# 日本語によるコード生成能力の正確な評価に向けて

種口 暁人 [1]　藤井 諒 [2,1]　森下 睦 [2,1]　鈴木 潤 [1,3,4]
[1] 東北大学　[2] フューチャー株式会社　[3] 理化学研究所　[4] 国立情報学研究所
taneguchi.akito.t1@dc.tohoku.ac.jp
{r.fujii.6d, m.morishita.pi}@future.co.jp　jun.suzuki@tohoku.ac.jp

## 概要

近年，ソースコード生成などのタスクにも大規模言語モデルを活用する動きが活発化している．しかし，日本語を用いたコード生成ベンチマークは限定的であり，特に Python 以外のプログラミング言語を用いた高品質なデータセットは存在しないことから，正確な評価はできていない．本稿では，既存のデータセットを拡張し，高品質な日本語タスク説明文を伴う，複数のプログラミング言語のコード生成評価ベンチマーク JMultiPL-E を構築した．実験により，日本語での継続事前学習が日本語を用いたコード生成において効果的であることの示唆を得た．

## 1　はじめに

近年，大規模言語モデル (Large Language Models, LLM) の飛躍的な発展により，プログラミング関連タスクにも LLM を活用する動きが活発化している [1, 2, 3]．コードを入力としたコメント生成，あるプログラミング言語から他のプログラミング言語へのコード翻訳など，様々なタスクが存在する [4] が，中でも自然言語の説明文，あるいはコードの一部を入力として続くコードを生成するタスクにおいては，数多くのベンチマークが提案されている [5, 6, 7]．しかしながら，これらのベンチマークでは，コード生成を指示するタスク指示文が英語で記述されていることが多く，特に日本語によるコード生成能力は Python 以外のプログラミング言語では十分に検証されていない．

本研究では，日本語によるコード生成能力を様々なプログラミング言語で評価するために，新たなデータセット **JMultiPL-E** を構築した．具体的には，複数のプログラミング言語に対応したコード生成ベンチマークである MultiPL-E を日本語化することにより，Java および C++の評価データセットを作成した (図 1)．さらに作成したデータセットを用いて，

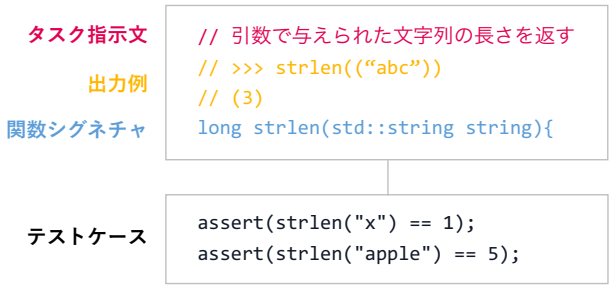


**図 1**　JMultiPL-E に含まれる C++のデータ例

日本語の継続事前学習が日本語でのコード生成能力にどのような影響を及ぼすのかを調査した．

調査の結果，どのプログラミング言語やモデルにおいても英語で指示した際にスコアが高くなった一方で，日本語で継続事前学習をしたモデルでは，日本語と英語の指示文の違いによるスコアの差が縮小し，一部のプログラミング言語では継続事前学習をしていないモデルを上回る日本語でのコード生成能力を示すことが明らかになった．

なお，本研究で構築した JMultiPL-E は今後のコード生成モデルの発展のため，Hugging Face Hub 上に公開する[1)]．

## 2　先行研究

### 2.1　コード生成に関するベンチマーク

今日，自然言語への応用だけでなく，プログラミング言語にも LLM を活用する動きが活発になっている [1, 2, 3, 8, 9]．LLM の開発にはその性能を測ることが不可欠であり，プログラミング言語に関連したタスクにおいても様々なベンチマークが提案されている [4, 10]．特に，表 1 に示すように，コード生成能力を測定するベンチマークは，評価対象とするプログラミング言語やタスク指示文として使われている自然言語によって，様々なものが存在する．

1) https://huggingface.co/datasets/tohoku-nlp/JMultiPL-E

**表 1**　コード生成ベンチマークの比較

| データセット名 | タスク指示文の言語 | | | 評価対象のプログラミング言語 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 英語 | 日本語 | 左記以外 | Python | Java | C++ | PHP | TS | C# | Bash | JS | 左記以外 |
| HumanEval [5] | ✓ | | | ✓ | | | | | | | | |
| MBPP [6] | ✓ | | | ✓ | | | | | | | | |
| MultiPL-E [7] | ✓ | | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 計 17 言語 |
| Multilingual HumanEval [11] | ✓ | | | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | 計 6 言語 |
| MBXP [11] | ✓ | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | 計 6 言語 |
| HumanEval-XL[12] | ✓ | | 計 22 言語 | ✓ | ✓ | | ✓ | ✓ | ✓ | | ✓ | 計 6 言語 |
| mHumanEval[13] | ✓ | ✓ | 計 202 言語 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | 計 17 言語 |
| JHumanEval [14] | | ✓ | | ✓ | | | | | | | | |
| JMultiPL-E (提案) | | ✓ | | | ✓ | ✓ | | | | | | |

例えば，Python のコード生成能力を測定するベンチマークとしては，HumanEval [5] や MBPP [6] などが提案されている．また，Python 以外にも広く使用されているプログラミング言語のコード生成能力を測るベンチマークとしては，MultiPL-E [7]，Multilingual HumanEval，MBXP [11] 等が存在している．さらに，これを自然言語の観点で多言語に拡張したものとして，HumanEval-XL [12]，mHumanEval [13] などが提案されており，mHumanEval には日本語も含まれている．しかしながら，当該ベンチマークの日本語指示文は翻訳品質が低く，その信頼性には難がある．一方で，日本語を用いたベンチマークとしては，他にも HumanEval のタスク指示文を邦訳した JHumanEval [14] が提案されている．JHumanEval は機械翻訳を人が後編集することで構築されているため，日本語の品質は高いものの，対象としているプログラミング言語は Python に限られている．本研究では高品質な日本語の指示文を伴う，幅広いプログラミング言語のための新たなコード生成能力評価ベンチマークを構築する．

## 2.2　事前学習言語とタスク精度の関係

LLM の事前学習に用いる言語と下流タスクの精度の関係については，いくつかの研究で調査が行われている．Raihan ら [13] は，多様な自然言語での事前学習が，コード生成タスクにおけるタスク指示言語への頑健性の向上に有効であることを示唆した．他方，齋藤ら [15] は日本語の継続事前学習によるタスク性能の変化を調査し，Python を用いたコード生成タスクにおいては継続事前学習の有用性が小さいことを指摘している．そこで本研究では，日本語の継続事前学習がコード生成に与える影響について，複数のプログラミング言語を用いて詳細に分析する．

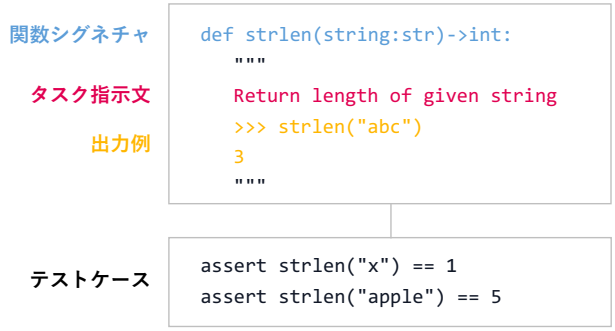


**図 2**　HumanEval のデータ例

# 3　JMultiPL-E の構築

本データセットは，HumanEval，JHumanEval および MultiPL-E をもとに構築した．3.1 節ではこれら 3 つのデータセットについて概説し，続く 3.2 節では JMultiPL-E の構築法について述べる．

## 3.1　使用したデータセットの概要

**HumanEval** は，164 個の問題で構成されており，図 2 に示すように各問題は関数シグネチャ，タスク指示文，出力例およびいくつかのテストケースを含む．モデルは関数シグネチャ，タスク指示文，出力例に続けて関数の中身を生成し，テストケースを通過した問題の割合を測定する．

**JHumanEval** は，HumanEval のタスク指示文を英語から日本語に翻訳したものであり，日本語のタスク指示文からのコード生成能力の評価に用いることが可能である．

**MultiPL-E** は，Python のコード生成能力を測定するベンチマークである HumanEval および MBPP を 20 以上のプログラミング言語に拡張する形で開発されたデータセットである．このデータセットは，各言語で同じ問題を並列的に提供することで一貫性のある評価を可能にしている．

### 3.2 構築方法

本データセットは，MultiPL-E の HumanEval に対応する Java および C++の英語タスク指示文を JHumanEval の日本語のタスク指示文に置き換えることで作成した．

まず，JHumanEval と HumanEval の日英タスク指示文の差分をとることで，英語の指示文に対応した日本語の指示文を抽出した．この際，JHumanEval と HumanEval のタスク指示文の差分には，Python 特有の単語や表現が含まれる場合がある．したがって，これらを Java および C++特有の単語や表現に置換する必要がある．具体例として，Python の list は，Java では array list に，C++では vector に修正した．また，JHumanEval と HumanEval のタスク指示文の差分には Python のテストケースが含まれる場合があり，それらが Java や C++のタスク指示文の中に混入することがあり，これらは人手によって修正した．

以上の点に注意し，最終的にはすべての問題に対して人手でチェックを行い，不適切な部分を修正することで JMultiPL-E を作成した．図 1 に作成した JMultiPL-E に含まれる C++の問題例を示す．問題数は Java が 158，C++が 161 である．ここで，HumanEval に比べて問題数が減少しているのは，164 問の問題のうち Python 固有のヘルパー関数が使われている問題や型注釈が翻訳できない問題などを除外しているためであることに注意する[2)]．

## 4 実験

### 4.1 実験設定

Python, Java, C++の 3 つのプログラミング言語で評価を行った．Python は，既存のベンチマークである HumanEval および JHumanEval，それ以外の 2 言語は MultiPL-E および今回作成した JMultiPL-E を用いて評価した．

評価対象のモデルには，Llama-3.1-8B-Instruct[3)] (Llama)，および Llama-3.1-Swallow-8B-Instruct-v0.1[4)] (Swallow) の 2 つを用いた．なお，Llama 3.1-Swallow-8B-Instruct-v0.1 は，Llama 3.1-8B[5)] に対して日本語で継続事前学習を行った後に，指示学習を行ったモデルである．

表 2　モデルごとの pass@1

| | Python | | Java | | C++ | |
|---|---|---|---|---|---|---|
| モデル | 英語 | 日本語 | 英語 | 日本語 | 英語 | 日本語 |
| Llama | 56.58 | 44.76 | 22.15 | 13.98 | 30.56 | 24.97 |
| Swallow | 38.54 | 35.49 | 29.62 | 24.72 | 31.30 | 27.83 |

### 4.2 評価方法

今回の評価では，Swallow プロジェクト 大規模言語モデル 評価スクリプト Ver. 202407[6)] を使用した．この評価方法では日英のタスク指示文からコードを生成し，生成されたコードの品質を pass@1 [5] を用いて測定する．

pass@$k$ は与えられたタスク指示文を基にモデルがコードを $k$ 個生成した時，いずれか一つ以上の候補がユニットテストを通過する期待値を計算した指標であり，以下の式で計算される．

$$\text{pass@}k = \frac{1}{N}\sum_{N}\left(1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right) \tag{1}$$

この式において，$N$ は設問数を，$n$ は生成された候補プログラムの総数を，$c$ は正しい候補プログラムの数をそれぞれ示す．これは，$n$ 個の候補プログラムから $k$ 個を選択する時，$k$ 個すべて不正解のプログラムを選択する事象の余事象であり，$k$ 個のうち少なくとも 1 つのプログラムがユニットテストを通過する確率を表す．本研究では $n = 10$ とすることで，各設問に対して 10 個のコードを生成し，pass@1 の期待値を計算した．

### 4.3 実験結果

表 2 に pass@1 の評価結果を示す．まず，自然言語の観点から比較すると，モデルやプログラミング言語に関わらず，タスク指示文が英語の場合に高いスコアとなっていることが確認された．これは，英語でタスク指示を行った場合に他の言語よりもタスク遂行性能が高くなりやすいという Ahuja らの主張 [16] とも符合する．また，Llama では日本語と英語の差が約 6-12 ポイントと大きいのに対し，日本語で継続事前学習をした Swallow ではその差が 3-5 ポイント程度となり，スコアの差にも違いが確認された．

2) MultiPL-E においても同様の問題が除外されている．
3) https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
4) https://huggingface.co/tokyotech-llm/Llama-3.1-Swallow-8B-Instruct-v0.1
5) https://huggingface.co/meta-llama/Llama-3.1-8B
6) https://github.com/swallow-llm/swallow-evaluation

表 3　英語のみ解けていた問題のタスク指示文

| | |
|---|---|
| **英語** | You're given a list of deposit and withdrawal operations on a bank account that starts with zero balance. Your task is to detect if at any point the balance of account fallls **below zero**, and at that point function should return True. Otherwise it should return False. |
| **日本語** | 銀行口座に対する入出金操作のリストが与えられます。あなたのタスクは、残高ゼロから始まて、口座の残高が**ゼロ以下**になったかどうかを検出し、その時点で関数が True を返すようにすることです。そうでなければ False を返すようにしてください。 |

次に，プログラミング言語の観点から比較すると，どちらのモデルにおいても Python でコードを生成させた場合のスコアが最も高くなった．また，Swallow のスコアは Python では英語，日本語ともに Llama よりも低い傾向にあったが，Java および C++ では Swallow の方が英日ともにスコアが高い結果となった．なお，Swallow のプログラミング言語別正解数を付録 A に示す．

## 5　考察

表 2 より，HumanEval および JHumanEval を用いて Python のコード生成能力のみを比較すると，日本語の継続事前学習によりコード生成能力が下がっているように見受けられる．しかしながら，今回作成した JMultiPL-E を使用した評価を行った結果，タスク指示文に使用した自然言語によらず，Swallow の Java や C++に対するスコアは Llama と比べて高いことがわかった．特に日本語側の性能に大きな改善が見られたことから，日本語の継続事前学習は日本語指示文によるコード生成能力の向上に寄与していると考えられる．これは，斎藤ら [15] とは異なる示唆を与えており，Python 以外のコード生成能力にも着目することの有用性が伺える．

次に，タスク指示文の言語による性能差を確認するため，Swallow が 3 つのプログラミング言語の全てにおいて (1) 日本語のタスク指示文のみで正解した問題，(2) 英語のタスク指示文のみで正解した問題を確認した．

日本語のタスク指示文でのみ正解している問題は 1 問存在していたが，タスク指示文や生成されたコードを確認しても特に目立った特徴は確認されなかった．一方，英語のタスク指示文のみで正解している問題は全 4 問存在した．表 3 および図 3 は，そのうちの 1 つのタスク指示文と，日本語のタスク指示文を用いた際の Python での生成例である．本事例では，英語側で "below zero" となっている部分が，日本語側で「ゼロ以下」と翻訳されているが，正しくは「ゼロ未満」と翻訳されるべきである．生成されたコードでは，ゼロ以下の条件で分岐するコードが記述されており，境界条件においてユニットテストに通過できなかったものと考えられる．残りの 3 つの問題についても同様に，「未満」と訳されるべきところを「以下」と訳していたり，訳そのものが曖昧で不自然だったりと，翻訳誤りが見受けられた．なお，これらの問題に対して翻訳誤りや曖昧なタスク指示文を修正し，再度 Swallow で評価すると，修正した 4 つの問題全てにおいて正しくコードを生成することができた．このことから，翻訳されたデータセットによる評価を行う際には，その翻訳文の正確性を確認することが適切な評価のために重要であると考えられる．

```python
def below_zero(operations: List[int]) -> bool:
    balance = 0
    for operation in operations:
        balance += operation
        if balance <= 0:
            return True
    return False
```

図 3　日本語のタスク指示文で生成されたコード

## 6　おわりに

本研究では，日本語のタスク指示文を伴うコード生成能力のより正確な評価のため，新たなデータセット JMultiPL-E を構築した．また，本データセットを用いて，日本語の継続事前学習が日本語指示文を用いたコード生成能力に及ぼす影響を調査した．

実験の結果，日本語で継続事前学習をしたモデルでは，Java および C++に関して性能の向上が見られるなど，従来のデータセットでは評価できていなかった側面が明らかになった．また，翻訳されたデータセットによる評価を行う際には，その翻訳文の正確性に起因して評価品質が低下する可能性があることが示唆された．

今後はコード補完以外のプログラミング関連タスクに対しても，日本語話者向けベンチマークの拡充を目指す．

## 謝辞

本研究は，JST ムーンショット型研究開発事業 JPMJMS2011-35 (fundamental research)，および，文部科学省の補助事業「生成 AI モデルの透明性・信頼性の確保に向けた研究開発拠点形成」の支援を受けたものです.

また，本研究成果の一部は，データ活用社会創成プラットフォーム mdx を利用して得られたものです.

## 参考文献

[1] Anton Lozhkov, et al. StarCoder 2 and The Stack v2: The next generation. arXiv preprint arXiv:2402.19173, 2024.

[2] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming – the rise of code intelligence. arXiv preprint arXiv:2401.14196, 2024.

[3] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. arXiv preprint arXiv:2409.12186, 2024.

[4] Shuai Lu, et al. CodeXGLUE: A machine learning benchmark dataset for code understanding and generation. arXiv preprint arXiv:2102.04664, 2021.

[5] Mark Chen, et al. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374, 2021.

[6] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. arXiv preprint arXiv:2108.07732, 2021.

[7] Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, Arjun Guha, Michael Greenberg, and Abhinav Jangda. MultiPL-E: A scalable and extensible approach to benchmarking neural code generation. arXiv preprint arXiv:2208.08227, 2022.

[8] Baptiste Rozière, et al. Code Llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2024.

[9] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with evol-instruct. arXiv preprint arXiv:2306.08568, 2023.

[10] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In The Twelfth International Conference on Learning Representations, 2024.

[11] Ben Athiwaratkun, et al. Multi-lingual evaluation of code generation models. arXiv preprint arXiv:2210.14868, 2022.

[12] Qiwei Peng, Yekun Chai, and Xuhong Li. HumanEval-XL: A multilingual code generation benchmark for cross-lingual natural language generalization. In Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pp. 8383–8394, 2024.

[13] Nishat Raihan, Antonios Anastasopoulos, and Marcos Zampieri. mHumanEval–A multilingual benchmark to evaluate large language models for code generation. arXiv preprint arXiv:2410.15037, 2024.

[14] 佐藤美唯, 高野志歩, 梶浦照乃, 倉光君郎. LLM は日本語追加学習により言語間知識転移を起こすのか？言語処理学会 第 30 回年次大会発表論文集, 2024.

[15] 齋藤幸史郎, 水木栄, 大井聖也, 中村泰士, 塩谷泰平, 前田航希, Youmi Ma, 服部翔, 藤井一喜, 岡本拓己, 石田茂樹, 高村大也, 横田理央, 岡崎直観. LLM に日本語テキストを学習させる意義. 研究報告自然言語処理 (NL), 2024.

[16] Kabir Ahuja, et al. MEGA: Multilingual evaluation of generative AI. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 4232–4267, 2023.

**表 4**　Python の統計量

| 日本語＼英語 | 正解 | 不正解 | 合計 |
|---|---|---|---|
| 正解 | 96 | 10 | 106 |
| 不正解 | 24 | 33 | 57 |
| 合計 | 120 | 43 | 164 |

**表 5**　Java の統計量

| 日本語＼英語 | 正解 | 不正解 | 合計 |
|---|---|---|---|
| 正解 | 73 | 7 | 80 |
| 不正解 | 22 | 56 | 78 |
| 合計 | 95 | 63 | 158 |

**表 6**　C++の統計量

| 日本語＼英語 | 正解 | 不正解 | 合計 |
|---|---|---|---|
| 正解 | 78 | 11 | 89 |
| 不正解 | 21 | 51 | 72 |
| 合計 | 99 | 62 | 161 |

## A　プログラミング言語別正解数

表 4, 5, 6 に Swallow を用いてコードを生成させた際の言語別の正解・不正解数をまとめたものを示す．正解数とは各タスクにおいて生成された 10 個のコードのうち，1 つでもテストケースを全て通ったコードがある問題の数であり，不正解数は生成された 10 個のコードのうち 1 つもテストケースを通るコードがなかった問題の数である．

Python では，タスク指示文の言語によらず正解することができた問題が 96 問存在し，英語のタスク指示文の時のみ正解することができた問題は 24 問，日本語のタスク指示文の時のみ正解することができた問題は 10 問，どちらも正解することができなかった問題は 33 問あった．

Java では，タスク指示文の言語によらず正解することができた問題が 73 問存在し，英語のタスク指示文の時のみ正解することができた問題は 22 問，日本語のタスク指示文の時のみ正解することができた問題は 7 問，どちらも正解することができなかった問題は 56 問あった．

C++では，タスク指示文の言語によらず正解することができた問題が 78 問存在し，英語のタスク指示文の時のみ正解することができた問題は 21 問，日本語のタスク指示文の時のみ正解することができた問題は 11 問，どちらも正解することができなかった問題は 51 問あった．

これらの表から Python では他の 2 つの言語に比べて，タスク指示文に使われている自然言語によらず正しいコードを生成できている数が多かった．また，どのプログラミング言語においても日本語より英語でタスク指示を与えた時の方が正確なコードを生成できている数が多かった．