特別研究報告

題目

ファインチューニングしたLLMを用いて
プログラミング問題の難易度を推定する手法の提案

指導教員
肥後 芳樹 教授

報告者
吉田 千優

令和6年2月7日

大阪大学 基礎工学部 情報科学科

令和 5 年度 特別研究報告

ファインチューニングした LLM を用いてプログラミング問題の難易度を推定する手法の提案

吉田 千優

内容梗概

プログラミング学習にプログラミングコンテストの問題を活用することがある．特に初心者にとっては，自分のレベルに応じた問題に取り組むことが重要である．しかし，自身のレベルより難しい問題に取り組んでしまうと，問題を解けないため自信を失ってしまい，学習へのモチベーションを損なう恐れがある．プログラミングコンテストでは，AtCoder や Codeforces などのように出題された問題に対して難易度が付与されている場合がある．このようなコンテストサイトの問題を活用することで，学習者は段階的に問題に取り組むことができると考えられる．しかし，現状ではコンテストサイトごとで難易度の基準が異なっており，サイト間で難易度を比較することができない．また，難易度が付与されていないコンテストサイトも多数存在する．このような課題を解決するために，与えられた問題の情報から難易度の推定を行うことができないか考えた．本研究では，問題文や解答例ソースコードの情報から，その問題の難易度を推定する手法について提案を行う．本手法では GPT-3.5 Turbo に対し，事前に用意した情報でファインチューニングしたモデルを用いて推定する．本手法による推定がどの程度正確であるか，ファインチューニングの効果を評価するために，推定結果の精度について調査を行った．その結果，問題文でファインチューニングしたモデルを用いて問題文の情報から難易度を推定した場合に最も精度が向上した．また，ファインチューニングデータとして，問題文と解答例ソースコードを用いた場合よりも，問題文のみのデータを用いた場合の方が難易度推定の精度が良くなることがわかった．

主な用語

GPT-3.5 Turbo
ファインチューニング
プログラミングコンテスト
プログラミング学習

目 次

1　まえがき

近年，プログラミングコンテストへの参加者が年々増加している [4]．要因としては，日本国内における小中学校でのプログラミング教育必修化など，プログラミングに対する社会的な関心の高まりなどが挙げられる．

プログラミングコンテストでは，プログラミングに関する知識を学び以下の能力を得ることができる．

- 仕様をソースコードに落とし込む能力
- 時間計算量や空間計算量を意識したコードを書く能力
- 解きたい問題に応じて，最も効率的にリソースを活用できるデータ構造やアルゴリズムを選択する能力

また，プログラミングコンテストでは，コンテストにおいて，いかに速く，正確に問題を解くことができるかを競う．また，各コンテストの成績に応じて自分のレーティングが向上するサイトも存在する．このようなサイトでは，ゲーム感覚で楽しみながらプログラミングを学ぶことができる．

参加者とともに発展してきたプログラミングコンテストでは，これまでのコンテストで出題された問題のデータが蓄積されている．これらの問題を解くことで，コンテストに備えるだけでなく，先ほどあげたようなプログラミング知識を学習することができる．特に，その問題が全問題の中でどのくらいの難易度であるかが示されているコンテストサイトでは，自身のレベルに応じた問題に取り組むことができる．コンテストに定期的に参加している学習者であれば，自身のレーティングを参考にして取り組む問題の優先順位をつけることも可能である．この方法は，特にプログラミング学習の初心者にとって有効である．初心者は，自身のレベルよりも難しすぎる問題に取り組むと，自信を損なったり，プログラミング学習へのモチベーションを失ったりしてしまう可能性が高い [3]．そのため，自分のレベルに応じた問題に取り組むことが重要である．このような問題を探すのに利用される代表的なサイトに，AtCoder Problems [5] や CodeForces Problems [6] などがある．

これらのサイトでは，問題の難易度の数値だけでなく，難易度の数値に応じた大まかな分類が色で表されている．そのため，自分のレベルに応じた難易度グループから解くべき問題の選択が可能である．しかしながら，これらのサイトでは独自の難易度基準が用いられている．難易度は各コンテストへの参加者の成績に応じて算出されている．そのため，コンテストの参加者層によって難易度数値は異なる．また，そもそも AtCoder では難易度を表す数値に理論上の上限は定められていないが，Codeforces では 28 段階と定められている．したがって，AtCoder Problems の問題と Codeforces Problems の問題の難易度を単純比較する

ことはできない．もし，同じ基準で難易度を比較することができれば，プログラミング学習者がより多くの問題に取り組む手助けができると考えられる．

本研究では，難易度推定を行う手法として，LLM のファインチューニングを提案する．LLM の能力について，競技プログラミングにおいてある程度の成績を示すことに注目した．GPT-4 の技術報告書においては，Codeforces においてレーティングが 392 の参加者と同程度の能力を示したことが報告されている [7]．これはプログラミング問題の情報を与えた場合に GPT が問題を解釈する能力の高さを示しているだけにとどまらない．問題を解くために必要なアルゴリズムや数学的知識，適切なデータ構造などをうまく選択し，さらにはソースコードに落としこめていると言える．本研究では，このような LLM の問題解釈能力に着目した．LLM は，問題の難しさを総合的に判断できるのではないかと考えたのである．しかし，この「難しさ」を数値として出力させるためには，難易度の基準を LLM に教える必要がある．

GPT に問題と難易度の数値の例を学習させる方法としては，以下の 2 つがある．

- プロンプトで教え込む
- ファインチューニングによって，モデルを追加学習させる

本研究では，大量のデータをまとめて学習させることが可能なファインチューニングを選択した．

難易度推定を行う目的はプログラミング学習者を支援することであるため，問題文のデータからその問題の難易度を推定できるようなモデルを作成する必要がある．したがって，ファインチューニングの際には問題文の情報を含んだデータを用いた．

実際に，ファインチューニングを行ったモデルを用いて難易度推定を行い，実際の難易度とどれくらい差があるかについて精度調査を行なった．また，ファインチューニング前のモデルに対しても同様の精度調査を行うことで，ファインチューニングによってどのくらい精度が向上したのかについて調査した．先行研究と同じメトリクスを用いて精度調査を行うことで，本研究における提案手法の有用性についても示している．

2　背景

本章では，本研究の背景として，プログラミングコンテストで出題される問題の難易度と，プログラミング学習において学習者が抱えている問題について説明する．それらを受けて，現在の課題と本研究の目的について述べる．

2.1　プログラミングコンテスト

プログラミングコンテストでは，出題された問題を，プログラミングを用いてできるだけスピーディーに，正確に解くことを競う．例えば，AtCoder[13] や Codeforces[14] などが有名である．

これらのプログラミングコンテストでは，出題された問題が蓄積されていて，幅広いレベルの問題に触れることができる．問題数が豊富であることと，幅広いレベルの問題に触れられることから，プログラミングコンテストで出題された問題によって，効果的なプログラミング学習が行える．

しかし，プログラミング問題を学習に用いるには注意すべき点がある．それは，経験やレベルに基づいたプログラミング問題にアプローチする必要があることだ．特に初心者にとっては，難しすぎる問題に無意識にアプローチしてしまうことによって，モチベーションの低下や自信の喪失などにつながる恐れがある [3]．

プログラミングコンテストに掲載されている問題の中から，自分のレベルに応じた問題を探すための方法として，プログラミング問題に付与されている難易度の活用が挙げられる．このように，レベル別で問題を探すために，AtCoder Problems [5] や Codeforces Problems [6] などが利用されている．

プログラミングコンテストの問題には，難易度が数値として定められている場合がある．ここでは，問題の難易度に影響する要素について考える．

問題を解くために必要な知識には，以下のようなものが挙げられる．

1. アルゴリズムに関する知識
2. 数学的知識
3. 問題文を読解し，ソースコードに落とし込む能力

これらの要素は，いずれも問題の難しさに影響する要素である．

また，AtCoder や Codeforces では，コンテスト参加者の成績に応じて問題の難易度が決められている．そのため，参加者の層が問題の難易度に与える影響が大きい．参加者の層が異なる要因の 1 つとして，時代とともにコンテスト参加者の全体的なレベルが高くなるこ

とが挙げられる．例えば，あるアルゴリズムに関する典型問題がコンテストで複数回出題されることで，参加者がそのアルゴリズムに慣れる場合などがある．難易度は参加者の能力に依存するため，昔の難易度と今の難易度とでは，同じ数値でも実際の難しさが異なる場合がある．

2.2 プログラミング学習における課題

先ほども触れたように，プログラミングコンテストの問題を学習に活用する場合，各学習者のレベルに応じた問題にアプローチすることが重要である．しかし，現状ではそれぞれのコンテストサイトの難易度が対応づけられていないという問題がある．

プログラミング学習者は，スキルの習得のためにプログラミングコンテストで過去に出題された問題を用いることがある．その中で，同じ難易度の問題をより多く解きたいと考えることがある．この場合，あるコンテストサイトの問題と同じような難易度の問題を別のコンテストサイトから探すという方法がある．しかし，それぞれのコンテストサイトが独自の難易度基準を設けているので，同じ難易度の問題を探すことには苦労する．

これを，AtCoder と Codeforces を例にして説明する．AtCoder Problems と Codeforces Problems を用いる場合を想定する．例えば，AtCoder で茶色レベルに分類されている問題と同じ難易度の問題をより多く解きたい場合を想定する．この場合に，Codeforces で同じレベルの問題を探そうとしても，AtCoder でいう茶色がどの色に該当するのか分からない (表 1)．このような状態は，AtCoder と Codeforces で難易度の基準が異なっているために生じる．

もし，難易度を一般化することができれば，統一された基準で問題の難易度を知ることができるようになるだろう．

2.3 先行研究

本節では，難易度推定に関する先行研究 [8] を紹介する．この研究の貢献の 1 つが，アルゴリズム問題の難易度を予測するためのクラス分類問題に初めて取り組んだことだ．

問題文の情報から，その問題の難易度をどれくらい推定できるかについて調査している．ここでは，アルゴリズム問題として Codeforces から収集した 7,976 件の問題を利用している．これらの問題の難易度の推定精度について，BigBird [9] をベースにしたモデルで実験を行った．結果は表 2 に示す通りである．

Codeforces では，それぞれの問題に対して 28 段階で難易度の数値が付与されている．精度測定に用いたメトリクスの説明は以下の通りである．

- accuracy: 推定難易度と実際の難易度が一致した割合

- CS($\theta = 3$): 推定難易度と実際の難易度との差の絶対値が 3 以下となる割合
- CS($\theta = 5$): 推定難易度と実際の難易度との差の絶対値が 5 以下となる割合
- MAE: 推定難易度と実際の難易度との差の絶対値の平均値

表 2 によると，accuracy は 10.59[%] となっている．したがって，10 問中 1 問は正確な難易度を推定できていることになる．このことから，先行研究で作成したモデルは正確な難易度を知りたい場合には有用ではないといえる．CS については，θ が 3 の場合に 24.85[%] となっている．例えば，実際の難易度が 28 段階中 8 であった場合に，5 以上 11 のいずれかの数値を推定したケースなどが当てはまる．この結果から，おおよその難易度を知りたい場合であっても，良い結果ではないと判断した．また，MAE は 5.08 となっている．これは，正解値から推定値が平均で 5 程度かけ離れていることを示している．

この結果を踏まえて，アルゴリズム問題の難易度推定の精度を向上させる方法について考える．1 つ目の方法として，用いるモデルを変更することが挙げられる．近年，次々と新しい LLM が発表されている [10]．より最新のモデルを用いることで，より良い結果を得られると予測した．もう 1 つの方法として，問題文の情報だけでなく，ソースコードの情報を活用することが挙げられる．プログラミングコンテストでは，参加者が提出したソースコードや，解答例ソースコードのデータが蓄積されている場合がある．問題文のみでは，使用するアルゴリズムなどの知識を推定することは難しい．一方で，ソースコードからアルゴリズムを推定することができるという研究結果 [11] が存在する．このことから，ソースコードのデータを用いることでより良い精度で難易度推定を行えるのではないかと考えた．

表 1: 問題の難易度と色の対応 ([1], [2] より作成)

AtCoder	Codeforces
3600-	
3200-3599	
2800-3199	
2400-2799	2400-
2000-2399	2100-2399
1600-1999	1900-2099
1200-1599	1600-1899
800-1199	1400-1599
400-799	1200-1399
-399	-1199

表 2: 難易度推定の精度

accuracy	CS ($\theta = 3$)	CS ($\theta = 5$)	MAE
10.59	24.85	35.21	5.08

3　提案手法

本章では，プログラミング問題の難易度推定の精度を高めるための手法を提案する．図 1 に，提案手法の概要を示す．

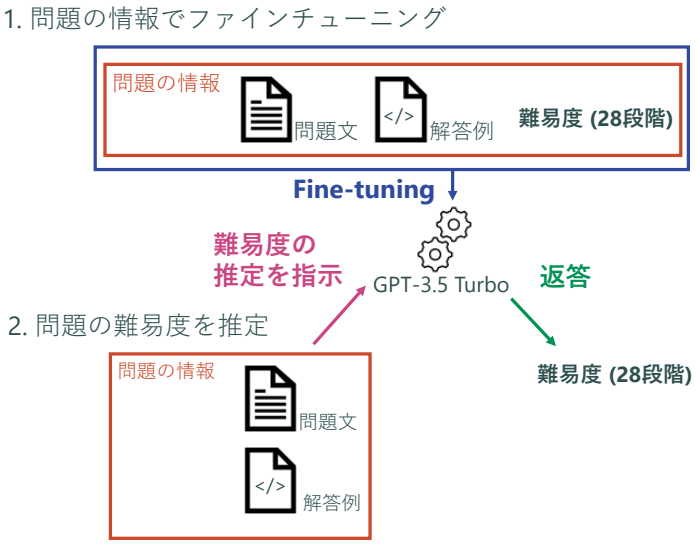

図 1: 提案手法

前章で説明した通り，プログラミング問題の難易度を推定するのにディープラーニングモデルを用いる研究 [8] は存在する．しかし，推定の精度は良くないのが現状である．そこで，本研究においては表 3 のような変更を行うことにする．

3.1　ファインチューニング

ファインチューニングとは，特定のタスクに合わせてモデルをカスタマイズする方法である．ここでは，GPT-3.5 Turbo のファインチューニングを前提とする．本節では，GPT-3.5 Turbo を GPT と呼ぶ．

まず，ファインチューニングを行った理由について説明する．当然のことであるが，難易度の基準を知らなければ，GPT は難易度を数値としてどのように表現して良いかがわから

表 3: 既存研究と本研究の手法の違い

	既存研究	本研究での変更点
モデル	BigBird	ファインチューニングした GPT-3.5 Turbo
データ	問題文のみ	問題文と解答例ソースコード

ない．そのため，難易度の基準を設定し，それを GPT に教える必要がある．プロンプトで例を教え込むのも方法の 1 つである．しかし，公式ドキュメント [12] にもあるように，大量のデータでトレーニングするなら，ファインチューニングが有効である．

次に，ファインチューニングによって GPT に教え込む内容について説明する．図 1 に示す通り，問題文や解答例ソースコードと問題の難易度を教える．本研究の目的は，プログラミング学習者がより快適にプログラミングコンテストの問題を利用できるようにすることだ．どのプログラミングコンテストにおいても，問題文の情報は提供されているので，問題文の情報から難易度の推定を行えることは学習者にとってのメリットとなる．解答例ソースコードについては，多くのコンテストサイトで提供されている情報というわけではない．Codeforces においても，コンテストによっては情報が提供されていないことがある．しかしながら，問題文の情報だけではその問題の難しさを決定する大きな要因の 1 つであるアルゴリズムについての情報を得ることは困難である．そこで，これらの情報を LLM に教えることで，問題の難しさを LLM に総合的に判断させることが可能になるのではないかと考えた．

問題文と解答例ソースコードのデータでファインチューニングを行う場合，以下の 3 種類の組み合わせがある．

1. 問題文のみ
2. 解答例ソースコードのみ
3. 問題文と解答例ソースコード

本研究では，1 と 3 のパターンでファインチューニングを行った．2 のパターンでファインチューニングを行わなかったのは，このパターンでは問題文の情報から難易度を推定することは難しいと考えたためである．もちろん，解答例ソースコードからは難易度を推定可能なモデルを作成できると思われる．しかし，解答例ソースコードが存在しない問題も多数存在する中，このようなモデルを作成したところで学習者にとってのメリットが小さくなってしまう．これに対して，1 と 3 のパターンでファインチューニングしたモデルでは，問題の情報から難易度を推定できるのではないかと考えた．

3.2 難易度の基準について

難易度は，Codeforces と同様に 28 段階の数値で表すことにする．この理由について説明する．

プログラミングコンテストで，出題された問題に対してその難易度を付与しているサイトとして以下の 2 つがある．

- AtCoder [13]

- Codeforces [14]

Codeforces では，各コンテストで出題されたプログラミング問題の難易度がコンテスト終了後に付与される．コンテスト参加者のレーティングとコンテストでの成績に応じて，その問題の難易度が決まる．AtCoder でも同様にして問題に難易度が付与されている．

これらのコンテストサイトにおける問題の難易度について，表 4 のようなことがいえる．表 4 からわかるように，AtCoder の場合は問題の難易度が何段階で表されているかを明確に表現することができない．一方で，Codeforces の場合，800 から 3500 の間で 28 段階で難易度が表されている．Codeforces の難易度基準を用いた方が，ファインチューニングの際にどのような基準で難易度を推定するべきかの指示が明確となる．

また，ファインチューニングの際には問題文やソースコード，難易度の情報が必要である．これらのデータセットを大量に準備することがファインチューニングにおける課題の 1 つである．Codeforces の場合，問題文と難易度に関するデータセットが先行研究で作成されている．

以上の理由から，難易度の基準として，Codeforces と同様の基準を用いることにした．

3.3 GPT のファインチューニング

ファインチューニングした GPT を活用できるようにするためには，以下のステップが必要である [12].

1. データセットを準備してアップロードする
2. モデルをファインチューニングする
3. ファインチューニングされたモデルを使用する

GPT でファインチューニングを行う場合，モデルを指定する必要がある．現在，API で利用可能なモデルは以下に示す通りである [15].

- gpt-3.5-turbo
- davinch-002

表 4: 問題の難易度

コンテストサイト	問題の難易度の表し方	難易度の下限	難易度の上限
Codeforces	800 から 3500 の 100 刻み	あり	あり
AtCoder	1 刻みで表される	あり	なし

- babbage-002

上記 3 つのモデルの中で gpt-3.5-turbo を採用したのは，結果と使いやすさの点で，ほとんどのユーザーにとって適切なモデルとなることが期待されている [12] と記述があったことが理由である．

また，ファインチューニングの際のエポック数についてはデフォルトの 3 を指定した．1 エポックとは学習において訓練データをすべて使い切ったときの回数 [16] を表す．

3.3.1 ファインチューニング用データセット

GPT でファインチューニングを行う場合，事前にデータセットを準備してアップロードする必要がある．GPT では，ファインチューニング用データとしてトレーニングデータの他，バリデーションデータも用いることができる．トレーニングデータは，パラメータの学習に利用する [16]．バリデーションデータは，ハイパーパラメータの性能を評価するために利用する [16]．

データのフォーマットは以下に示すような json 形式である必要がある．

```
{"messages": [
  {"role": "system", "content": "Answer with a number on a scale
  of 28 from 0 to 27. You must output only a number."},
  {"role": "user", "content": プロンプト},
  {"role": "assistant", "content": difficulty}
]}
```

各 “role” において，モデルに学習させる内容を表 5 に示す．

どのようなデータでファインチューニングするかによって，プロンプトの内容が変わる．本研究では 2 種類のファインチューニングを実施する．

1. 問題文のみでファインチューニングするパターン
2. 問題文と解答例ソースコードでファインチューニングするパターン

表 5: “role” と学習内容

role	学習内容
system	どういう答え方をすべきかをモデルに指示.
user	プロンプトとして与えられる情報．ここでは問題の情報.
assistant	どんな答えを出力すべきか．ここでは問題の難易度.

これらを実行するために，プロンプトの内容が異なる 2 種類のデータセットを準備する．

問題文のみでファインチューニングするパターンでは，プロンプトを次のようにした．

```
prompt =
"Please tell me the difficulty of
this competitive programming problem.: 問題文"
```

問題文と解答例ソースコードでファインチューニングするパターンでは，プロンプトを次のようにした．

```
prompt =
"Please tell me the difficulty of
this competitive programming problem. Problem: 問題文
Answer code: 解答例ソースコード"
```

作成したデータセットの情報を表 6 にまとめる．データの集め方は，以下の 2 種類である．

1. 先行研究で作成されたデータセットから収集
2. 問題文のデータと同じ問題の解答例ソースコードを Codeforces の各コンテストの tutorial から収集

表 6 において，解答例ソースコードを含むデータセットのデータ数が，問題文のみのデータ数の約半数となっているのは，公式 tutorial に解答例ソースコードが掲示されていない場合が多数存在したためである．

ファインチューニングには少なくとも 10 個の例が必要で，通常 50 から 100 個のデータで明らかな改善が見られる [12]．表 6 の訓練データに着目すると，問題文でファインチューニングする場合の訓練データ数は 5,501 となっている．また，問題文と解答例ソースコードでファインチューニングする場合は 2,898 となっている．以上より，精度を向上させるのに十分な量のデータを準備できたといえる．

表 6: ファインチューニング用データセット

用途	データの集め方	データ数
問題文でトレーニング	1	5,501
問題文でバリデーション	1	1,346
問題文と解答例ソースコードでトレーニング	2	2,898
問題文と解答例ソースコードでバリデーション	2	695

4　実験手法

本章では，前章で提案した手法で作成したモデルがどのくらいの精度で難易度推定を行えるかを調査する手法について説明する．

4.1　調査内容

本研究では，以下の 3 つが難易度推定の精度に与える影響について調査した．

- 調査 1: ファインチューニングによって難易度推定の精度は向上するか．
- 調査 2: 解答例ソースコードを用いて難易度推定すると精度は向上するか．
- 調査 3: ファインチューニング用データの種類は難易度推定の精度に影響するか．

調査 1 では，問題文で難易度推定する場合について，ファインチューニングをしたモデルとファインチューニングしていないモデルについて精度を比較する．また，先行研究とも比較を行い，精度が向上したかどうかを調べる．

調査 2 では，問題文と解答例ソースコードで難易度推定する場合について，ファインチューニングを行ったモデルとそうでないモデルについて精度を調査する．これらの結果と，問題文のみで難易度推定を行なっている先行研究の結果とを比較することで，解答例ソースコードが精度に与える影響について調査する．

調査 3 では，問題文で難易度推定する場合について，ファインチューニング用のデータが異なる 2 つのモデルの精度を比較する．これによって，解答例ソースコードを含むファインチューニング用データが難易度推定の精度に与える影響について調査する．

4.2　実験の概要

ここでは，本研究での実験の概要について説明する．実験の流れは以下の通りである．

1. テストデータを準備する．
2. 各問題について，難易度を推定する．
3. 推定値と実際の値を比較することで，精度を調査する．

この流れを図示したものが，図 2 である．

この実験を表 7 に示す 5 つの条件で行った．それぞれの条件で行われた実験は，先ほど述べた調査 1，2，3 を調べるために実施した．それぞれの調査と条件との対応を表 8 に示す．

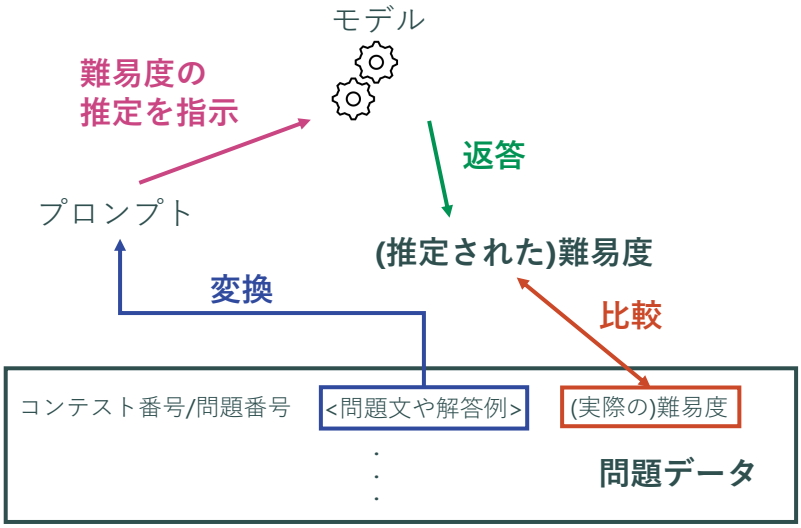

図 2: 実験概要

表 7: 実験の条件

条件	ファインチューニング	テスト
条件 1	なし	問題文
条件 2	なし	問題文と解答例ソースコード
条件 3	問題文	問題文
条件 4	問題文と解答例ソースコード	問題文
条件 5	問題文と解答例ソースコード	問題文と解答例ソースコード

表 8: 調査内容と実験の条件の対応

調査内容	条件	ファインチューニング	テスト
調査 1	条件 1	なし	問題文
	条件 3	問題文	問題文
	条件 4	問題文と解答例ソースコード	問題文
調査 2	条件 2	なし	問題文と解答例ソースコード
	条件 5	問題文と解答例ソースコード	問題文と解答例ソースコード
調査 3	条件 3	問題文	問題文
	条件 4	問題文と解答例ソースコード	問題文

4.3 実験

ここでは，実験の詳細について説明する．実験の流れは，以下に示す通りである．

1. テストデータの各問題データからプロンプトを作成
2. プロンプトを用いて難易度の推定をモデルに指示
3. モデルから返答された推定難易度のデータを収集

各実験で，全く同じテストデータを用いたテストを 10 回ずつ行った．これは，同じデータであっても，試行ごとにモデルの返答が異なるという GPT-3.5 Turbo の性質を考慮したためである．各試行で，どのくらいデータにばらつきがあるのかについて調べた結果を表 9 に示す．

条件 1 で実験した場合の accuracy に着目すると，平均値が 3.60 であるのに対し，標準偏差が 0.765 となっている．これは割合にして 21.3%ほどで，試行ごとのばらつきが大きいことを示唆している．また，LLM の結果は非常に不安定で，同じプロンプトに対して非決定論的に異なるコードを返すことがある．Shuyin らは [17] において，ChatGPT において非決定性が確かに高いことを示しており，LLM に基づく研究において結論を導く際には，この非決定性を考慮するべきだと述べている．以上のことから，1 回のテストのみで精度を調査することは不適切であると考え，各実験につき 10 回テストを行なった．

実験に用いたデータセットの情報を表 10 に示す．データの集め方は，以下の 2 種類である．

1. 先行研究で作成されたデータセットから収集
2. 問題文のデータと同じ問題の解答例ソースコードを Coderorces の各コンテストの tutorial から収集

これらのデータセットから，プロンプトを作成する．問題文のみのテストデータを用いる場合，プロンプトは以下のような形式とした．

表 9: メトリクスのばらつき

実験		accuracy[%]	CS($\theta = 3$)[%]	CS($\theta = 5$)[%]	MAE
条件 1	標準偏差	0.765	0.676	0.107	0.105
	平均	3.60	24.8	37.6	8.64
条件 3	標準偏差	0.426	1.05	1.34	0.106
	平均	8.18	45.7	64.1	4.91

```
prompt = "Please tell me the difficulty of
this competitive programming problem.: 問題文"
```

問題文と解答例ソースコードのテストデータを用いる場合，プロンプトは以下のような形式とした．

```
prompt = "Please tell me the difficulty of
this competitive programming problem. Problem description: 問題文 \\n
 Answer code of this problem: 解答例ソースコード"
```

いずれかの形式のプロンプトを用いて，以下のようにしてモデルへの指示を行った．

```
messages=[
  {"role": "system", "content": "Answer with a number on
  a scale of 28 from 0 to 27. You must output only a number."},
  {"role": "user", "content": prompt},
],
```

モデルには，0 から 27 の 28 段階で難易度を答えるように指示した．このような指示を行い，モデルからの返答として得られた難易度のデータを収集した．

4.4 メトリクス

ここでは，精度を表すために用いた指標について説明する．提案手法が有効であることを示すためには，どれだけ正確に難易度を推定できたかを表すような指標を用いる必要がある．以下では，指標のことをメトリクスと呼ぶ．

本研究で用いたメトリクスの一覧を表 11 に示す．先行研究で用いていたものと同様のメトリクスに加えて，CS で，θ の値のみ変更したものについても調査した．

それぞれのメトリクスの見方について説明する．accuracy と CS は，高い方が精度が良いことを示すメトリクスである．accuracy が高いと，それだけ正確な値を推定できているということを示す．CS($\theta = n$) については，n がより小さい場合に CS が高い数値を示すと，よ

表 10: 実験用データセット

用途	データの集め方	データ数
問題文のみで難易度推定	1	847
問題文と解答例ソースコードで難易度推定	2	340

り正確な値に近い値を推定できているということを示す．最も望ましいのは accuracy の値が高くなることである．次に望ましいのは，n の値がより小さい CS($\theta = n$) の値が高くなることである．一方，MAE(Mean Absolute Error) は推定難易度と実際の難易度とにどの程度差があるかを示す値である．MAE はより低い数値であることが望ましい．それぞれのメトリクスの計算式を以下に示す．

- $accuracy = num\_of\_same/sum$
  - $num\_of\_same =$ (実際の難易度と推定難易度が一致した個数)
  - $sum =$ (テストデータの総数)
- $CS(\theta = 1) =$ (実際の難易度と推定難易度の差の絶対値が 1 以下の個数)$/sum$
- $CS(\theta = 2) =$ (実際の難易度と推定難易度の差の絶対値が 2 以下の個数)$/sum$
- $CS(\theta = 3) =$ (実際の難易度と推定難易度の差の絶対値が 3 以下の個数)$/sum$
- $CS(\theta = 4) =$ (実際の難易度と推定難易度の差の絶対値が 4 以下の個数)$/sum$
- $CS(\theta = 5) =$ (実際の難易度と推定難易度の差の絶対値が 5 以下の個数)$/sum$
- $MAE = sum\_of\_ae/sum$
  - ae: absolute error
  - $sum\_of\_ae =$ (実際の難易度と推定難易度の差の絶対値の総和)

上記の式を用いる際，GPT の失敗も含めて計算を行なっている．ファインチューニングをしていない GPT-3.5 Turbo で実験した際，テストの過程で，GPT が失敗するケースがあった．失敗例を以下に示す．

表 11: 精度調査に用いたメトリクスの種類と説明

メトリクス	説明
accuracy	推定難易度と実際の難易度が一致した割合
CS($\theta = 1$)	推定難易度と実際の難易度との差の絶対値が 1 以下に収まる割合
CS($\theta = 2$)	推定難易度と実際の難易度との差の絶対値が 2 以下に収まる割合
CS($\theta = 3$)	推定難易度と実際の難易度との差の絶対値が 3 以下に収まる割合
CS($\theta = 4$)	推定難易度と実際の難易度との差の絶対値が 4 以下に収まる割合
CS($\theta = 5$)	推定難易度と実際の難易度との差の絶対値が 5 以下に収まる割合
MAE	推定難易度と実際の難易度との差の絶対値の平均値

- 数値を返さない.
- 0-27 という制約を無視して，それ以外の数字を返す.
- 問題に対する解答例を答えてしまう.

それぞれの試行で得られた結果について，これらのメトリクスを計算する．最後に，10 回の試行それぞれで得られたメトリクスの平均値を取り，これを実験結果とした.

5　実験結果

本章では，プログラミング問題の難易度推定の精度に影響する要素について調査した結果について述べる．前章で説明した通り，調査内容を 3 つに分けて調べた．これらそれぞれについて，調査結果を示す．

まずは，結果の見方について説明する．5 つの条件で実験を行い，メトリクスを算出している．特に，CS($\theta = n$) の値を見る際は，難易度推定は 28 段階で行なっていることに留意すべきである．例えば，CS($\theta = 2$) について考える．これは，実際の難易度が 2000 であった場合に，推定難易度が 1800， 1900， 2000， 2100， 2200 のいずれかであるようなケースの割合を表す．Codeforces Problems において，難易度が 1800 の場合は青色で，2200 の場合は橙色と，2 段階の差がある (表 1)．

次に，実際にテストを行って，結果を得ることができたデータ数について述べる．表 12 からわかるように，解答例ソースコードを含むテストデータを用いた場合，テストデータ数と結果が得られたデータ数に差分が生じている．解答例ソースコードを含むデータセットからプロンプトを作成する場合，プロンプトのトークン数が非常に大きくなってしまうことがある．この場合，トークン数が 4,097 を超えると，プロンプトのトークン制限に引っかかってしまう．このような場合を除いた結果，結果が得られたデータ数が 267 となった．

5.1　調査 1: ファインチューニングによって難易度推定の精度は向上するか

本節では，ファインチューニングを行うことで，難易度推定の精度が向上するかについて調べた結果について説明する．それぞれの実験の条件について表 13 に示す．これらの実験を行った結果を図 3 に示す．詳細な数値は，表 14 に示す通りである．

accuracy については，先行研究，条件 3，条件 4，条件 1 の順で良い数値を示している．先行研究のモデルが最も高い割合で正確な難易度を推定できている．ファインチューニングをしない場合よりは，ファインチューニングした場合の方が正確な値を推定できている．

表 12: 結果が得られたデータ数

条件	FT	テスト	テストデータ	結果が得られた
条件 1	なし	問題文	847	847
条件 2	なし	問題文と解答例	340	267
条件 3	問題文	問題文	847	847
条件 4	問題文と解答例	問題文	847	847
条件 5	問題文と解答例	問題文と解答例	340	267

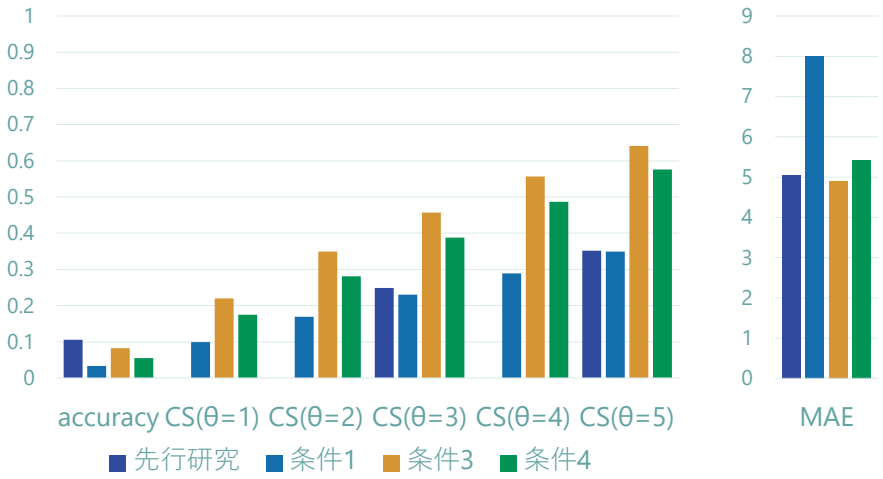

図 3: 調査 1 の実験結果

CS($\theta = 3, 5$) については，条件 3，条件 4，先行研究，条件 1 の順で良い数値を示している．CS($\theta = 1, 2, 4$) については，条件 3，条件 4，条件 1 の順で良い数値を示している．また，いずれの条件の場合においても，θ の値が大きくなるほど精度が向上している．以上より，ファインチューニングによって，正解値に近い値を推定する割合が増加しており，正解値とのずれに対する許容範囲を大きくすればするほど，精度が向上していると言える．

MAE については，条件 3，先行研究，条件 4，条件 1 の順に良い数値を示している．このメトリクスからは，どれくらい正解値からかけ離れた値を推定しているかについて調べることができ，問題文でファインチューニングした場合に最も良い精度となっている．

以上の結果を踏まえて，ファインチューニングによって，正解値に近い値を推定する割合が増加していると推測することができる．正解値と完全に一致する難易度を推定することはできていないが，大まかな難易度の分類については，先行研究よりも有用なモデルを得ることができたと言える．

表 13: 調査 1 で実施した実験

実験を行う条件	モデル	ファインチューニング	テスト
先行研究	BigBird	-	問題文のみ
条件 1	GPT-3.5 Turbo	なし	問題文のみ
条件 3	GPT-3.5 Turbo	問題文	問題文のみ
条件 4	GPT-3.5 Turbo	問題文と解答例ソースコード	問題文のみ

表 14: 調査 1: 各メトリクスの数値

	accuracy	CS($\theta = 1$)	CS($\theta = 2$)	CS($\theta = 3$)	CS($\theta = 4$)	CS($\theta = 5$)	MAE
先行研究	**0.106**	-	-	0.249	-	0.352	5.06
条件 1	0.033	0.099	0.169	0.230	0.289	0.349	8.00
条件 3	0.082	**0.220**	**0.349**	**0.457**	**0.557**	**0.641**	**4.91**
条件 4	0.055	0.175	0.281	0.388	0.487	0.576	5.42

5.2 調査 2: 解答例ソースコードを用いて難易度推定すると精度は向上するか

本節では，解答例ソースコードを用いて難易度推定を行うと，その精度が向上するかどうかについて調べた結果について説明する．それぞれの実験の条件を表 15 に示す．これらの実験を行った結果を図 4 に示す．詳細な数値は，表 16 に示す通りである．

accuracy については，先行研究，条件 5，条件 2 の順に良い数値を示した．先行研究のモデルが最も高い割合で正確な難易度を推定できている．また，ファインチューニングをしない場合よりは，ファインチューニングした場合の方が正確な値を推定できている．

CS($\theta = 3, 5$) については，条件 5，先行研究，条件 2 の順で良い数値を示している．CS($\theta = 1, 2, 4$) については，条件 5，条件 2 の順で良い数値を示している．また，いずれの条件の場合においても，θ の値が大きくなるほど精度が向上している．これらのことから，解答例ソースコードを難易度推定に用いる場合，ファインチューニングによって，難易度の基準を教えると，先行研究よりも正解値に近い値を推定する割合が増加することがわかった．

MAE については，先行研究，実験 5，実験 2 の順に良い数値を示している．解答例ソースコードを難易度推定に用いることで，平均的に正解値からかけ離れた数値を推定するようになったことがわかる．一方で，CS からわかるように，正解値に近い値を推定する割合は増加している．以上を踏まえると，正解値からかけ離れた数値を推定しているケースの割合

表 15: 調査 2 で実施した実験

実験を行う条件	モデル	ファインチューニング	テスト
先行研究	BigBird	-	問題文のみ
条件 2	GPT-3.5 Turbo	なし	問題文と解答例ソースコード
条件 5	GPT-3.5 Turbo	問題文と解答例ソースコード	問題文と解答例ソースコード

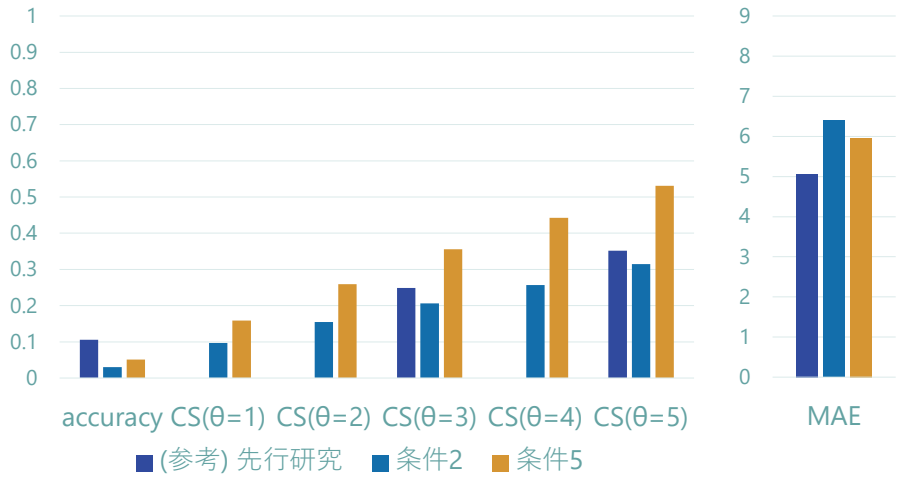

図 4: 調査 2 の実験結果

表 16: 調査 2: 各メトリクスの数値

	accuracy	CS($\theta = 1$)	CS($\theta = 2$)	CS($\theta = 3$)	CS($\theta = 4$)	CS($\theta = 5$)	MAE
先行研究	**0.106**	-	-	0.249	-	0.352	**5.06**
条件 2	0.030	0.097	0.155	0.206	0.257	0.315	6.40
条件 5	0.051	**0.159**	**0.259**	**0.356**	**0.443**	**0.531**	5.96

が増加したのではないかと考えられる．

以上より，解答例ソースコードを難易度推定に用いることで，正解値に近い値を推定する割合が増加した一方で，正解値からかけ離れた難易度を推定する割合が増加したと言える．

5.3 調査 3: ファインチューニング用データの種類は難易度推定の精度に影響するか

本節では，ファインチューニング用のデータを問題文のみにした場合と，問題文と解答例ソースコードにした場合とで難易度推定の精度にどのような影響があるかについて調べた．実施した実験の条件について表 17 にまとめる．これらの実験を行った結果を図 5 に示す．詳細な数値は，表 18 に示す通りである．

全てのメトリクスにおいて，条件 3 の方が条件 4 よりも良い数値を示した．よって，問題文のみでファインチューニングする方が，問題文と解答例ソースコードでファインチューニングした場合よりも良い結果を示したとわかった．

ここで，ファインチューニングで使用した 2 種類のデータセットについて，有意差があ

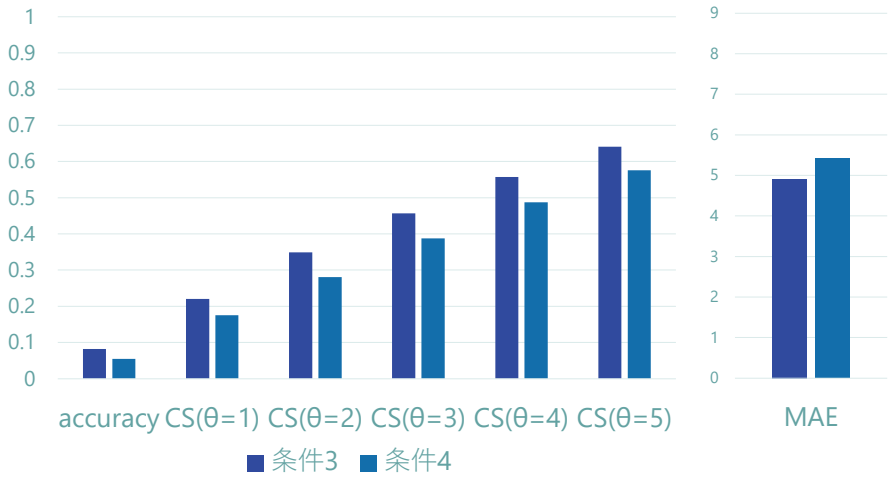

図 5: 調査 3 の実験結果

るかどうかについて調べた．異なるグループのデータを比較したときに，その差が偶然によって生じた誤差の範囲に収まるものではなく，理由や原因があって生じた差である場合に，データ間に有意差があるという．これを調べるために，以下の結果について t 検定を行った．

1. 問題文でファインチューニングしたモデルを用いて，問題文から推定した難易度の一覧 (10 回分)
2. 問題文と解答例ソースコードからファインチューニングしたモデルを用いて，問題文から推定した難易度の一覧 (10 回分)

この結果，P 値が 0.00019651 となり，これは 0.05 より小さい．一般的に，t 検定を行った結果，P 値が 0.05 より小さい場合に有意差が認められたとすることが多い．したがって，ファインチューニング用データが異なるモデルから推定された難易度間で生じた差は，偶然によるものではないと推測される．

以上のことを踏まえて，ファインチューニング用データに解答例ソースコードを用いると，何らかの要因によって難易度推定の精度が低下することがわかった．この理由については，

表 17: 調査 3 で実施した実験

実験を行う条件	モデル	ファインチューニング	テスト
条件 3	GPT-3.5 Turbo	問題文	問題文
条件 4	GPT-3.5 Turbo	問題文と解答例ソースコード	問題文

次章で考察する．

5.4 考察

ここでは，ファインチューニング用データと精度の関係について考察する．本研究では，2 種類のファインチューニングデータを用いた．

1. 問題文のみのデータセット

2. 問題文と解答例ソースコードのデータセット

調査 3 の結果からわかるように，ファインチューニングデータ 1 を用いた場合の方が精度が良かった．解答例ソースコードを用いると精度が向上するという予測に対し，実際には精度が悪化した．この要因として考えうるものを挙げる．

1 つ目の要因として，各プログラミング言語のデータ数の少なさが挙げられる．データセットに含まれる解答例ソースコードを記述するプログラミング言語は，C++，Java，Python，Kotlin などである．言語の多様なことで，1 言語あたりのファインチューニングデータが少なくなってしまう．これによって，GPT の性能改善に十分なだけのデータ数を確保できなかった言語があるかもしれない．今後の展望としては，まずプログラミング言語ごとのファインチューニング用データ数を調査する必要がある．次に，ファインチューニング用データ数の少なさと，その言語で記述されたソースコードから難易度推定した場合の精度とに相関があるかどうかについて調べたい．

2 つ目の要因として，難易度別のデータ数の違いが挙げられる．Codeforces から解答例ソースコードのデータを収集する際に，容易な問題については解答例を省略しているコンテストや，逆に難易度の高い問題について解答例の記載がされていないコンテストなどが見受けられた．このように，難易度の数値が低めの問題や，高めの問題について，十分な量のデータが確保できていなかった可能性がある．そのため，難易度別にファインチューニング用データを分割したときに，極端にデータ数が少ないものがあるかどうかを調べる必要がある．次に，正解値の難易度で分割したグループに対して，各グループの難易度推定精度について調べ，ファインチューニング用データ数と精度との相関について調べたい．

表 18: 調査 3: 各メトリクスの数値

	accuracy	CS($\theta = 1$)	CS($\theta = 2$)	CS($\theta = 3$)	CS($\theta = 4$)	CS($\theta = 5$)	MAE
条件 3	**0.082**	**0.220**	**0.349**	**0.457**	**0.557**	**0.641**	**4.91**
条件 4	0.055	0.175	0.281	0.388	0.487	0.576	5.42

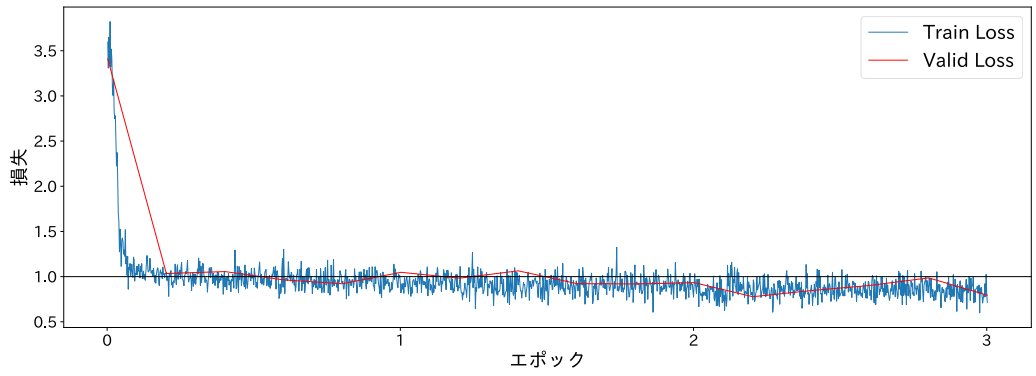

図 6: 問題文のみでファインチューニングした場合の損失関数

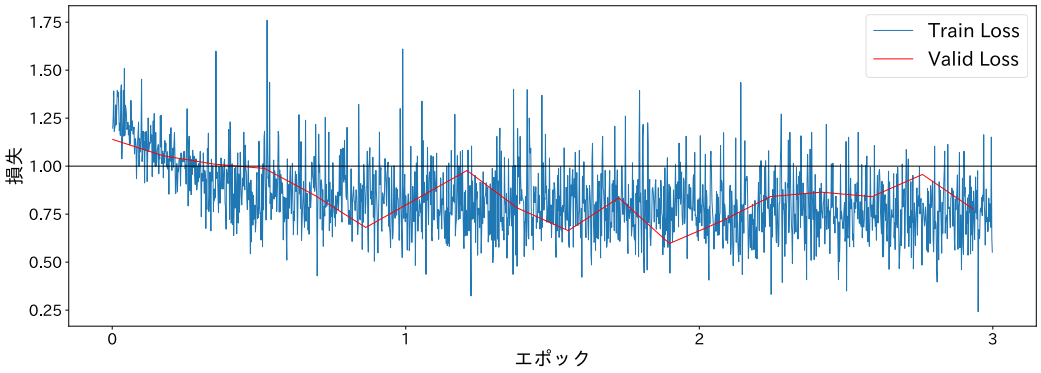

図 7: 問題文と解答例ソースコードでファインチューニングした場合の損失関数

3 つ目の要因として，問題文と解答例ソースコードでファインチューニングした場合のエポック数が不十分だった可能性がある．図 6 は，問題文でファインチューニングした場合の損失関数のグラフである．エポック数が増えるにつれて損失関数の値が小さくなっており，学習が進んでいることがわかる．これに対して，問題文と解答例ソースコードでファインチューニングした場合の損失関数のグラフが図 7 である．損失関数の値があまり減少していないことがわかる．今後，エポック数をより増やして再度ファインチューニングを行い，そのモデルに対して精度を調査したい．

6　まとめと今後の課題

本研究では，プログラミングコンテストをプログラミング学習に効果的に用いるために，プログラミング問題の難易度推定に取り組んだ．問題文，解答例ソースコードの情報から問題の難易度推定を行えるようにするために，問題文のみ，または問題文と解答例ソースコードでファインチューニングした GPT-3.5 Turbo の難易度推定精度を調査した結果，以下のことが明らかになった．第一に，問題文でファインチューニングした GPT-3.5 Turbo はこれまでに先行研究で作成されたモデルよりも，正解値に近い値を推定する割合が高いといえる．第二に，解答例ソースコードは難易度推定の精度向上に貢献しなかった．ただし，これは現状の手法における結果であり，手法の改善によって精度改善につながるのではないかと考えられる．第三に，ファインチューニング用データとして問題文と解答例ソースコードのデータセットを用いた場合よりも，問題文のみのデータセットを用いる場合の方が精度が高いといえる．これは，ソースコードの記述言語が複数種類あったことや，難易度別のデータ数の違い，学習時のエポック数が不十分だったことなどが原因であると考えられ，今後調査する必要がある．

ここからは，本研究の今後の展望について述べる．各メトリクスについて最も数値が良かったものを表 19 にまとめる．accuracy については，先行研究の数値から改善することができなかった．一方，CS と MAE については条件 3 の場合に改善が見られた．条件 3 での結果は，問題文でファインチューニングしたモデルに対して，問題文で難易度推定を行いその精度を計算したものである．したがって，GPT をファインチューニングする方法では，28 段階で表された難易度について正解値に近い値を推定する能力が上がったと言える．以下では，この事実の応用に関する今後の展望と，解答例ソースコードを精度改善に活用する方法について説明する．

表 19: 最も良い数値を示したメトリクス

メトリクス	条件	数値
accuracy	先行研究	0.106
CS($\theta = 1$)	条件 3	0.220
CS($\theta = 2$)	条件 3	0.349
CS($\theta = 3$)	条件 3	0.457
CS($\theta = 4$)	条件 3	0.557
CS($\theta = 5$)	条件 3	0.641
MAE	条件 3	4.91

はじめに，現在と同様の手法を用いてより精度を良くする方法について考える．現在の手法の問題点として以下の 2 つが挙げられる．

1. accuracy が改善されていない

2. 解答例ソースコードを活用できていない

これらの問題点を解決するためのデータセット改善の方針を示す．まずは，accuracy に改善が見られなかった理由について考える．これは，28 段階の難易度に分類するにはファインチューニング用データ数が足りなかったためではないかと予測される．これを確かめるために，難易度別で，ファインチューニングデータ数と難易度推定の精度について調査する必要がある．必要があれば，難易度に応じて問題を追加したデータセットを作成する．次に，解答例ソースコードを活用する方法について考える．6.1 節でも述べたように，問題ごとに別の言語で記述されていた．そのため，言語によってはデータ数が少なくなっている可能性がある．各言語と難易度推定の精度に相関があるかどうかについては調査する必要があるが，プログラミング言語を 1 種類に限定したデータセットを用いてファインチューニングを行うことで精度が改善すると予測される．

最後に，難易度推定の精度を改善するためのアイデアについて説明する．1 つ目は，難易度の基準を 3 段階や 5 段階に変更して，与えられた問題の難易度の大まかなレベルを推定するツールである．これは，ファインチューニング後の GPT が，おおよその難易度であれば推定精度が高かったことを利用している．難易度の基準を大まかにするほど，このツールの精度は高くなると考えられる．今後，このツールの作成と精度測定を行いたいと考えている．2 つ目は，学習者のレベルに応じて，与えられた問題が解けるか解けないかを判定するツールである．このツールでは，学習者がどのような問題を解けるのかについて学習する．その上で，与えられた問題を学習者が解けるかどうかを解ける・解けないの 2 値分類で判定する．このツールの応用として，解けない場合であっても，学習者にとってどの程度難しいのかを数値で表すツールの作成にも取り組みたいと考えている．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 肥後 芳樹 教授には，中間報告会等において，研究の方針に対するご助言や，発表資料に対する的確なご指摘を賜りました．また，研究の途中経過について発表する機会をいただくことで，研究への意欲を保ち続けることができました．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授には，研究を進めるにあたり，方針の立て方から，作業における優先順位の付け方，研究を進めるのに必要な考え方など研究のいろはについて教えていただきました．研究活動に不慣れな中で至らない点ばかりでしたが，毎週のミーティングをはじめとする様々な機会で熱心なご指導により，無事に卒業論文を完成させることができました．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 神田 哲也 助教には，研究室内の発表機会において，貴重なご意見，ご助言を賜りました．特に，発表資料の細部について相談にのっていただきました．

九州大学大学院システム情報科学研究院 亀井 靖高 教授には，研究会におきまして，論文を完成させるために何をすべきかについて，様々なご助言を賜りました．

東京工業大学情報理工学院 林 晋平 准教授には，研究会におきまして，本研究における本質的な課題について貴重なご意見を賜りました．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松本 真佑 助教には，研究会におきまして，発表資料や論文の質をよりよくするための貴重なご意見を賜りました．

九州大学大学院システム情報科学研究院 近藤 将成 助教には，研究会におきまして，本研究で課題となりそうな点について貴重なご意見を賜りました．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 博士前期課程 川渕 皓太 氏には，研究テーマの策定から，日頃の研究活動を進めるうえでの困りごとなど，様々な点で相談にのっていただきました．特に，私の研究におけるキーアイデアや方針について共に考えていただきました．また，資料の添削においても，基本的な考え方から教えていただきました．

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 事務職員 軽部 瑞穂 氏は，私たち学生が快適な環境で研究を行うため，研究室の環境整備や事務的作業といったご支援を賜りました．また，私が悩みを抱えていた際には，親身になって相談に乗ってくださり，解決に向けて手助けをしていただきました．

最後に，その他様々なご指導，ご助言等を賜りました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻 肥後研究室の皆様に，心より深く感謝申し上げます．

参考文献

[1] Algorithm 部門のレーティングと業務における期待できる活躍. https://info.atcoder.jp/utilize/jobs/rating-business-impact. Accessed: Feb 2024.

[2] Codeforces のすすめ for AtCoder ユーザー. https://ywmt.hatenablog.com/entry/2019/10/17/120914. Accessed: Feb 2024.

[3] Chowdhury Md Intisar and Yutaka Watanobe, "Cluster Analysis to Estimate the Difficulty of Programming Problems,"pp.23-28, Aizu-Wakamatsu, Japan, 2018. DOI: 10.1145/3274856.3274862

[4] "日本最大のプログラミングコンテストサイト AtCoder　全世界での登録者数が 50 万人を突破！". PR TIMES. https://prtimes.jp/main/html/rd/p/000000038.000028415.html. Accessed: Feb 2024.

[5] AtCoder Problems. https://kenkoooo.com/atcoder/. Accessed: Feb 2024.

[6] Codeforces Problems. https://cf.kira924age.com/. Accessed: Feb 2024.

[7] OpenAI, "GPT-4 Technical Report, "arXiv.2303.08774, 2023.

[8] Juntae Kim and Eunjung Cho and Dongwoo Kim and Dongbin Na, "Problem-Solving Guide: Predicting the Algorithm Tags and Difficulty for Competitive Programming Problems, "arXiv.2310.05791, 2023.

[9] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santi- ago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al, "Big bird: Transformers for longer sequences. Advances in neural information processing systems, "33:17283-17297, 2020.

[10] Wayne Xin Zhao and Kun Zhou and Junyi Li and Tianyi Tang and Xiaolei Wang . . . and Ji-Rong Wen, "A Survey of Large Language Models, "arXiv.2303.18223, 2023.

[11] Chourasia, Pranshu and Ramakrishnan, Ganesh and Apte, Varsha and Kumar, Suraj, "Algorithm identification in programming assignments, "Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, pp.471-481, 2022.

[12] "Fine-tuning". Welcome to the OpenAI developer platform. https://platform.openai.com/docs/guides/fine-tuning. Accessed: Feb 2024.

[13] AtCoder. https://atcoder.jp/. Accessed: Feb 2024.

[14] Codeforces. https://codeforces.com/. Accessed: Feb 2024.

[15] “Pricing”. OpenAI. https://openai.com/pricing. Accessed: Feb 2024.

[16] 斎藤 康毅, ゼロから作る Deep Learning—Python で学ぶディープラーニングの理論と実装, 株式会社オライリー・ジャパン, 東京, 2016 年.

[17] Shuyin Ouyang and Jie M. Zhang and Mark Harman and Meng Wang, “LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation, ”arXiv.2308.02828, 2023.