一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

GPT-4 における自動コード翻訳タスクの精度調査

川渕　皓太†　　松下　　誠†　　肥後　芳樹†

† 大阪大学大学院情報科学研究科　〒565-0871 大阪府吹田市山田丘 1-5
E-mail: †{k-kawabt,matusita,higo}@ist.osaka-u.ac.jp

あらまし　あるプログラミング言語で書かれたソースコードを別の言語に翻訳する自動コード翻訳の研究において，機械学習を用いるニューラル機械翻訳が注目されている．ニューラル機械翻訳は，事前に翻訳ルールを逐一定義するルールベース機械翻訳と比べて，人間が読みやすいコードが生成されやすいというメリットがある一方で，翻訳の精度が低いという問題がある．本研究では様々な分野で高い性能を発揮している大規模言語モデル GPT-4 を対象として，C++，COBOL，Go，Java，Rust，Python3 のソースコードに対する自動コード翻訳タスクの精度を調査した．既存のデータセットは GPT-4 が事前学習済みである可能性が高いため，新たにデータセットを作成して調査を行った．その結果，GPT-4 が既存の事前学習言語モデルよりも精度が高いこと，翻訳元言語よりも翻訳先言語のほうが精度への影響が大きいこと，プロンプトの記述言語が日本語の場合と英語の場合で精度に大きな差がないことがわかった．
キーワード　自動コード翻訳，GPT-4

Measuring the Accuracy of Code Translation Tasks in GPT-4

Kota KAWABUCHI†, Makoto MATSUSHITA†, and Yoshiki HIGO†

† Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka, 565-0871 Japan
E-mail: †{k-kawabt,matusita,higo}@ist.osaka-u.ac.jp

Abstract　Neural machine translation using machine learning has been attracting attention in research on automatic code translation that translates source code written in one programming language into another language. Neural machine translation provides the benefit of readily generating human-readable code compared to rule-based machine translation, which requires pre-defined rules for translation. On the other hand, neural machine translation has the problem of low translation accuracy. In this study, we investigated the accuracy of automatic code translation tasks for C++, COBOL, Go, Java, Rust, and Python3 source code on GPT-4, large language models that have demonstrated high performance in various fields. We created a new dataset for the study because GPT-4 is likely to have been pre-trained on an existing dataset. The results show that GPT-4 is more accurate than the existing pre-trained model, the impact on accuracy is more significant with the target language than with the source language, and there is no significant difference in accuracy when the prompts are in Japanese and in English.
Key words　Code Translation, GPT-4

1. はじめに

自動コード翻訳とはあるプログラミング言語で書かれたソースコードを別の言語に翻訳する技術のことである．COBOL から Java などのレガシーな言語からモダンな言語への翻訳や，Python3 から C++ など低速高水準言語から高速低水準言語への翻訳などに利用されている [1], [15].

自動コード翻訳の手法の 1 つとして，ルールベース機械翻訳がある (Cython [1]，opensourceCOBOL4j [15] など)．これはルールや辞書に沿って翻訳を行う手法のことである．この手法はルールの範囲内であれば正確に翻訳を行うことができる．しかし，翻訳のために大量のルール登録を手動で行う必要があり，このルール登録には非常に労力がかかる．また，人間にとって読みにくい不自然な翻訳が出力される場合もある [17].

近年，ルールベース機械翻訳に代わる手法としてニューラル機械翻訳が提案された [3]．これは機械学習を用いて翻訳を行う手法であり，メリットとして，人間が作成したコードを学習しているため人間が理解しやすい翻訳が出力されやすいこと，ルールベース機械翻訳と比較してルール登録の手間がかからないということが挙げられる．しかし最新のニューラル機械翻訳

モデルの 1 つである TransCoder-IR [17] においても，C++，Go，Java，Rust 間を相互に翻訳した場合の精度が平均 44.2% と低く，正確に翻訳ができないことが多いという問題がある．

2023 年 3 月に OpenAI 社から最新の大規模言語モデル (Large Language Model, LLM) である GPT-4 [13] が公開された．LLM とは，事前学習言語モデル (Pre-trained Language Model, PLM) のモデルサイズ，学習データサイズをスケーリングすることにより，通常の PLM よりも複雑なタスクを解く能力 (創発的能力 [19]) を獲得したモデルの呼称である [22]．GPT-4 は学習に自然言語，ソースコード，画像を使用しており，Uniform Bar Examination(米国統一司法試験) を受験者の上位 10 %の成績で合格 [13]，自然言語翻訳において既存の商用製品レベルのスコアを達成 [8]，既存モデルよりも高精度なコード理解，コード生成を実現 [2], [13]，といった多くの分野で成果を残している．

GPT-4 は高精度でコード理解，コード生成を行うことができるため，自動コード翻訳タスクも高精度で行うことができると予想できる．しかし，GPT-4 のコード翻訳精度を調査する研究はまだ存在しない．

そこで本研究では GPT-4 の自動コード翻訳タスクの精度を調査した．対象の言語として C++，COBOL，Go，Java，Rust，Python3 を選択した．

本研究の主な貢献は以下の通りである．

- GPT-4(gpt-4-0613) が事前学習していないデータから新たにコード翻訳タスクのテスト用データセットを作成した．
- GPT-4 が既存の自動コード翻訳モデルよりも高精度でコード翻訳を行えることを明らかにした．
- 翻訳先言語，翻訳元言語の組み合わせによって翻訳の精度が変化することを明らかにした．
- プロンプトの言語が日本語である場合と英語である場合の翻訳の精度の変化を調査し，大きな差がないことを明らかにした．

2. 背　　景

2.1 LLM

近年，大規模なデータセットを用いて学習された LLM が注目を集めている [22]．その中でも，OpenAI 社が開発した LLM である GPT-3.5 [14]，GPT-4 [13] が高精度であるとして特に注目されている．GPT-3.5，GPT-4 は基本的に入力としてテキストを与え，その入力テキストの内容によって出力が変化する．この入力テキストを一般的にプロンプトと呼び，プロンプトによって出力の質が変化する [18] ため，より良い出力を得るためにこのプロンプトを工夫することが重要視されている．これをプロンプトエンジニアリング [11], [20] と呼び，例えば思考の連鎖を促すことで出力の質を上げる Chain-of-Thought [18] などのテクニックがある．また，プロンプトの言語が精度に影響を与えるという報告も行われている [4], [9], [13].

2.2 LLM とコード翻訳

関連研究 [7] では StarCoderBase [10] と GPT-3.5-turbo [14] のコード翻訳精度の評価を行っている．Java，Python3，C++，C#，JavaScript を含む新たなデータセットである G-TransEval を作成して評価した結果，これらの LLM は既存の事前学習言語モデルよりも高い精度を発揮し，特に GPT-3.5-turbo が良い性能を発揮したと報告されている．

しかし作成されたデータセットに含まれるデータは LLM が事前学習済みの可能性があり，そのことが精度に影響を及ぼしている可能性についても同研究では述べられている．また，COBOL などのレガシーな言語を含む翻訳性能の評価は行われていない．

2.3 調 査 内 容

本研究では，GPT-4 の自動コード翻訳タスクの精度に関する 4 つの調査を行った．

調査 1: ニューラル機械翻訳モデルと GPT-4 の比較

GPT-4 がニューラル機械翻訳モデルと比較して有用かどうかを調査する．比較対象として最新のニューラル機械翻訳モデルの 1 つである TransCoder-IR [17] を選択する．TransCoder-IR は関数単位の翻訳を行うモデルであるため，GPT-4 でも関数単位で翻訳を行い精度を比較する．

調査 2: 異なる特徴の言語間における翻訳の精度

C++，COBOL，Go，Java，Rust，Python3 のソースコードを翻訳し，各言語間の精度の傾向を調査する．同時にコード単位での翻訳の精度も調査する．

調査 3: プロンプトの言語が翻訳の精度に与える影響

MMLU データセット [5], [6] においてプロンプトの言語が日本語より英語の場合のほうが 5.6 %精度が高いと報告されている [13] 一方，安全なコードの生成においてはプロンプトの言語が日本語の場合も英語の場合も精度は同等であるという報告 [21] があるなど，タスクによってプロンプトの言語の影響が異なることが報告されている．調査 3 では自動コード翻訳タスクにおけるプロンプトの言語の影響を調査する．

調査 4: 外部ツールが翻訳の精度に与える影響

調査 2 の結果において，翻訳先言語が Go である場合のコンパイルエラーに import されているが使用されていないパッケージが存在することによるエラーが多く含まれていたが，これは goimports[1] という静的解析ツールを使うことで自動解決できる．同様に，翻訳先言語が Rust である場合のコンパイルエラーも cargo-fix[2] を用いることで自動解決できる．調査 4 ではこれらの静的解析ツールを用いることでどのように精度が向上するか調査する．

3. 調　　査

本研究で使用する GPT-4 のモデル (gpt-4-0613) は 2021 年 9 月以前のデータで学習を行っている [13]．コード翻訳タスクの評価は通常 CodeXGLUE [12] 等のデータセットが使用されるが，これらのデータセットは GPT-4 が既に学習している可能性がある．そこで，本研究では GPT-4 が事前学習していないデータからデータセットを新たに作成して調査を行う．データセットは C++，COBOL，Go，Java，Python3，Rust の並列ソー

(注1)：https://pkg.go.dev/golang.org/x/tools/cmd/goimports
(注2)：https://doc.rust-lang.org/cargo/commands/cargo-fix.html

表 1　データセットのソースコード情報

言語	ソースコード数	平均行数
C++	140	22.5
COBOL	98	35.5
Go	140	19.7
Java	140	24.0
Python3	140	9.1
Rust	140	16.5

スコードとテストケースを含む．

3.1　データセットの作成

データセットは競技プログラミングサイトである AtCoder[3]の問題に提出されたソースコードと，AtCoder が公式に公開している各問題のテストケース[4]から作成した．

対象の問題は 2022 年 1 月 ~2023 年 6 月までの期間に AtCoder において開催された AtCoder Beginner Contest の A 問題と B 問題とし，対象の言語は C++，COBOL，Go，Java，Python3，Rust とする．

対象の全ての問題に対して，対象の各言語で以下の処理を行いソースコードを取得する．

（1） 問題に提出された正解ソースコードを行数で昇順にソートし，上位 25% を除いたソースコードのうち行数最小のものを取得．この処理を行うことで，極端に短いソースコードや極端に冗長なソースコードを除外できる．

（2） 取得したソースコードを実行し，全てのテストケースに通過することを確認．

作成したデータセットのソースコード情報を表 1 に示す．COBOL のみソースコード数が少ないのは AtCoder の対象問題のうち，COBOL の解答が提出されていない問題が存在したためである．

このデータセットに含まれるソースコードを翻訳し，翻訳の精度を調査する．本研究における評価指標は全てテストの合格率を指標とする計算精度 [16] を採用する．

3.2　調　査　1

GPT-4 と最新のニューラル機械翻訳モデルである TransCoder-IR との精度比較を行う．TransCoder-IR は関数単位での翻訳を行うモデルであるため，GPT-4 でも同様に関数単位で翻訳を行う．対象の言語は TransCoder-IR が学習に使用している C++，Go，Java，Rust とする．

3.2.1　データセットの加工

関数単位の翻訳の精度を調査するために，先述したデータセットの C++，Go，Java，Rust のソースコードを加工する．

（1） 各ソースコードにおいて，入力を関数 main で行い，処理と出力を関数 solve で行うように手作業で書き換える．このとき，書き換え後のソースコードは関数 main と関数 solve 以外の関数を含まないようにする．例を図 1 に示す．

（2） 書き換え後のソースコードを実行し，全てのテストケースに通ることを確認する．

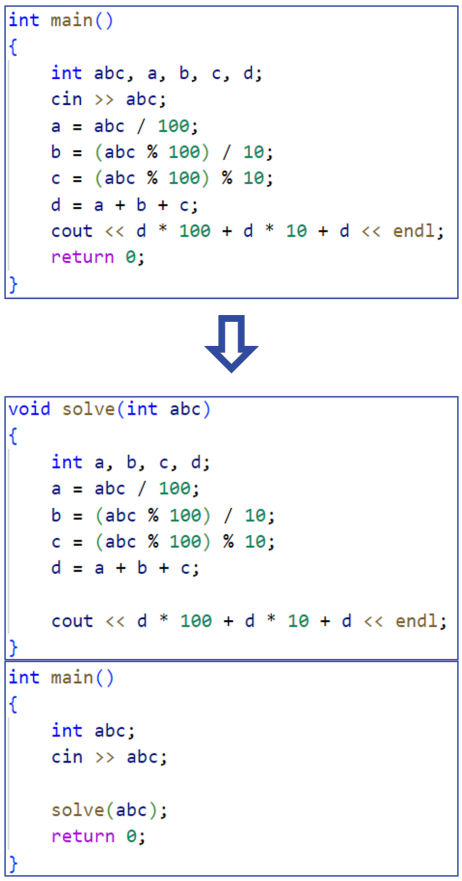

図 1　C++ のソースコードを関数 solve と関数 main に分ける処理の例

表 2　調査 1 で使用するデータセット

言語	ソースコード数	関数 solve の平均行数
C++	140	17.2
Go	140	10.7
Java	140	14.1
Rust	140	11.3

加工後のデータセットのソースコード情報を表 2 に示す．このデータセットに含まれる各ソースコードの関数 solve を翻訳して精度を調査する．

3.2.2　調 査 方 法

データセットの各ソースコードの関数 solve を TransCoder-IR，GPT-4 で翻訳し，翻訳の精度の調査を行う．C++ から Rust に翻訳する場合の実験の流れを例として示す．

（1） データセットに含まれる C++ のソースコードの関数 solve を取得する．

（2） 関数 solve を TransCoder-IR，あるいは GPT-4 を用いて Rust で記述された関数に翻訳する．GPT-4 においては，出力の自然言語部分は削除する．ここで出力された関数を関数 solve’ とする．

（3） 関数 solve’ を呼び出す関数 main とライブラリのインポートを手動で補完する．このとき関数 solve’ は一切変更しない．

（4） テストケースを全て実行し，全てのテストケースに通

(注3)：https://atcoder.jp/
(注4)：https://www.dropbox.com/sh/nx3tnilzqz7df8a/AAAYlTq2tiEHl5hsESw6-yfLa?dl=0

表 3　TransCoder-IR と GPT-4 の精度比較

	C++		Go		Java		Rust		平均	
	TransCoder-IR	GPT-4	TransCoder-IR	GPT-4	TransCoder-IR	GPT-4	TransCoder-IR	GPT-4	TransCoder-IR	GPT-4
C++	-	-	0.45	**0.93**	0.54	**0.93**	0.24	**0.85**	0.41	**0.90**
Go	0.29	**0.89**	-	-	0.33	**0.86**	0.16	**0.84**	0.26	**0.86**
Java	0.41	**0.91**	0.34	**0.94**	-	-	0.19	**0.92**	0.31	**0.92**
Rust	0.07	**0.84**	0.05	**0.88**	0.04	**0.83**	-	-	0.05	**0.85**
平均	0.26	**0.88**	0.28	**0.92**	0.30	**0.87**	0.20	**0.87**	0.26	**0.88**

過すれば翻訳は正しいとし，正しい翻訳が出力される精度を調査する．このとき，実行時間を 10 秒に制限し，これを超えた場合はテストケースに失敗したとみなす．

3.2.3　モデルの設定

TransCoder-IR は配布されているモデル[5]をそのまま使用した．GPT-4 は API を使用し，モデルは gpt-4-0613 を選択した．C++ から Rust への翻訳を行う際のプロンプトを以下に例示する．

> あなたは優秀なソフトウェアエンジニアです．C++ で記述された以下の関数を Rust の関数に翻訳してください．必ず翻訳結果のみを出力してください．
> ```
> {C++ で記述された関数 }
> ```

3.2.4　結果と考察

結果を表 3 に示す．表における縦軸は翻訳元言語，横軸は翻訳先言語を示す．結果から，GPT-4 が全てのペアにおいて TransCoder-IR よりも精度が高いことがわかる．

今回の TransCoder-IR の精度は TransCoder-IR の論文 [17] で述べられている精度よりも全体的に低くなっているが，これは本研究で作成したデータセットが TransCoder-IR の評価に使用されたデータセットよりも翻訳における難易度が高いためであると考えられる．

3.3　調　査　2

C++，COBOL，Go，Java，Rust，Python3 のソースコードを翻訳し，レガシーな言語とモダンな言語間や，静的型付け言語と動的型付け言語間などの異なる特性を持つ言語間で翻訳を行った場合の精度の傾向の調査を行う．また，同時にコード単位での翻訳の精度の調査を行う．

3.3.1　調 査 方 法

先述したデータセットに含まれるソースコードを翻訳し，精度を調査する．C++ から Rust に翻訳する場合の実験の流れを以下に例示する．

（1） データセットに含まれる C++ のソースコードを GPT-4 を用いて Rust で記述されたソースコードに翻訳．このとき，出力の自然言語部分は削除する．

（2） 各問題のテストケースを実行し，全てのテストケースに通過すれば翻訳は正しいとし，正しい翻訳が出力される精度を調査する．このとき，実行時間を 10 秒に制限し，これを超えた場合はテストケースに失敗したとみなす．

表 4　GPT-4 を用いて 6 言語間で翻訳した際の精度

GPT-4	C++	COBOL	Go	Java	Python3	Rust	平均
C++	-	0.01	0.70	0.90	0.63	0.63	0.59
COBOL	0.53	-	0.29	0.65	0.58	0.54	0.52
Go	0.90	0.01	-	0.94	0.67	0.58	0.62
Java	0.91	0.01	0.60	-	0.56	0.64	0.54
Python3	0.84	0.02	0.60	0.87	-	0.66	0.60
Rust	0.86	0.00	0.62	0.90	0.74	-	0.62
平均	0.81	0.01	0.56	0.85	0.65	0.61	0.58

3.3.2　モデルの設定

GPT-4 は API を使用し，モデルは gpt-4-0613 を選択した．C++ から Rust への翻訳を行う際のプロンプトを以下に例示する．

> あなたは優秀なソフトウェアエンジニアです．C++ で記述された以下のソースコードを Rust のソースコードに翻訳してください．必ず翻訳結果のみを出力してください．
> ```
> {C++ で記述されたソースコード }
> ```

3.3.3　結果と考察

実験結果を表 4 に示す．表における縦軸は翻訳元言語，横軸は翻訳先言語を示す．表 4 から，翻訳元言語よりも翻訳先言語の方が精度に影響することがわかる．特に，C++，Java が翻訳先言語である場合は比較的精度が高いのに対して，COBOL が翻訳先言語である場合は精度が低いことがわかる．これは GPT-4 が COBOL のようなレガシーな言語を生成する能力が他言語と比べて低いためであると考えられる．

また関連研究 [7] において，一般的な事前学習言語モデルでは動的型付け言語から静的型付け言語への翻訳の精度が低下すると報告されているが，GPT-4 においては精度の大幅な低下は見られなかった．

3.4　調　査　3

コード翻訳タスクにおけるプロンプトに用いる言語の影響を調査するために，プロンプトの言語が英語の場合と日本語の場合の翻訳の精度を比較する．

3.4.1　調 査 方 法

プロンプトの言語を英語に変更して実験を行い，調査 2 の結果と比較する．GPT-4 に与えるプロンプト以外の条件は調査 2 と同じとする．C++ から Rust の翻訳を行う際のプロンプトを

(注5)：https://github.com/facebookresearch/CodeGen/blob/main/docs/TransCoder-IR.md

表 5　プロンプトを英語に変更した場合の精度

GPT-4	C++	COBOL	Go	Java	Python3	Rust	平均
C++	-	0.01 (±0.00)	0.66 (-0.04)	0.91 (+0.01)	0.67 (-0.04)	0.58 (-0.05)	0.57 (-0.02)
COBOL	0.51 (-0.02)	-	0.17 (-0.12)	0.58 (-0.07)	0.54 (-0.04)	0.46 (-0.07)	0.45 (-0.06)
Go	0.91 (+0.01)	0.03 (+0.02)	-	0.91 (-0.03)	0.63 (-0.04)	0.56 (-0.02)	0.61 (-0.01)
Java	0.92 (+0.01)	0.01 (±0.00)	0.58 (-0.02)	-	0.50 (-0.06)	0.63 (-0.01)	0.53 (-0.01)
Python3	0.79 (-0.05)	0.00 (-0.02)	0.39 (-0.21)	0.84 (-0.03)	-	0.69 (+0.03)	0.54 (-0.06)
Rust	0.84 (-0.02)	0.01 (+0.01)	0.59 (-0.03)	0.89 (-0.01)	0.57 (-0.17)	-	0.58 (-0.04)
平均	0.79 (-0.02)	0.01 (±0.00)	0.49 (-0.07)	0.83 (-0.02)	0.58 (-0.07)	0.58 (-0.03)	0.55 (-0.03)

以下に例示する．

> You are an excellent software developer. Translate the following source code, written in C++, into Rust source code. Make sure that you only output the results of the translation.
> ```
> {C++ で記述されたソースコード }
> ```

3.4.2　結果と考察

プロンプトを英語に変更して実験した結果を表 5 に示す．括弧内の数値はプロンプトが日本語の場合の結果との差分を示し，精度が上がった場合は背景色を緑，精度が下がった場合は背景色を赤，精度が変化しなかった場合は背景色を灰としている．結果から，プロンプトが英語の場合と日本語の場合で精度に大きな差はないことが確認できた．つまり，コード翻訳タスクに関しては英語が不得手な日本語話者が無理にプロンプトを英語に翻訳する必要がないことを示す．

プロンプトが英語の場合と日本語の場合でのどちらでも生成されるコードの安全性は変化しないという報告 [21] と本実験の結果を加味すると，コード生成に関するタスクにおいてはプロンプトの言語は大きく影響しないということが考えられる．

3.5　調　査　4

調査 4 では調査 2 の結果に静的解析ツールを適用した場合の精度の向上について調査する．

3.5.1　調 査 方 法

調査 2 の結果に以下の静的解析ツールを適用した場合の精度の変化を調査する．

- goimports: Go のパッケージの import を自動で解決するツール
- cargo-fix: Rust のコンパイラが提案した修正を自動で適用するツール

3.5.2　結果と考察

調査 2 で出力された Go コードと Rust コードにそれぞれ goimports と cargo-fix を適用した場合の精度を表 6 に示す．左の表は goimports を適用した場合の精度，右の表は cargo-fix を適用した場合の精度である．また，括弧内の数値はプロンプトが日本語の場合の結果との差分を示し，精度が上がった場合は背景色を緑，精度が下がった場合は背景色を赤，精度が変化しなかった場合は背景色を灰としている．結果から，静的解析ツールを利用することで精度が上がることがわかる．特に，

表 6　調査 2 の結果に静的解析ツールを適用した場合の精度

GPT-4 (goimports)	Go
C++	0.82 (+0.12)
COBOL	0.32 (+0.03)
Java	0.70 (+0.10)
Python3	0.68 (+0.08)
Rust	0.71 (+0.09)
平均	0.65 (+0.09)

GPT-4 (cargo-fix)	Rust
C++	0.65 (+0.01)
Go	0.58 (±0.00)
COBOL	0.54 (±0.00)
Java	0.66 (+0.02)
Python3	0.66 (+0.01)
平均	0.62 (+0.01)

goimports を適用した場合平均 +0.09 となり，cargo-fix の +0.01 と比べてより向上している．

この結果は，GPT-4 などのコード生成可能な LLM に静的解析ツールを組み込むことで，さらに高精度にコード生成を行うことができるようになることを示す．

4.　妥当性への脅威

4.1　内的妥当性

データセットに含まれる AtCoder のテストケースの網羅性が低いと実験の信頼性が失われるが，本研究におけるデータセットにはコンテスト中に発覚したテストケースの不備の対策としてコンテスト後に追加されたテストケースも含んでいるため，テストケースの不備による妥当性の脅威は小さいと考えられる．

4.2　外的妥当性

4.2.1　再現性の問題

GPT-4 は同一プロンプトであっても複数回実行することで出力が変化する．そのため，同じモデルとプロンプトであっても本研究の結果を完全に再現することは不可能である．また，GPT-4 は定期的に新しいモデルが発表され，古いモデルは非推奨になる．今後非推奨のモデルが非公開になる可能性も考えられ，その場合には本研究で使用したモデルを用いた実験を行うことは不可能になる．

4.2.2　通常のプロジェクトにおける精度

データセットのドメインが競技プログラミングに限られているため，通常のプロジェクトにおけるソースコードに適用する場合の精度は，本研究で報告した精度よりも低くなる可能性が考えられる．

5.　まとめと今後の展望

本研究では GPT-4 が事前学習していないデータからデータセットを新たに作成して GPT-4 のコード翻訳タスクの精度の

調査を行った．具体的には TransCoder-IR と GPT-4 の精度比較 (調査 1)，異なる特徴の言語間における翻訳の精度の調査 (調査 2)，プロンプトに使用する言語が翻訳の精度に与える影響の調査 (調査 3)，外部ツールの利用が翻訳の精度に与える影響の調査 (調査 4) を行った．

調査 1 では GPT-4 が TransCoder-IR よりも高精度でコード翻訳を行えることがわかった．調査 2 ではコード翻訳においては翻訳元言語よりも翻訳先言語の影響が大きいことを明らかにし，特に翻訳先言語が COBOL である場合の精度が低いことがわかった．調査 3 ではプロンプトの言語が英語の場合も日本語の場合も精度に大きな差がないことがわかった．調査 4 では goimports や cargo-fix などの静的解析ツールを使用することで，エラーの一部を解決できることがわかった．

今後は，Fine-tuning を実施した場合の精度の調査や，プロンプトに Chain-of-Thought [18] を導入した場合の精度の調査を行う予定である．

謝辞　本研究は JSPS 科研費 JP21K18302 の助成を受けたものです。

文　献

[1] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D.S. Seljebotn, and K. Smith, "Cython: The Best of Both Worlds," Computing in Science & Engineering, vol.13, no.2, pp.31-39, Mar. 2011, DOI:10.1109/MCSE.2010.118.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y.T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M.T. Ribeiro, and Y. Zhang, "Sparks of artificial general intelligence: Early experiments with gpt-4," eprint arXiv:2303.12712, Mar. 2023. DOI:10.48550/arXiv.2303.12712.

[3] X. Chen, C. Liu, D. Song, "Tree-to-Tree Neural Networks for Program Translation," Neural Information Processing Systems, Montréal, Canada, pp.2552–2562, Dec. 2023

[4] J. Etxaniz, G. Azkune, A. Soroa, O.L. de Lacalle, and M. Artetxe, "Do Multilingual Language Models Think Better in English?," eprint arXiv:2308.01223, Aug. 2023. DOI:10.48550/arXiv.2308.01223.

[5] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring Massive Multitask Language Understanding," The International Conference on Learning Representations, Virtual Event, https://openreview.net/forum?id=d7KBjmI3GmQ, May 2021.

[6] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song, and J. Steinhardt, "Aligning AI With Shared Human Values," The International Conference on Learning Representations, Virtual Event, https://openreview.net/forum?id=dNy_RKzJacY, May 2021.

[7] M. Jiao, T. Yu, Xuan Li, G.J. Qiu, X. Gu, and B. Shen. "On the Evaluation of Neural Code Translation: Taxonomy and Benchmark," The International Conference on Software Engineering, pp.1597–1608, Pittsburgh, USA, May 2022. DOI:10.1145/3510003.3510060.

[8] W. Jiao, W. Wang, J. Huang, X. Wang, S. Shi, and Z. Tu, "Is ChatGPT A Good Translator? A Preliminary Study," eprint arXiv:2301.08745, Jan. 2023. DOI:10.48550/arXiv.2301.08745.

[9] V.D. Lai, N.T. Ngo, A.P.B. Veyseh, H. Man, F. Dernoncourt, T. Bui, and T.H. Nguyen, "Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning," eprint arXiv:2304.05613, Apr. 2023. DOI:10.48550/arXiv.2304.05613.

[10] R. Li, L.B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T.Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M. Yee, L.K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. Sankalp Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C.J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C.M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and Harm de Vries, "StarCoder: may the source be with you!," eprint arXiv:2305.06161, May 2023. DOI:10.48550/arXiv.2305.06161.

[11] V. Liu and L. B. Chilton, "Design guidelines for prompt engineering text-to-image generative models," CHI2022, New Orleans, USA, no.384, pp.1–23, Apr. 2022. DOI:10.1145/3491102.3501825.

[12] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C.B. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tufano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S.K. Deng, S. Fu, and S. Liu, "CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation," eprint arXiv:2102.04664, Feb. 2021. DOI:10.48550/arXiv.2102.04664.

[13] OpenAI, "GPT-4 Technical Report," eprint arXiv:2303.08774, Mar. 2023. DOI:10.48550/arXiv.2303.08774.

[14] OpenAI. "Models". https://platform.openai.com/docs/models/gpt-3-5, Dec. 2023.

[15] OSS Consortium, "opensource COBOL ダウンロード", https://www.osscons.jp/osscobol/download/, Dec, 2023.

[16] B. Roziere, M. Lachaux, L. Chanussot, and G. Lample, "Unsupervised Translation of Programming Languages," The International Conference on Neural Information Processing Systems, no.1730, pp20601-20611, Virtual Event, Dec. 2020.

[17] M. Szafraniec, B. Roziere, H. Leather, F. Charton, P. Labatut, and G. Synnaeve, "Code translation with Compiler Representations," The International Conference on Learning Representations, https://openreview.net/forum?id=XomEU3eNeSQ, May 2023.

[18] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," eprint arXiv:2201.11903, Jan. 2022. DOI: 10.48550/arXiv.2201.11903;

[19] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," eprint arXiv:2206.07682, June 2022. DOI: 10.48550/arXiv.2206.07682.

[20] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D.C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," eprint arXiv:2302.11382, Feb. 2023. DOI:10.48550/arXiv.2302.11382.

[21] 山岸伶，笹晋也，藤井翔太，"入力時の日本語と英語の差異が ChatGPT で生成するコードの安全性に与える影響の考察，" コンピュータセキュリティシンポジウム 2023 論文集，pp.1544-1551, Oct. 2023.

[22] W.X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, and J. Wen, "A Survey of Large Language Models," eprint arXiv:2303.18223, Mar. 2023. DOI:10.48550/arXiv.2303.18223.