4) Hasil penelitian ini dapat menjadi studi kasus untuk penelitian lebih lanjut, dalam melakukan pendekatan dengan *Large Language Model* (LLM) dan memungkinkan peneliti lain untuk membangun dan memperdalam konsep yang diusulkan.

1.5 Batasan Masalah

1) Penelitian difokuskan untuk melakukan *fine-tuning* pada model *pre-trained Llama2*, *CodeLlama* dan *CodaLlama Instruct* dengan Chat dengan varian model 7B parameter.
2) Penelitian ini berfokus pada perbaikan penulisan sintaks kode yang berpotensi menyebabkan kesalahan logika dalam penulisan kode JavaScript.
3) Mengoptimasi pelatihan model dengan menggunakan *Parameter Efficient Fine-Tuning (PEFT)* dan QLoRA.

1.6 Struktur Organisasi Skripsi

Struktur Organisasi pada penelitian ini menggunakan sistematika dengan detail sebagai berikut :

1) Bab I Pendahuluan

Pada bab pertama yakni Pendahuluan, terdapat penyajian lengkap mengenai kerangka dasar penelitian meliputi latar belakang penelitian yang menjelaskan alasan dibalik penelitian, rumusan masalah yang didasarkan pada latar belakang, tujuan yang ingin dicapai melalui penelitian ini, serta manfaat yang diharapkan dari hasil penelitian. Terakhir, bagian ini juga mencakup penjelasan mengenai sistematika organisasi penulisan penelitian.

2) Bab II Kajian Pustaka

Bab kedua menyajikan penjelasan kerangka teoritis yang digunakan pada penelitian, yang didasarkan pada kajian literatur serta temuan dan rekomendasi dari penelitian terdahulu. Hasil dan saran dari penelitian terdahulu dijadikan sebagai rancangan model referensi dan acuan dalam pembuatan model pada penelitian ini.

3) Bab III Metode Penelitian

Bab ketiga merinci metode-metode yang telah diuraikan dalam kajian pustaka, yang akan diterapkan sebagai solusi untuk mengatasi permasalahan dalam penelitian ini. Bagian ini menguraikan struktur penelitian, termasuk desain penelitian, alur penelitian, instrumen penelitian, serta analisis data yang akan diterapkan dalam penelitian ini.

4) Bab IV Temuan dan Pembahasan

Pada Bab keempat menjelaskan tahapan pengembangan model deteksi bugs pada kode JavaScript. Dalam bab ini, hasil perbandingan dan analisis kinerja dari model deteksi bugs yang telah dilatih dengan menggunakan metode yang telah dijelaskan dalam metodologi penelitian dipresentasikan. Eksperimen ini bertujuan untuk meningkatkan akurasi model dan mencapai temuan yang dapat memberikan dampak positif dalam penggunaan model ini.

5) Bab IV Penutup

Bab kelima, yang merupakan bab terakhir, merangkum temuan dari penelitian ini menjadi kesimpulan yang kuat. Hasil penelitian ini akan digunakan untuk memberikan saran selanjutnya yang dapat memberikan manfaat dan menjadi acuan bagi penelitian-penelitian yang akan datang.