At Which Training Stage Does Code Data Help LLMs Reasoning?

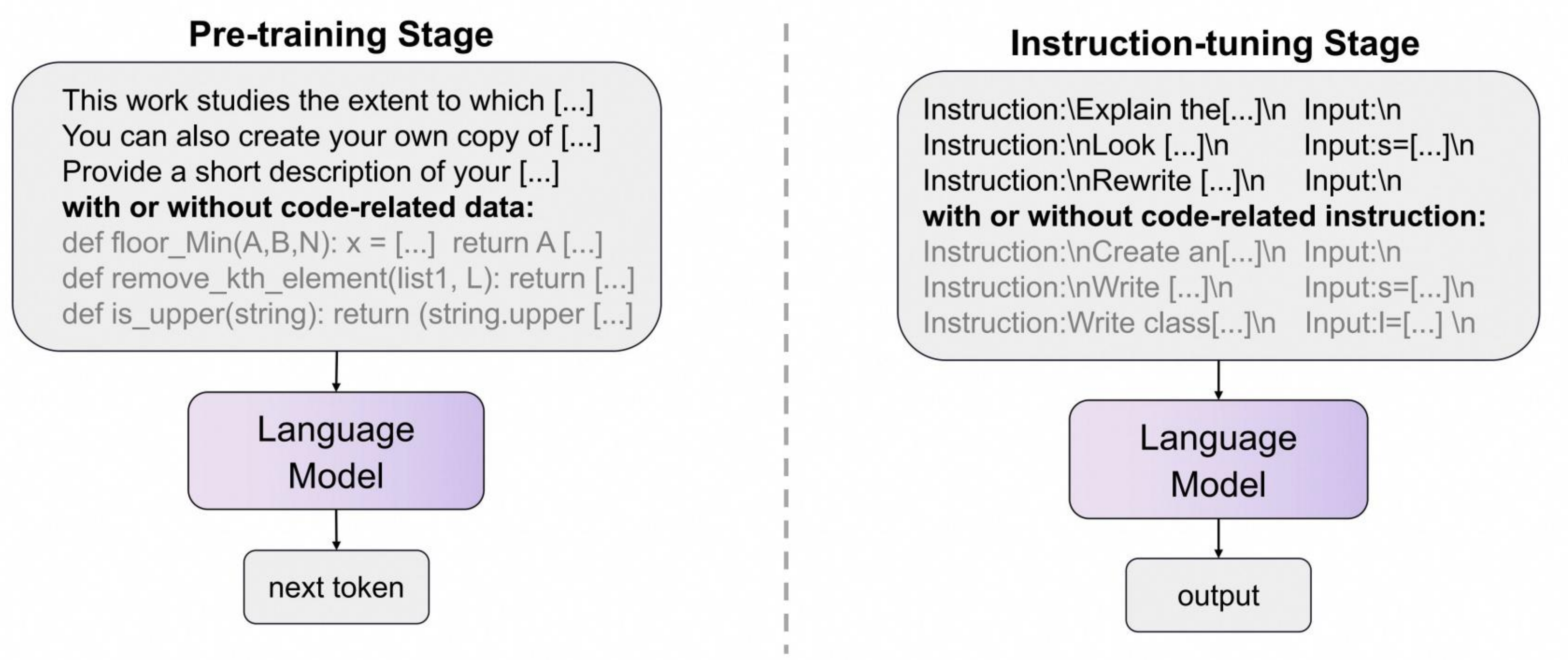
Yingwei Ma* , Yue Liu* , Yue Yu , Yuanliang Zhang , Yu Jiang , Changjian Wang , Shanshan Li

Abstract

Large Language Models (LLMs) have exhibited remarkable reasoning capabilities. Inspired by the great success of code data in training LLMs, we naturally wonder at which training stage introducing code data can help LLMs reasoning. To this end, this paper systematically explores the impact of code data on LLMs at different stages.

Concretely, we introduce the code data at the pre-training stage, instruction-tuning stage, and both of them, respectively. Then, the reasoning capability of LLMs is comprehensively and fairly evaluated via six reasoning tasks in five domains. We analyze the experimental results and provide conclusions with insights. First, pre-training LLMs with the mixture of code and text can enhance LLMs' general reasoning capability almost without negative transfer on other tasks. Besides, at the instruction-tuning stage, code data endows LLMs the task-specific reasoning capability. Moreover, the dynamic mixing strategy of code and text data assists LLMs to learn reasoning capability step-by-step during training.

Introduction


Pre-training Stage
This work studies the extent to which [...]
You can also create your own copy of [...]
Provide a short description of your [...]
with or without code-related data:
def floor_Min(A,B,N): x = [...] return A [...]
def remove_kth_element(list1, L): return [...]
def is_upper(string): return (string.upper [...]
Language Model
next token
Instruction-tuning Stage
Instruction:\Explain the[...]\n Input:\n
Instruction:\nLook [...]\n Input:s=[...]\n
Instruction:\nRewrite [...]\n Input:\n
with or without code-related instruction:
Instruction:\nCreate an[...]\n Input:\n
Instruction:\nWrite [...]\n Input:s=[...]\n
Instruction:Write class[...]\n Input:l=[...] \n
Language Model
output

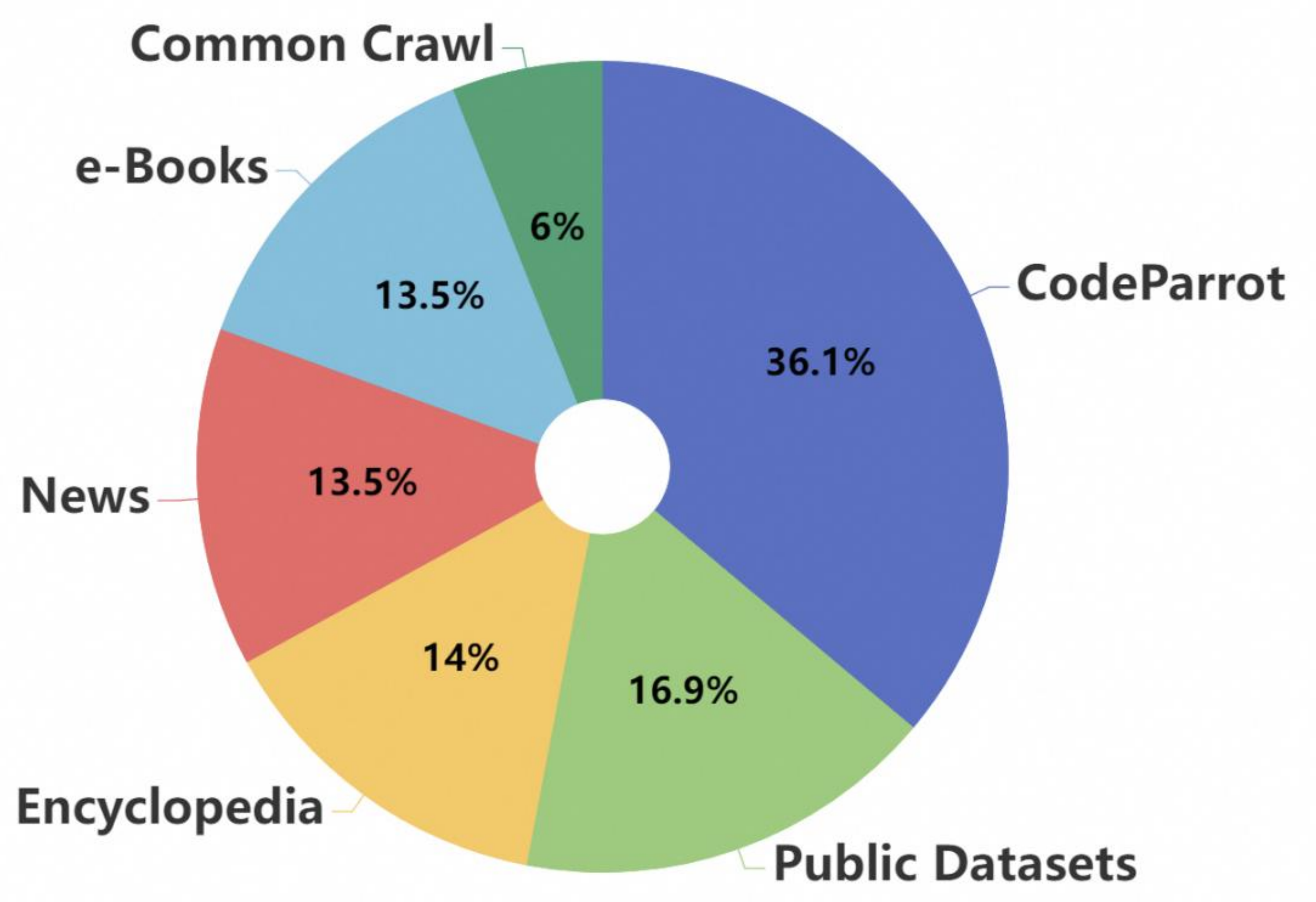


Figure 1: Demonstration of the pre-training and tuning phase.

Problem. We mainly discuss the impact of code data on model reasoning capabilities in the pre-training node and fine-tuning stages.

Evaluation. we evaluate LLMs on six tasks in five domains, including logical reasoning, code reasoning, legal reasoning, scientific reasoning, and analogical reasoning.

Result. Pre-training LLMs with the mixture of code and text can enhance LLMs' general reasoning capability almost without negative transfer on other tasks.

Conclusion

We point out that simply adding code data in the pre-training phase can effectively improve the general reasoning ability of the model. Furthermore, we find that adding code instructions in the instruction tuning stage can make the model follow human instructions for output and improve specific code reasoning capabilities.

Contact

Code: https://github.com/yingweima2022/codellm
Email: yingwei.ywma@gmail.com

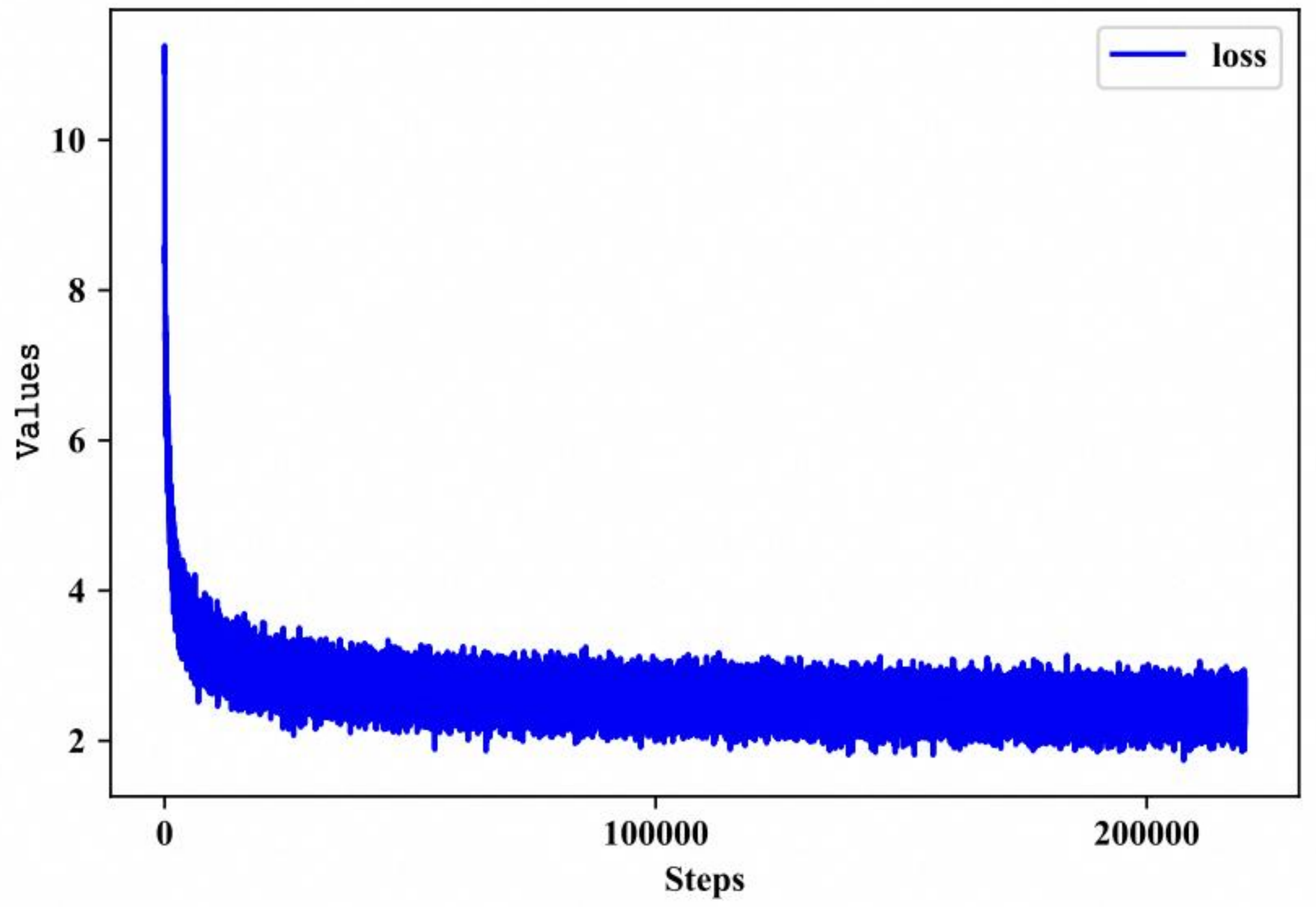
Experiments

Dataset	Task	Metric	NL (2.6B)	NL (13B)	CODE (2.6B)	p-value
Logic*	Logical Reasoning	ACC	36.36	**45.45**	40.90	4.197e-06
JEC-QA*	Legal QA	ACC	27.00	27.00	**28.70**	1.956e-25
ScienceQA*	Scientific QA	ACC	45.93	45.18	**46.06**	0.014
E-KAR*	Analogical Reasoning	ACC	32.24	35.52	**36.12**	7.013e-07
CosQA†	Code QA	ACC	47.02	46.85	**50.50**	1.066e-40
MBPP†	Code Generation	BLEU	0.52	1.34	**5.06**	-

Table 2: Results on pre-training stage. Bold values indicate the best performance. * denote the general reasoning task, and † denote the code-related reasoning task.

Dataset	NN (2.6B)	NC (2.6B)	NN (13B)	NC (13B)	CC (2.6B)
Logic*	36.36	**40.90**	**40.90**	**40.90**	**40.90**
JEC-QA*	25.20	26.10	24.50	26.40	**27.10**
ScienceQA*	**44.45**	43.44	42.94	43.41	41.90
E-KAR*	**30.45**	28.66	26.27	27.46	27.20
CosQA†	45.20	48.18	47.52	51.99	**52.48**
MBPP†	0.00	5.61	0.00	1.88	**24.88**

Table 3: Results on instruction-tuning stage. Bold values indicate the best performance. * denote general reasoning task, and † denote the code-related reasoning task.


loss
Values
Steps