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　王 棟　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，言語間の学習データ量の違いに着目し，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．その中の一つに，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の必要がなく，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 の性能に与える影響に関して調査を行った．

以降，第 2 章で関連研究および本研究の目的，第 3 章で実験設計について述べる．第 4 章で調査結果と追加実験の必要性を示し，第 5 章で考察を示す．最後に第 6 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

Investigation of the effect of input languages on code recommendation using GitHub Copilot

Kei Koyanagi, 九州大学, Kyushu University.

Kotaro Noguchi, Dong Wang, Masanari Kondo, Yasutaka Kamei, Naoyasu Ubayashi, 九州大学, Kyushu University.

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，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 関連研究

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，次にどのようなコードを生成するかを予測する．

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．このエントロピーベースの手法では，Global Entropy(以下GlobalE)とLocal Entropy(以下LocalE)が用いられている．GlobalEは各入力に対して極端にアンバランスな予測を避けるプロンプトを特定するため，LocalEは学習データに対して過度に高性能な予測を避けるプロンプトを特定するために使用されている．調査の結果，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．

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では全てのプログラミング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，プログラミング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

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．

このコンテストは開催番号1~307(2023/06/24現在)が公開されており，各コンテストにはA, B, C, D, E, F, G, Ex(H)までの最大8段階の，難易度の異なる問題が存在し，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．A問題は簡単な文法の確認，B問題は簡単な文法に加えてif文やfor文の処理，C問題は計算量を意識した処理，D問題はアルゴリズムを考慮した処理，E問題以降は複数のアルゴリズムを考慮した処理を行う問題である．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，開催番号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，開催番号はテストケースが存在している問題を使用するため，問題難易度はE問題以降，複数のアルゴリズムを考慮した複雑な処理が必要となるため，D問題までの問題を使用する．

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．

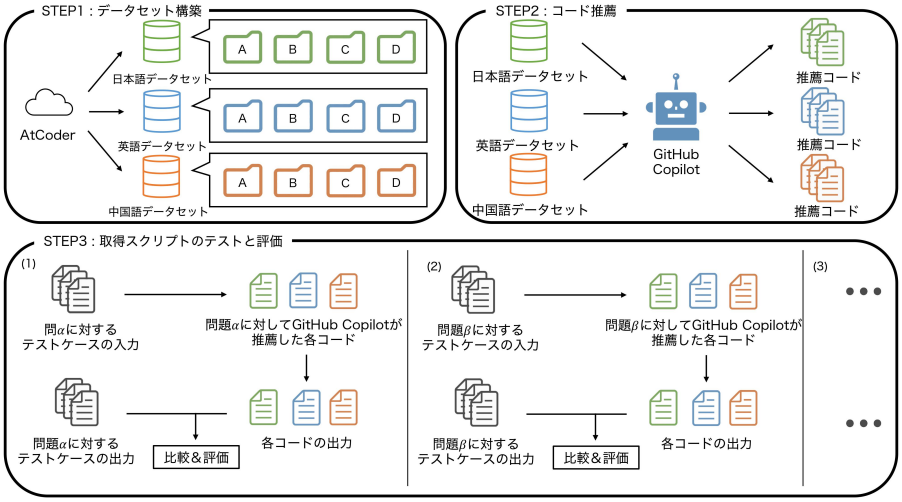

図 1: 実験設計の概要

最終的に，開催番号が 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 コード推薦

3.1 節で作成したデータセットに対して，Copilot でコード推薦を行う．3.1 節で作成した各言語のデータセットの中には，開催番号が 99~287，各問題の難易度が A,B,C,D 存在するため，合計で 756 問ある．これらの各問題に対して，それぞれコード推薦を行う．

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 によるコード推薦を行い，x 個の推薦コードを得る．このとき，x は $0 \leq x \leq 10$ を満たす．また，推薦コード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 の推薦コード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 1: 英語における *Accuracy*

推薦	A		B		C		D	
1st	63.7	864/1357	51.0	860/1686	28.3	482/1702	10.4	172/1657
2nd	65.4	878/1343	50.5	858/1698	26.7	471/1761	10.0	182/1813
3rd	63.6	856/1346	50.9	811/1594	26.6	438/1648	9.31	151/1622
4th	60.4	819/1355	51.6	842/1631	31.3	499/1596	11.0	167/1524
5th	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*

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

3.3 取得コードのテストと評価

3.2 節で取得したコードに対して，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，取得したコードを実行し，テストケースの出力と一致するかを確認する．そして，各推薦コード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，推薦された全コードの内，全てのテストケースを通過したコードの割合を示す．ただし，Copilot が推薦したコードの順番は考慮せずに評価を行う．

4 結果

本章では，RQ と結果，および本実験で得られた結果に対する追加実験の必要性を示す．なお，本実験は

[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

2023/4~2023/7 に実施した.

4.1 RQ への回答

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

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．表の各行は，各推薦で得られた総推薦コード数と全てのテストケースを通過した推薦コード数を表している．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 が毎回異なるコード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 3: 中国語における *Accuracy*

推薦	A		B		C		D	
1st	52.7	739/1402	40.1	705/1757	22.7	405/1787	7.70	136/1766
2nd	52.5	728/1386	43.1	734/1703	21.9	376/1718	7.81	134/1715
3rd	52.1	737/1415	39.7	687/1730	22.7	405/1788	8.11	143/1764
4th	51.8	732/1412	41.5	717/1728	23.2	413/1783	8.35	146/1748
5th	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%

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コード数を平均すると，英語のデータセットでは，A 問題が 1355 個，B 問題が 1661 個，C 問題が 1696 個，D 問題が 1686 個，日本語のデータセットでは，A 問題が 1247 個，B 問題が 1607 個，C 問題が 1596 個，D 問題が 1484 個，中国語のデータセットでは，A 問題が 1410 個，B 問題が 1731 個，C 問題が 1774 個，D 問題が 1747 個であった．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，推薦された全コード数が少なかった．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 の中央値を示している．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，日本語のデータセットとの *Accuracy* の差が大きかった．

GitHub Copilot の推薦コードの正答率は，入力言語によって変化し，日本語，英語，中国語の順に *Accuraccy* が高かった．

4.2 追加実験の必要性

本研究では，AtCoder [1] の開催番号 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．その結果，日本語を用いてコード中にコメントを入力する場合の方が，英語や中国語を用いてコメントを入力する場合よりも，正答率が高いことがわかった．現在の世の中において，世界で最も使用されている言語は英語である [3] ため，入力に使用する言語が英語である場合の方が，正答率が高いことが経験的に知られている．しかし今回の結果より，ある特定のタスクにおいては英語以外の言語を使用した方が正答率が高い可能性が示唆された．ただし，本研究では特定のデータセットに対して調査を行ったため，調査結果をより一般化する必要がある．そこで，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今後調査を行う所存である．

5 考察

4 章で示した結果について，現時点での筆者の考察を述べる．これらは筆者の考察であるため，今後の実験により検証されなければならない．

5.1 推薦コードの生成数と正答率の関係

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，推薦された全コード数が 5 回のうち最も少なく，このことから，推薦 1 回につき推薦コードを最大 10 個取得することで，かえって *Accuracy* が低下した可能性が考えられる．

また，難易度別に推薦された全コード数に着目すると，A 問題は B, C, D 問題に比べて，推薦された全コード数が最も少なかった．これは A 問題が基本的な文法を問う問題であるため，処理が単純であり，類似した文面のコードが推薦されると重複として除外されている可能性があると考えた．

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

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

図 3: 問題 212-A の中国版に対する推薦コードの一例

さらに，言語別で推薦された全コード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 がより最適なコードのみを推薦していることがわかる．これは，AtCoder [1] が日本で運営されているため，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よりアップロードされ，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．

5.2 中国語データセットにおける正答率の低下

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コード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コード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，中国語のコメントを含んだ正解コードの数が少なかった可能性がある．これに対して中国語を基とするプログラミングコンテストのデータセットを使用して調査を行う必要がある．

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コードがアッ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，全ての推薦コード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，全ての推薦コード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 3 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コードである．図 3 のような，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．

6 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 によるコード推薦を行い，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，タスクごとに最適な言語が異なる可能性に関して，データセットを変更した調査はもちろん，推薦コードの品質の評価，および Copilot が推薦するコードの順番を考慮した調査が必要である．

謝辞

本研究の一部は JSPS 科研費 JP21H04877，JP22K17874，JP22K18630，稲盛財団の稲盛科学研究機構（InaRIS: Inamori Research Institute for Science）フェローシップの助成を受けた．

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.