대규모 언어 모델(LLM)의 종류별 프로그래밍 성능 비교

한동근, 최승빈, 김보성, 배성민, 송성민, 신병철, 박세진*
계명대학교 컴퓨터공학과
e-mail : ehdrms001030@naver.com, csb8226@naver.com, kimbosung22@naver.com, smin9e@naver.com , song47166@naver.com, sb5811c98@gmail.com, baksejin@kmu.ac.kr

Comparison of programming performance by type of Large Language Model (LLM)

Dong Geun Han, Seung Bin Choi, Bo Sung Kim, Seong Min Bae, Sung Min Song, Byung Cheol Shin, Se Jin Park*
Department of Computer Engineering
Keimyung University

Abstract

With the development of AI, language models such as Chat GPT, a leading player in large-scale language models (LLM), are widely used in various fields, including programming. The purpose of the study is to compare and analyze which language models are more helpful in performing programming (coding) tasks, as it is expected that using large-scale language models will increase work efficiency.

I. 서론

4차 산업혁명의 핵심 기술로 꼽히는 AI 기술은 다양한 방향으로 발전을 거듭하던 중, 인간이 자연어로 요구사항을 작성하여 요청하면 그에 맞는 답이나 해결책을 제공하는 대규모 언어 모델(Large Language Model)이 등장하게 되었다. 대규모 언어 모델이란[1] 방대한 양의 텍스트 데이터를 학습하여 이용자의 특정 요구에 따라 결과를 생성하는 생성형 AI(Generative AI)의 일종이다. LLM은 트랜스포머(Transformer) 아키텍처를 기반으로 하며, 셀프 어텐션(Self-attention) 메커니즘을 통해 문맥 정보를 효과적으로 파악할 수 있는 특징이 있다[2].

현재 대규모 언어 모델의 종류로는 ChatGPT, PaLM, LLaMa 등이 있다. 특히 'ChatGPT'는 인간과 비슷한 수준으로 작문하고, 자연어로 질문한 내용에 대해서 이해하는 수준이 점차 높아지고 있으며 이해한 내용을 바탕으로 사용자에게 원하는 답이나 해결책 등을 제공하고 있다. LLM은 프로그래밍 및 코딩 분야에서도 활용되고 있다.

본 논문에서는 GPT를 필두로 하여 생겨난 다양한 LLM 종류 중 OpenAI의 GPT-4, Google의 Gemini, Anthropic의 Claude3 Opus, Github Copilot 총 4가지를 선정하여 알고리즘 구현 능력을 비교 분석하고자 한다. 이를 통해 알고리즘 학습에 도움 되는 모델을 분석하고, 가장 효율적으로 활용될 수 있는 모델을 파악하는 것이 주요 목적이다.

II. 본론

2.1 연구 방법

비교분석에 사용할 LLM 4가지로 GPT-4, Gemini, Claude3 Opus, Github Copilot을 선정하였다. 알고리즘 구현에 사용할 데이터 셋은 국내 코딩 테스트 사이트인 '백준'[3]의 문제 난이도에 따른 단계를 분류해놓은 solved.ac[4]에서 활용하였다. 백준은 국제 대학생 프로그래밍 대회인 ICPC, 한국정보올림피아드 등 각종 대회에서 출제된 알고리즘 문제들을 모아놓은 사이트로 다양한 유형의 문제들을 보유하고 있다. 각 문제에는 문제의 설명, 입력과 출력, 그리고 어떤 자료구조나 알고리즘이 사용되었는지 알려준다. 문제에 따른 답안을 제출하면 그 즉시 채점이 이루어지고, 정답 시 코드의 공간 복잡도, 소요 시간, 코드 길이 등을 분석해준다. 오답 시에는 컴파일 에러, 런타임 에러 등 실패 사유를 간단하게 알려준다.

각 LLM에게 질의할 문제는 120개를 선정하였다. 문제 선정 과정은 관련 논문[5] 을 참고하여 해당 논문의 기준과 유사하게 맞추어 연구를 진행했다. 문제의 난이도는 브론즈, 실버, 골드, 플래티넘, 다이아몬드, 루비 총 6개의 단계로 분류된다. 또 분류된 각 단계에서 V~I까지 5개의 세부 단계로 분류된다. 각 세부 단계에서 정답자가 많은 문제들이 LLM 모델에 학습이 많이 됐을 것이라 판단해 푼 사람이 많은 순으로 4개의 문제를 선택하였고, 결과적으로 총 120문제를 선정하게 되었다(6개의 단계 X 5개의 세부 단계 X 4개 문제). 문제 설명에 그림이 포함된 것 중 문제 해결에 필수적으로 그림이 필요한 문제, 영어로 된 문제들은 제외하였다.

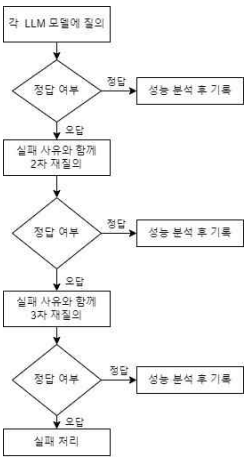

그림 1. 알고리즘 질의 순서도

그림 1은 LLM 모델에게 알고리즘 문제를 질의하는 순서도이다. 질의할 프롬프트는 프롬프트 조합에 관련된 논문[6]을 참고하였다. 각 문제의 설명과 어떤 값이 입력되고 출력되어야 하는지 LLM에게 1차 질의한다. 1차 질의에서 정답 처리가 될 시 그에 따른 성능 분석과 기록을 한다. 오답 처리 시 실패 사유를 포함해 재질의를 하고, 3차 질의까지도 오답 처리가 된다면 해당 문제에 대한 알고리즘 구현 능력이 없는 것으로 판단하고 실패 처리한다.

표 1. 성능 분석표 작성 예시

LLM	시도 횟수	소요 시간 (ms)	공간 복잡도 (KB)	코드 간결성 (Byte)	정답 여부
GPT-4	1	512	1000	684	1
Github Copilot	1	648	1684	591	1
Claude3 Opus	1	598	1870	798	1
Gemini	1	515	1684	900	1

표 1은 정답 처리가 될 경우 모델별로 기록할 성능 기준표의 예시이다. 정답으로 채점이 될 시 사이트에서 분석한 성능 데이터인 소요 시간(ms), 공간 복잡도(KB), 코드 간결성(Byte), 정답 여부가 나오게 된다. 정답 여부는 정답일 시 1, 오답일 시 0으로 구분한다. 이에 추가로 LLM에게 질의한 횟수를 추가해 시도 횟수를 기록한다.

2.2 연구 결과

정답자가 많은 순으로 선정한 120개 문제들의 채점 결과에 따른 LLM 별 성능 분석표를 기록한 후 6개 단계별로 평균 성능 분석표를 작성하였다.

표 2. 브론즈 단계 평균 성능 분석표

LLM	시도 횟수	소요 시간 (ms)	공간 복잡도 (KB)	코드 간결성 (Byte)	정답률 (%)
GPT-4	1	332	40193	795	100
Github Copilot	1	423	50532	438	100
Claude3 Opus	1.05	299	33203	513	100
Gemini	1.5	510	46419	517	80

브론즈 단계에서는 조건문, 반복문, 문자열 등 기본적인 문법과 수학적 사고를 이용한 구현 문제들이 출제되었다. Gemini를 제외한 3개의 LLM 모두 대부분의 문제를 1차 시도에 풀어내는 높은 성공률을 보였다. Gemini의 경우 문자열을 활용하는 일부 문제에서

알고리즘 구현에 어려움을 겪으며 실패처리 되는 모습을 보였다. 코드 간결성 면에서는 Github Copilot이 가장 우수했으며, 소요 시간과 공간 복잡도는 Claude3 Opus가 가장 효율적이었다. 브론즈 단계에서 LLM 간의 큰 성능 격차는 발견되지 않았다.

표 2. 실버 단계 평균 성능 분석표

LLM	시도 횟수	소요 시간 (ms)	공간 복잡도 (KB)	코드 간결성 (Byte)	정답률 (%)
GPT-4	1	373	30268	1331	100
Github Copilot	1.2	480	38654	930	95
Claude3 Opus	1.05	460	38446	1047	100
Gemini	2.4	299	19918	1098	40

실버 단계에서는 자료구조, 재귀함수, 브루트 포스, 그리디 알고리즘 등의 유형들이 출제되었다. GPT-4와 Claude3 Opus는 100%의 정답률을 유지한 반면, Github Copilot은 소폭 하락하였고 Gemini는 40%까지 하락하였다. Gemini의 경우 평균 시도 횟수도 2.4회로 증가하였는데, 실패 처리된 문제를 분석해 본 결과 재귀 호출이나 브루트 포스와 같은 반복 패턴을 인식하고 적용하는 능력이 학습되지 않은 것으로 보인다. 정답률 대비 코드 간결성은 여전히 Github Copliot이 우수하고, 소요 시간과 공간복잡도는 GPT-4가 가장 효율적이었다.

표 3. 골드 단계 평균 성능 분석표

LLM	시도 횟수	소요 시간 (ms)	공간 복잡도 (KB)	코드 간결성 (Byte)	정답률 (%)
GPT-4	1.25	921	96532	2076	95
Github Copilot	1.15	1068	113160	1735	95
Claude3 Opus	1.1	1299	145332	1891	100
Gemini	2.7	2348	113211	1531	20

골드 단계에서는 동적 프로그래밍, 백트래킹, 다익스트라, 벨만-포드 등의 고난도 알고리즘 유형들이 출제되었다. Gemini를 제외한 3개 LLM 은 준수한 정답률을 유지하였다. Gemini의 경우 대부분의 문제를 풀지 못하고 20%까지 정답률이 하락하는 모습을 보여주었는데, 이는 고난도 알고리즘에 대한 학습이 되어있지 않은 것으로 보인다. Claude3 Opus는 정답률 면에서 100%로 가장 우수했으나, 소요 시간과 공간 복잡도는 GPT-4에 비해 높게 나타났다. 코드 간결성 면에서는 여전히 Github Copilot이 우수하고, 소요 시간과 공간 복잡도 면에서도 GPT-4가 가장 효율적인 구현 능력을 보여주었다.

표 4. 플래티넘 단계 성능 분석표

LLM	시도 횟수	소요 시간 (ms)	공간 복잡도 (KB)	코드 간결성 (Byte)	정답률 (%)
GPT-4	2.15	1480	127030	2324	50
Github Copilot	2.15	2090	200580	1775	45
Claude3 Opus	2.5	1422	186738	2010	35
Gemini	3	-	-	-	0

플래티넘 단계에서는 세그먼트 트리, 네트워크 플로우, 고급 동적 프로그래밍, 고난도 그래프 알고리즘 등 복합적인 알고리즘과 자료구조를 활용해야 하는 문제들이 출제되었다. 이 단계에서는 모든 LLM의 성능이 저하되는 모습을 보였다. 틀리는 유형은 모델 간 비슷하게 분석되었는데, 이분 매칭, 고난이도의 동적 프로그래밍과 그래프 탐색 유형에서 모두 실패 처리가 되었다. Gemini의 경우 단 하나의 문제도 해결하지 못하였는데, 이는 Gemini의 알고리즘 구현 능력이 고난도 문제를 다루기에는 모델 학습이 부족하다는 것을 나타낸다. 코드 간결성 면에서는 여전히 Github Copilot이 우수하고, 소요 시간과 공간 복잡도를 종합적으로 봤을 때 GPT-4이 가장 효율적이다.

다이아몬드 단계에서 GPT-4와 Github Copilot은 단 하나의 문제만을 해결했으며, Claude3 Opus와 Gemini는 모든 문제를 해결하지 못하였다. 나아가 루비 단계에서는 모든 LLM이 한 문제도 해결하지 못하는 모습을 보여주었다

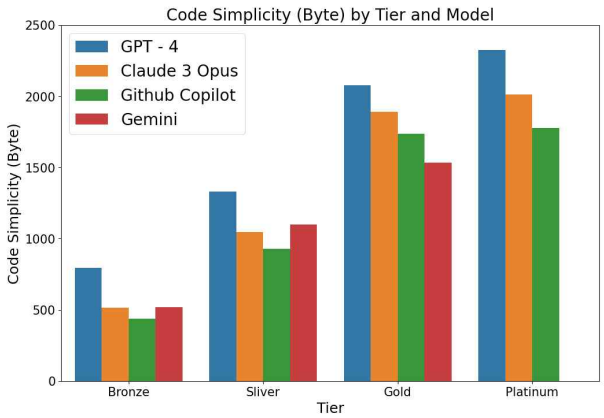

그림 2. 단계별 평균 코드 간결성 시각화

그림 2는 6개 단계별 평균 코드 간결성을 시각화한 자료다. 다이아몬드와 루비 단계는 성능을 분석하는데 유의미한 데이터가 분석되지 않아 제외하였다. 코드의 간결성은 가독성과 직결되는 만큼 알고리즘 구현 능력을 평가하는 주요 지표 중 하나로 볼 수 있다. 코드의 길이는 단계가 상승함에 따라 점점 증가하는데, 이는 난이도 상승에 따른 자연스러운 결과로 해석된다. 정답률 대비 코드의 간결성을 보면 Github Copliot이 모든 단계에서 전반적으로 우수적이다. 이는 Github Copliot이 자체적으로 방대한 오픈소스 코드를 학습한 결과로 판단된다.

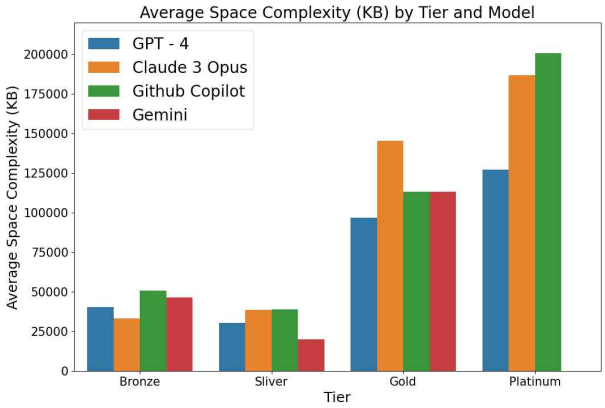

그림 3. 단계별 평균 공간 복잡도 시각화

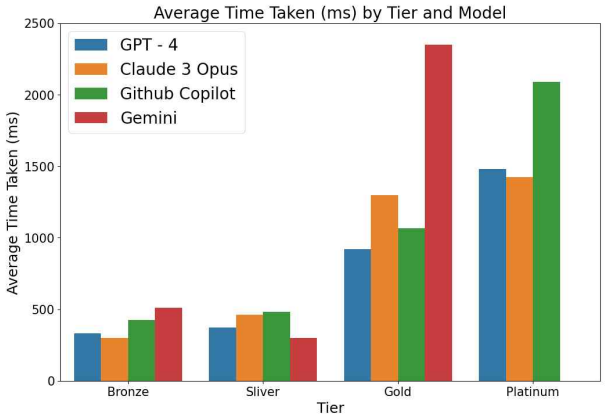

그림 4. 단계별 평균 소요 시간 시각화

그림 3과 그림 4는 각각 6개 단계별 평균 공간 복잡도, 소요 시간을 시각화한 자료다. 앞선 코드 간결성 시각화 자료에서 Github Copilot이 가장 우수한 성능을 보인 반면, 소요 시간과 공간 복잡도 측면에서는 다소 떨어지는 모습을 보였다. 플래티넘 단계에서는 2090ms로 가장 긴 실행 시간을 기록하였다. 이는 Github Copilot이 간결한 코드를 생성하는 대신 실행 속도 면에서는 상대적으로 최적화되지 않은 알고리즘을 구현하고 있음을 보여준다.

공간 복잡도 또한 그림 4에서 보는 바와 같이 Github Copilot이 전반적으로 높은 수치를 나타냈다. 브론즈와 실버 단계에서는 각각 50532KB, 38654KB로 가장 많은 메모리를 사용하였고, 플래티넘 단계에서도 두 번째로 높은 공간 복잡도를 기록하였다.

GPT-4의 경우 코드의 길이는 길지만 모든 단계에서 소요 시간과 공간 복잡도 면에서 우수한 성능을 보였다. 이는 GPT-4가 알고리즘 구현에 있어 실행 속도와 메모리 효율성을 고려한 최적화된 코드를 생성해 낼 수 있음을 보여준다.

Claude3 Opus는 브론즈 단계에서 가장 빠른 소요 시간, 실버 단계에서 GPT-4에 이어 두 번째로 빠른 실행 속도를 보였으나 골드와 플래티넘 단계에서는 소요 시간이 크게 증가하는 모습을 나타냈다. 공간 복잡도 면에서는 중상위권의 성능을 유지하였다.

Gemini는 모든 단계에서 전반적으로 좋지 않은 성능을 보여주었다. 이는 Gemini가 알고리즘 구현에 있어 데이터 학습이 부족하다고 판단된다.

분석 결과를 종합해볼 때, Github Copilot은 간결한 코드 생성 능력에 비해 소요 시간과 공간 복잡도 면에서는 다소 미흡한 모습을 보였다. 반면 GPT-4는 코드 간결성은 다소 부족하나 소요 시간과 공간 복잡도 양측면에서 가장 우수한 성능을 나타내며, 알고리즘 구현에 있어 속도와 메모리 사용의 최적화를 고려한 코드를 생성해 낼 수 있는 것으로 판단된다.

Ⅲ. 결론

본 논문에서는 GPT-4, Github Copilot, Claude3 Opus, Gemini 등 4개의 대규모 언어 모델(LLM)을 대상으로 알고리즘 문제 해결 능력을 비교 분석하였다. 연구 과정에서 코드의 간결성뿐만 아니라 실행 속도와 메모리 효율성 등 다양한 측면을 함께 고려해야 하는 점을 확인할 수 있었다.

분석 결과, LLM의 알고리즘 문제 해결 능력은 난이도가 높아질수록 전반적으로 저하되는 경향을 보였다. 특히 플래티넘 이상의 고난도 문제에서는 모든 LLM이 현저한 성능 하락을 나타냈으며, 루비 단계에서는 단 한 문제도 해결하지 못하였다. 이는 현재 상용화된 LLM 기술로는 아직까지 복잡하고 창의적인 알고리즘 문제를 다루는 데 한계가 있음을 보여준다.

개별 LLM의 성능을 살펴보면, GPT-4가 전반적으로 가장 우수한 문제 해결 능력을 보였다. GPT-4는 플래티넘 단계까지 안정적인 정답률을 유지하였으며, 코드 간결성은 다소 부족하지만 소요 시간과 공간 복

잡도 면에서 가장 효율적인 것으로 나타났다.

Github Copilot은 코드 간결성 측면에서 우수한 성능을 나타냈다. 그러나 소요 시간과 공간 복잡도는 상대적으로 높게 측정되어, 실행 속도와 메모리 사용 효율성 면에서는 개선의 여지가 있는 것으로 판단된다.

Claude3 Opus는 골드 단계까지 안정적인 정답률을 보였으나, 플래티넘 이상의 고난도 문제에서는 급격한 성능 저하를 나타냈다. 소요 시간과 공간 복잡도는 중상위 수준을 유지하였다.

Gemini는 실버 단계부터 현저한 성능 하락을 보였으며, 플래티넘 이상의 문제는 단 하나도 해결하지 못하였다. 소요 시간과 공간 복잡도 또한 가장 좋지 않은 것으로 측정되어, 전반적인 알고리즘 구현 능력이 다른 LLM에 비해 부족한 것으로 나타났다.

최근 알고리즘 문제나 코딩 학습을 수행함에 있어 LLM을 활용하는 사람이 많아지고 있다. LLM은 문제 해결을 위한 아이디어를 제공하고 코드 작성을 보조하는 유용한 도구로 활용될 수 있다. 그러나 본 논문에서 확인된 바와 같이 LLM이 제시하는 답안은 특정 성능 지표 중 어느 하나에 치우칠 수 있으며, 모델의 한계로 인해 최적의 솔루션이 아닐 수 있다는 점을 유념해야 한다.

본 논문의 결과를 기반으로 향후 LLM을 활용한 알고리즘 학습 방안을 모색하고, 프로그래밍 업무에 LLM을 적용하는 데 있어 유용한 기초 근거 자료로 활용되기를 기대한다.

참고문헌

[1] 양은영, "생성형 AI의 개발 및 이용에 관한 규제의 필요성 - 대규모 언어모델에 기반한 대화형 인공지능 서비스(LLMs AI)를 중심으로", 성균관법학, vol. 35, no. 2, pp. 293-325, 2023.

[2] https://www.elastic.co/kr/what-is/large-language-models

[3] https://www.acmicpc.net/

[4] https://solved.ac/

[5] 최수지 외, “한국어 코딩 테스트에서의 인간 대 ChatGPT 3.5 & 4.0 성능 비교 및 평가 체계”, 디지털콘텐츠학회논문지, vol. 24, no. 12, pp. 3,167-3,178, 2023.

[6] 여상엽 외, “LLM 기반 코드 생성을 위한 프롬프트 조합 방법”, 한국정보과학회 학술발표논문집, pp. 633-635, 2023.