Exploring the Effect of Multiple Natural Languages on Code Suggestion Using GitHub Copilot

Kei Koyanagi
Dong Wang
Kotaro Noguchi
Masanari Kondo
koyanagi@posl.aist.kyushu-u.ac.jp
d.wang@ait.kyushu-u.ac.jp
noguchi@posl.aist.kyushu-u.ac.jp
kondo@ait.kyushu-u.ac.jp
Kyushu University
Japan

Alexander Serebrenik
a.serebrenik@tue.nl
Eindhoven University of Technology
The Netherlands

Yasutaka Kamei
Naoyasu Uabayashi
kamei@ait.kyushu-u.ac.jp
ubayashi@ait.kyushu-u.ac.jp
Kyushu University
Japan

ABSTRACT

GitHub Copilot is an AI-enabled tool that automates program synthesis. It has gained significant attention since its launch in 2021. Recent studies have extensively examined Copilot’s capabilities in various programming tasks, as well as its security issues. However, little is known about the effect of different natural languages on code suggestion. Natural language is considered a social bias in the field of NLP, and this bias could impact the diversity of software engineering. To address this gap, we conducted an empirical study to investigate the effect of three popular natural languages (English, Japanese, and Chinese) on Copilot. We used 756 questions of varying difficulty levels from AtCoder contests for evaluation purposes. The results highlight that the capability varies across natural languages, with Chinese achieving the worst performance. Furthermore, regardless of the type of natural language, the performance decreases significantly as the difficulty of questions increases. Our work represents the initial step in comprehending the significance of natural languages in Copilot’s capability and introduces promising opportunities for future endeavors.

KEYWORDS

Code Suggestion, GitHub Copilot, Empirical Study

1 INTRODUCTION

In recent years, the expansion of IT demand has led to the use of various support tools, such as task management tools and project management tools, to improve development efficiency. One of these tools is GitHub Copilot, which was introduced by GitHub and OpenAI in June 2022 [8]. Copilot is a code suggestion tool powered by a large-scale language model. It suggests code snippets and libraries in different programming languages to developers based on comments that describe specifications and the code being written. As a result, developers could save time by not starting from scratch and further reduce development costs.

It is well-known that the output of large language models can vary significantly depending on the input [16]. Several studies have been conducted to empirically examine the impact of input on the accuracy of suggested code in terms of Copilot. For example, Yetistiren et al. [28] reported that the utilization of proper explanations of the given problem is important in terms of acquiring correct and valid code. Nguyen and Nadi [19] evaluated Copilot’s performance by running LeetCode’s provided tests. The results showed that Copilot’s performance varies across different languages, with Java having the highest correctness. Mastropaolo et al. [17] revealed that paraphrasing the input description results in different code suggestions with substantial variations, indicating the central role played by the model’s input.

To reduce barriers for non-native English speakers and assist developers of all backgrounds, Copilot supports not only English but also other languages such as Chinese and Japanese. In addition to languages of specific domains, the performance of large language models can also be influenced by the characteristics of natural languages [12, 25]. Recent studies have proposed a benchmark that spans multiple natural languages in order to address the technology gap in code suggestion. For example, Wang et al. [25] developed a multilingual dataset namely MCoNala, annotating a total of 896 NL-Code pairs in Spanish, Japanese, and Russian. However, little is known about the effect of diverse natural languages on the Copilot tool. We hypothesize that variations in natural language input could also substantially affect the performance of Copilot.

To minimize the potential bias caused by natural languages in the future, we conduct an empirical study on Copilot to explore the effect of multiple natural languages on code suggestion. Specifically,
we use 756 AtCoder (a programming contest) questions from 189 contests to create queries for Copilot in English, Japanese, and Chinese. We then evaluate the correctness of the corresponding Copilot suggestions against the test cases provided by AtCoder. The preliminary results reveal that natural languages do have an effect on the correctness of the code suggestions when using Copilot. We find that in the context of AtCoder questions, Japanese has the highest level of correctness, followed by English, while Chinese performs the worst. More specifically, there is a difference ranging from 2.6% to 11.5% observed between Chinese and English across four levels of question difficulty. In addition, the correctness tends to decrease dramatically with the increase of difficulty for all studied natural languages.

In summary, the contributions of this work are as follows: (1) we are the first work to investigate the ability of Copilot to suggest code, taking into account the distinction of natural languages; (2) the results emphasize the significance of addressing bias in natural languages when using generative AI techniques for programming tasks, in addition to efforts aimed at addressing political [15], sentiment [10], and gender [23] aspects; (3) Our work opens up several future research directions.

2 RELATED WORK

Studies on GitHub Copilot. Several studies have focused on the securities of Copilot. Pearce et al. [21] examined the security issues in code suggested based on queries created from the 25 top CWE vulnerabilities with a total of 89 scenarios, with approximately 40% of them being vulnerable. Asare et al. [4] demonstrated that Copilot, despite performing differently across various vulnerability types, is not as bad as human developers at introducing vulnerabilities. At the same time, another group of studies have been conducted to evaluate the various capabilities of Copilot. To name a few, Nguyen and Nadi [19] validated the correctness of 132 LeetCode questions across four different programming languages. Mastropaolo et al. [17] analyzed whether different but semantically equivalent natural language descriptions result in the same recommended function. Dakhel et al. [6] investigated the quality of the code by Copilot and suggested that Copilot can become an asset for experts, but a liability for novice developers. Sobania et al. [22] compared the Copilot and Genetic Programming on standard program synthesis benchmark problems. Al Madi [3] reported that the suggested code is comparable in complexity and readability to code written by human pair programmers.

Studies on Social Bias in Language Models. Several studies have been carried out toward understanding and mitigating social biases in language models [10, 13, 24]. Regarding gender bias, Treude and Hata [23] examined the extent to which 56 tasks related to software development are affected by implicit gender bias embedded in large language models. Their findings revealed a clear pattern of gender bias. Regarding religion bias, Abid et al. [1] investigated the anti-Muslim bias, demonstrating that it appears consistently and creatively in different uses of the model and that it is severe even compared to biases about other religious groups. Regarding political bias, Liu et al. [15] described metrics for measuring political bias and proposed a reinforcement learning framework for mitigating such biases in the generated text. Regarding sentiment bias, Huang et al. [10] quantified sentiment bias by adopting individual and group fairness metrics from the fair machine learning literature. In contrast to previous work, we present a novel perspective to examine the Copilot capability within the context of diverse natural languages on code suggestion. This would offer valuable insights into optimizing the usage of Copilot and complement the knowledge of social bias in large language models.

3 STUDY DESIGN

In this section, we describe the study design, including the research question, data collection, and code suggestion and its evaluation.

3.1 Research Question

RQ - How does the input of different natural languages affect the performance of Copilot? We aim to study the effectiveness of natural languages (i.e., English, Japanese, and Chinese) in the code suggestions synthesized by the Copilot. Answering this research question will shed light on the usage of Copilot for developers from different backgrounds in practice.

3.2 Data Collection

To answer our research question, we collected the programming questions from the AtCoder. AtCoder is one of the largest programming contest sites, catering to individuals ranging from beginners to experts. It has been extensively studied in previous works analyzing programming [2, 18]. Hence, it is a suitable starting point for examining the impact on code suggestion.

We used the dataset that was publicly available at the end of March 2023, consisting of 1,624 questions from 287 contests. Each contest has up to eight levels of difficulty, labeled A, B, C, D, E, F, G, and Ex (H). The higher the letter’s position in the alphabet, the more difficult the question. Some contests did not contain all eight levels. To minimize potential bias in our validation, we established the following two criteria for selecting the questions:

- We focus on levels A to D since questions E and higher require complex processing that involves multiple algorithms.
- We excluded the contests where the test cases were not provided.

After applying the criteria above, we collected a total of 756 questions (189 per level) from 189 contests (from the 99th contest to the 287th contest). We chose the 99th because the following contests consistently feature four levels of difficulty. Note that AtCoder only offers questions in English and Japanese. Therefore, we need to
translate the questions into Chinese. Chinese is a representative language in this context study [5, 14, 26, 27] and serves as the comparative language for our goal. To ensure accuracy, the second author, a native Chinese speaker with over five years of experience in software engineering research inspected all 756 questions translated by DeepL. API1 and made significant modifications. A similar method has also been adopted by Wan et al. [24]. Below we present four examples of question descriptions with levels A to D from the 212th contest2 for a better understanding of code suggestion tasks.

The question with A level:

Takahashi melted and mixed A grams of gold and B grams of silver (0 ≤ A, B, 0 < A + B) to produce new metal. What metal did he produce: pure gold, pure silver, or an alloy? Formally, the product is called as follows. Pure gold, if 0 < A and B = 0. Pure silver, if A = 0 and B < 0. An alloy, if 0 < A and 0 < B.

The question with B level:

You are given a 4-digit PIN: \(X_1X_2X_3X_4\), which may begin with a 0. The PIN is said to be weak when it satisfies one of the following conditions: All of the four digits are the same. For each integer \(i\) such that 1 ≤ \(i\) ≤ 3, \(X_{i+1}\) follows \(X_i\). Here, \(j+1\) follows \(j\) for each 0 ≤ \(j\) ≤ 8, and 0 follows 9. If the given PIN is weak, print Weak; otherwise, print Strong.

The question with C level:

You are given two sequences: \(A = (A_1, A_2, ..., A_N)\) consisting of \(N\) positive integers, and \(B = (B_1, ..., B_M)\) consisting of \(M\) positive integers. Find the minimum difference of an element of \(A\) and an element of \(B\), that is, \(\min_{1\leq i \leq N} \min_{1\leq j \leq M} |A_i - B_j|\).

The question with D level:

Takahashi has many balls, on which nothing is written, and one bag. Initially, the bag is empty. Takahashi will do \(Q\) operations, each of which is one of the following three types.

Type 1: Write an integer \(X_i\) on a blank ball and put it in the bag. Type 2: For each ball in the bag, replace the integer written on it with that integer plus \(X_i\). Type 3: Pick up the ball with the smallest integer in the bag (if there are multiple such balls, pick up one of them). Record the integer written on this ball and throw it away. For each 1 ≤ \(i\) ≤ \(Q\), you are given the type \(P_i\) of the \(i\)-th operation and the value of \(X_i\) if the operation is of Type 1 or 2. Print the integers recorded in the operations of Type 3 in order.

3.3 Code Suggestion and Evaluation

**Code Suggestion.** We invoked Copilot to suggest the code using a dataset of 756 questions from three different natural languages. In this study, our focus is on code written in Python, which is currently a widely used programming language. During the input, in addition to the question description, we provided complementary information to make the query more precise. This included question constraints, the format of input and output, as well as examples of input and output. For example, given the aforementioned question with A level, the complementary information is:

```
[Constraint] 0 <= A, B <= 100 and 1 <= A+B and A, B are integers
[Input] A B
[Output] Gold or Silver or Alloy
[Input Example] 50 50
[Output Example] Alloy
```

Based on the input, Copilot suggested \(x\) code snippets, \(0 \leq x \leq 10\). Unlike the prior work [19] where they manually invoked Copilot for each query, to mitigate the human bias, we established an environment using Keyboard Maestro3 to automatically manipulate the input/suggestion. Note that the process from input to output is executed five times for every queried question. The suggested code by Copilot may vary each time, hence we take into account the potential randomness bias in our subsequent evaluation.

**Evaluation.** We evaluated the suggested code using the corresponding test cases provided for each question. Specifically, we examined whether the output of the suggested code matches the expected output of the test case. We used Accuracy to measure the correctness of the suggested code, which indicates the percentage of code that passed all test cases out of all suggested code. To statistically confirm the significant differences among the studied natural languages at each difficulty level of questions, we perform the one-way ANOVA test [7].

4 RESULTS

Tables 1, 2 and 3 show the accuracy of suggested code for five rounds under the setting of English, Japanese, and Chinese, respectively. Each row displays the proportion of suggested code snippets that passed all test cases in each round. Note that the number of suggested snippets varies depending on the questions.

When the query is written in Chinese, Copilot is less likely to suggest the correct code. As shown in Table 3, we observe that the accuracy of Chinese is relatively lower than that of English and Japanese for all levels of questions, which indicates that the suggested code has a lower chance of passing the provided test cases. For example, the accuracy of the A, B, C, D level, varies from 51.5% to 52.7%, 39.7% to 43.1, 21.9% to 23.2%, and 7.6% to 8.4%. However, as shown in Table 1, when the query is written in English, the accuracy for the four levels, ranges from 60.4% to 65.4%, 50.5% to 51.6%, 26.6% to 31.3%, and 9.3% to 12.0%, respectively. On the other hand, we find that the Japanese achieved the best performance comparatively. Specifically, the accuracy of the A, B, C, D level varies from 67.4% to 68.7%, 52.8% to 57.6%, 28.4% to 33.1%, and 10.0% to 12.8%. Moreover, the statistical tests indicate that there are significant differences among the three languages at each difficulty level. The p-values of the statistical tests for the A, B, C, and D levels among the three languages are 9.924e-10, 1.258e-10, 7.661e-10, and 0.000256, respectively.

The accuracy varies as the number of execution rounds changes. Our repetitive experiment suggests that to some extent there is

---

1https://www.deepl.com/docs-api
2https://atcoder.jp/contests/abc212/tasks
3https://www.keyboardmaestro.com/main/
suggest incorrect code in situations where the correct code should handle multiple conditions, complex conditions, or string outputs. The below example displays one of the suggested incorrect code snippets for Chinese in question 212-A.

```python
def main():
A, B = map(int, input().split())
    if A == 0:
        print("Silver")
    elif B == 0:
        print("Gold")
```

The correct code should handle three conditions: Gold, Silver, and Alloy, based on the combination of variables A and B. However, this code only has two conditions. This tendency may be because the output format of AtCoder strings uses Roman characters or English words, and the Chinese dataset used as input included Roman characters or English words. These characters may not have been recognized as symbols.

(II) Relationship between the number of suggested code and the correctness. When analyzing Tables 1, 2, and 3 and focusing on maximum accuracy, if there is a difference of 3% or more from the lowest accuracy across five rounds within the same language and difficulty level, the total number of suggested code snippets will be the lowest. This result suggests that increasing the maximum number of suggested code snippets may decrease accuracy in comparison to suggesting a smaller number.

In terms of the number of code snippets suggested for each difficulty level, as shown in the tables, we notice that level A had the lowest number of code snippets suggested compared to levels B, C, and D. This might be because Copilot does not output duplicated code snippets in the maximum of 10 suggested code snippets. We deduce that, compared to higher-level questions, level A questions, which typically involve basic grammar that is simple to process, may result in a higher number of duplicated code snippets.

Comparing the average number of suggested code snippets across the three languages, we found that Japanese had the lowest number. As observed in Section 4, Japanese also leads to the highest accuracy. Therefore, there might be a relationship between the number of suggested code snippets and the accuracy. The possible reason for this is that AtCoder is primarily operated in Japan and is widely used by Japanese people. Consequently, there is a larger pool of Japanese-coded responses with accompanying comments that were utilized during the training of the large language models.

### 6 THREATS TO VALIDITY

**External Validity** The evaluation results may not be generalized to other contest questions, programming languages other than Python, and natural languages other than English, Japanese, and Chinese. However, we believe that the representativeness of our study is highlighted by the dominance of Python and the popularity of the natural languages chosen. Furthermore, the questions in AtCoder are designed for learners. Therefore, future studies should investigate the impact of code suggestions on professional tasks.

**Construct Validity** During the data preparation, the questions may not be translated precisely from English to Chinese [14]. To address this threat, we have included a native Chinese researcher with a strong research background in this work to manually validate the Chinese translation generated by DeepL.
We gratefully acknowledge the financial support of: (1) JSPS for the InaRIS Fellowship. Research Institute for Science for supporting Yasutaka Kamei via the Bilateral Program grant JPJSBP120239929; and (2) the Inamori KAKENHI grants (JP21H04877, JP22K17874, JP23K18630, JP23K16864), and the complementary results.

We gratefully acknowledge the financial support of: (1) JSPS for the InaRIS Fellowship. Research Institute for Science for supporting Yasutaka Kamei via the Bilateral Program grant JPJSBP120239929; and (2) the Inamori KAKENHI grants (JP21H04877, JP22K17874, JP23K18630, JP23K16864), and the complementary results.

**ACKNOWLEDGMENTS**

We gratefully acknowledge the financial support of: (1) JSPS for the KAKENHI grants (JP21H04877, JP22K17874, JP22K18630, JP23K16864), and Bilateral Program grant JPJSBP120239929; and (2) the Inamori Research Institute for Science for supporting Yasutaka Kamei via the InaRIS Fellowship.

**DATA AVAILABILITY**

To encourage the replication study in the future, we have made our replication package available [11], including the experiment scripts and the complementary results.

**REFERENCES**


