MapCoder: Multi-Agent Code Generation for Competitive Problem Solving

Md. Ashraful Islam, et al

Bangladesh University of Engineering and Technology

May 18, 2024

Abstract

- LLMs have limited ability for coding
  - Code generation is harder than just NLP

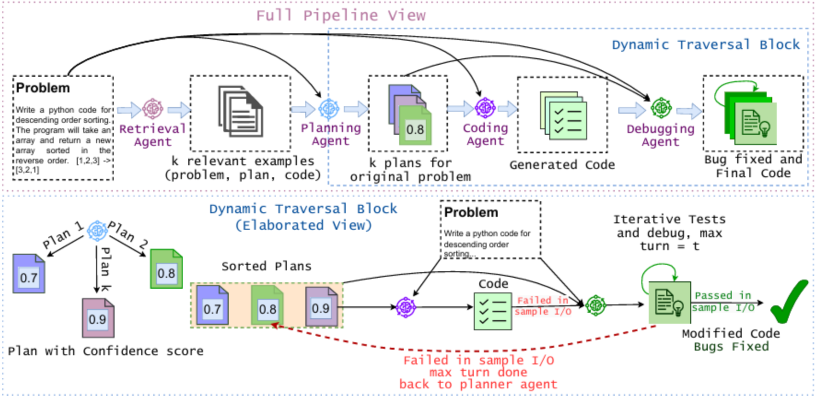
- New framework MapCoder, which has 4 stages
  - Retrieval, Planning, Coding, Debugging
  - It is open source

Previous works

- Chain-of-Thought
  - Pseudo code-based generation
  - Fail to pass test cases, No bug fixing
- Retrieval-based approach
  - Leverage relevant problems and solutions
  - Fail to pass test cases, No bug fixing
- Self-reflection
  - Iteratively evaluates generated code against test case
  - Only leverage the problem description itself in a zero-shot manner

Multi Agent Prompting Coder

- Retrieval agent
  - Generates relevant examples itself
- Dynamic Traversal Block
  - Plans and considers the confidence of the generated plans
  - Code generation and debugging

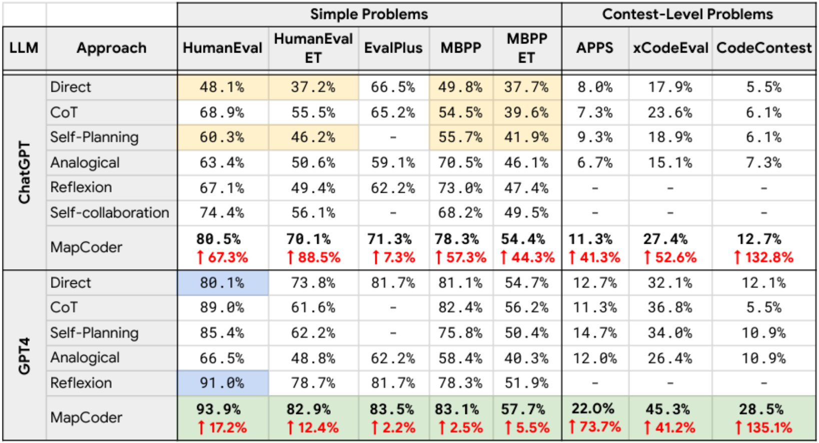


Write a python code for descending order sorting. The program will take an array and return a new array sorted in the reverse order. [1,2,3] -> [3,2,1]

Write a python code for descending order sorting...

Experiment

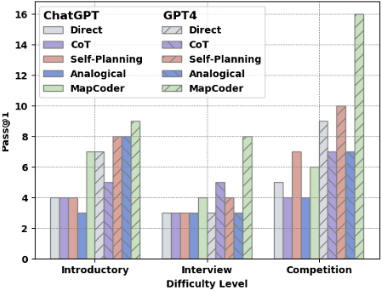
- 5 basic-level benchmark + 3 contest-level benchmark
- GPT-3.5-Turbo and GPT-4 as foundation models
- Various prompting baselines including MapCoder

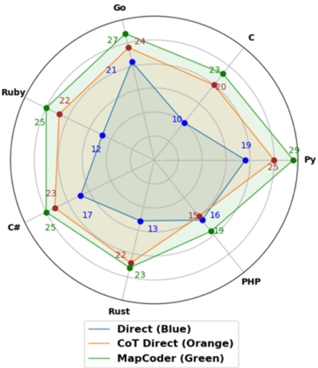


		Simple Problems					Contest-Level Problems		
LLM	Approach	HumanEval	HumanEval ET	EvalPlus	MBPP	MBPP ET	APPS	xCodeEval	CodeContest
ChatGPT	Direct	48.1%	37.2%	66.5%	49.8%	37.7%	8.0%	17.9%	5.5%
	CoT	68.9%	55.5%	65.2%	54.5%	39.6%	7.3%	23.6%	6.1%
	Self-Planning	60.3%	46.2%	-	55.7%	41.9%	9.3%	18.9%	6.1%
	Analogical	63.4%	50.6%	59.1%	70.5%	46.1%	6.7%	15.1%	7.3%
	Reflexion	67.1%	49.4%	62.2%	73.0%	47.4%	-	-	-
	Self-collaboration	74.4%	56.1%	-	68.2%	49.5%	-	-	-
	MapCoder	**80.5%** ↑ 67.3%	**70.1%** ↑ 88.5%	**71.3%** ↑ 7.3%	**78.3%** ↑ 57.3%	**54.4%** ↑ 44.3%	**11.3%** ↑ 41.3%	**27.4%** ↑ 52.6%	**12.7%** ↑ 132.8%
GPT4	Direct	80.1%	73.8%	81.7%	81.1%	54.7%	12.7%	32.1%	12.1%
	CoT	89.0%	61.6%	-	82.4%	56.2%	11.3%	36.8%	5.5%
	Self-Planning	85.4%	62.2%	-	75.8%	50.4%	14.7%	34.0%	10.9%
	Analogical	66.5%	48.8%	62.2%	58.4%	40.3%	12.0%	26.4%	10.9%
	Reflexion	91.0%	78.7%	81.7%	78.3%	51.9%	-	-	-
	MapCoder	**93.9%** ↑ 17.2%	**82.9%** ↑ 12.4%	**83.5%** ↑ 2.2%	**83.1%** ↑ 2.5%	**57.7%** ↑ 5.5%	**22.0%** ↑ 73.7%	**45.3%** ↑ 41.2%	**28.5%** ↑ 135.1%

Experiment Results

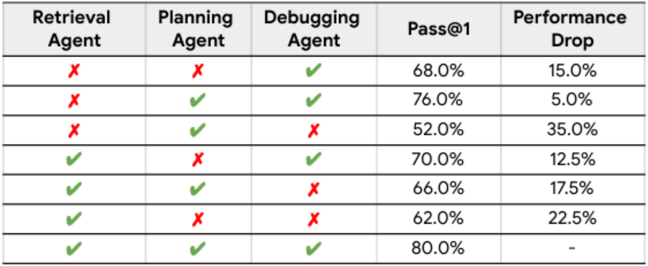
- Performance gain on varying difficulty levels
- Performance gain on different programming languages





Ablation study

- Removed each agent
  - Showed that every agent has its role in the pipeline
  - Debugging Agent has the most significant impact



Retrieval Agent	Planning Agent	Debugging Agent	Pass@1	Performance Drop
✗	✗	✔	68.0%	15.0%
✗	✔	✔	76.0%	5.0%
✗	✔	✗	52.0%	35.0%
✔	✗	✔	70.0%	12.5%
✔	✔	✗	66.0%	17.5%
✔	✗	✗	62.0%	22.5%
✔	✔	✔	80.0%	-

Conclusion

- MapCoder outperformed many SOTA approaches
  - Performance gain on varying difficulty levels
  - Performance gain on different programming languages

- Limitation
  - It generates a large number of tokens
  - Challenging in resource-constrained environment