1. Pendahuluan

Latar Belakang

Pengembangan perangkat lunak adalah proses yang kompleks dan rentan terhadap kesalahan. *Bug* atau cacat pada perangkat lunak dapat menimbulkan konsekuensi serius, mulai dari ketidaknyamanan pengguna hingga kerugian finansial yang signifikan [1]. Oleh karena itu, perbaikan *bug* menjadi tugas penting dalam siklus pengembangan perangkat lunak. Namun, proses perbaikan *bug* secara manual memakan waktu dan membutuhkan sumber daya yang besar [2]. Di sinilah Automated Program Repair (APR) berperan penting dalam meningkatkan efisiensi dan mengurangi biaya perbaikan *bug*.

APR telah menjadi bidang penelitian yang aktif dalam beberapa tahun terakhir. Berbagai pendekatan telah diusulkan, seperti *search-based, constraint-based, template-based,* dan *learning-based* [3]. Salah satu perkembangan terbaru dalam APR adalah pemanfaatan Large Language Models (LLM) seperti GPT-3 dan Codex. Codex adalah model yang dirancang oleh OpenAI untuk menangani berbagai tugas pemrograman, termasuk pembuatan kode, penyelesaian masalah, dan perbaikan *bug* [4]. Selain itu, Codex juga merupakan model yang menjadi inti dari GitHub Copilot, sebuah alat yang membantu pengembang menulis kode dengan lebih cepat dan efisien [5].

Namun, performa LLM dalam memperbaiki *bug* masih bisa ditingkatkan. Prenner et al. [6] melakukan evaluasi terhadap kemampuan Codex dalam memperbaiki *bug* pada dataset QuixBugs. Mereka menemukan bahwa *prompt* memiliki pengaruh yang signifikan terhadap performa Codex. *Prompt* yang tidak tepat dapat menyebabkan Codex menghasilkan perbaikan yang tidak benar atau bahkan memperburuk *bug* yang ada. Oleh karena itu, pengembangan teknik *prompting* yang efektif menjadi kunci untuk meningkatkan performa LLM dalam APR.

Prompt adalah sekumpulan instruksi yang diberikan kepada LLM untuk menghasilkan respons atau output tertentu [7]. Dalam konteks penggunaan LLM, *prompt* sering kali dirancang dengan teknik khusus yang disebut *prompt engineering*. *Prompt engineering* adalah metode untuk merancang *prompt* secara sistematis untuk mengarahkan LLM menghasilkan output atau mencapai tujuan tertentu [7]. Teknik ini melibatkan desain pola atau template yang efektif untuk membantu model memahami tugas yang diberikan, terutama pada kasus yang membutuhkan *reasoning* atau penyelesaian masalah yang kompleks.

Salah satu konsep *prompt engineering* yang belum banyak dieksplorasi dalam APR adalah Chain-of-Thought (CoT) Prompting. CoT Prompting adalah teknik *prompting* yang memandu LLM untuk menghasilkan penjelasan *step-by-step* sebelum memberikan jawaban akhir [8]. Teknik ini telah terbukti efektif dalam meningkatkan performa LLM pada berbagai tugas, seperti aritmatika dan penyelesaian masalah [9]. CoT Prompting memungkinkan LLM untuk memecah masalah kompleks menjadi sub-masalah yang lebih sederhana dan memberikan penjelasan yang lebih terstruktur. Dalam konteks APR, penggunaan CoT Prompting dapat membantu LLM untuk lebih baik memahami konteks *bug*, mengidentifikasi penyebab *bug*, dan menghasilkan perbaikan yang benar. Namun, potensi CoT Prompting dalam APR masih belum banyak dieksplorasi.

Sehingga, penelitian ini bertujuan untuk mengekplorasi potensi dan merancang teknik CoT Prompting yang efektif dengan mengacu pada literatur yang relevan untuk meningkatkan performa LLM dalam APR. Pada penelitian ini, struktur *prompt* yang dirancang akan dibandingkan dengan pendekatan Standard Prompting untuk mengevaluasi peningkatan performa serta efisiensi biaya. Standard Prompting adalah pendekatan sederhana di mana model diberikan pertanyaan atau masalah, lalu secara langsung menghasilkan jawaban akhir. Sementara itu, CoT Prompting memandu model untuk memberikan penjelasan *step-by-step* sebelum menghasilkan jawaban akhir. Evaluasi dilakukan dengan menggunakan dataset QuixBugs [10], sebuah *benchmark* yang dirancang untuk menguji kemampuan APR. Dataset ini telah digunakan secara luas dalam berbagai penelitian APR untuk mengevaluasi teknik perbaikan kode [11]. Hasil evaluasi mencakup analisis performa LLM dalam menghasilkan perbaikan kode yang benar pada dataset QuixBugs, estimasi biaya yang dihitung berdasarkan jumlah token yang digunakan, serta identifikasi pola kegagalan model dalam memperbaiki berbagai jenis bug.

Topik dan Batasannya

Topik atau permasalahan yang diteliti dalam penelitian ini adalah bagaimana merancang teknik CoT Prompting yang efektif untuk APR. Selain itu, penelitian ini juga membandingkan performa beberapa model LLM dengan menguji dua mekanisme perancangan *prompt*, yaitu Standard Prompting dan CoT Prompting. Adapun batasan dalam penelitian ini mencakup penggunaan dataset yang terbatas hanya pada versi Python dari dataset QuixBugs, serta model LLM yang digunakan hanya model yang dapat diakses secara publik melalui Application Programming Interface (API). Penelitian ini menggunakan 10 model LLM, meliputi gpt-4o, ol-preview, ol-mini, claude-3.5-sonnet, llama-3.3-70b, gemini-1.5-pro, gemini-1.5-flash, grok-beta, grok-2, dan deepseek-v3.

Tujuan

Tujuan dari penelitian ini adalah merancang teknik CoT Prompting yang efektif untuk APR dan membandingkan performa berbagai model LLM dalam menerapkan pendekatan Standard Prompting dan CoT Prompting yang telah dirancang.

Organisasi Tulisan

Organisasi tulisan dalam penelitian ini dimulai dengan bagian pendahuluan yang mencakup latar belakang, topik dan batasan, serta tujuan penelitian. Bagian studi terkait memaparkan literatur review yang relevan dengan penelitian ini. Bagian sistem yang dibangun menjelaskan metodologi yang digunakan dan bagaimana evaluasi dilakukan. Bagian evaluasi memuat hasil eksperimen dan analisisnya. Penelitian diakhiri dengan kesimpulan yang diambil berdasarkan hasil evaluasi.