대규모 언어 모델(LLM)을 활용한 코드 생성 성능 비교 연구

김현주[1], 우상욱[1], 정보원[1], 조관우[1], 김형수[2], 정설영[1]
[1]경북대학교 컴퓨터학부
[2]유아이패스 코리아
rnfma115@gmail.com, sangwook0808@gmail.com, qh0614dd@knu.ac.kr,
qzwsx5123@gmail.com, hyungsoo.kim@uipath.com

Comparative Analysis of Code Generation Performance Using Large Language Models

Hyeonju-Kim[1], Sanguk-Woo[1], Bowon-Jeong[1], Kwanwoo-Jo[1], Hyungsoo-Kim[2]
Seolyoung-Jung[1]
[1]Dept. of Computer Science, Kyung-Pook National University
[2]Principal Solution Enginer of UiPath

요 약

최근 GPT 시리즈, Anthropic의 Claude, Google의 Gemini 등 첨단 대규모 언어 모델(LLM)이 소프트웨어 개발에서 코드 생성에 활발히 활용되고 있다. 본 연구에서는 Claude Sonnet 4, Gemini 2.5 Pro, GPT-5 세 모델이 동일한 요구사항으로 생성한 코드를 비교 평가하였다. “가챠 무인 판매기 - 장치 등록 시스템” 요구사항을 1차(간략 설명)와 2차(세부 명세)로 나누어 각 모델에 제공하였으며, 입력검증, 오류처리, 보안, 로직 구현의 정확성·가독성, 모듈화, RESTful URI 설계, 응답 구조, 확장성 등 항목별로 10점 척도 평가 후 가중치를 적용하여 종합점수를 산출하였다. 실험 결과 Claude Sonnet 4가 가장 높은 성능을 보였으며, GPT-5는 요구사항이 상세해질수록 급격히 향상되었고, Gemini 2.5 Pro는 단순 구현 특성으로 인해 개선폭이 작았다. 본 연구는 대규모 언어 모델의 코드 생성 능력을 정량적으로 비교함으로써 적절한 모델 선택과 활용 지침을 제공하는 것을 목적으로 한다.

1. 서론

소프트웨어 개발 분야에서 대규모 언어 모델(LLM)을 통한 자동 코드 생성에 대한 관심이 높아지고 있다. 특히 GPT 시리즈와 Claude 등의 첨단 AI 모델은 주어진 요구사항만으로도 동작 가능한 프로그램 코드를 생성할 수 있다고 알려져 있다. 본 연구에서는 “가챠 무인 판매기 - 장치 등록 시스템”이라는 동일한 요구사항을 세 가지 서로 다른 LLM 기반 코드 생성 모델에 제공하고, 생성된 코드의 품질을 비교하였다. 비교 대상 모델은 Gemini 2.5 Pro, GPT-5, Claude Sonnet 4 이다.

본 연구의 목적은 동일한 요구사항에 대해 각 모델이 산출하는 소프트웨어 구현의 완성도, 아키텍처 적합성, 코드 품질 등을 객관적 기준으로 살펴보는 것이다. 또한 이전의 코드 생성 연구들과 달리 실제 요구사항 시나리오를 활용하여 최신 LLM 모델들의 구현력을 정량 평가하였고, 요구사항 명세의 구체성이 결과에 미치는 영향도 함께 관찰하였다.

2. 실험 설계

실험에서 사용한 1차 요구사항은 개략적 기능 설명 중심으로 구성되었고, 이후 2차 요구사항으로 상세한 API 규격과 예시를 추가 제공하여 모델의 코드 생성 변화도 관찰하였다[1]. 각 모델에는 동일한 텍스트 요구사항을 프롬프트로 입력하였으며, 출력된 코드는 항목별 평가 기준에 따라 점수화했다. 평가 기준은 입력값 검증(필수값 확인, 형식/범위 검증), 오류 처리(예외 구체성, 로깅, HTTP 코드), 보안 요소(인증·인가, 민감정보 보호), 로직 디테일(요구사항 구현 정확성, 코드 가독성, 주석), 모듈화 구조(Controller/Service/Repository 분리, 의존성 주입, 재사용성), API URI/메소드(RESTful 설계), 응답 일관성(응답 구조 통일성, 필드 명명 규칙), 확장성(하드코딩 최소화) 등을 포함하며, 각 범주에 가중치를 부여하였다. 세부 기준마다 1~10점 척도로 평가한 후, 가중 평균을 계산하여 모델별 총점을 산출하였다. 이 과정에서 점수화된 각 항목별 우수 사항 및 미비점을 별도 기록하여 질적 비교도 병행하였다[2].

3. 실험 결과 및 분석

1차 요구사항 평가 결과 Claude Sonnet 4의 점수가 가장 높았고, Gemini 2.5 Pro와 GPT-5가 그 뒤를 이었다. Claude Sonnet 4는 Express.js 기반의 엔터프라이즈급 아키텍처를 활용해 DI 패턴(의존성 주입)을 완전 구현하고, 상세 예외 클래스와 감사 로깅을 갖추는 등 안정성과 구조 품질에서 우수했다. 입력 검증과 오류처리가 철저하여 응답 완성도가 높았으며, 주석은 상대적으로 적었으나 핵심 로직 설명이 충실했다. Gemini 2.5 Pro는 NestJS 프레임워크의 데코레이터 기반 검증과 의존성 주입을 효과적으로 활용하여 모던한 아키텍처를 구현했으나, 관리자 승인/거부 API가 누락되어 있고 보안 인증 체계가 상대적으로 미흡하였다. GPT-5는 최신 NestJS 프레임워크와 데코레이터 기반 구조를 활용했으나, 입력 검증과 에러 처리의 완성도가 낮아 실제 운영 환경에서 안정성 우려가 존재하였다. 1차 실험을 바탕으로 요구사항을 상세화한 2차 실험도 진행하였다. 표 1은 1차 요구사항과 2차 요구사항 명세의 주요 기능을 비교한 것이다.

항목	1차 버전 요구사항	2차 버전 요구사항
요구사항 수준	간략한 기능 설명 중심	상세 API 명세 및 동작 흐름 포함
코드 생성 대상	추상적인 지시문 중심	SRP 기반 계층형 아키텍처 명시
API 명세	거의 없음	메서드, URI, 본문, 응답 예시 포함
비즈니스 로직	장치 등록 및 승인 여부	등록 → 상태 조회 → 승인/거부 → 큐 발급 로직 명확화
보안 고려	관리자 기능만 JWT 인증	관리자 JWT + 장치 인증 제외 명확 분리
확장성 고려	없음	SQS 모듈 추상화로 Kafka 전환 가능성 고려
장애 처리	미정	중복 등록, 실패 응답 코드 및 메시지 설계 포함

<표 1> 차 요구사항과 2차 요구사항 명세 비교

요구사항을 상세화한 2차 실험에서는 모든 모델의 성능이 상승하였는데, 특히 GPT-5가 가장 크게 향상되어 70.32점까지 상승하였다. 이는 GPT-5가 초기에는 일부 기능(예: JWT 인증, 등록 승인/거부 로직) 누락이 있었으나 2차 요구사항 제공 후 입력 DTO 검증 및 DI 개선으로 점수가 크게 상승하였다. 이는 구체적 명세로 보다 완전한 구현을 수행했음을 보여준다. Claude Sonnet 4는 기존에 높은 수준의 완성도를 보였으나 추가 세부 요구사항으로 코드 품질이 더 정교해져 82.21점으로 증가하였다. Gemini 2.5 Pro는 이미 단순하고 직관적이었기 때문에 개선 폭이 미미하여 70.72점에 머물렀다.

이와 같이 Claude Sonnet 4는 처음부터 충실한 구현과 안정성을 바탕으로 최고 점수를 유지했으며, GPT-5는 신중한 설계 철학에 따라 복잡한 기능 구현에서 강점을 보였다. Gemini 2.5 Pro는 코드 작성 속도와 문서화에 장점이 있지만 복잡도 높은 요구사항에 대한 대응은 제한적이었다. 표 2는 세 모델의 구현 특징과 평가 점수를 요약한 것이다.

구현체(모델)	주요 특징	평가 점수(1차-> 2차)
Claude Sonnet4	엔터프라이즈급 아키텍처, 철저한 입력 검증 및 예외 처리	75.91점 -> 82.21점
Gemini 2.5 Flash	교육용으로 최적화된 단순 구현, 풍부한 주석	70.57점 -> 70.72점
GPT-5	최신 프레임워크 활용, 데코레이터 기반 구조	54.31점 -> 70.32점

<표 2> 모델별 구현 특징 및 1차, 2차 평가 점수 비교

4. 결론

본 연구에서는 장치 등록 시스템이라는 현실적인 시나리오에 대해 세 종류의 대규모 언어 모델이 출력한 결과물을 비교 평가하였다. Claude Sonnet 4는 엔터프라이즈급 아키텍처 및 철저한 검증 로직으로 모든 평가 항목에서 우수한 점수를 받았고, GPT-5는 상세 명세 제공 시 특히 보안·유효성 검사 부문이 크게 향상되며 경쟁력을 갖추었다. Gemini 2.5 Pro는 상대적으로 간결하고 속도 지향적인 구현을 보여 모델별 특색이 뚜렷했다.

이 연구는 제한적으로 하나의 시나리오에 대해 진행되었다. 향후 연구에서는, 다양한 규모와 복잡도의 다른 프로젝트에도 동일한 평가를 적용하여 일반화된 경향을 파악하고, 모델 성능을 향상시키기 위한 프롬프트 최적화 기법, 인간-AI 협업 방식 등에 대해 추가로 탐구할 예정이다. 본 연구가 AI 코딩 비서의 실용성 평가에 기여하고, 소프트웨어 공학 분야에서 LLM 활용의 장단점을 이해하는 데 도움이 되기를 기대한다.

사사

본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음.(2021-0-01082)

참고문헌

[1] Austin et al. "Program synthesis with large language models." arXiv preprint arXiv:2108.07732 .(2021)

[2] Chen et al, "Evaluating large language models trained on code." arXiv preprint arXiv:2107.03374. (2021)