MATHSENSEI: A Tool-Augmented Large Language Model for Mathematical Reasoning

实验分析

韩子坚

华中师范大学计算机学院

2024 年 12 月 27 日

Content

1 实验概览

2 分析 wa

1 实验概览
自己跑的实验 - 闭源模型
自己跑的实验 - 开源模型
2 分析 wa

论文中的实验结果

Method	Alg	P.Cal	P.Alg	Geom	Prob	N.Th	Int.Alg	O.Acc
Baselines with `gpt-3.5-turbo`								
CoT-LTP (Guo et al., 2023)	49.6	16.3	52.3	22.5	30.2	29.8	16.9	31.1
ComplexCoT (Fu et al., 2023)	49.1	16.8	53.8	22.3	29.7	33.4	14.6	34.1
ComplexCoT+PHP (Zheng et al., 2023)	51.1	16.1	57.7	25.4	33.7	35.1	17.1	36.5
SKiC (Chen et al., 2023a)	57.9	23.0	62.0	30.1	38.2	35.5	17.8	40.6
Baselines with `GPT-4`								
CoT (Zhou et al., 2024)	70.8	26.7	71.6	36.5	53.1	49.6	23.4	50.4
PHP (Zhou et al., 2024)	74.3	29.8	73.8	41.9	56.3	55.7	26.3	53.9
Ours								
SG (🟢)	46.7	18.1	55.7	25.3	32.9	30.2	16.2	34.5
KR+SG (🟪+🟢)	49.1	15.0	58.0	24.4	34.3	29.6	12.0	34.4
BS+SG (⚫+🟢)	51.6	20.1	63.3	27.1	36.1	39.6	16.3	38.7
PG+SG (🔵+🟢)	60.0	26.5	66.1	30.7	42.1	40.5	21.1	44.6
PG+CR+SG (🔵+🟫+🟢)	59.7	25.2	63.9	26.9	48.3	43.0	26.9	44.8
PG′[🔵]+SG (🔵+🟢)	55.4	23.5	58.0	22.9	32.7	42.2	17.9	39.6
WA+SG (🔴+🟢)	57.8	26.1	58.5	26.3	37.6	37.8	31.5	42.6
PG+BS+SG (🔵+⚫+🟢)	53.1	20.7	58.7	28.6	37.8	36.6	19.9	39.0
BS+PG+SG (⚫+🔵+🟢)	55.0	23.1	61.2	27.5	35.4	35.4	20.5	39.8
WA+PG+SG (🔴+🔵+🟢)	62.5	28.9	61.5	27.1	42.6	45.7	33.4	46.3
PG+WA+SG (🔵+🔴+🟢)	61.6	28.7	64.7	30.5	42.8	49.1	35.0	47.6
BS+WA+SG (⚫+🔴+🟢)	56.2	22.9	61.0	29.8	37.5	44.0	28.9	42.9
WA+BS+SG (🔴+⚫+🟢)	60.0	27.0	65.0	29.0	40.5	42.2	31.4	45.4
BS+PG+WA+SG (⚫+🔵+🔴+🟢)	60.2	26.4	65.0	31.3	44.7	48.7	31.6	46.7

图 1: 论文中的实验结果

1 实验概览
自己跑的实验 - 闭源模型
自己跑的实验 - 开源模型
2 分析 wa

4o-mini: 不使用 WA （只使用 sg）

wa : Wolfram Alpha sg : Solution Generation

ACC	Value
Accuracy Math data	65.0
Total No of examples	100
Accuracy by level	
Accuracy Level 3	70.0
Accuracy Level 1	87.5
Accuracy Level 4	68.0
Accuracy Level 2	94.11764705882352
Accuracy Level 5	36.666666666666664
Accuracy by type	
Accuracy Algebra	73.07692307692307
Accuracy Number Theory	81.81818181818183
Accuracy Counting & Probability	66.66666666666666
Accuracy Intermediate Algebra	60.0
Accuracy Prealgebra	75.0
Accuracy Precalculus	35.714285714285715
Accuracy Geometry	42.857142857142854
No of errors in output format	12

4o-mini: wa + sg（wa 最大尝试 3 次）

- 100 例样本中有 12 例 3 次 wa 尝试均失败.

ACC	Value
Accuracy Math data	73.0
Total No of examples	100
Accuracy by level	
Accuracy Level 3	85.0
Accuracy Level 2	94.11764705882352
Accuracy Level 5	53.333333333333336
Accuracy Level 1	87.5
Accuracy Level 4	68.0
Accuracy by type	
Accuracy Precalculus	50.0
Accuracy Algebra	73.07692307692307
Accuracy Number Theory	72.72727272727273
Accuracy Prealgebra	85.0
Accuracy Counting & Probability	83.33333333333334
Accuracy Intermediate Algebra	70.0
Accuracy Geometry	71.42857142857143
No of errors in output format	0

4o-mini: wa + sg（wa 最大尝试 10 次）

- 100 例样本中有 7 例 10 次 wa 尝试均失败.

ACC	Value
Accuracy Math data	69.0
Total No of examples	100
Accuracy by level	
Accuracy Level 5	46.666666666666664
Accuracy Level 3	90.0
Accuracy Level 4	64.0
Accuracy Level 2	82.35294117647058
Accuracy Level 1	87.5
Accuracy by type	
Accuracy Intermediate Algebra	50.0
Accuracy Counting & Probability	75.0
Accuracy Prealgebra	75.0
Accuracy Precalculus	50.0
Accuracy Algebra	73.07692307692307
Accuracy Geometry	71.42857142857143
Accuracy Number Theory	81.81818181818183
No of errors in output format	0

4o-mini: wa + sg（wa 最大尝试 20 次）

- 100 例样本中有 3 例 19 次 wa 尝试均失败.
- Increasing max attempts has almost no effect.

ACC	Value
Accuracy Math data	72.0
Total No of examples	100
Accuracy by level	
Accuracy Level 1	75.0
Accuracy Level 3	85.0
Accuracy Level 5	56.666666666666664
Accuracy Level 2	82.35294117647058
Accuracy Level 4	72.0
Accuracy by type	
Accuracy Prealgebra	85.0
Accuracy Counting & Probability	75.0
Accuracy Number Theory	81.81818181818183
Accuracy Algebra	80.76923076923077
Accuracy Intermediate Algebra	40.0
Accuracy Precalculus	42.857142857142854
Accuracy Geometry	85.71428571428571
No of errors in output format	0

1 实验概览
自己跑的实验 - 闭源模型
自己跑的实验 - 开源模型
2 分析 wa

qwen2.5:32b-instruct-fp16: 不使用 WA （只使用 sg ）

ACC	Value
Accuracy Math data	72.0
Total No of examples	100
Accuracy by level	
Accuracy Level 1	75.0
Accuracy Level 2	100.0
Accuracy Level 3	75.0
Accuracy Level 5	56.666666666666664
Accuracy Level 4	68.0
Accuracy by type	
Accuracy Algebra	69.23076923076923
Accuracy Intermediate Algebra	70.0
Accuracy Prealgebra	80.0
Accuracy Counting & Probability	91.66666666666666
Accuracy Number Theory	72.72727272727273
Accuracy Precalculus	42.857142857142854
Accuracy Geometry	85.71428571428571
No of errors in output format	0

qwen2.5:32b-instruct-fp16: wa + sg（wa 最大尝试 3 次）

- 100 例样本中有 11 例 3 次 wa 尝试均失败.

ACC	Value
Accuracy Math data	77.0
Total No of examples	100
Accuracy by level	
Accuracy Level 3	95.0
Accuracy Level 1	100.0
Accuracy Level 5	63.33333333333333
Accuracy Level 4	56.00000000000001
Accuracy Level 2	100.0
Accuracy by type	
Accuracy Number Theory	72.72727272727273
Accuracy Algebra	84.61538461538461
Accuracy Precalculus	71.42857142857143
Accuracy Prealgebra	85.0
Accuracy Geometry	85.71428571428571
Accuracy Counting & Probability	75.0
Accuracy Intermediate Algebra	50.0
No of errors in output format	0

1 实验概览

2 分析 wa

代码中处理 wa 存在的问题

- 代码中的 WA 处理方式存在问题，它没有对 WA 返回的结果进行有效处理。
- 大量无关信息被直接传递给大模型，可能会影响大模型的推理。

WA 返回信息示例

例如: 对于计算 $2 * 17 - 1$ 时，wa 的返回结果如下：

{'success': True, '@error': 'false', '@xml:space': 'preserve', '@numpods': '5', '@datatypes': 'Math', '@timedout': '', '@timedoutpods': '', '@timing': '0.422', {'@parsetiming': '0.154', '@parsetimedout': 'false', '@recalculate': '', '@id': 'MSP3881da7a1e87baf001a008057d4a999h4e17e77', '@host': 'https://www6b3.wolframalpha.com', '@server': '19', '@related': 'https://www6b3.wolframalpha.com/api/v1/relatedQueries.jsp?id=MSPa3891da7a1e87baf001a00803bfe5041455g9d9071799157805137f692', '@version': '2.6', '@inputstring': '2*17-1', 'pod': [{'@title': 'Input', '@scanner': 'Identity', '@id': 'Input', '@position': 100.0, '@error': 'false', '@numsubpods': 1, 'subpod': {'@title': '', 'img': {'@src': 'https://www6b3.wolframalpha.com/Calculate/MSP/MSP3981da7a1e87baf001a80083f2274i85g2gfdff?MSPStoreType=image/gif&s=19', '@alt': '2×17 - 1', '@title': '2×17 - 1', '@width': 62, '@height': 19, '@type': 'Default', '@themes': '1,2,3,4,5,6,7,8,9,10,11,12', '@colorinvertable': 'true', '@contenttype': 'image/gif'}, 'plaintext': '2×17 - 1'}, 'expressiontypes': {'@count': '1'}, 'expressiontype': {'@name': 'Default'}}}, {'@title': 'Result', '@scanner': 'Simplification', '@id': 'Result', '@position': 200.0, '@error': 'false', '@numsubpods': 1, '@primary': True, 'subpod': {'@title': '', 'img': {'@src': 'https://www6b3.wolframalpha.com/Calculate/MSP/MSP3911da7a1e87baf001a800021hd6c15493i894e?MSPStoreType=image/gif&s=19', '@alt': '33', '@title': '33', '@width': 16, '@height': 19, '@type': 'Default', '@themes': '1,2,3,4,5,6,7,8,9,10,11,12', '@colorinvertable': 'true', '@contenttype': 'image/gif'}, 'plaintext': '33'}, 'expressiontypes': {'@count': '1', 'expressiontype': {'@name': 'Default'}}}, {'@title': 'Number line', '@scanner': 'NumberLine', '@id': 'NumberLine', '@position': 300.0, '@error': 'false', '@numsubpods': 1, 'subpod': {'@title': '', 'img': {'@src': 'https://www6b3.wolframalpha.com/Calculate/MSP/MSP3921da7a1e87baf001a80002g28e722f95c4f3c?MSPStoreType=image/gif&s=19', '@alt': 'Number line', '@title': '', '@width': 330, '@height': 60, '@type': '2DMathPlot_1', '@themes': '1,2,3,4,5,6,7,8,9,10,11,12', '@colorinvertable': 'true', '@contenttype': 'image/gif'}, 'plaintext': None}, 'expressiontypes': {'@count': '1', 'expressiontype': {'@name': 'Default'}}}, {'@title': 'Number name', '@scanner': 'Integer', '@id': 'NumberName', '@position': 400.0, '@error': 'false', '@numsubpods': 1, 'subpod': {'@title': '', 'img': {'@src': 'https://www6b3.wolframalpha.com/Calculate/MSP/MSP3931da7a1e87baf001a800038e195gf9ded0h96?MSPStoreType=image/gif&s=19', '@alt': 'thirty-three', '@title': 'thirty-three', '@width': 74, '@height': 19, '@type': 'Default', '@themes': '1,2,3,4,5,6,7,8,9,10,11,12', '@colorinvertable': 'true', '@contenttype': 'image/gif'}, 'plaintext': 'thirty-three'}, 'expressiontypes': {'@count': '1', 'expressiontype': {'@name': 'Default'}}}, {'@title': 'Percent decrease', '@scanner': 'Numeric', '@id': 'PercentIncrease/decrease', '@position': 500.0, '@error': 'false', '@numsubpods': 1, 'subpod': {'@title': '', 'img': {'@src': 'https://www6b3.wolframalpha.com/Calculate/MSP/MSP3941da7a1e87baf001a8008108dhhhe2h84bbc2?MSPStoreType=image/gif&s=19', '@alt': '2 17 - 1 = 33 is 2.941% smaller than 2 17 = 34.', '@title': '2 17 - 1 = 33 is 2.941% smaller than 2 17 = 34.', '@width': 331, '@height': 19, '@type': 'Default', '@themes': '1,2,3,4,5,6,7,8,9,10,11,12', '@colorinvertable': 'true', '@contenttype': 'image/gif'}, 'plaintext': '2 17 - 1 = 33 is 2.941% smaller than 2 17 = 34.'}, 'expressiontypes': {'@count': '1', 'expressiontype': {'@name': 'Default'}}}}}

实际上，返回结果中只有 `@inputstring` 和 `'plaintext'` 是有用的信息。

调用 wa 的常见错误

- **生成的查询过于复杂**：不符合 wa 语法规范，无法获得答案。
  - 例如，对于查询语句：`Calculate the distances PQ, QR, PR; check which sides form a right angle; use ((1)/(2)) × base × height to find the area of △ PQR.`

调用 wa 的常见错误

- **生成的查询过于复杂**: 不符合 wa 语法规范，无法获得答案。
  - 例如，对于查询语句：`Calculate the distances PQ, QR, PR; check which sides form a right angle; use ((1)/(2)) × base × height to find the area of △ PQR.`

- **混杂自然语言**: 生成了正确的 wa 查询语句，但是混杂在自然语言中。
  - 例如，对于查询语句：`Find the minimum x greater than 10 such that PrimeQ[x + 7] && PrimeQ[x + 13] && PrimeQ[x + 15].`
  - wa 返回结果：`'@success': False, Did you mean: PrimeQ[x + 7] && PrimeQ[x + 13] && PrimeQ[x + 15] ?`
  - 如果按照 wa 的提示，把 `PrimeQ[x + 7] && PrimeQ[x + 13] && PrimeQ[x + 15]` 作为查询语句再次调用 wa，wa 是可以理解的，但这和我们的问题没有关联。
  - 如何把 `Find the minimum x greater than 10 such that PrimeQ[x + 7] && PrimeQ[x + 13] && PrimeQ[x + 15].` 转换为一个有效的 wa 查询语句？这似乎非常困难。

wa 的纠错方式

- 代码中有纠错的方式，如果返回结果中 `@success` 为 false，则再生成新的查询并调用 wa，最多尝试三次。

wa 的纠错方式

- 代码中有纠错的方式，如果返回结果中 `@success` 为 false，则再生成新的查询并调用 wa，最多尝试三次。
- 但是，三次都失败的概率并不低。
  - 对于 qwen2.5:32b-instruct-fp16，100 个测试样例有 11 个样例出现 3 次调用 wa 均失败的情况。
  - 对于 4o-mini，100 个测试样例有 12 个样例出现 3 次调用 wa 均失败的情况。

wa 的纠错方式

- 代码中有纠错的方式，如果返回结果中 `@success` 为 false，则再生成新的查询并调用 wa，最多尝试三次。
- 但是，三次都失败的概率并不低。
  - 对于 qwen2.5:32b-instruct-fp16，100 个测试样例有 11 个样例出现 3 次调用 wa 均失败的情况。
  - 对于 4o-mini，100 个测试样例有 12 个样例出现 3 次调用 wa 均失败的情况。
- 增加最大尝试次数对于提高正确率帮助很小，并且显著增加推理时间。

wa 的纠错方式

- 代码中有纠错的方式，如果返回结果中 `@success` 为 false，则再生成新的查询并调用 wa，最多尝试三次。
- 但是，三次都失败的概率并不低。
  - 对于 qwen2.5:32b-instruct-fp16，100 个测试样例有 11 个样例出现 3 次调用 wa 均失败的情况。
  - 对于 4o-mini，100 个测试样例有 12 个样例出现 3 次调用 wa 均失败的情况。
- 增加最大尝试次数对于提高正确率帮助很小，并且显著增加推理时间。
- 性能较差的模型还有其他问题，例如，使用 qwen2.5:7b-instruct-fp16，发现 7b 模型有很多次出现生成的内容没有 `Final Query:` 字符串，导致提取失败而多次尝试的情况（4omini 和 qwen2.5:32b-instruct-fp16 没有观察到），并且有四例十次最终没有生成 wa 查询语句的情况（不是生成了错误的查询，是 llm 生成的内容没有相应的关键词而无法提取出查询）

- 对于性能较差的模型来说，正则匹配或者关键词提取可能不是一个很好的方式，可以考虑结构化输出的方式。

- 对于性能较差的模型来说，正则匹配或者关键词提取可能不是一个很好的方式，可以考虑结构化输出的方式。
- 例如，查阅 qwen api 文档，发现结构化输出功能支持 qwen2.5 系列所有模型（除了 math 与 coder 模型）。

```python
from openai import OpenAI
import os

def get_response():
    client = OpenAI(
        api_key=os.getenv("DASHSCOPE_API_KEY"), # 如果您没有配置环境变量，请在此处用您的API Key进行替换
        base_url="https://dashscope.aliyuncs.com/compatible-mode/v1",  # 填写DashScope服务的base_url
    )
    completion = client.chat.completions.create(
        model="qwen2.5-72b-instruct",
        messages=[
            {'role': 'system', 'content': 'You are a helpful assistant.'},
            {'role': 'user', 'content': '请用json格式输出一个学生的信息，姓名是张三，学号是12345678"'}],
        response_format={
            "type": "json_object"
        }
        )
    print(completion.model_dump_json())


if __name__ == '__main__':
    get_response()
```

图 2: qwen api 文档结构化输出示例请求

```json
{
    "id": "chatcmpl-756a4ce7-64fc-986f-bfe3-cdba914d38d5",
    "choices": [
        {
            "finish_reason": "stop",
            "index": 0,
            "logprobs": null,
            "message": {
                "content": "{\n  \"姓名\": \"张三\",\n  \"学号\": \"12345678\"\n}",
                "refusal": null,
                "role": "assistant",
                "function_call": null,
                "tool_calls": null
            }
        }
    ],
    "created": 1726654530,
    "model": "qwen2.5-72b-instruct",
    "object": "chat.completion",
    "service_tier": null,
    "system_fingerprint": null,
    "usage": {
        "completion_tokens": 25,
        "prompt_tokens": 57,
        "total_tokens": 82
    }
}
```


图 3: qwen api 文档结构化输出示例响应

问：有什么方式增加生成正确查询的概率？

- 在 prompt 中加入一些合法的 wa 查询作为 few-shot 是否可行？大概需要多少条才能有效提升生成正确查询的概率？
- 如何分解问题，多次调用 wa？
- 如何结合 CoT 、ToT 等方式？

问：有什么方式增加生成正确查询的概率？

- 在 prompt 中加入一些合法的 wa 查询作为 few-shot 是否可行？大概需要多少条才能有效提升生成正确查询的概率？
- 如何分解问题，多次调用 wa？
- 如何结合 CoT 、ToT 等方式？
- 对于类似于 `Find the minimum x greater than 10 such that PrimeQ[x + 7] && PrimeQ[x + 13] && PrimeQ[x + 15]`. 这种很难转化为 wa 查询的问题，是否可以结合 python 程序去解决？什么样的问题适合用 python 程序解决？什么样的问题适合用 wa 解决？如果 wa 和 python 都使用，对于一个具体的问题，LLM 如何决策选择哪一种工具？更复杂地，如果一个问题经过分解后，有一部分适合用 wa 解决，有一部分适合用 python 解决，LLM 如何决策？

Thank you!