Non-English Code Generation with Cross-Lingual Chain of Thought

Yuha Nishigata, Waka Ito, Kimio Kuramitsu

Proceedings of the 39th Pacific Asia Conference on
Language, Information and Computation (PACLIC 39)

Emmanuele Chersoni, Jong-Bok Kim (eds.)

2025

© 2025. Yuha Nishigata, Waka Ito, Kimio Kuramitsu. Non-English Code Generation with Cross-Lingual Chain of Thought. In Emmanuele Chersoni, Jong-Bok Kim (eds.), *Proceedings of the 39th Pacific Asia Conference on Language, Information and Computation* (PACLIC 39), 443-451. Institute for the Study of Language and Information, Kyung Hee University. This work is licensed under the Creative Commons Attribution 4.0 International License.

Non-English Code Generation with Cross-Lingual Chain of Thought

Yuha Nishigata
Japan Women's University
Tokyo, Japan
m2116061ny@ug.jwu.ac.jp

Waka Ito
Japan Women's University
Tokyo, Japan
m2016013iw@ug.jwu.ac.jp

Kimio Kuramitsu
Japan Women's University
Tokyo, Japan
kuramitsuk@fc.jwu.ac.jp

Abstract

Large language models (LLMs) that support multiple languages are becoming increasingly important. A key challenge in building such models is the performance gap between English and non-English languages, largely caused by the imbalance of pre-training data across languages.

We introduce Cross-Lingual Chain-of-Thought (CL-CoT), a fine-tuning approach designed to improve code generation in low-resource languages through cross-lingual transfer. CL-CoT works by adding English chain-of-thought prompts to instruction texts in low-resource languages, such as Japanese, encouraging the model to reason in English while generating code.

We evaluated CL-CoT on code generation tasks in three low-resource languages. Our method outperformed conventional approaches in most cases, indicating that CL-CoT effectively promotes cross-lingual transfer.

1 Introduction

Large language models (LLMs) have become foundational components of modern information systems. For widespread adoption and practical implementation, users must be able to utilize LLMs smoothly in their native languages. However, developing LLMs that operate effectively in non-English languages presents significant challenges.

The fundamental cause is the imbalance of language resources in pre-training datasets (Li et al., 2025; Zhang et al., 2024).LLM pre-training relies heavily on web text data, which is predominantly English(Penedo et al., 2025). This tendency is particularly pronounced in specialized domains such as engineering and medicine (Zhao et al., 2024).

Cross-lingual transfer has attracted attention as one approach to address this challenge. Cross-lingual transfer refers to transferring knowledge acquired through learning in a specific language to another language (Wu and Dredze, 2019). In particular, by transferring knowledge from high-resource languages such as English to low-resource languages such as Japanese, performance improvements can be expected for instructions in low-resource languages (Xu et al., 2025). However, many aspects of the mechanisms underlying cross-lingual transfer remain unclear.

We propose Cross-Lingual Chain-of-Thought (CL-CoT), an instruction tuning method based on the internal translation hypothesis. This hypothesis suggests that LLMs internally perform reasoning in English when processing non-English instructions(Schut et al., 2025). CL-CoT aims to promote cross-lingual transfer by explicitly encouraging English-based reasoning.

In this study, we evaluated the effectiveness of CL-CoT in promoting cross-lingual transfer using code generation tasks. Code generation has been reported to significantly improve programming efficiency and reduce the burden on developers (Paradis et al., 2024). If we consider programming languages as a type of language, code generation can be viewed as a translation task from natural language to code. However, what distinguishes code generation from translation tasks is that established methods exist for quantitatively evaluating the correctness of generated code(Chen et al., 2021).

In code generation tasks, it is known that significant performance gaps exist between instructions in English and instructions in low-resource languages (Li et al., 2024; Sato et al., 2024). Additionally, since the output is quantitatively evaluable code, this is a task that enables the analysis of cross-lingual transfer. In this paper, we analyze cross-lingual transfer in code generation not only between English and Japanese, which we have previously investigated, but also with other Asian languages (Vietnamese and Korean) to verify the general applicability of CL-CoT.

The remainder of this paper is organized as follows. Section 2 defines the problem regarding language resource quantity and cross-lingual performance gaps, and explains why we focused on cross-lingual transfer and internal translation. Section 3 proposes the CL-CoT method. Section 4 reports the evaluation of code generation performance using the CL-CoT method. Section 5 summarizes related work, and finally, Section 6 concludes this research.

2 Background

In this section, we describe code generation tasks, language resource imbalances, and cross-lingual transfer methods that inform our approach.

2.1 Code Generation

Code generation converts natural language instructions into executable code (Jiang et al., 2024). Similar to machine translation, it involves converting natural language descriptions into another representation system (programming languages). However, generating code requires understanding the intent behind natural language descriptions and producing algorithmic solutions. Therefore, code generation tasks serve as a testbed for fundamental NLP capabilities such as compositional understanding and abstract reasoning.

Code generation tasks have robust semantic evaluation metrics that directly measure whether a model truly understands and solves given problems.

Traditional evaluation metrics like BLEU scores often have low correlation with human judgment, making objective assessment challenging. Code generation tasks address this limitation by enabling robust semantic evaluation through execution-based metrics such as Pass@k (Chen et al., 2021), providing clear, quantitative performance evaluation independent of subjective interpretation.

2.2 Language Resources

Code generation faces significant language resource imbalances. The latest information sources about code are disseminated through documentation, tutorials, and academic papers written in English. Additionally, discussions about code implementation methods and error resolution are primarily conducted in English on developer-oriented platforms such as GitHub and Stack Overflow. According to Kocetkov et al. (2022), in The Stack, one of the code datasets used for pre-training, 94% of code comments are written in English as shown in Figure 1, with the proportion of low-resource languages being extremely limited. Given this background, English is the only high-resource language, while many languages including Chinese are considered as low-resource languages.

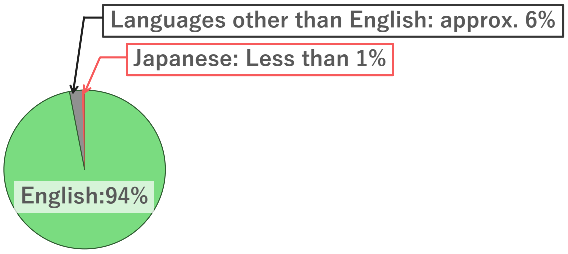


Figure 1: Language distribution of code comments in The Stack dataset as a percentage-based pie chart.

The imbalance in language resource quantity also affects actual code generation performance, and clear performance differences have been confirmed between instructions in English and non-English languages. In a study by Wang et al. (2024), a comparison of code generation performance for instructions in English and Chinese showed that performance decreased by more than 13% for Chinese instructions. The study identified the lack of language resources for non-English languages as the primary factor behind this performance degradation.

2.3 Cross-Lingual Transfer

Cross-lingual transfer enables knowledge learned in one language to be utilized in another language (Wu and Dredze, 2019). It has attracted attention as a means to resolve language resource imbalances and cross-lingual performance gaps. In code generation, utilizing knowledge acquired in English enables efficient performance improvements even for instructions in low-resource languages.

The underlying principles and mechanisms of cross-lingual transfer remain unclear. The internal translation hypothesis suggests that when LLMs receive instructions in languages other than English, they internally translate the instructions into English and then generate responses using knowledge learned in English (Wendler et al., 2024; Schut et al., 2025). We focus on an instruction tuning method based on the internal translation hypothesis to promote cross-lingual transfer.

3 Proposed Method

In this section, we describe existing instruction tuning methods and our proposed method, CL-CoT. Figure 2 shows an overview of the proposed method using Japanese as an example.

3.1 Instruction Tuning

Instruction tuning is supervised fine-tuning using paired data consisting of instruction texts and corresponding outputs, enabling LLMs to follow given instructions (Wei et al., 2021).

A key characteristic is that effective improvements can be achieved with far less training data compared to pre-training (Zhou et al., 2023).

Below is an example of instruction tuning data in Japanese:

Instruction

2つの数の乗算を計算する関数をかきなさい.

Output

```
def calculate_multiple(a, b):
    return a * b
```

Multilingual LLMs are created by composing instruction texts in target languages (Wang et al., 2024) and applying tuning. However, adding instruction texts in multiple languages can cause catastrophic forgetting (Fujii et al., 2024), leading to performance degradation in certain languages.

3.2 Para-lingual SFT

Sato et al. proposed Para-lingual supervised fine-tuning with parallel instruction texts in high-resource and low-resource languages to promote cross-lingual transfer (Sato et al., 2025). We refer to this approach as Para SFT. For convenience, we refer to the conventional monolingual SFT as Mono SFT.

Below is an example of Para SFT when Japanese is the low-resource language:

Instruction

2つの数の乗算を計算する関数をかきなさい.
Write a function that calculates the multiplication of two numbers.

Output

```
def calculate_multiple(a, b):
    return a * b
```

Para SFT is based on the internal translation hypothesis. By providing instructions in the order of low-resource language followed by English, Para SFT implicitly encourages translation from the low-resource language to English, and further expects code generation utilizing English knowledge.

3.3 Cross-Lingual Chain-of-Thought

Para SFT implicitly encourages switching from low-resource languages to English through parallel translation. In contrast, we propose Cross-Lingual Chain-of-Thought (CL-CoT), which explicitly promotes cross-lingual transfer by adding Chain-of-Thought(CoT) instruction text that induces reasoning.

The CoT instruction text is a cross-lingual transfer version of the CoT prompt ("Let's think step by step"): "Let's think the instruction in English."

Below is an example of CL-CoT for a low-resource language. The part enclosed by `<reason>` `</reason>` tags is the CoT instruction text.

Instruction

2つの数の乗算を計算する関数をかきなさい.
`<reason>`
Let's think the instruction in English.
`</reason>`

Output

```
def calculate_multiple(a, b):
    return a * b
```

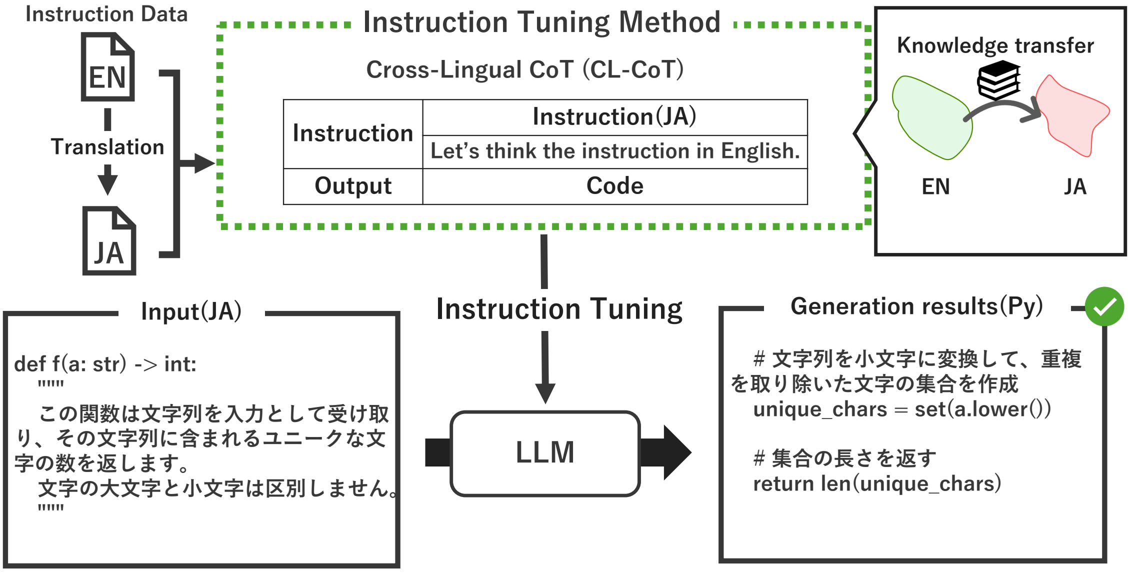
4 Experiments

This section reports on the effectiveness of the CL-CoT proposed in the previous section. The purpose of this experiment is to verify whether CL-CoT is effective in promoting cross-lingual transfer and improving code generation performance in low-resource languages.

4.1 Experimental Settings

4.1.1 Target Languages

We selected Japanese, Vietnamese, and Korean as target low-resource languages. According to the analysis of The Stack pre-training corpus by Kocetkov et al. (2022), natural-language descriptions



```
def f(a: str) -> int:
    """
    この関数は文字列を入力として受け取り、その文字列に含まれるユニークな文字の数を返します。
    文字の大文字と小文字は区別しません。
    """
```

```
    # 文字列を小文字に変換して、重複を取り除いた文字の集合を作成
    unique_chars = set(a.lower())

    # 集合の長さを返す
    return len(unique_chars)
```

Figure 2: Overview of Cross-Lingual Chain-of-Thought (CL-CoT) instruction tuning method with Japanese input example.

account for less than 1 % in Japanese and Korean, while Vietnamese is not included at all. These languages are therefore well suited to evaluating the effectiveness of instruction tuning in low-resource settings.

4.1.2 Instruction Tuning Methods for Comparison

We investigated the instruction tuning methods described in Section3.

1. **Mono SFT**: Standard instruction tuning method that uses only instruction texts in the low-resource language.

2. **Para SFT**: A method where parallel instruction texts are arranged in the order of low-resource language followed by English.

3. **CL-CoT**: A method where a CoT prompt that encourages reasoning in English is added after the instruction text in the low-resource language.

4.1.3 Instruction Tuning Dataset

We created our dataset using the educational_instruct dataset[1] from the OpenCoder project:

1. From the 118,278 samples in the educational_instruct dataset, extract the top 1,000 English instruction–code pairs, sorted in descending order of instruction length.

2. For the above English instruction texts, we machine-translated them into low-resource languages using the DeepL API[2]. The code was used as-is without translation.

High-quality machine translation is crucial for multilingual LLM development.

Following Zhou et al. (2023), we adopt a sample size of 1,000 pairs, which has been shown effective for instruction tuning experiments.

We compared several translation services by translating a subset of instruction data and checking for errors and omissions. Based on translation fidelity and proven performance in previous multilingual LLM projects, we selected DeepL.

4.1.4 Target LLMs for Evaluation

We evaluated three LLMs with different pre-training data compositions:

- **meta-llama/Meta-Llama-3-8B**: A model developed by Meta, pre-trained on over 15 trillion tokens of data primarily in English.

- **Qwen/Qwen2.5-7B**: A model developed by Alibaba that supports multilingual generation. This model is pre-trained on 18 trillion tokens

[1] OpenCoder-LLM/opc-sft-stage2

[2] https://www.deepl.com

of multilingual data, and the officially supported languages include English, Japanese, Vietnamese, and Korean.

- **infly/OpenCoder-8B-Base**: A code-specialized model jointly developed by INF Technology and M-A-P. This model is pre-trained on 2.5 trillion tokens with a composition ratio of 9:1 between code and code-related web data.

We performed instruction tuning on these models using the entire English dataset from educational_instruct to create baseline models. This baseline served to improve English code generation performance under the same conditions and subsequently examine cross-lingual transfer capabilities.

4.1.5 Evaluation Method

We evaluated code generation performance for low-resource languages using CL-HumanEval (Sato et al., 2024), which is based on HumanEval (Chen et al., 2021). The benchmark includes English, Japanese, Vietnamese, and Korean, with 164 problems provided for each language. CL-HumanEval differs from the original HumanEval in two main ways. First, CL-HumanEval replaces English-derived function and variable names with language-independent identifiers. Second, this benchmark removes hints such as code execution examples. We used Pass@1 as the evaluation metric. Similar to HumanEval, this metric indicates whether the model can generate correct code when generating a single code sample.

4.2 Experimental results

Table 1 reports Pass@1 scores for the three instruction tuning methods across three LLMs. The highest score for each model–language pair is highlighted in bold. CL-CoT achieved the highest scores in most combinations, outperforming the baseline in seven out of nine combinations of three models and three low-resource languages. Our results show that the proposed method, CL-CoT, improves code generation performance for low-resource languages.

In contrast, Para SFT underperformed the baseline in four of the nine model–language combinations. Furthermore, CL-CoT matched or outperformed Mono SFT in six of the nine model–language combinations.

These results suggest that instruction tuning methods that explicitly encourages reasoning in English are effective in promoting cross-lingual transfer.

4.2.1 Performance Changes by Instruction Tuning Data Size

Based on the previous section's results, we investigated how the size of instruction tuning data affects performance. Figure 3 shows the evaluation results for Japanese with data sizes of 1,000, 5,000, and 10,000 samples. The results indicate that increasing the data size from 1,000 to 10,000 samples yields no significant performance improvement. Based on these findings, we conclude that approximately 1,000 instruction tuning samples are sufficient to achieve the observed performance gains.

4.2.2 Proportion of Comments in Target Languages

CL-CoT is an instruction tuning method that encourages reasoning in English. Therefore, even when instructions are given in low-resource languages, there is a possibility that comments in the generated code may be written in English. To address this concern, we conducted an additional analysis of the proportion of comments in the generated code that were written in the same language as that used in the instruction. Table 2 shows the proportion of problems where comments in the generated code were written in the instruction language out of all 164 problems in CL-HumanEval. Bold values represent the highest values for each model.

From the results in Table 2, CL-CoT showed a higher proportion of comment generation in the instruction language than the baseline in seven out of nine combinations of three models and three low-resource languages. This result demonstrates that while CL-CoT promotes reasoning in English, it can appropriately maintain the instruction language in the output. This suggests that promoting cross-lingual transfer and maintaining output language consistency can be compatible, which is a result that supports the effectiveness of CL-CoT.

5 Related Work

The work most closely related to ours is xCoT, proposed by Chai et al. (2024) for mathematical-reasoning tasks. xCoT uses cross-lingual CoT prompting during inference by adding the instruction "Let's think the question in Language and then think step by step in English" to the input, which encourages LLMs to handle questions in low-resource languages while reasoning and producing answers

Table 1: Evaluation results on CL-HumanEval using the Pass@1 metric. The table compares three LLMs with three instruction tuning methods (Mono SFT, Para SFT, and CL-CoT). The highest score for each model–language pair is highlighted in bold.

model	instruction tuning method	EN	JA	VI	KR
Llama3-8B	**Baseline**	50.6	39.6	42.1	43.3
	Mono SFT	-	**45.7**	43.9	**45.7**
	Para SFT	-	38.4	43.9	36
	CL-CoT	-	43.9	**45.1**	45.1
Qwen2.5-7B	**Baseline**	61	54.3	51.2	48.2
	Mono SFT	-	54.3	**54.9**	**52.4**
	Para SFT	-	51.8	51.2	49.4
	CL-CoT	-	**54.9**	**54.9**	**52.4**
OpenCoder-8B	**Baseline**	59.8	41.5	**40.9**	43.9
	Mono SFT	-	**45.1**	39.6	42.7
	Para SFT	-	43.9	37.2	**44.5**
	CL-CoT	-	43.3	39.6	43.9

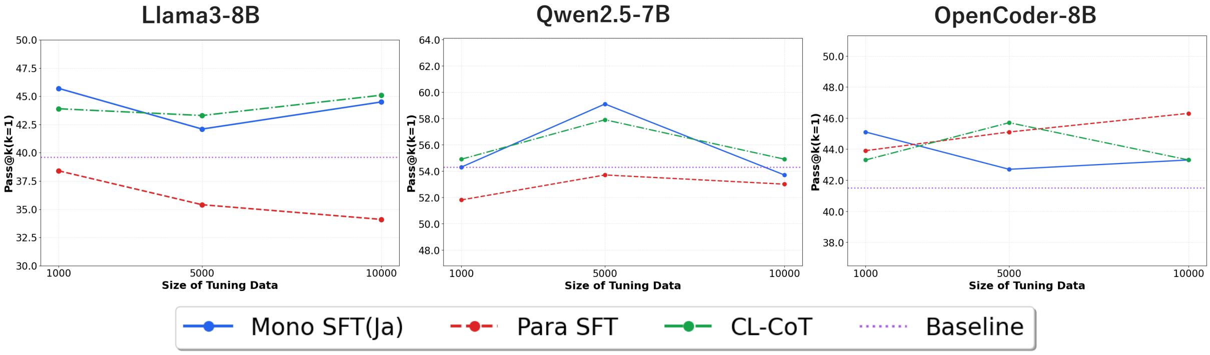


Figure 3: Changes in Pass@1 performance when the size of instruction tuning data is increased.

in English. In contrast, our method is based on instruction tuning that explicitly appends such CoT prompts during tuning while preserving the instruction language in the output for cross-lingual code generation.

In recent years, various methods have been proposed to promote cross-lingual transfer. In the following, we organize related work from three perspectives: approaches other than instruction tuning, instruction tuning-based approaches, and prompting methods.

5.1 Methods for promoting cross-lingual transfer other than instruction tuning

We first describe methods other than instruction tuning for promoting cross-lingual transfer. Representative methods include continual pre-training and model merging.

Continual pre-training refers to further training a pre-trained LLM on data specific to a target language or task, enabling the model to adapt to that domain (Fujii et al., 2024). Although this approach improves comprehension in the target language, it requires substantially more data than instruction tuning to achieve comparable gains.

Model merging is a method that integrates parameters from multiple models with different capabilities (Yang et al., 2024). This method can improve performance without requiring additional training data or large-scale computation, and its effectiveness in Japanese code generation has also been reported (Nakano et al., 2025). However, there is a constraint that integration is difficult between models with different architectures.

Our instruction tuning method has the advantage of being applicable to existing models with a small amount of data compared to these methods.

5.2 Instruction tuning methods promoting cross-lingual transfer

We describe instruction tuning methods that promote cross-lingual transfer.

Table 2: Percentage of benchmark problems in which comments in the generated code were written in the instruction language (low-resource language). The highest value for each model–language pair is highlighted in bold.

Model	Instruction tuning method	JA	VI	KR
Llama3-8B	**Baseline**	**3.7**	1.8	**5.5**
	Mono SFT	0	0	0.6
	Para SFT	0.6	1.2	0
	CL-CoT	1.8	**4.3**	0
Qwen2.5-7B	**Baseline**	26.8	39.6	27.4
	Mono SFT	23.1	33.5	47
	Para SFT	40.2	46.3	40.9
	CL-CoT	**47.5**	**54.3**	**64.6**
OpenCoder-8B	**Baseline**	74.3	93.3	88.4
	Mono SFT	**92.7**	**97**	94.5
	Para SFT	89.6	95.7	95.7
	CL-CoT	92	96.3	**95.7**

Chen et al. (2023) compared monolingual and multilingual instruction tuning. They created multilingual data using the Alpaca dataset and its machine-translated versions, demonstrating that multilingual tuning achieves performance equal to or better than individual language-specific tuning under the same computational constraints.

Shaham et al. (2024) investigated the effects of small amounts of multilingual data. They showed that adding just 40 multilingual examples to an English tuning set significantly improves multilingual instruction-following capabilities.

Research focusing on language mixing within instructions includes Yoo et al. (2024), who proposed dividing instruction text into multiple languages at the sentence or word level. Ranaldi et al. (2024) proposed CrossAlpaca, which improves cross-lingual semantic consistency through demonstrations combining cross-lingual instruction following and translation following.

Compared to these approaches, our method is distinguished by incorporating prompts that encourage English reasoning within low-resource language instructions.

5.3 Promoting cross-lingual transfer using CoT

Many methods for promoting cross-lingual transfer using CoT have also been proposed.

Shi et al. (2022) demonstrated that providing LLMs with prompts for step-by-step reasoning in English achieves high performance regardless of the language of the problem statement.

Qin et al. (2023) proposed Cross-Lingual Prompting (CLP). This method employs a two-stage approach that first aligns cross-lingual understanding and then performs task-specific reasoning, thereby improving multilingual reasoning performance.

Our method differs from these prompting approaches by incorporating CoT elements that encourage English reasoning during instruction tuning.

6 Conclusion

This study aims to narrow code generation performance gaps for low-resource languages by leveraging cross-lingual transfer. Imbalanced language resources in LLM pre-training create performance disparities across languages. Based on the internal translation hypothesis, we propose CL-CoT, which adds English reasoning prompts to low-resource language instructions.

Evaluation results confirmed that the proposed method can promote cross-lingual transfer using minimal data and improve code generation performance in low-resource languages. In particular, evaluation in three languages—Japanese, Vietnamese, and Korean—demonstrated the versatility of CL-CoT. Additionally, the elements in CL-CoT that encourage reasoning in English were found to function effectively for code generation from low-resource languages.

These findings provide valuable insights for developing LLMs aimed at enhancing code generation performance in non-English speaking regions and for designing instruction tuning methods that promote cross-lingual transfer.

Future work will focus on improving performance in multilingual code generation tasks through collecting extensive data in other languages and refining the CL-CoT method. Furthermore, we plan to extend the insights gained from our proposed method to other tasks and validate the generalizability of our approach.

References

Linzheng Chai, Jian Yang, Tao Sun, Hongcheng Guo, Jiaheng Liu, Bing Wang, Xiannian Liang, Jiaqi Bai, Tongliang Li, Qiyao Peng, and 1 others. 2024. xcot: Cross-lingual instruction tuning for cross-lingual chain-of-thought reasoning. *arXiv preprint arXiv:2401.07037*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Pinzhen Chen, Shaoxiong Ji, Nikolay Bogoychev, Andrey Kutuzov, Barry Haddow, and Kenneth Heafield. 2023. Monolingual or multilingual instruction tuning: Which makes a better alpaca. *arXiv preprint arXiv:2309.08958*.

Kazuki Fujii, Taishi Nakamura, Mengsay Loem, Daiki Iida, Seiya Ohi, Kakeru Hattori, Shota Hirai, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. 2024. Building a japanese-centric large language model via continued pre-training. In *Proceedings of the 30th Annual Meeting of the Association for Natural Language Processing (NLP)*. The Association for Natural Language Processing (NLP). This work is licensed under CC BY 4.0.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2024. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, and 1 others. 2022. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*.

Mingda Li, Abhijit Mishra, and Utkarsh Mujumdar. 2024. Bridging the language gap: Enhancing multilingual prompt-based code generation in llms via zero-shot cross-lingual transfer. *arXiv preprint arXiv:2408.09701*.

Zihao Li, Yucheng Shi, Zirui Liu, Fan Yang, Ali Payani, Ninghao Liu, and Mengnan Du. 2025. Language ranker: A metric for quantifying llm performance across high and low-resource languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 28186–28194.

Noriko Nakano, Yuha Nishigata, Waka Ito, Nao Soma, Miyu Sato, and Kimio Kuramitsu. 2025. Enhancing cross-lingual transfer in code generation capabilities through model merging. In *Proceedings of the Annual Conference of the Japanese Society for Artificial Intelligence*.

Elise Paradis, Kate Grey, Quinn Madison, Daye Nam, Andrew Macvean, Vahid Meimand, Nan Zhang, Ben Ferrari-Church, and Satish Chandra. 2024. How much does ai impact development speed? an enterprise-based randomized controlled trial. *arXiv preprint arXiv:2410.12944*.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. 2025. Fineweb2: One pipeline to scale them all–adapting pre-training data processing to every language. *arXiv preprint arXiv:2506.20920*.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. *arXiv preprint arXiv:2310.14799*.

Leonardo Ranaldi, Giulia Pucci, and Andre Freitas. 2024. Empowering cross-lingual abilities of instruction-tuned large language models by translation-following demonstrations. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7961–7973, Bangkok, Thailand. Association for Computational Linguistics.

Miyu Sato, Yuha Nishigata, Yuka Akinobu, Toshiyuki Kurabayashi, and Kimio Kuramitsu. 2025. Does instruction tuning with parallel structure promote cross-lingual transfer? In *Proceedings of the Thirty-First Annual Meeting of the Association for Natural Language Processing*. The Association for Natural Language Processing.

Miyu Sato, Yui Obara, Nao Souma, and Kimio Kuramitsu. 2024. Cl-humaneval: A benchmark for evaluating cross-lingual transfer though code generation. In *Proceedings of the 38th Pacific Asia Conference on Language, Information and Computation*, pages 656–664.

Lisa Schut, Yarin Gal, and Sebastian Farquhar. 2025. Do multilingual llms think in english? *arXiv preprint arXiv:2502.15603*.

Uri Shaham, Jonathan Herzig, Roee Aharoni, Idan Szpektor, Reut Tsarfaty, and Matan Eyal. 2024. Multilingual instruction tuning with just a pinch of multilinguality. *arXiv preprint arXiv:2401.01854*.

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, and 1 others. 2022. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*.

Chaozheng Wang, Zongjie Li, Cuiyun Gao, Wenxuan Wang, Ting Peng, Hailiang Huang, Yuetang Deng, Shuai Wang, and Michael R Lyu. 2024. Exploring multi-lingual bias of large code models in code generation. *arXiv preprint arXiv:2404.19368*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. 2024. Do llamas work in english? on the latent language of multilingual transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15366–15394.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of bert. *arXiv preprint arXiv:1904.09077*.

Yuemei Xu, Ling Hu, Jiayi Zhao, Zihan Qiu, Kexin Xu, Yuqi Ye, and Hanwen Gu. 2025. A survey on multilingual large language models: Corpora, alignment, and bias. *Frontiers of Computer Science*, 19(11):1911362.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*.

Haneul Yoo, Cheonbok Park, Sangdoo Yun, Alice Oh, and Hwaran Lee. 2024. Code-switching curriculum learning for multilingual transfer in llms. *arXiv preprint arXiv:2411.02460*.

Hongbin Zhang, Kehai Chen, Xuefeng Bai, Yang Xiang, and Min Zhang. 2024. Lingualift: An effective two-stage instruction tuning framework for low-resource language reasoning. *arXiv preprint arXiv:2412.12499*.

Jun Zhao, Zhihao Zhang, Luhui Gao, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. Llama beyond english: An empirical study on language capability transfer. *arXiv preprint arXiv:2401.01055*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36:55006–55021.