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，開発効率向上のため，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．また，調査結果によって明らかとなった大規模言語モデルに対する問題点を示す．

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，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．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 [4]．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．

現在世界では 7000 語以上の言語が存在する [3] 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，言語間の学習データ量の違いに着目し，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 章で考察および拡大質問を示す．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

急速に進む IT 化に伴い，事前学習済み大規模言語モデルの活用が進んでおり，今後ますます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ことが予想される．その中の一つである Copilot に対しても研究が盛んに行われている [4] [5] [6] [2] [7]．一方で，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 [4]．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．コメントは主要な要素として入力言語と入力内容で構成される

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.

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．そこで，入力言語の違いによる性能への影響を調査することで，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それぞれの 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　現状，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 [4] 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．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，サンプルの特定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優れた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，各モデルに対して優れた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優れた性能を示した．

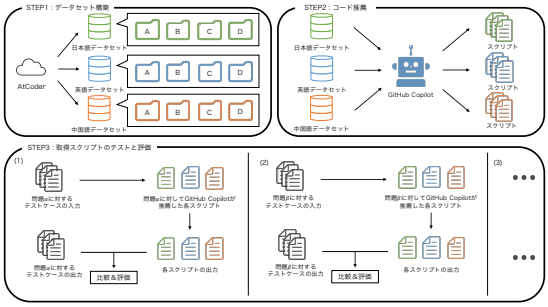

図 1: 実験設計の概要

Nguyen [5] 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．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．

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．このコンテストは問題番号 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．この問題の中から，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，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，翻訳した問題をソフトウェア工学における研究歴が約 6 年ある中国人研究者によって校正することで，中国語版のデータセットを作成する．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 3 つのデータセットを取得し，これらを本実験のデータセットとする．

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 756 問存在する．これらの各問題に対して，それぞれプログラム推薦を行う．

図 2 は問題 212-A の日本語版のデータセットの中身である [1]．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，入力例，出力例がコメントとして記述されている．ただし，問題によって入力例および出力例の数は異なり，また注釈等の追加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 個の推薦スクリプトを得る．このとき，x は $0 \leq x \leq 10$ を満たす．また，推薦スクリプトで出力するプログラミング言語は Python を使用する．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．

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標準入力として与え，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．ただし，Copilot が推薦したスクリプトの順番は考慮せずに評価を行う．

[問題文]
高橋くんは A グラムの純金と B グラムの純銀 $(0 \leq A, B, 0 < A + B)$ をよく溶かした上で
混ぜ合わせ、新たな金属を生成しました。
生成された金属は「純金」「純銀」「合金」の
いずれでしょうか？
なお、生成された金属は
$0 < A$ かつ $B = 0$ なら「純金」
$A = 0$ かつ $0 < B$ なら「純銀」
$0 < A$ かつ $0 < B$ なら「合金」
であるとみなします。
[制約]
$0 \leq A, B \leq 100$
$0 < A + B$
A, B は整数
[入力]
入力は以下の形式で標準入力から与えられる。
A B
[出力]
生成された金属が「純金」なら Gold と、
「純銀」なら Silver と、
「合金」なら Alloy と出力せよ。
[入力例 1]
50 50
[出力例 1]
Alloy
[入力例 2]
100 0
[出力例 2]
Gold

図 2: 問題 212-A の日本語版

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 1: 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 2: 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 3: 中国語における *Accuracy*

	A		B		C		D	
1	52.7	739/1402	40.1	705/1757	22.7	405/1787	7.70	136/1766
2	52.5	728/1386	43.1	734/1703	21.9	376/1718	7.81	134/1715
3	52.1	737/1415	39.7	687/1730	22.7	405/1788	8.11	143/1764
4	51.8	732/1412	41.5	717/1728	23.2	413/1783	8.35	146/1748
5	51.5	739/1436	41.9	728/1738	22.0	395/1798	7.57	132/1744

表 4: 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	52.1%	41.5%	22.7%	7.81%

結果：表 1，表 2，および表 3 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 1 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 2 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 3 より，中国語のデータセットにおいては，A 問題で最大 1.2%，B 問題で最大 3.4%，C 問題で最大 1.3%，D 問題で最大 0.78%の差が生じた．

また，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 1410 問，B 問題が 1731 問，C 問題が 1774 問，D 問題が 1747 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．さらに，表 4 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 4 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．

5 考察と拡大質問

4 章で示した結果について，考察および拡大質問を示す．表 1, 2, 3 のうち，各言語の *Accuracy* 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の平均と，日本語のデータセットの *Accuracy* が最も高かったことより，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がわかる．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 GitHub 上にアップロードされ，それら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，英語や中国語は日本

```
=======
Suggestion 1

def alloy(a,b):
    if a == 0:
        return 'Silver'
    elif b == 0:
        return 'Gold'
    else:
        return 'Alloy'

=======
Suggestion 2

def problem212_a():
    a, b = map(int, input().split())
    if a > 0 and b == 0:
        print("Gold")
    elif a == 0 and b >

=======
Suggestion 3

def main():
    a,b = map(int, input().split())
    if a > 0 and b == 0:
        print("Gold")
    elif a == 0 and b > 0:
```

図 3: 問題 212-A の英語版に対する推薦スクリプトの一例

語より多く使用されているため，より多くのスクリプトから学習を行ったことで，かえって *Accuracy* が低下した可能性がある．

表 4 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，日本語や英語のコメントを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．さらに中国語の生成スクリプトが最も多かった原因としては，AtCoder の中国語版が存在しないため，GitHub 上に中国語のコメントを含んだ正解プログラムが少なく，推薦プログラムが絞りきれず，多くのスクリプトが推薦された可能性がある．

また，A 問題に関しては特に大きな差が生じ，日本語との差が 16.0%，英語との差が 11.5%であった．A 問題は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．

その他の原因探索のため，実際に *Accuracy* の差が大きかった問題をいくつか確認した．図 2 の問題は，英語および日本語のデータセットでは，全ての推薦スクリプトが全てのテストケースを通過しているが，中国語のデータセットでは，全ての推薦スクリプトが全てのテストケースを通過していない例である．また，図**??**は実際に問題 212-A における中国語のデータセットに対する推薦スクリプトである．図**??**より，単純な条件のみで構成されているものや，条件分岐の途中で推薦が打ち切られているもの，条件分岐の数が少ないものであった．

この問題のように，条件が複数ある場合や，条件が複雑である場合，出力が文字列である場合に特に英語および日本語との *Accuracy* の差が大きくなる傾向があった．この原因として，AtCoder の文字列の出力形式がローマ字や英単語であるため，入力として使用した中国語のデータセットの文章中にローマ字や英単語が含まれており，それらがシンボルとして認識されなかった可能性や翻訳してデータセットを作成した影響が考えられる．

大規模言語モデルはブラックボックスであり，今後も調査を行うことが必要である．

6 妥当性への脅威

内的妥当性．本研究では，中国語のデータセット作成の際に，DeepL を使用して英語のデータセットを翻訳することで中国語のデータセットを作成した．また，作成後のスクリプトに対して 1 人のネイティブのみによる校正を行った．そのため，翻訳の精度や校正者の主観による影響が考えられる．

外的妥当性．本研究では，AtCoder [1] の問題番号 99〜287，各問題の難易度 A, B, C, D，各問題の言語を日本語，英語，中国語を対象として調査を行ったが，調査結果をより一般化するためには，英語圏で行われているプログラミングコンテストの問題を日本語および中国語に翻訳，また中国語圏で行われているプログラミングコンテストの問題を英語および日本語に翻訳したデータセットを使用して調査を行う必要がある．

7 おわりに

本稿では日本語，英語，中国語のデータセットを使用して，GitHub Copilot によるプログラム推薦を行い，その正答率を比較した．調査の結果，日本語，英語，中国語の順に正答率が高く，英語と中国語には A 問題において約 11.5%の差が生じた．今後の課題として，データセットを変更した調査，推薦プログラムの品質を評価する調査，および Copilot が推薦するプログラムの順番に応じて重みを付与した調査が必要である．

謝辞

本研究の一部は JSPS 科研費 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] Dakhel, A. M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Ming, Z., and Jiang: GitHub Copilot AI pair programmer: Asset or Liability?, *arXiv e-prints*, (2022).

[3] Ethnologue: `https://www.ethnologue.com/browse/names/` [Accessed: 2023-7-8].

[4] 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.

[5] 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.

[6] Sobania, D., Briesch, M., and Rothlauf, F.: Choose Your Programming Copilot: A Comparison of the Program Synthesis Performance of Github Copilot and Genetic Programming, *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, NY, USA, Association for Computing Machinery, 2022, pp. 1019–1027.

[7] Vaithilingam, P., Zhang, T., and Glassman, E. L.: Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models, *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, Association for Computing Machinery, 2022.