2025 年度 卒業論文

論文題目
プログラミング教育におけるフィードバック効率化のための LLM 擬似学習者による検証

指導教員
舟橋　健司　准教授
伊藤　宏隆　助教

名古屋工業大学　工学部　情報工学科
2022 年度入学　34714142 番

山内　将紀

目 次

第1章　はじめに

教育において適切なフィードバックが学習成果の向上に効果的であることが、Hattie and Timperley [1] らによって示されており、これはプログラミング教育においても同様である。単に正誤を示すだけでなく、誤りの原因を指摘するフィードバックや、改善の方向性を具体的に示すフィードバックは、学習者が自身の理解不足を認識し、知識を定着させるうえで重要な役割を果たす。ただし、情報量が多いフィードバックや直接的な答えを示すフィードバックは、学習者の自主的な学習を阻害する可能性がある。このため、適切な情報量で質の高いフィードバックを提供することは、プログラミングを含む多くの教育分野において重要である。一方で、近年のプログラミング教育では、大学の講義やオンライン学習環境を中心に受講者数が増加している。プログラムコードを確認し、学習者に適したフィードバックを生成するには相応の時間と労力を必要とするため、受講者数や課題数が増えるほど、教師の負担は大きくなる。その結果、教師が学習者一人ひとりに適切なフィードバックを提供することは容易ではなくなっている。

このような背景から、近年ではプログラミング課題に対するフィードバックを支援する手法として、大規模言語モデル (Large Language Model: 以下 LLM と呼ぶ) をはじめとするフィードバック自動生成技術の活用が注目されている。例えば、Roest, Keuning and Jeuring らは LLM による次ステップヒント生成の実装と評価を行い、その有効性と課題を報告している [2]。これらの技術により、実際の教育現場でも複数の異なるフィードバック案を容易に生成することが可能となる。しかし、生成された複数のフィードバック案の中から、どれが学習者にとって適切であるか選別することは容易ではない。また従来の研究では、フィードバックの有用性や質を評価するために、実際の学習者を対象としたユーザスタディや専門家による内容分析が主に用いられてきた。これらの手法は、信頼性は高いものの、多くの時間的・人的コ

ストが必要となる。このため、多数のフィードバック案を対象とした反復的な検証や比較が困難である。このように、複数のフィードバックから適切なフィードバックの効率的な選別手法の確立は、重要な課題となっている。

ところで、LLM による自動ヒント生成時のヒントの正確性を上げるために GPT-4 にヒントを生成させ、そのヒントを GPT-3.5 を用いて正しいヒントか検証し、判別する手法を用いた研究が報告されている [3]。この研究では、低いレベルのモデルをヒントの有効性の検証に用いることで、初学者として機能したとされている。

本研究では、プログラミング課題に対する適切なフィードバックを選別する手法として、LLM を用いた擬似学習者による選別手法を検証する。具体的には、低いレベルの LLM モデルを初級学習者と同等のプログラミングスキルレベルとみなし、擬似学習者として用いる。異なるフィードバックを擬似学習者に与えた際に生じる修正成功率 (正しく修正できた割合) の違いに着目し、その結果をもとに、より適切であると考えられるフィードバックを選別できるかどうか検証する。それにより、適切なフィードバック選別の効率化の可能性を明らかにすることを目的とする。検証の結果、直接的な答えを示すようなフィードバックや情報量が多いフィードバックを与えた場合、修正成功率が高くなり、情報量が少ない、曖昧なフィードバックを与えた場合、修正成功率が低くなる傾向が確認された。LLM 擬似学習者のフィードバック間の修正成功率の違いが実際の初級者のフィードバック間の修正能力の傾向と類似する可能性を示し、LLM 擬似学習者による適切なフィードバック選別の効率化の可能性を明らかにした。

また、学習者の能力の違いに応じた適切なフィードバックを選別するために、擬似学習者により実際の学習者の能力の違いを実現可能か検証する。そのために、次の 2 つの方法を提案する。1 つ目は、異なるパラメータ数のモデルを用いる方法である。パラメータ数の違いによる性能差から、同様のフィードバックにおける修正成功率に生じる違いにより、学習者の能力の違いが実現できることを期待する。2 つ目は、LLM に対して異なるプロンプトを用いる方法である。プロンプトにおいて、指示を追加することによる出力時の計算過程の変化から、同様のフィードバックにおける修正成功率に生じる違いにより、学習者の能力の違いが実現できることを期待する。学習者の能力の違いを実現可能か検証した結果、限定的ではあるものの実

現できる可能性が示唆された。

本文では、第 2 章において背景知識として LLM について、第 3 章において LLM 擬似学習者を用いた適切なフィードバック選別の効率化の可能性の検証について、第 4 章において異なる能力を持つ学習者の実現の検証について、最後に第 5 章において各検証結果のまとめと今後の課題について述べる。

第2章　背景知識

2.1　大規模言語モデル (LLM) 概要

LLM とは、膨大なテキストデータを用いて事前学習されたニューラルネットワークモデルであり、入力された文章に対して、次に続く語や文を確率的に生成する能力があるモデルである。近年では、Transformer と呼ばれる注意機構を中心とするニューラルネットワーク構造が Vaswani らにより提案された [4]。この構造により、長い文脈を考慮した言語理解が可能となった。このような構造を基盤とした LLM は、文章の生成や理解、質問応答などの幅広い自然言語処理タスクに対応できる能力を備えている。

2.2　性能の違い

LLM の性能は、主に学習に用いられるデータ量、パラメータ数、および計算量に強く依存していることが Kaplan らにより報告されている [5]。パラメータとは、モデルがニューラルネットワーク内部で学習によって調整される重みのことである。一般に、パラメータ数が多いほど、モデルはより複雑な言語パターンや文脈を表現できるようになり、多様な入力に対して柔軟に対応できる傾向がある。一方で、パラメータ数が多いほど、必要な計算資源や実行時間が増加するという問題も存在する。さらに、性能が高いとされている LLM であっても、常に正確な出力を生成できるわけではなく、部分的に正しいが不完全な回答や、もっともらしいが事実とは異なる内容 (ハルシネーション) を生成することも報告されている [6]。

2.3　プロンプト設計

プロンプトとは、LLM に与える入力文のことで、LLM はプロンプトに基づいて出力を生成する。与えるプロンプトの内容や表現により出力が大きく変化することが

Gan and Mori らにより報告されている [7]。したがって、目的に沿った出力を取得するには、それに応じた適切なプロンプトを設計することが重要である。また、プロンプトはシステムプロンプトとユーザプロンプトに大別される。システムプロンプトでは、LLM に与える立場や役割、実行させるタスク、出力の制約条件などを指定する。例えば、特定の専門家として回答させたい場合や出力の形式を一定にしたい場合などに用いられる。ユーザプロンプトでは、LLM に与える具体的な指示を指定する。

第3章　LLM 擬似学習者によるフィードバック効率化の検証

3.1　LLM 擬似学習者

本研究では、LLM を用いてプログラミング初級者に相当する擬似学習者を定義する。まず、選別するフィードバックの対象となる課題内容と誤ったコード、および「間違っています。」という簡単なフィードバックを LLM に与え、同一条件で 10 回コードを修正させる。その結果、1 回も正しく修正できなかった LLM モデルを、本研究における擬似学習者として定義する。これにより、間違っているとだけ示された場合では、正しく修正できない初級者相当の能力であるモデルを擬似学習者として定義できる。本検証では、上記条件を満たすことが確認された Qwen3-8B (パラメータ数 約 82 億個) [8] を擬似学習者として用いた。Qwen3 はローカル LLM である。ローカル LLM とは、ユーザ個人の PC 上などのローカル環境で動作する LLM のことである。したがって、データはローカルでのみ使用されるので安全である。

3.2　プロンプト

LLM 擬似学習者には、図 3.1 に示すシステムプロンプト、図 3.2 に示すユーザプロンプトを入力として与えた。システムプロンプトでは、振る舞いとしてプログラミングを学ぶ学生であることを定義し、タスクの内容としてフィードバックを読んで提出したコードを修正することを、出力の要件として修正したコードを出力することを指定した。ユーザプロンプトでは、プログラミングの課題内容、擬似学習者が以前提出したコードとして誤ったコード、およびその提出したコードに対するフィードバックを与えた。また、フィードバックをもとにコードを修正することを指示した。システムプロンプトは、全ての試行で共通とし、ユーザプロンプトは、フィードバック以外は全ての試行で共通とした。

```
You are a student learning programming.

You previously submitted code for a programming assignment and received feedback.
Your task is to fix the code after reading the feedback.

=== Output Requirements ===
Output the following item:
1. Fixed code
```

図 3.1: 使用したシステムプロンプト

```
課題内容:
{problem_text}

以前に提出したコード:
{previous_code}

フィードバック:
{feedback}

フィードバックをもとに以前に提出したコードを修正してください。
```

図 3.2: 使用したユーザプロンプト

3.3　課題内容

検証に使用したプログラミング課題は、実際の Java プログラミング授業で提示された課題である。課題の内容は次のとおりである。

年と月を入力すると、カレンダーを表示するプログラムを作成せよ。
一例：

年を入力してください：2004
月を入力してください：6
2004 年 6 月のカレンダー

```
S  M  Tu W  Th F  S
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

本課題は、条件分岐や繰り返し処理などのプログラミング初級者が学習初期に扱う要素を複数含んでいる。複数を組み合わせてコードを作成する必要があるため、初級者向けの中で少し難易度の高い課題である。この課題におけるコードの正当性は、次の複数の入力パターンに対し、すべての出力の日付の配置が正しいことで判定した。入力パターンは 2000 年 2 月、2000 年 3 月、2000 年 12 月、2026 年 1 月の 4 パターンである。

3.4　修正前の誤ったコード

検証に使用した誤ったコードと誤った出力例とそれに対する正しい出力例は次の通りである。

```
// 年・月を読み込んでカレンダーを表示

import java.util.Scanner;

class ex4_2 {
```

```
public static void main(String[] args) {
    Scanner stdIn = new Scanner(System.in);

    System.out.print("年を入力してください：");
    int year = stdIn.nextInt();

    int month,a,b,A,B,c,dow;
    do {
        System.out.print("月を入力してください：");
        month = stdIn.nextInt();
    } while (month <= 0 || month >= 13);

    if (month == 1 || month == 2) {
        a = (year - 1) / 100;
        b = (year - 1) % 100;
        A = a / 4;
        B = b / 4;
        c = (month + 12 + 1) * 26 /10;
        dow = (1 + c + b + B + A - 2 * a) % 7;
        if (dow == 0)
            dow = 7;

    } else {
        a = year / 100;
        b = year % 100;
        A = a / 4;
        B = b / 4;
        c = (month + 1) * 26 /10;
        dow = (1 + c + b + B + A - 2 * a) % 7;
        if (dow == 0)
            dow = 7;
    }

    System.out.println(year + "年" + month + "月のカレンダー");
    System.out.println("␣Su␣␣M␣Tu␣␣W␣Th␣␣F␣Sa");

    if (month == 1 || month == 3 || month == 5 || month == 7
    || month == 8 || month == 10 || month == 12) {
        int day = 31;

        for (int s = 1; s <= dow - 1; s++)
            System.out.print("␣␣␣");
        for (int l = 1; l <= 8 - dow; l++)
            System.out.print("␣␣" + l);
        System.out.println();
        for (int i = 9 - dow; i <= day; i++) {
            if (i <= 9)
                System.out.print("␣␣" + i);
            else
                System.out.print("␣" + i);
            if ((i + dow - 1) % 7 == 0)
                System.out.println();
        }
        if ((day + dow - 1) % 7 != 0)
            System.out.println();
    }
```

```
        else if (month == 4 || month == 6 || month == 9
    || month == 11) {
            int day = 30;

            for (int s = 1; s <= dow - 1; s++)
                System.out.print("␣␣␣");
            for (int l = 1; l <= 8 - dow; l++)
                System.out.print("␣␣" + l);
            System.out.println();
            for (int i = 9 - dow; i <= day; i++) {
                if (i <= 9)
                    System.out.print("␣␣" + i);
                else
                    System.out.print("␣" + i);
                if ((i + dow - 1) % 7 == 0)
                    System.out.println();
            }
            if ((day + dow - 1) % 7 != 0)
                System.out.println();
        }

        else {
            int day;
            if (year % 4 == 0)
                if (year % 100 == 0)
                    if (year % 400 == 0)
                        day = 29;
                    else
                        day = 28;
                else
                    day = 29;
            else
                day = 28;

            for (int s = 1; s <= dow - 1; s++)
                System.out.print("␣␣␣");
            for (int l = 1; l <= 8 - dow; l++)
                System.out.print("␣␣" + l);
            System.out.println();
            for (int i = 9 - dow; i <= day; i++) {
                if (i <= 9)
                    System.out.print("␣␣" + i);
                else
                    System.out.print("␣" + i);
                if ((i + dow - 1) % 7 == 0)
                    System.out.println();
            }
            if ((day + dow - 1) % 7 != 0)
                System.out.println();
        }
    }
}
```

誤った出力例 (2000 年 3 月のカレンダー)

```
Su  M Tu  W Th  F Sa
 1  2  3  4  5  6  7  8  9  10  11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

正しい出力例 (2000 年 3 月のカレンダー)

```
Su  M Tu  W Th  F Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

このコードは上に示すように、ある特定の入力の際に誤った出力となっている。この誤出力の原因は、dow 変数にある。このコードでは、dow が 1 から 7 の値になる前提でカレンダーを表示する部分のループが作成されている。しかし、Java では演算子「%」の結果が負の値になる場合がある。そのため、dow が負の値になる入力では、誤ったカレンダーが出力される。

3.5　フィードバック

検証に使用したフィードバック群を表 3.1 に示す。

表 3.1: フィードバック

番号	フィードバックの内容
f1	間違っています。
f2	曜日の計算が間違っています。

番号	フィードバックの内容
f3	dow = (1 + c + b + B + A - 2 * a) % 7; の計算の際に dow が負の値になることがあり、それが原因で間違った出力になります。
f4	日付の配置ロジックが不正確で、最初の曜日から正しい位置にない。

f5	dow = (1 + c + b + B + A - 2 * a) % 7;
	これを以下の通りに変更してください。
	dow = (1 + c + b + B + A + 5 * a) % 7;

f6	dow が負の値になることがあり、それが原因で間違った出力になります。
f7	dow = (1 + c + b + B + A - 2 * a) % 7; ここに間違った出力となる原因があります。
f8	間違っています。演算子%では、負の値を割った余りは負の値になります。
f9	間違っています。dow が取りうる値の範囲を考慮してください。
f10	間違っています。あなたのコードは、dow が負の値にならないことを前提に作成されています。
f11	あなたのコードは、dow が負の値にならないことを前提に作成されていますが、dow が負の値になることがあります。その際に、誤った出力となっています。

f12	間違っている出力例を教えます。
	正しい 2000 年 3 月のカレンダーは以下の通りです。
	`Su  M Tu  W Th  F Sa`
	`1  2  3  4`
	`5  6  7  8  9 10 11`
	`12 13 14 15 16 17 18`
	`19 20 21 22 23 24 25`

番号	フィードバックの内容
	26 27 28 29 30 31 あなたのコードによる 2000 年 3 月のカレンダーは以下の通りです。 Su M Tu W Th F Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
f13	今のコードでは、dow が負の値になることがあるので、そうならない dow の計算方法に変えてください。
f14	dow が‘- 2 * a‘により、負の値になることがあるので、そこを変えると正しくなります。
f15	今のコードは、変数 dow が 1 から 7 の整数値となる前提になっていますが、dow は-6 から-1 までの整数値となることもあります。それにより、表示処理のループで意図しない挙動が発生しています。
f16	今のコードでは、‘dow = (1 + c + b + B + A - 2 * a) % 7‘ により、表示するカレンダーの最初の日 (1 日) の曜日を求めています。その方法では、dow が 1, -6：日曜日、2, -5：月曜日、3, -4：火曜日、4, -3：水曜日、5, -2：木曜日、6, -1：金曜日、0：土曜日となります。しかし現状では、dow の値の範囲が 0 から 6 までででしか考慮されておらず、それにより間違った出力結果となることがあります。
f17	変数 dow が負の値になることがあるので、表示処理のループでは dow の絶対値を用いてください。

番号	フィードバックの内容
f18	変数 dow が負の値になるときの処理が不十分です。負の値であるときに 7 を足すように修正してください。
f19	変数 dow が負の値になるときの処理が不十分です。
f20	‘dow‘が 1 から 7 の値になるように正規化してください。
f21	曜日を表す ‘dow‘は ‘% 7‘を使って計算されていますが、Java では ‘%‘の結果が負の値になることがあります。 現在のコードでは ‘dow == 0‘の場合のみ補正していますが、負の値がそのまま使われる可能性があります。 計算後に ‘dow‘が必ず同じ範囲に収まるように処理を追加すると、表示のずれが解消されます。
f22	一部の年月で表示がずれており、その原因は曜日を表す数値の扱い方にあります。 特に、曜日を計算した直後の値が、想定している範囲に本当に収まっているかを確認してください。
f23	現在のコードでは ‘dow = (1 + c + b + B + A - 2 * a) % 7;‘の計算結果をそのまま使っていますが、Java では ‘%‘演算子が負の値を返すことがあり、そのままだとカレンダー表示がずれる原因になります。計算後に ‘dow‘が必ず 0 から 6 の範囲に収まるように正規化してください。

3.6　検証方法

本節では、前節までに示したデータと Qwen3-8B を LLM 擬似学習者として用いた検証方法について述べる。LLM に入力として、先述したシステムプロンプトとユーザプロンプトを与えた。ユーザプロンプトの {problem_text} には先述した課題内容を、{previous_code} には先述した誤ったコードを対応させ、これらとシステムプロンプトは全ての試行で共通とした。ユーザプロンプトの {feedback} には、先述したフィードバック群の各フィードバックを対応させた。つまり、プロンプト

表 3.2: 生成時ハイパーパラメータ

パラメータ	値
temperature	1.0
top_p	1.0
top_k	50

において各試行で変更されたのはフィードバックのみである。表 3.2 に示す生成時ハイパーパラメータを用い、同一のプロンプトに対して LLM に 50 回出力を生成させた。これを 1 試行とし、各フィードバックに対して行った。その結果、各フィードバックに対し 50 個の修正されたコードを全部で 1150 個取得した。取得したコード群から、修正成功率 (正しく修正されたコードの割合) を算出した。そして、フィードバック間の修正成功率の違いと、フィードバック間の情報量の違いとの関係を分析した。それにより、LLM 擬似学習者による修正成功率のフィードバック間の違いと実際の初級者によるフィードバック間の修正能力の傾向が類似するか調査し、適切なフィードバックを選別できるか検証した。

3.7　修正成功率

Qwen3-8B による各フィードバックの修正成功率を表 3.3 に示す。100、98 %と非常に高い修正成功率となるフィードバックは、f5 や f18 の直接的な答えが含まれるフィードバックとなっており、0 %と非常に低い修正成功率となるフィードバックは、f1 や f12 の間違っている原因に関する情報が含まれていないフィードバックとなっている。修正成功率が 10 %未満のフィードバックは、f2、f4、f7、f9 のような間違っている原因やコードの場所に関する情報が含まれているものの、情報量が足りず、曖昧なフィードバックとなっている。高い修正成功率となるフィードバックは、十分な情報が含まれているフィードバックである傾向があり、低い修正成功率となるフィードバックは、少ない情報のフィードバックである傾向がある。

表 3.3: Qwen3-8B による各フィードバックの修正成功率

番号	修正成功率 (単位：%)	番号	修正成功率 (単位：%)
f1	0	f13	42
f2	2	f14	10
f3	46	f15	70
f4	4	f16	36
f5	100	f17	26
f6	22	f18	98
f7	2	f19	70
f8	32	f20	30
f9	8	f21	42
f10	18	f22	16
f11	54	f23	66
f12	0		

3.8　考察

フィードバックと修正成功率の関係から、LLM 擬似学習者によるフィードバック間の修正成功率の違いが実際の初級者のフィードバック間の修正能力の傾向と類似する可能性が期待できる。そして、今回のフィードバックの中で、修正成功率が半分程度の 54 % である f11「あなたのコードは、dow が負の値にならないことを前提に作成されていますが、dow が負の値になることがあります。その際に、誤った出力となっています。」が情報量が多くも少なくもない適切なフィードバックの例といえる。このように、LLM 擬似学習者により複数のフィードバックの中から、適切なフィードバックを効率的に選別できる可能性が示唆される。

第4章　擬似学習者による異なる能力を持つ学習者の実現の検証

4.1　パラメータ数の違いによる実現

4.1.1　検証方法

本小節では、パラメータ数の違いにより学習者の能力の違いを実現可能か調査するために行った検証方法について述べる。3 章の修正成功率の算出方法を使用する LLM のモデルのみを変更し、実施した。使用したモデルは Qwen3-4B (パラメータ数 約 40 億個) [9] であり、3 章で使用したモデルのパラメータ数が約 82 億個に対し、少なくなっている。Qwen3-4B と Qwen3-8B の修正成功率の違いを比較し、学習者の能力の違いを実現可能か検証した。

4.1.2　Qwen3-4B と Qwen3-8B の比較

Qwen3-4B による各フィードバックの修正成功率を表 4.1 に示す。Qwen3-4B による各フィードバックの修正成功率 (表 4.1) と Qwen3-8B による各フィードバックの修正成功率 (表 3.3) を比較する。パラメータ数別の各フィードバックの修正成功率の分布を図 4.1 に示す。図 4.1 は、Qwen3-8B による修正成功率で昇順にソートされている。同一のフィードバックに対しても、パラメータ数の違いにより修正成功率に大きな差が生じることが確認された。しかし、f5 のような修正方法が明示されたフィードバックに対しては、どちらのモデルも共に非常に高い修正成功率を示し、f1 の誤っていることのみを指摘するフィードバックに対しては、どちらのモデルも共に非常に低い修正成功率を示した。このように、一部のフィードバックでは、パラメータ数の違いによる差は見られなかった。数値範囲の考慮や誤りの原因とその結果生じる不具合を示す f11 や f15 のようなフィードバックでは、Qwen3-8B の方が一貫して高い修正成功率を示した。全体的な傾向としてパラメータ数が多い Qwen3-8B の

表 4.1: Qwen3-4B による各フィードバックの修正成功率

番号	修正成功率 (単位：%)	番号	修正成功率 (単位：%)
f1	0	f13	10
f2	0	f14	10
f3	76	f15	34
f4	0	f16	2
f5	100	f17	4
f6	8	f18	78
f7	0	f19	58
f8	98	f20	14
f9	0	f21	98
f10	2	f22	0
f11	18	f23	66
f12	0		

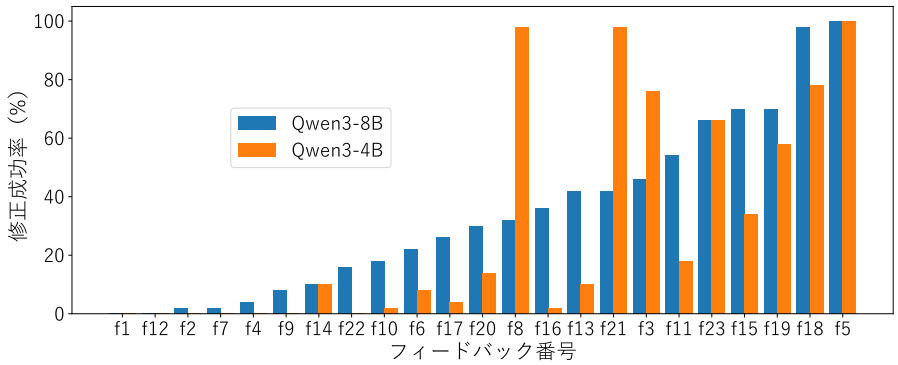

図 4.1: パラメータ数別の各フィードバックの修正成功率の分布

方が高い修正成功率を示すフィードバックが多く見られた。しかし、すべてのフィードバックにおいて Qwen3-8B の方が高かったわけではない。Java の仕様に基づく事実を提示する f8、f21 のフィードバックや、原因が示されており、修正方法が限定されている f3 のフィードバックにおいては、Qwen3-4B の方が高い修正成功率を示した。

4.1.3　考察

比較結果は、モデルのパラメータ数の違いが修正成功率の優劣としてではなく、フィードバックの解釈および修正方針の立て方の違いである可能性を示唆した。Qwen3-

4B は、フィードバックに含まれている内容をそのまま受け取り、局所的かつ直接的な修正を行う傾向が強いと考えられる。そのため、修正方針がほぼ一意に定まるフィードバックに対しては、高い修正成功率を示したと考えられる。一方で、Qwen3-8B は、フィードバックに含まれる内容をより広く解釈することにより、修正方針が明確でないフィードバックに対しても、Qwen3-4B より比較的高い修正成功率を示したと考えられる。しかし、解釈の幅が広がることにより修正方針が複数考えられる状況では、適切な修正に至らない場合があり、修正方針がほぼ一意に定まるフィードバックに対して、Qwen3-4B よりも比較的低い修正成功率を示す場合があったと考えられる。以上より、モデルのパラメータ数の違いが学習者のフィードバックを理解・利用して修正に至る能力の差を実現できる可能性が示唆された。

4.2　プロンプト設計の違いによる実現

4.2.1　検証方法

本小節では、プロンプト設計の違いにより学習者の能力の違いを実現可能か調査するために行った検証方法について述べる。3 章の修正成功率の算出方法をシステムプロンプトのみを変更し、実施した。変更したシステムプロンプトを図 4.2 に示し、プロンプト 2 と呼ぶ。変更前のシステムプロンプト (図 3.1) をプロンプト 1 と呼ぶ。変更点として、フィードバックを注意深く読んで、ステップバイステップでフィードバックの意味などを考えるようになどの指示を追加した。プロンプト 1 とプロンプト 2 の修正成功率の違いを比較し、学習者の能力の違いを実現可能か検証した。

4.2.2　プロンプト 1 とプロンプト 2 の比較

プロンプト 2 による各フィードバックの修正成功率を表 4.2 に示す。プロンプト 1 による各フィードバックの修正成功率 (表 3.3) とプロンプト 2 による各フィードバックの修正成功率 (表 4.2) を比較する。プロンプト別の各フィードバックの修正成功率の分布を図 4.3 に示す。図 4.3 は、プロンプト 1 による修正成功率で昇順にソートされている。プロンプト 1 の修正成功率が f16 の 36 %以上のフィードバックにおい

```
You are a student learning programming.

You previously submitted code for a programming assignment and received
feedback.
Your task is to fix the code after carefully reading and reasoning about the
feedback.

Think step by step about what the feedback means and how the code should be
corrected.
Do not include your reasoning in the output.

=== Output Requirements ===
Output the following item:
1. Fixed code
```

図 4.2: 変更したシステムプロンプト

て、f19 を除いてプロンプト 2 の方が高かった。それらのフィードバックには、情報量が多く、複雑であるものが多く含まれていた。特に、f21 と f23 のフィードバックにおいて、プロンプト間の修正成功率に顕著な差を示した。対して、プロンプト 1 の修正成功率が f16 の 36 %未満のすべてのフィードバックにおいて、プロンプト 1 の方が高かった。それらのフィードバックには、単純なものが多く含まれていた。直接的な答えを示すようなフィードバック f5、f18 において、両プロンプトとも非常に高い修正成功率を示し、f1 の誤っていることのみを指摘するフィードバックにおいて、両プロンプトとも非常に低い修正成功率を示した。これらのフィードバックではプロンプトの違いによる影響は見られなかった。

4.2.3　考察

情報量が多く、複雑なフィードバックに対し、プロンプト 2 の修正成功率が高くなったのは、ステップバイステップでの推論の影響によるものだと考えられる。特に、f21 と f23 が他のフィードバックよりもプロンプトの違いによる修正成功率の差が大きいのは、フィードバックが長く多くの情報を含むため、ステップバイステップでの推論の影響が強く及んだと考えられる。一方、単純なフィードバックに対しプロンプト 2 の方が修正成功率が低くなったのは、ステップバイステップでの推論に

表 4.2: プロンプト 2 による各フィードバックの修正成功率

番号	修正成功率 (単位：%)	番号	修正成功率 (単位：%)
f1	0	f13	56
f2	0	f14	0
f3	58	f15	82
f4	0	f16	52
f5	100	f17	10
f6	12	f18	100
f7	0	f19	68
f8	18	f20	8
f9	0	f21	84
f10	4	f22	2
f11	70	f23	100
f12	0		

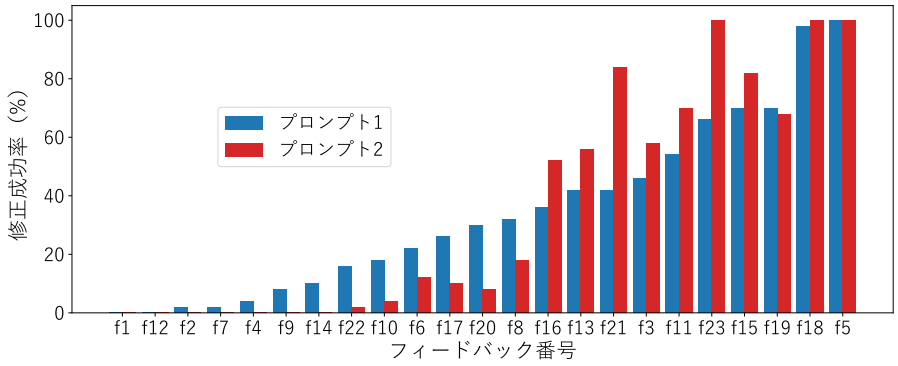

図 4.3: プロンプト別の各フィードバックの修正成功率の分布

よるフィードバックの拡大解釈や考えすぎが影響した可能性がある。以上より、プロンプト設計の違いによりモデルが複雑なフィードバックを理解し修正する能力の差を実現できる可能性が示唆された。また、プロンプトの改良により、単純なフィードバックに対しても学習者の能力差を実現できる可能性が期待される。

第5章　むすび

本研究では、性能が十分でない LLM を擬似学習者として用い、擬似学習者のフィードバックの違いによる修正成功率の違いに着目することで、適切なフィードバック選別の効率化の可能性を検証した。検証の結果、LLM 擬似学習者による各フィードバックの修正成功率の違いが実際の初級学習者の各フィードバックの修正能力の傾向と類似する可能性が期待でき、適切なフィードバック選別を効率化できる可能性が示唆された。また、学習者の能力の違いに応じた適切なフィードバックの選別を目的とし、能力が異なる学習者を 2 つの手法で実現可能か検証した。1 つ目として、LLM モデルのパラメータ数の違いにより、実現する手法である。この手法は結果として、学習者のフィードバックの活用能力の差をある程度実現できる可能性が示唆された。2 つ目として、プロンプト設計の違いにより、実現する手法である。この手法は結果として、限定的ではあるものの、複雑なフィードバックでの学習者の能力の差を実現できる可能性が示唆された。しかし、これら 2 つの手法においては、実現できる可能性が示唆されたのは限定的な学習者の能力の違いであった。今後の課題として、プロンプトの改良により能力の違いの実現範囲を拡大することが挙げられる。また、本研究では、使用した課題内容や誤ったコードが 1 種類のみであったため、汎用性に課題がある。そのため、異なる課題や、別の誤ったコードでの検証を実施したい。

謝辞

本研究において、日頃からご指導頂いた名古屋工業大学、舟橋健司 准教授、伊藤宏隆 助教に心から感謝致します。また、本研究を進めるにあたり多大なご協力をいただきました舟橋研究室の皆様に深く感謝いたします。

参考文献

[1] John Hattie, Helen Timperley, “The Power of Feedback”, Review of Educational Research, Vol.77, No.1, pp.81-112, 2007.

[2] Lianne Roest, Hieke Keuning, Johan Jeuring, “Next-Step Hint Generation for Introductory Programming Using Large Language Models”, Proceedings of the 26th Australasian Computing Education Conference (ACE 2024), Association for Computing Machinery, pp.144-153, 2024.

[3] Tung Phung, Victor-Alexandru Pădurean, Anjali Singh, Christopher Brooks, José Cambronero, Sumit Gulwani, Adish Singla, Gustavo Soares, “Automating Human Tutor-Style Programming Feedback: Leveraging GPT-4 Tutor Model for Hint Generation and GPT-3.5 Student Model for Hint Validation”, Proceedings of the 14th Learning Analytics and Knowledge Conference, pp.12-23, 2024.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”, Advances in Neural Information Processing Systems, Vol.30, 2017.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, Dario Amodei, “Scaling Laws for Neural Language Models”, arXiv:2001.08361, 2020.

[6] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen Weihua Peng, Xiaocheng Feng, Bing Qin, Ting Liu, “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions”, ACM Transactions on Information Systems, Vol.43, No.2, pp.1-55, 2025.

[7] Chengguang Gan, Tatsunori Mori, “Sensitivity and Robustness of Large Language Models to Prompt Template in Japanese Text Classification Tasks”, Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation, pp.1-11, 2023.

[8] Qwen3-8B
https://huggingface.co/Qwen/Qwen3-8B

[9] Qwen3-4B
https://huggingface.co/Qwen/Qwen3-4B