

LLM によるプログラミング初学者の スキル分析と問題作成による学習支援

長谷川 駿一

(木更津工業高等専門学校 専攻科 制御・情報システム工学専攻)

LLM-based Skill Analysis and Problem Generation for Learning Support of Novice Programmers

Shunichi Hasegawa

(Advanced Control and Information Engineering Course, National Institute of Technology, Kisarazu College)

In programming education for beginners, the varying levels of student understanding make early identification of struggling learners crucial. Prior work employed random forest models to classify source code but faced challenges with higher-order assessments, such as evaluating problem-solving approaches. This study leverages the NLP(Natural Language Processing) capabilities of LLMs(Large Language Models) for comprehensive learner assessment and personalized problem generation. Unlike quantitative feature-based methods, this study assigns LLMs the role of an educator, enabling qualitative source code analysis through prompt engineering techniques, such as Role-Play Prompting, Chain-of-Thought (CoT), Few-Shot Learning, and Self-Consistency. In evaluating problems generated by Gemini 1.5 Flash, GPT-4o mini, and GPT-4o, Gemini excelled in comprehensibility but demonstrated reduced code quality at higher difficulty levels. GPT-4o mini demonstrated strengths in engagement and problem quality, with the potential to boost learner motivation. GPT-4o performed consistently well. However, the 'time appropriateness' metric was consistently rated low, highlighting the importance of balancing problem complexity with time constraints. These results suggest that LLM-based problem generation can enhance learner understanding and motivation, contingent on the specific characteristics of the LLM employed.

Keywords: LLM(Large Language Model), Prompt Engineering, Source Code Analysis

1 まえがき

プログラミング初学者の能力判定に関する先行研究では、千枝ら⁽¹⁾が決定木を活用してソースコードの特徴量を抽出し、学習者の理解度を定量的に評価する可能性を示した。一方、飯棲ら⁽²⁾はIRMを用いることでソースコードの構造的特徴に焦点を当て、より詳細な分析を試みた。先行研究は定量的な学習者理解度把握の可能性を示唆する点で共通するが、特徴量や分析粒度に違いがある。しかし、これらの研究で抽出された特徴量のみでは、学習者の問題解決アプローチといった高次元側面の評価は困難である。

本研究では、従来の機械学習モデルに代わり大規模言語モデル (Large Language Model, 以下 LLM) を活用し、より詳細かつ多面的な学習者評価の実現を目指す。近年、LLM は目覚ましい発展を遂げており、適切なプロンプト設計によってその分析能力を最大限に引き出せることが様々な研究で実証されている⁽³⁾。先行研究では、k-means 法と決

定木、そして IRM といった手法を用いて特徴量抽出が行われているが、本研究では、LLM の持つ自然言語処理能力を最大限に活用し、学習者のスキル分析と個別最適化された問題生成を同時に行うことを目指す。具体的にはプロンプトエンジニアリングの手法を組み合わせることで、LLM に教育者としての役割を与え、学習者のソースコードを分析させ、その分析結果に基づいて個々の学習者に最適な問題を自動生成するシステムの構築を試みる。これは、従来の機械学習アプローチでは困難であった、学習者の問題解決プロセスに対する質的評価と、それに基づく個別最適化された学習支援の実現という、教育工学上の重要課題に対する新たな解決アプローチを提示することを目的とする。

本研究は、プロンプト最適化を通じてプログラミング初学者のスキル分析を効率化し、得られた分析結果をもとに、学習者の能力に応じた問題を自動生成するシステムの構築を目指している。ま

た、LLM を活用した学習者評価の有効性を実証することで、プログラミング教育における LLM の可能性をさらに拡大することを目的としている。

2 提案手法

本研究では、LLM の高度な自然言語処理能力を活用することで、より包括的な学習者評価と個別最適化された学習支援の実現を目指す。具体的には、LLM に教育者の役割を付与し、ソースコードを直接分析させることで、従来の機械学習アプローチでは困難であった質的な評価を可能にする。

Fig.1 は、本研究で提案する問題生成システムにおけるプロンプト設計の概要を示している。このシステムは、後述する LLM のプロンプトエンジニアリング手法を統合的に実装している。

2.1 プロンプトエンジニアリング手法

このシステムの中核となるのは、LLM の能力を最大限に引き出すためのプロンプトエンジニアリングである。本研究では、以下のプロンプトエンジニアリング手法を組み合わせたアプローチを採用した。

- **Role-Play Prompting:**⁽⁴⁾ LLM に教育者としての役割を付与し、学習者評価の専門家としての文脈を確立する。これにより、LLM は教育的な観点からソースコードを分析し、適切なフィードバックや問題生成を行うことが期待される。
- **Chain-of-Thought (CoT):**⁽⁵⁾ LLM に段階的な推論を促し、分析プロセスの明示化を行う。これにより、LLM がどのように分析に至ったかの過程を追跡可能にし、結果の妥当性を検証しやすくする。
- **Few-Shot Learning:**⁽⁶⁾ 少数の例題とその分析結果、あるいは問題とその解答例を LLM に提示することで、望ましい出力形式や分析の基準を学習させる。これにより、LLM の出力の質と一貫性を向上させることが期待される。
- **Self-consistency:**⁽⁷⁾ LLM に複数の問題候補を生成させ、その中から統計的に最も整合性の高い問題を選択するメカニズムを実装する。これにより、LLM の出力の多様性を活用し、より質の高い問題を選択することが可能となる。

これらのプロンプト手法を活用した問題生成システムは、以下の3つの段階で構成される。

1. 役割付与とフィードバック指示

LLM にプログラミング教師としての役割と期待するフィードバックの種類を明示的に伝

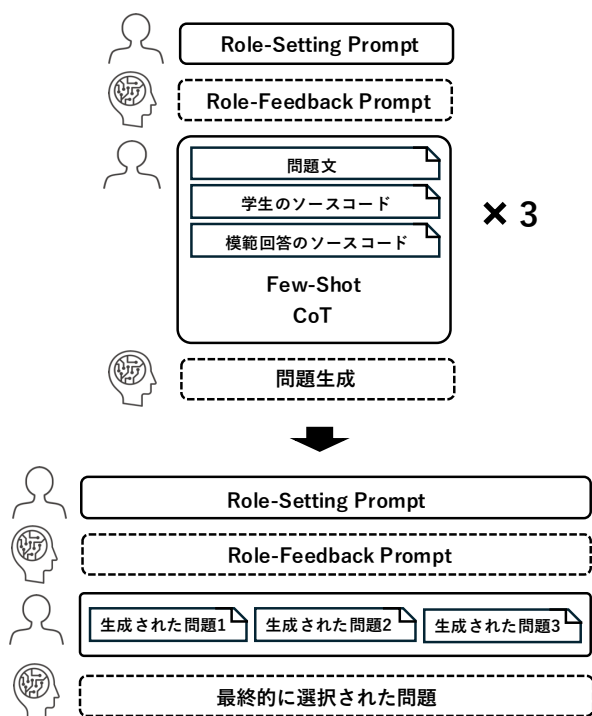


Fig.1. Overview of the Problem Generation System

え、教育的な観点からの分析と問題生成を誘導する。

2. Few-Shot Learning と CoT による問題生成

Few-Shot Learning を用いて、過去の学習者の解答例（問題文、ソースコード、模範解答）を LLM に提示し、Chain-of-Thought (CoT) によりソースコードの分析過程とスキル判断の根拠を段階的に出力させる。これにより、学習者の解答傾向や誤りやすい箇所を分析し、適切な問題生成のための基礎情報を獲得する。Few-Shot の例には、学習者のレベルやソースコードの評価情報を含めることで、より適切な問題選定・生成を可能とする。

3. Self-Consistency による問題選定と最終出力

Few-Shot Learning と CoT に基づき生成された複数の問題から、Self-Consistency の考え方に基づき、同一の入力に対して独立に3つの問題を生成する。その後、再度役割付与とフィードバック指示を行い、LLM に教師としての役割を再認識させた上で、問題の難易度、明確性、独創性などを考慮し、最適な問題を最終出力として選定する。

2.2 本研究で利用した LLM

提案した問題作成システムを実装する LLM の選定は重要な項目の一つである。本研究では性能特性の異なる3種の LLM を比較評価する。Figure.1 に示す共通のプロンプトを各 LLM への入力として

Table1. List of Questionnaire Items

質問項目	評価尺度
問題の文章は理解しやすかったですか？	10段階評価 (1：わかりにくい，10：わかりやすい)
新しいことを学ぶ機会になりましたか？	10段階評価 (1：ならない，10：なった)
難易度はどうでしたか？	10段階評価 (1：簡単すぎる，10：難しすぎる)
この問題を解いて、楽しかったですか？	10段階評価 (1：つまらなかった，10：楽しかった)
「いい問題だな」と思いましたか？	10段階評価 (1：あてはまらない，10：あてはまる)
この問題の解答時間である15分は適切でしたか？	10段階評価 (1：短かった，10：長かった)

用い、生成された問題を評価することで、各 LLM の特性を分析する。以下では、各 LLM の概要について述べる。

- **Gemini 1.5 Flash** (Google): 大規模なテキストおよびコードベースの文脈理解に優れている。
- **GPT-4o mini** (OpenAI): 比較的軽量の構成ながら、高速な推論処理と多様なタスクへの適応性を特徴とする。
- **GPT-4o** (OpenAI): 高度な推論能力とマルチモーダル処理機能を備え、4o-mini と比較してより精緻な自然言語処理が可能である。

各 LLM の生成した問題の評価は、プログラミング学習者に実際に問題を解答させ、その解答コードとアンケート調査の結果に基づいて行う。本評価では、解答コードの正確性や効率性などを分析するとともに、アンケート調査を通じて被験者の主観的な評価を収集する。

3 実験概要

3.1 実験データ

本研究では、2024 年度木更津工業高等専門学校情報工学科 2 年の「プログラミング基礎 II」後期中間試験の問題と、試験に出席した学生 41 名が提出したソースコードをデータとして用いた。この試験は筆記試験を含む計 7 問で構成されており、本研究ではそのうち学生に実際にプログラミングをさせる 3 問を対象とした。これらの問題文、学生が実際に解いたソースコード、および各問題に対して定義された模範解答のソースコードを用意し、実際の問題生成用プロンプトに組み込んだ。

3.2 アンケート調査

Table.1 は、本実験で使用したアンケート項目を示す。問題生成システムで生成された問題を解いた学生に本アンケートを実施し、問題に対する評価を収集する。

Table2. Evaluation Criteria for Source Code

評価項目	説明
正確性	問題を解決できているかを評価。
効率性	アルゴリズムの選択や実装の最適性を評価。
可読性	コードの読みやすさ、命名規則の適切さ、構造の整理などを評価。
基礎概念	コードが適切なプログラミング概念を適切に活用しているかを評価。

Table3. Average Scores of Survey Items for Each Model

アンケート項目	Gemini 1.5 Flash	GPT-4o mini	GPT-4o
理解しやすさ	7.40	6.00	6.97
新しい学び	5.69	5.37	5.89
難易度	5.51	4.92	5.75
楽しさ	6.51	6.55	6.03
良問評価	6.43	6.63	6.22
時間の適切さ	4.49	4.34	4.56

3.3 ソースコードの評価

本研究では、Table.2 の評価項目を用いて、GPT-4o による自動コード評価を行った。各項目は 5 段階で評価し、5 点を最高評価、1 点を最低評価とした。5 段階評価を採用したのは、評価の粒度と評価のしやすさのバランスを考慮したためである。

4 実験結果

4.1 各モデルのアンケート回答傾向

Table.3 に各モデルのアンケート回答の平均値を示す。Gemini 1.5 Flash は「理解しやすさ」で最も高い平均値 ($M = 7.40$) を示し、問題の一貫性を示唆する結果となった。GPT-4o mini は「楽しさ」($M = 6.55$) と「良問評価」($M = 6.63$) で若干高い評価を得ており、学習意欲を高める問題生成に強みを持つ可能性が示唆された。GPT-4o は全体的に中間的な評価を示し、バランスの取れた問題生成を行う傾向が見られた。全モデルで「時間の適切さ」の評価が比較的低く ($M = 4.34 \sim 4.56$)、設定時間 (15 分) が短すぎた可能性が示唆された。また、ソースコード評価は全モデルで低い傾向が見られ、コード品質の改善の余地が示された。

4.2 モデル比較(一元配置分散分析と Tukey の HSD 検定)

各モデルの評価を比較するため、アンケート調査の各指標に対して一元配置分散分析と Tukey の HSD 検定を実施した。Table.4 に示すように、「理解しやすさ」において統計的に有意な差 ($F = 3.0928, p = 0.0495$) が確認された。Tukey の HSD 検定 (Table.5) の結果、Gemini 1.5 Flash と GPT-4o mini 間に有意差 ($p = 0.0456$) が認められ、Gemini 1.5 Flash が生成する問題が最も理解しやすいことが示された。他のアンケート指標では有意な差は

Table4. Results of One-Way ANOVA on Survey Data

評価項目	F 値	p 値	有意性 (5% 水準)
理解しやすさ	3.0928	0.0495	有意差あり
新しい学び	0.3899	0.6781	有意差なし
難易度	1.5028	0.2272	有意差なし
楽しさ	0.5317	0.5891	有意差なし
良問評価	0.2966	0.7440	有意差なし
時間の適切さ	0.1000	0.9049	有意差なし

Table5. Results of Tukey's HSD Test for "Ease of Understanding

比較	平均差	p 値	有意性 (5% 水準)
GPT-4o vs. GPT-4o mini	-0.9722	0.2142	有意差なし
GPT-4o vs. Gemini 1.5 Flash	0.4278	0.7474	有意差なし
GPT-4o mini vs. Gemini 1.5 Flash	1.4000	0.0456	有意差あり

Table6. Correlation of key elements

評価指標の比較	Gemini 1.5 Flash	GPT-4o mini	GPT-4o
難易度 vs. 正確性	-0.738	-0.328	-0.559
難易度 vs. 効率性	-0.693	-0.488	-0.377
難易度 vs. 可読性	-0.657	-0.468	-0.447
難易度 vs. 基本概念	-0.759	-0.488	-0.428
時間の適切さ vs. 正確性	0.614	0.213	0.327
時間の適切さ vs. 効率性	0.600	0.241	0.246
時間の適切さ vs. 可読性	0.444	0.236	0.308
時間の適切さ vs. 基本概念	0.609	0.210	0.421
新しい学び vs. 正確性	-0.566	-0.224	-0.131
新しい学び vs. 効率性	-0.417	-0.254	-0.188
新しい学び vs. 可読性	-0.511	-0.263	-0.206
新しい学び vs. 基本概念	-0.485	-0.312	-0.274

注：相関係数の絶対値が 0.4 以上を中程度以上の相関として太字で示している。

見られなかった。

4.3 各指標との相関分析

各モデルにおけるアンケート調査指標とソースコード評価指標間の相関分析を行った結果 (Table.6), 特に Gemini 1.5 Flash において顕著な傾向が認められた。「難易度」とソースコードの各評価指標の間には強い負の相関 ($r = -0.657 \sim -0.759$) が観察され、難易度が高いと判断された問題ほどコード品質が低下する傾向が示された。また、「時間の適切さ」とソースコードの各評価指標の間には中程度の正の相関 ($r = 0.444 \sim 0.614$) が見られ、問題に取り組む時間が適切であると感じるほどコード品質が向上する傾向が示唆された。さらに、「新しい学び」とソースコードの各評価指標の間には中程度の負の相関 ($r = -0.417 \sim -0.566$) が見られ、新しい学びが多いと感じるほどコード品質が低下する傾向が示された。これは、新しい概念学習時にコードの完成度よりも学習に重点が置かれるためと考えられる。他のモデルでは、Gemini 1.5 Flash で見られたような顕著な相関は見られなかった。

5 考察

Gemini 1.5 Flash は、生成された問題の理解しやすさに優れている一方で、難易度が上昇するに

つれてコード品質が低下する傾向が見られた。この結果は、問題の難易度調整とコード品質の維持を両立させることの困難さを示唆している。一方、GPT-4o mini は、学習者の動機づけを高めるような問題作成に強みを持つ可能性が示された。これは、問題の提示方法を工夫することで学習意欲を効果的に高められることを示している。また、GPT-4o は各評価項目において安定した性能を示しており、多様な学習ニーズに対応できる可能性を示唆している。しかし、全モデルで時間配分が課題として浮上しており、問題の複雑さと制限時間との関係性を適切にモデリングする必要性が明らかになった。

6 まとめ

本研究は 3 種類の LLM を用いて個々の学習者に合わせたプログラミング問題生成を試み、学習支援への応用可能性を探った。実験で LLM は可能性を示したものの、教育的視点に基づく問題設計が不可欠と示唆された。今後の研究では、問題の多様性の拡大、コード自動評価、学習履歴との連動、各 LLM の特性を活かした問題生成手法の確立を目指し、より効果的な学習支援システムの開発を進める予定である。

謝辞

本研究は JSPS 科研費 23K17604 および 24K00460 の助成を受けたものです。

参考文献

- (1) 千枝睦実, 大枝真一, “プログラミング授業での決定木を用いたドロップアウト原因の可視化,” 第18回情報科学技術フォーラム(FIT2019), pp.87-90(第3分冊), 2019.
- (2) 飯棲俊介, 大枝真一, “IRMと決定木を用いたプログラミング初学者の能力判定のための特徴量の抽出,” 第21回情報科学技術フォーラム(FIT2022), pp.235-236(第4分冊), 2022.
- (3) Schulhoff, Sander, et al. "The Prompt Report: A Systematic Survey of Prompting Techniques." arXiv preprint arXiv:2406.06608 (2024).
- (4) Kong, Aobo, et al. "Better zero-shot reasoning with role-play prompting." arXiv preprint arXiv:2308.07702 (2023).
- (5) Suzgun, Mirac, et al. "Challenging big-bench tasks and whether chain-of-thought can solve them." arXiv preprint arXiv:2210.09261 (2022).
- (6) Brown, Tom, et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
- (7) Wang, Xuezhi, et al. "Self-consistency improves chain of thought reasoning in language models." arXiv preprint arXiv:2203.11171 (2022).