GitHub Copilotを用いたコード推薦における入力言語の影響調査

小柳 慶　野口 広太郎　近藤 将成　亀井 靖高　鵜林 尚靖

近年，IT 需要の拡大に伴って，開発効率の向上のために開発支援ツールを活用して開発が行われている．その中の一つとして，2022 年に GitHub が公開した GitHub Copilot がある．GitHub Copilot は大規模言語モデルをベースとしたコード推薦ツールの一種であり，仕様を記述したコメントや，記述中のプログラムをもとに開発者に対してコードやライブラリを推薦する．一方，大規模な事前学習済み言語モデルは，入力によって出力が大きく異なることが知られている．そこで，本稿では，言語間のニュアンスの違いに着目し，入力言語の違いが Copilot の性能にどのような影響を与えるのか調査を行った．調査の結果，入力言語の違いによって GitHub Copilot の性能に差が生じることが明らかになった．

I will write English abstract here after the review of Japanese abstract.

1　はじめに

近年，IT 需要の拡大に伴って，開発効率の向上のためにタスク管理ツールやプロジェクト管理ツールをはじめ，様々な支援ツールを活用して開発が行われている．その中の一つとして，2022 年に GitHub が公開した GitHub Copilot がある（以下，Copilot と記述する）．Copilot は大規模言語モデルをベースとしたコード推薦ツールの一種であり，使用を記述したコメントや，記述中のプログラムをもとに開発者に対してコードやライブラリを推薦する．そのため，フルスクラッチする必要がなく，開発コストの削減が見込まれる．

一方，大規模な事前学習済み言語モデルは，入力によって出力が大きく異なることが知られている [2]．そのため，言語モデルの能力を最大限引き出すためには，適切なコメントを入力する必要がある．

また，現在世界では 7000 語以上の言語が存在すると言われているが，言語によって使用頻度は異なる．そのため，コメントに異なる言語を使用することで，学習データ数の不均衡などにより学習結果のバイアスにつながる可能性があると考えた．そこで本研究では，言語間のニュアンスの違いに着目し，言語の違いが Copilot の性能にどのような影響を与えるのか調査を行った．

以降，第 2 章で関連研究および本研究の目的，第 3 章で本研究の実験設計について述べる．第 4 章で調査結果を示し，第 5 章で考察を行う．第 6 章では，妥当性の脅威を説明し，最後に第 7 章で結論と今後の課題について述べる．

2　背景と目的

2.1　本研究の目的

Copilot に対して研究が盛んに行われているが，入力言語の違いによる性能への影響については調査が行われていない．入力言語の違いによる性能への影響を調査することで，今後ますます事前学習済み大規模言語モデルが組み込まれたシステムが拡大していくにあたって，システムの性能を最大限引き出すための手がかりを得ることができると考えた．本稿では，日本語，英語，および中国語の 3 言語を入力とした場合の

Investigation of the effect of input languages on code recommendation using GitHub Copilot
Kei Koyanagi, 九州大学, Kyushu University.
Kotaro Noguchi, Masanari Kondo, Yasutaka Kamei, Naoyasu Ubayashi, 九州大学, Kyushu University.

Copilot の性能を比較する．調査課題を以下に示す．

RQ 入力する言語の違いによって，**Copilot** の性能にどのような影響を与えるのか

目的 近年，大規模な事前学習済み言語モデルを用いたシステムが拡大しており，今後も拡大していくことが予想される．しかし，現状の大規模な事前学習済み言語モデルは，入力によって出力が大きく異なり，主な要素としては入力内容および入力言語がある．本研究では，後者の言語に着目し，入力言語によって Copilot の性能に差が生じるのか明らかにすることで，今後の Copilot の最適な活用についての知見を得る．

2.2 プロンプトエンジニアリング

コード生成におけるプロントエンジニアリングとは，モデルに対してどのようなプロンプトを入力として与えれば生成精度がより向上するかを探索する手法である．プロンプトとはモデルに対して与える入力のことで，モデルに対して与える入力としてはコードの仕様やプログラムそのものなどがある．モデルは与えられたプロンプトをもとに，次にどのようなをコードを生成するかを予測する．

2.3 関連研究

Yao らは，プロンプトとして入力するサンプルの入力順序の違いによって，GPT-3 のような大規模な事前学習済み言語モデルの性能にどのような影響を与えるのか調査を行った [2]．その結果，サンプルの入力順序によって性能にばらつきが生じ，これがモデルサイズに関係なく発生すること，サンプルの特性のセットに関係なく発生すること，およびあるモデルにはよい学習順序であっても別のモデルには適用できないことを示した．また，各モデルに対してよい性能を示すプロンプトの探索手法としてエントロピーベースでの探索手法を提案した．その結果，エントロピーベースでの探索手法は，ランダムにプロンプトを選択するよりもよい性能を示した．

Nguyen らは，Copilot のコード推薦の精度および推薦されたコードの品質について, 異なるプログラミング言語で調査を行った [3]．LeetCode の問題に対して Easy, Medium, Hard の 3 つの難易度で調査を行ったところ，Easy では全ての言語が全テストケースを通過し，全体として，Java, Python, C, Javascript の順に高い精度を示した．また，推薦されたコードの品質については，言語間における差はほとんど生じず，判読性は高いことを示した．

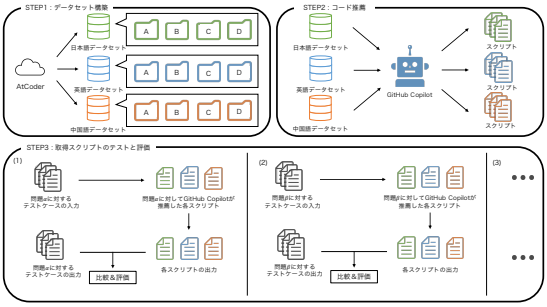

図 1: 実験設計の概要

3 実験設計

本章では，本研究で行った実験の概要について説明する．

3.1 データセット構築

本研究では日本国内において最大の競技プログラミングコンテストである AtCoder [1] の問題を使用する．中でも，AtCoder Beginner Contest という毎週開催されているコンテストの問題を使用する．このコンテストは 6/24 までに 1~307(2023/06/24 現在) の問題が公開されており，各問題の難易度は A, B, C, D, E, F, G, Ex(H) までの最大 8 段階でアルファベットの語順が後になるほど難易度が高くなる．また，問題は日本語版と英語版が準備されている．この問題から，問題番号 99~287，各問題の難易度 A~D の日本語版および英語版を使用する．問題を限定する理由としては，問題番号は十分なテストケースの数が確保されている問題を使用するため，問題難易度は事前実験にて Copilot が正しいプログラムを生成できる境目であったためである．続いて，中国語版の問題は AtCoder には存在しないため，英語版のデータセットを DeepL API を使用して翻訳し，翻訳した問題をネイティブによる校正を行うことで，中国語版

表 1: 問題 102-A の英語版

```
#Problem Statement
#You are given a positive integer N.
#Find the minimum positive integer
#divisible by both 2 and N.
#Constraints
#1 ≦ N ≦ 10^9
#All values in input are integers.
#Input
#Input is given from Standard Input
#in the following format:
#N
#Output
#Print the minimum positive integer
#divisible by both 2 and N.
#Sample Input 1
#3
#Sample Output 1
#6
```

のデータセットを作成した．最終的に，問題番号が 99∼287，各問題の難易度が A,B,C,D，問題の言語が英語，日本語，中国語の三つのデータセットが取得でき，これらを本実験のデータセットとした．

3.2 プログラム推薦

3.1 節で作成したデータセットに対して，Copilot でプログラム推薦を行う．3.1 節で作成した各言語のデータセットの中には，問題番号が 99∼287，各問題の難易度が A,B,C,D 存在するため，合計で 2,268 問存在する．これらの問題に対して，プログラム推薦を行う．

表 1 は問題 102-A の英語版のデータセットの中身である．このように各問題のデータセットには，問題，制約，入力，出力，サンプル入力，サンプル出力がコメントとして記述されている．ただし，問題によってサンプル入力およびサンプル出力の数は異なり，また注釈等の記述がある場合もある．このデータセットを入力として，Copilot によるプログラム推薦を行い，x 個の推薦プログラムを得る．このとき，x は $0 \leq x \leq 10$ を満たす．また，推薦プログラムで出力するプログラミング言語は Python を使用した．この入力から出力までの流れを各問題に対して 5 回実行する．その理由としては，Copilot の推薦スクリプトが毎回変化するため，このランダム性を考慮して，評価を行うためである．

表 2: 英語における *Accuracy*

	A		B		C		D	
1	63.7	864/1357	51.0	860/1686	28.3	482/1702	10.4	172/1657
2	65.4	878/1343	50.5	858/1698	26.7	471/1761	10.0	182/1813
3	63.6	856/1346	50.9	811/1594	26.6	438/1648	9.31	151/1622
4	60.4	819/1355	51.6	842/1631	31.3	499/1596	11.0	167/1524
5	60.4	831/1375	50.5	857/1698	31.1	552/1774	12.0	217/1812

表 3: 日本語における *Accuracy*

	A		B		C		D	
1	68.7	857/1248	52.8	872/1651	28.9	482/1668	10.0	170/1697
2	67.4	840/1246	52.9	824/1559	28.4	454/1597	10.2	155/1526
3	68.1	842/1237	54.3	879/1619	28.5	449/1577	11.0	153/1391
4	68.4	860/1258	57.6	872/1515	33.1	439/1328	12.8	125/974
5	68.0	849/1249	54.5	922/1693	32.7	592/1813	11.4	208/1832

表 4: 中国語における *Accuracy*

	A		B		C		D	
1	52.8	758/1435	41.2	724/1756	23.2	415/1792	8.11	144/1775
2	52.5	737/1403	44.0	749/1704	22.5	388/1724	8.00	138/1727
3	52.1	747/1433	40.4	701/1734	23.4	420/1793	8.44	150/1777
4	52.1	743/1426	42.3	736/1738	23.9	427/1789	8.99	158/1758
5	52.2	756/1447	43.1	751/1744	22.6	406/1800	7.84	137/1748

3.3 取得プログラムのテストと評価

3.2 節で取得したプログラムに対して，テストを行う．このテストは，各問題に対して，テストケースの入力を入力として取得したプログラムを実行し，テストケースの出力と一致するかを確認する．そして，各推薦スクリプトが全てのテストケースを通過したか否かで評価を行う．使用した評価指標は，*Accuracy*(正答率) で，推薦された全スクリプトの内，全てのテストケースを通過したスクリプトの割合を示す．

4 結果

本章では，RQ と結果を示す．

RQ 入力する言語の違いによって，Copilot の性能にどのような影響を与えるのか

表 5: 各言語の *Accuracy* の中央値

	A	B	C	D
英語	63.6%	50.9%	28.3%	10.4%
日本語	68.1%	54.3%	28.9%	11.0%
中国語	52.2%	42.3%	23.2%	8.11%

結果 表 2, 表 3, および表 4 はそれぞれ英語，日本語，および中国語における *Accuracy* を示したものである．これらの表から，生成毎に *Accuracy* にばらつきが生じているため，Copilot が毎回異なるスクリプトを推薦していることがわかる．また，推薦回数を重ねるほど *Accuracy* の値が単調に増加したり，減少したりすることはなかった．表 2 より，英語のデータセットにおいて，A 問題では最大 5.0%，B 問題では最大 1.1%，C 問題では最大 4.7%，D 問題では最大 2.7%の差が生じた．また，表 3 より，日本語のデータセットにおいて，A 問題では最大 1.3%，B 問題では最大 4.8%，C 問題では最大 4.7%，D 問題では最大 2.8%，表 4 より，中国語のデータセットにおいては，A 問題で最大 0.70%，B 問題で最大 3.6%，C 問題で最大 1.4%，D 問題で最大 1.2%の差が生じた．各言語の差が 3.0%以上だった問題の最大値に着目すると，推薦された全スクリプト数が 5 回のうち最も少なく，このことから，推薦スクリプトを最大 10 個取得することで，かえって *Accuracy* が低下する可能性があることが考えられる．また，難易度別に推薦された全スクリプト数に着目すると，A 問題は B, C, D 問題に比べて，推薦された全スクリプト数が最も少なかった．これは問題が単純であるが故に，Copilot がスクリプトを推薦する際に，生成されるスクリプトの多様性が低くなるためであると考えられる．さらに，言語別で推薦された全スクリプト数を平均すると，英語のデータセットでは，A 問題が 1355 問，B 問題が 1661 問，C 問題が 1696 問，D 問題が 1686 問，日本語のデータセットでは，A 問題が 1247 問，B 問題が 1607 問，C 問題が 1596 問，D 問題が 1484 問，中国語のデータセットでは，A 問題が 1428 問，B 問題が 1735 問，C 問題が 1779 問，D 問題が 1757 問であった．これらの結果から，日本語のデータセットでは，英語のデータセットと中国語のデータセットに比べて，推薦された全スクリプト数が少なかった．加えて，全難易度で日本語のデータセットの *Accuracy* が最も高かった．これは，日本語での入力により，Copilot がより最適なスクリプトのみを推薦していることを示している．この理由として，AtCoder [1] が日本で運営されているため，より多くの日本人が AtCoder を使用しており，日本語のコメントを含んだ回答が学習時に多く使用されたためであると考えられる．また，表 5 は，各言語の難易度別の *Accuracy* の中央値を示したものである．表 5 より，中国語のデータセットは，英語のデータセットに比べて，日本語のデータセットとの *Accuracy* の差が大きかった．これは，AtCoder [1] において，日本語と英語の問題が準備されているため，日本語や英語をコメントの含んだより多くの正解プログラムが学習時に使用されたためであると考えられる．

5 考察

4 節で示した結果について，考察を行う．表 5 より，中国語のデータセットは，英語のデータセットに比べて，日本語のデータセットとの *Accuracy* の差が大きかった．中でも，A 問題に関しては，日本語との差が 15.9%，英語との差が 11.4%であった．今回使用したデータセットの難易度の中で最も簡単な問題であるため，*Accuracy* の値に大きな差が生じにくいと予測していたが，予測とは異なる結果となった．その理由として，AtCoder [1] において，日本語と英語の問題が準備されているため，簡単な問題であっても，これらの言語をコメントに含んだより多くの正解プログラムが GitHub 上にアプロードされており，それらが学習に使用された可能性がある．

6 妥当性への脅威

内的妥当性． 本研究では，各プロジェクトの ‘requirements.txt’ からライブラリ使用情報を取得して調査を行った．ただしファイル名は任意に設定可能な

ため，別のファイル名を使用している，あるいは，‘requiments.txt’ ファイルを設定していない場合がある．そのため，対象にすべきファイルを取得できていない可能性を考慮し，実際のプログラムからも情報を抽出して実験を行う必要がある．

外的妥当性．本研究では，Python の機械学習ライブラリの使用率上位 12 件のライブラリを最低 1 つ使用しているプロジェクトのうち，GitHub でのスター数各上位 100 件までを対象に調査を行ったが，調査結果をより一般化するためには，依存ライブラリ数，および対象のプロジェクト数を増加させて調査を行う必要がある．

7 おわりに

本稿では協調フィルタリングを用いて Python における機械学習関連ライブラリを推薦し，5 つの評価指標を用いてシステムの評価を行った．調査の結果，各評価指標値の推移は対象が Java の場合と同様の傾向を示したが，先行研究に比べて精度は同等またはそれ以下であった．今後の課題として，データセットの変更や拡張を行った調査，および推薦システムの構造変更が必要である．

謝辞

本研究の一部は JSPS 科研費 JP18H04097, JP20H04167, JP21H04877, JP22K17874, JP22K18630 の助成を受けた．

参考文献

[1] AtCoder: `https://atcoder.jp/contests/archive?ratedType=1&category=0&keyword=` [Accessed: 2023-7-8].

[2] Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P.: Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, May 2022, pp. 8086–8098.

[3] Nguyen, N. and Nadi, S.: An Empirical Evaluation of GitHub Copilot's Code Suggestions, *Proceedings of the IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, 2022, pp. 1–5.