
REFTOOL: Enhancing Model Reasoning with Reference-Guided Tool Creation

Xiao Liu¹ Da Yin² Zirui Wu¹ Yansong Feng^{1*}

¹Wangxuan Institute of Computer Technology, Peking University

²University of California, Los Angeles

{lxlisa,ziruiwu,fengyansong}@pku.edu.cn, da.yin9712@gmail.com

Abstract

Tools enhance the reasoning capabilities of large language models (LLMs) in complex problem-solving tasks, but not all tasks have available tools. In the absence of predefined tools, prior works have explored instructing LLMs to generate tools on their own. However, such approaches rely heavily on the models' internal knowledge and would fail in domains beyond the LLMs' knowledge scope. To address this limitation, we propose REFTOOL, a reference-guided framework for automatic tool creation that leverages structured external materials such as textbooks. REFTOOL consists of two modules: (1) tool creation, where LLMs generate executable tools from reference content, validate them using illustrative examples, and organize them hierarchically into a toolbox; and (2) tool utilization, where LLMs navigate the toolbox structure to select and apply the appropriate tools to solve problems. Experiments on causality, physics, and chemistry benchmarks demonstrate that REFTOOL outperforms existing tool-creation and domain-specific reasoning methods by 11.3% on average accuracy, while being cost-efficient and broadly generalizable. Analyses reveal that grounding tool creation in references produces accurate and faithful tools, and that the hierarchical structure facilitates effective tool selection. RefTool enables LLMs to overcome knowledge limitations, demonstrating the value of grounding tool creation in external references for enhanced and generalizable reasoning. Code is in <https://github.com/xxxiaol/RefTool>.

1 Introduction

Tools play a critical role in enhancing the reasoning capabilities of large language models (LLMs), particularly in complex problem-solving tasks like mathematical reasoning [14, 32]. By integrating external tools, LLMs can use off-the-shelf modules to complete subtasks and execute precise computations, thereby improving their performance.

Despite their importance, such tools are not universally available across all scenarios. A prominent line of work attempts to mitigate this limitation by instructing LLMs to generate their own tools based on given problems [20, 2, 25]. However, these methods would fall short when models lack the relevant expert knowledge, especially for specialized and novel domains. For example, if an LLM is unfamiliar with *how to estimate the causal effect from a treatment variable to an outcome variable*, it can hardly generate appropriate tools for such tasks.

To address this challenge, we propose REFTOOL, a reference-guided framework for automatic tool creation. Unlike existing methods that rely on LLMs' internal knowledge, REFTOOL extracts and generates tools from structured reference materials, such as textbooks and technical documents. As shown in Figure 1, REFTOOL consists of two modules: tool creation and tool utilization. During

* Corresponding author.

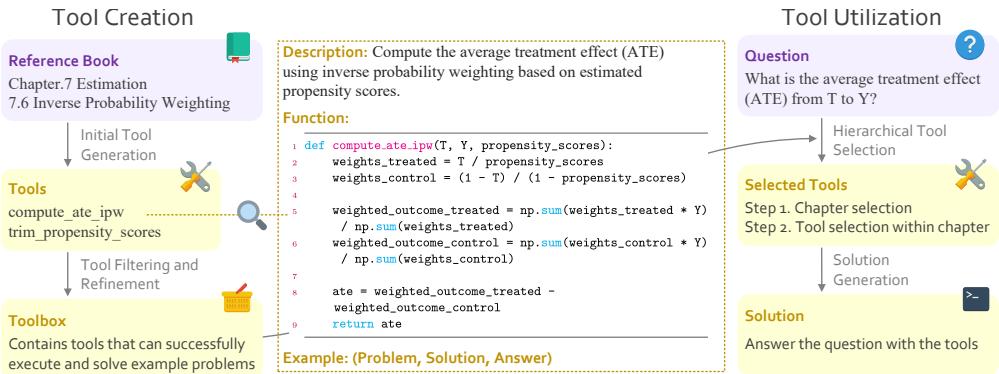


Figure 1: Overview of the REFTOOL framework, which consists of two modules: tool creation (left) and tool utilization (right).

tool creation, the framework employs LLMs to generate executable tools from reference content. In the example, given a section on *Inverse probability weighting*² from a causal inference textbook, the LLM produces tools like `compute_ate_ipw` according to the content of the section. The generated tools consist of descriptions, functions, alongside illustrative examples of how to use the tools. These examples also serve as validation cases, filtering out incorrect or non-functional tools while retaining those that successfully solve the example problems. The validated tools are organized into a hierarchical toolbox, mirroring the structure of the reference material.

During inference, REFTOOL guides the LLM to select tools from the toolbox hierarchically and apply tools to solve problems. For an input question like *what is the average treatment effect from T to Y*, the LLM navigates the toolbox hierarchy, selecting the *Estimation* chapter and then the `compute_ate_ipw` tool within the chapter. Finally, the LLM generates the solution with the help of the selected tool. By grounding tool creation and selection in external references rather than internal knowledge, REFTOOL can construct and deploy tools beyond the model’s original capabilities, enabling it to tackle tasks that would otherwise be infeasible.

We evaluate REFTOOL across three challenging domains: causality, physics, and chemistry. Experimental results on QRData [13], TheoremQA [4], and SciBench [24] show that REFTOOL outperforms general-reasoning methods like Program-of-Thoughts [3] and ReAct [30], highlighting the value of incorporating external knowledge. Notably, retrieval-augmented generation with the same textbook references fails to match this performance, revealing REFTOOL’s unique strengths in transforming knowledge into tools and utilizing hierarchical organization for tool selection.

REFTOOL achieves an average accuracy improvement of 11.3% over existing tool creation and domain-specific methods [20, 19, 22]. Unlike prior works that depend on manually constructed toolsets or extensive trial-and-error on validation data, REFTOOL achieves greater efficiency in both time and computational cost. The generated tools also exhibit strong generalization. Rather than being dataset-specific, they maintain robust performance across diverse datasets, akin to human learning from textbooks.

Further analyses highlight the benefits of reference-guided tool creation and hierarchical tool selection. Human evaluation reveals that with the help of reference books, most tools created by REFTOOL are correct, faithful to the books, and useful in solving downstream problems. On the other hand, the hierarchical organization of tools according to the books’ structures enables effective tool selection, outperforming similarity-based retrieval by an average of 3.3% in accuracy.

To summarize, we propose REFTOOL, a reference-guided framework for tool creation. REFTOOL has the following advantages: (1) By leveraging structured reference materials, REFTOOL enables LLMs to generate tools beyond their internal knowledge, while the reference hierarchy naturally serves as a structure for organizing and selecting tools. (2) Experiments on three complex problem-solving benchmarks demonstrate that REFTOOL consistently improves model reasoning performance over

²Inverse probability weighting is a common method for estimating causal effects.

existing baselines. (3) REFTOOL generates dataset-agnostic tools in a cost-efficient and human-free manner, demonstrating the potential for application in diverse scenarios.

2 The REFTOOL Framework

As shown in Figure 1, REFTOOL operates in two stages: (1) constructing a hierarchical toolbox T from reference material R , and (2) selecting and applying tools $t \subset T$ to answer the input question q during inference.

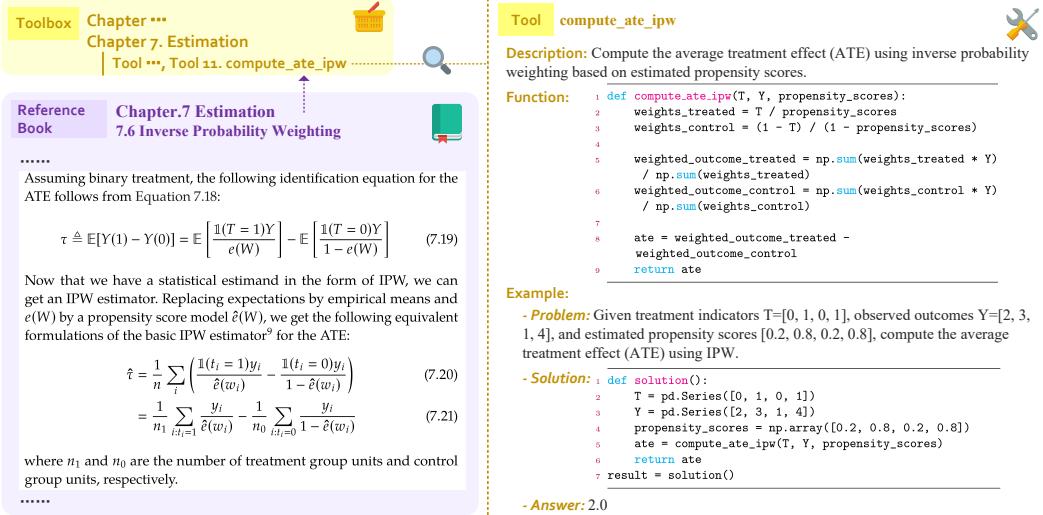


Figure 2: Example of a generated tool and its corresponding reference content. Code comments are omitted due to space limits.

2.1 The Tool Creation Module

Structure Extraction The tool creation process begins with extracting the structure of reference materials. Reference materials are first converted to \LaTeX format, preserving their inherent structure.

These documents such as textbooks typically exhibit a clear hierarchical structure that supports systematic knowledge acquisition. At the highest level, the documents are usually organized into several *chapters* (e.g., `\chapter{Estimation}` in the causal inference book), and chapters are further decomposed into *sections* (e.g., `\section{Inverse Probability Weighting}`), where each section addresses a particular technique, theorem, or application within the chapter’s broader context.

Initial Tool Generation Given the content of each section $s_i \in \mathcal{R}$, the LLM is instructed to generate executable tools based on the content. Each tool contains the following components, as shown in Figure 2:

- **Description:** Natural language summary of the tool’s purpose.
- **Function:** Python implementation of the tool, with comments describing the parameters and returns.
- **Example:** A demonstration example of how to use the tool, consisting of a problem, a piece of solution code where the tool is called, and the expected answer. The model prioritizes examples from the reference text when available, otherwise generating an appropriate example by itself.

The LLM is asked to generate at most m tools for each section. To ensure proper formatting, the prompt includes a human-written tool example from a different domain.

Tool Filtering and Refinement Each tool is validated through *execution testing* and *output verification* using the model-generated demonstration example. The solution code should run without errors, and the output should match the expected answer.

Failed tools trigger a refinement step, where the failure information is provided to the LLM to refine the tool. Finally, the valid tools are organized hierarchically into a toolbox, mirroring the source document’s structure.

2.2 The Tool Utilization Module

Hierarchical Tool Selection During inference, REFTOOL performs hierarchical retrieval to select tools for question q through two phases:

- **Chapter Selection:** Given the reference material’s table of contents C , the model is instructed to select at most n_c relevant chapters $c \subset C$ for the question q .
- **Tool Selection within Chapter:** For each selected chapter c_i , the model is given access to all tools from the toolbox T associated with that chapter, including their descriptions, functions, and demonstration examples. It is then prompted to select up to n_t relevant tools t , or none if no tools are deemed applicable.

Solution Generation The selected tools are then integrated into the reasoning process. We incorporate the tools with two reasoning paradigms: single-turn Program-of-Thoughts (PoT) reasoning [3] and multi-turn ReAct-style agent reasoning [30]. For both paradigms, the model receives selected tools in the initial prompt and is instructed to invoke them when appropriate. When no suitable tools are identified, REFTOOL defaults to standard PoT or ReAct reasoning, ensuring graceful degradation for questions outside the reference domain.

3 Experiments

We conduct experiments on three complex problem-solving domains: causality, physics, and chemistry. This section introduces the experimental setup and presents the performance of REFTOOL across these domains.

3.1 Experimental Setup

Datasets We employ the following evaluation benchmarks:

- **Causality:** QRData-causal [13] (269 questions). Each question is accompanied by one or multiple datasheets, and models are asked to analyze the datasheets and answer causal questions.
- **Physics:** TheoremQA-physics [4] (114 questions), covering broad topics of university-level physics.
- **Chemistry:** SciBench-chemistry [24], focusing on three sub-datasets (chemmc, quan, and matter) related to physical and quantum chemistry (118 questions in total). We omit the other sub-dataset atkins because our reference material overlaps with the question source of this sub-dataset.

We maintain consistent evaluation protocols (like answer extraction methods and tolerance rates) with the original benchmarks (see Appendix A for details) and report accuracy as our primary metric.

Reference Materials Analogous to humans preparing for an exam by reading relevant textbooks, we select reference materials that have a similar domain of knowledge to the evaluation datasets.

For causality, we choose *Introduction to Causal Inference* [16], which provides a detailed description of main causal inference topics like causal discovery and estimation. For physics, as university physics is a broad domain, we choose the three-volume textbook *University Physics* [11], which covers the core concepts of physics like mechanics, thermodynamics, and modern physics. For chemistry, as the evaluation benchmark is in the domain of physical chemistry, we choose a famous physical chemistry textbook *Atkins’ Physical Chemistry* [1].³

Table 1 (left) provides detailed statistics. Note that none of the evaluation questions originate from these books, and none of these books contain code directly.

Implementation Details We employ GPT-4o [8] for tool creation in the main paper, and evaluate four powerful LLMs for tool utilization: Llama-3.1-70B [6], Gemini-1.5-Pro [23], GPT-4 [17], and GPT-4o. The specific versions are Llama-3.1-70B-Instruct, gemini-1.5-pro-002, gpt-4-1106-preview and gpt-4o-2024-11-20.

³Quantum chemistry is a subdomain of physical chemistry, and is also introduced in this textbook.

Table 1: Statistics of the reference materials and created tools. “Avg. Lines” indicates the average lines of tool functions.

Domain	Book	# Chapters	# Sections	# Tools	Avg. Lines
Causality	Introduction to Causal Inference [16]	11	55	84	24
Physics	University Physics [11]	44	284	515	16
Chemistry	Atkins’ Physical Chemistry [1]	19	90	158	17

During tool creation, we set $m = 2$ tools per section across all domains. This can be adjusted based on each section’s length and information density. For tool utilization, we employ a default configuration of selecting $n_c = 1$ chapter and $n_t = 1$ tool. As QRData and TheoremQA do not have a validation set, we use the default setting for the causality and physics domains. For chemistry, we perform grid search over $n_c \in [1, 2]$ and $n_t \in [1, 2]$ on the validation set of SciBench, and choose $n_c = 1$ and $n_t = 2$. Additional details and prompt templates can be found in Appendices A and F.

Baseline Methods We compare against the following baselines, with more details in Appendix A:

- **General reasoning methods:** We implement Program-of-Thoughts (PoT) for single-turn reasoning and ReAct for multi-turn reasoning.⁴ We also experiment with direct reasoning and Chain-of-Thought [28] on GPT-4 in Appendix B.1, but they are excluded from main comparisons due to inferior performance.
- **Retrieval-augmented generation (RAG) methods:** To investigate if LLMs can directly learn from reference materials, we enhance both PoT and ReAct with RAG, using the same reference books employed for tool creation. The books are segmented into subsections, and the segment with the highest similarity to the question embedding is retrieved.
- **General-purpose tool creation methods:** We include Creator [20] as a representative method, which dynamically creates tools for each question and performs error correction. We exclude methods that rely on training data [2, 31] or require multiple test-time predictions [25] to maintain a fair zero-shot comparison.
- **Domain-specific reasoning methods:** Our evaluation includes: (1) Physics Reasoner [19], which manually constructs a formula set and instructs LLMs to retrieve formulas during reasoning. (2) For chemistry, both StructChem [18], which instructs LLMs to generate formulas before reasoning; and ChemAgent [22], which builds a library of memories by extensive trial-and-error on validation data. We do not compare with methods that use web search as some answers may be searched directly from the Internet.

3.2 Results

Toolbox Construction Table 1 (right) demonstrates statistics of tools created. On average, 73% of initially generated tools pass validation directly, with an additional 14% tools succeeding after refinement. We assess tool quality through human evaluation in §4.1.

Main Results The performance comparison in Table 2 demonstrates REFTOOL’s superior performance across all domains, achieving the highest average accuracy.⁵ Notably, REFTOOL surpasses Creator by an average margin of 12.3%, highlighting the advantage of reference-based tool creation over tools created solely by LLMs.

While RAG incorporates the same reference materials, it fails to consistently enhance the performance, showing an average accuracy decrease of 0.2% over corresponding PoT/ReAct baselines. This suggests that direct retrieval struggles to effectively extract and apply relevant knowledge, while REFTOOL’s tool format and hierarchical organization enable better utilization of reference materials. The ablation study in §4.2 further analyzes the impact of both components.

⁴While ReAct demonstrates effectiveness on QRData by allowing error correction through multi-turn interactions [13], our preliminary experiments (Appendix Table 6) show limited benefits for physics and chemistry domains, likely due to the simpler code solutions without data analysis and fewer execution errors. Consequently, we omit ReAct for these domains.

⁵Chemistry results are averaged across three sub-datasets, with detailed per-sub-dataset performance shown in Appendix Table 7.

Table 2: Performance of REFTOOL and baseline methods in causality (QRData), physics (TheoremQA), and chemistry (SciBench) domains. Numbers are in percentages (%), with the best performance for each model shown in bold.

Method	Accuracy				
	Llama-3.1-70B	Gemini-1.5-Pro	GPT-4	GPT-4o	Average
Causality					
PoT	33.1	41.3	34.2	39.8	37.1
PoT + RAG	29.7	36.4	37.5	42.0	36.4
PoT + REFTOOL	36.8	43.9	38.7	46.8	41.6
Creator	14.9	29.7	39.4	39.8	31.0
ReAct	30.1	47.6	50.9	46.5	43.8
ReAct + RAG	32.3	46.8	48.0	49.1	44.1
ReAct + REFTOOL	33.5	48.3	51.3	52.0	46.3
Physics					
Physics Reasoner	48.2	50.9	42.1	33.3	43.6
Creator	40.4	57.0	35.1	40.4	43.2
PoT	48.2	57.9	45.6	57.0	52.2
PoT + RAG	44.7	57.0	44.7	57.9	51.1
PoT + REFTOOL	53.5	58.8	49.1	57.9	54.8
Chemistry					
Creator	40.1	60.0	46.9	43.3	47.6
StructChem	37.9	50.2	29.7	40.5	39.6
ChemAgent	48.2	65.5	52.5	58.9	56.3
PoT	46.9	62.3	51.8	58.9	55.0
PoT + RAG	48.1	63.7	54.1	56.6	55.6
PoT + REFTOOL	49.5	66.4	53.4	61.3	57.7

Among domain-specific methods, Physics Reasoner and StructChem perform inferior to PoT, with their complex format requirements leading to suboptimal adaptation on some models. Although ChemAgent approaches REFTOOL’s performance, it needs significantly higher computational costs, as discussed in §4.3.

Performance on Reasoning Models We also conduct a small-scale experiment to evaluate if REFTOOL works for reasoning models. We apply REFTOOL on o1-mini [9] (with the specific version o1-mini-2024-09-12), and Appendix Table 8 shows that its average accuracy improves by 4.3% over PoT. This indicates that REFTOOL is also compatible with reasoning models, supplementing their knowledge and skills.

Robustness of Tool Creation We validate whether REFTOOL remains effective when alternative LLMs are used in the tool creation module. As shown in Appendix Table 9, employing Gemini-1.5-Pro for tool creation also achieves superior performance compared to baseline methods.

Tool Reusability We conduct experiments on another physics dataset SciBench-fund [24] to validate the generalizability of tools created by REFTOOL. Appendix Table 10 shows that on SciBench-fund, REFTOOL outperforms all zero-shot baseline methods and matches 4-shot Physics Reasoner, using the same tools as in the evaluation of TheoremQA. As REFTOOL is dataset-agnostic, tools created for one domain could be applied to different datasets in that domain.

4 Analysis

In this section, we further analyze the effectiveness of REFTOOL through multiple perspectives: human evaluation of tool quality and selection consistency (§4.1), ablation study of key components (§4.2), computational cost analysis (§4.3), and case study of how REFTOOL helps LLMs to answer questions (§4.4).

Table 3: Results of human evaluation. Numbers are in percentages (%).

(a) Tool Quality Assessment. Example correctness is assessed only when the function is correct.

Domain	Faithful	Function Correct	Example Correct	Useful
Causality	95	95	100	90
Physics	90	90	100	90
Chemistry	90	90	89	95

(b) Consistency of Tool Selection with Humans. Chapter selection consistency is calculated as the fraction of questions where human and model chapter choices match (when both select a chapter). Tool selection consistency is the fraction where their tools overlap, given they both choose tools from the same chapter.

Domain	Consistency	Llama-3.1-70B	Gemini-1.5-Pro	GPT-4	GPT-4o
Causality	Chapter Selection	100	95	100	100
	Tool Selection within Chapter	94	100	91	94
Physics	Chapter Selection	80	80	75	76
	Tool Selection within Chapter	53	56	44	69
Chemistry	Chapter Selection	55	65	72	75
	Tool Selection within Chapter	60	67	40	90

4.1 Human Evaluation

We conduct human evaluation to assess (1) the quality of created tools and (2) the alignment between LLM-selected tools and those chosen by domain experts.

Tool Quality Assessment We evaluate tools along four dimensions. (1) *Faithfulness*: Whether the tool accurately reflects the source material. (2) *Function Correctness*: Whether the tool function meets the tool description and is implemented correctly. (3) *Example Correctness*: Whether the example solution uses the function properly and returns the right answer. (4) *Usefulness*: The practical utility of the tool for solving relevant problems without being too narrow. For each domain, we randomly sample 20 tools from the toolbox, and ask a human expert who has studied corresponding courses to annotate them.

As shown in Table 3a, all aspects are satisfied by $\geq 89\%$ tools, indicating that most tools are faithfully derived from the references, correctly implemented, and useful in application. Chemistry tools show slightly lower quality due to the domain’s complexity. When LLMs lack foundational knowledge, they may misinterpret nuanced concepts, like mistaking the meaning of a coefficient.

Consistency of Tool Selection with Humans We compare human and LLM tool selection by having experts simulate the hierarchical selection process. For each domain, we randomly sample 20 questions where models consistently use tools (see Appendix C for more details).

Table 3b shows the consistency between LLMs and human experts. We observe higher agreement in chapter selection compared to tool selection, supporting our hierarchical selection step which narrows down the tool search space with a high consensus. Comparing between domains, causality achieves the highest consistency in both chapter and tool selection. This stems from the domain’s more straightforward questions containing keywords like *average treatment effect* that directly point to specific knowledge. Physics and chemistry questions, by contrast, often involve more indirect formulations that may challenge models in identifying the required knowledge.

Gemini-1.5-Pro and GPT-4o demonstrate better alignment with human experts, mirroring their superior PoT performance which reflects stronger internal domain knowledge. While Llama-3.1-70B and GPT-4 show weaker consistency with humans in physics and chemistry tool selection, their chosen tools are still valuable as relevant knowledge is recalled. In cases where these models select the same chapter as humans but different tools, we observe a 17% accuracy improvement from tool usage compared to the PoT baseline. Appendix D provides a concrete example.

Table 4: Ablation results. (sim) indicates selecting with text similarity. Numbers are in percentages (%), with the best performance for each model shown in bold.

Method	Accuracy				
	Llama-3.1-70B	Gemini-1.5-Pro	GPT-4	GPT-4o	Average
Causality					
PoT + RAG	29.7	36.4	37.5	42.0	36.4
PoT + Hierarchical RAG	36.8	38.7	43.5	36.8	39.0
PoT + REFTOOL (sim)	30.5	36.1	46.1	39.4	38.0
PoT + REFTOOL	36.8	43.9	38.7	46.8	41.6
Physics					
PoT + RAG	44.7	57.0	44.7	57.9	51.1
PoT + Hierarchical RAG	44.7	64.0	44.7	55.3	52.2
PoT + REFTOOL (sim)	45.6	62.3	43.0	56.1	51.8
PoT + REFTOOL	53.5	58.8	49.1	57.9	54.8

4.2 Ablation Study

We design two variants of REFTOOL to analyze its key components: code-form tool creation and hierarchical selection.

- **PoT + Hierarchical RAG:** Substitutes REFTOOL’s code-form tools with raw text segments while preserving the hierarchical structure. This maintains the three-step reasoning process of chapter selection, intra-chapter text retrieval, and solution generation, allowing us to isolate the impact of tool representation.
- **PoT + REFTOOL (sim):** Retains the tool creation but replaces hierarchical selection with similarity-based retrieval. Tool descriptions are encoded into embeddings, with the most similar tool selected for each problem, mirroring standard RAG approaches but using tools instead of text.

Table 4 shows the ablation results. Due to computational constraints, we focus on causality and physics domains with single-turn reasoning. By comparing PoT + REFTOOL with PoT + Hierarchical RAG, as well as PoT + REFTOOL (sim) with PoT + RAG, we observe an average 1.9% accuracy gain of tools over textual knowledge, confirming that the code form of tools enhances model understanding and application of knowledge.

By comparing PoT + REFTOOL with PoT + REFTOOL (sim), as well as PoT + Hierarchical RAG with PoT + RAG, we find that hierarchical selection outperforms similarity-based retrieval by 2.6% on average, demonstrating its effectiveness in knowledge retrieval.

4.3 Cost Analysis

Table 5: Cost analysis of tool-augmented methods (with GPT-4o as the base model). “-” indicates that the step is done by humans and the cost is unknown.

Domain	Method	Time (min.)		Cost (\$)	
		Toolbox Construction	Inference	Toolbox Construction	Inference
Physics	Physics Reasoner	-	75	-	3.5
	PoT + REFTOOL	5	2	6.9	1.5
Chemistry	ChemAgent	1233	536	79.3	41.3
	PoT + REFTOOL	3	6	3.5	1.4

Table 5 shows that REFTOOL greatly saves cost compared with other tool-augmented methods. Compared with Physics Reasoner which iteratively refines the reasoning process, REFTOOL reduces inference time by 97% and cost by 57%. The improvements are even more pronounced when compared to ChemAgent’s divide-and-retry strategy: REFTOOL cuts both toolbox construction time and inference time by 99%. The guidance of references enables REFTOOL to achieve great performance without repeatedly trying, offering a scalable solution for complex reasoning tasks.

Input

Data Description: The data set in flow.csv offers continuous measurements of expression levels of multiple phosphorylated proteins and phospholipid components in human immune system cells ...

Question: Which cause-and-effect relationship is more likely?
 A. pakts473 causes pmek B. pmek causes pakts473
 C. No causal relationship exists
 Please answer with A, B, or C.

Program of Thoughts (PoT)

```

1 def solution():
2     data = pd.read_csv('flow.csv')
3     models = [LinearRegression().fit(data[[x]], data[y])
4               for x, y in [('pakts473', 'pmek'), ('pmek',
5                             'pakts473')]]
6     r2 = [r2_score(data[y], m.predict(data[[x]]))
7               for m, (x, y) in zip(models, [('pakts473', 'pmek',
8                                 ), ('pmek', 'pakts473'))]]
9     return "A" if r2[0] > r2[1] else "B" if r2[1] > r2[0]
   else "C"

```

Answer: A ❌ **The magnitude of the R-squared value does not indicate causality!**

Tool **Causal Direction Fit**

Fit a linear model in the causal direction and compute residuals to test for independence between the input variable and residuals.

```

1 def causal_direction_fit(X, Y):
2     X, Y = np.array(X).reshape(-1, 1), np.array(Y)
3     model = LinearRegression().fit(X, Y)
4     residuals = Y - model.predict(X)
5     independence = abs(pearsonr(X.flatten(), residuals)[0]) < 0.1
6     return {k: v for k, v in zip(
7         ["coefficients", "intercept", "residuals",
8          "independence"], [
9             model.coef_, model.intercept_, residuals,
10            independence])}

```

PoT + RefTool

Selected Chapter: Chapter.11 Causal Discovery from Observational Data
Selected Tool: Causal Direction Fit

```

1 def solution():
2     data = pd.read_csv('flow.csv')
3     r1, r2 = causal_direction_fit(data[x], data[y])
4     for x, y in [('pakts473', 'pmek'), ('pmek',
5                   'pakts473')]:
5         a, b = r1['independence'], r2['independence']
6     return "A" if a and not b else "B" if b and not a else
   "C"

```

Answer: C ✓

(a) Wrong solution when reasoning with PoT

(b) Correct solution when reasoning with PoT + RefTool

Figure 3: Example case of GPT-4o with (right) and without (left) REFTOOL.

4.4 Case Study

Figure 3 demonstrates a case where GPT-4o correctly answers a question with REFTOOL. When presented with the causal discovery problem, the model successfully navigates to the relevant chapter *Causal discovery from observational data* and selects the appropriate `causal_direction_fit` tool, generating the correct solution code. In contrast, without tool assistance, the model incorrectly uses R-squared values to infer causal relationships, leading to a wrong prediction. The detailed version of the case, along with physics and chemistry cases, are in Appendix D.

5 Related Work

Automatic Tool Creation The automatic creation of tools for LLMs aims to overcome the limitations of relying solely on pre-existing tools. Most works generate tools in code format, while some generate skills or workflows in the format of abstract actions [29] or non-executable text [26]. Existing methods can be broadly categorized into two paradigms: (1) generating temporary, task-specific tools for individual queries [20], and (2) constructing reusable toolsets based on training or testing data [2, 25]. These methods have demonstrated success in mathematical reasoning [20, 2], visual question answering [31, 25], and agent-based tasks [27, 33]. Unlike previous works, which primarily rely on LLMs’ internal knowledge, our method utilizes external references to create tools, enabling applications beyond the models’ inherent knowledge scope.

Tool-Augmented Reasoning Tool-augmented reasoning enhances LLMs’ problem-solving capabilities by integrating external tools, particularly for tasks requiring specialized knowledge or complex computation. Some studies manually curate a small set of high-quality tools [7, 15], while others [21, 12] utilize large-scale APIs from platforms like RapidAPI or API-Bank [10]. However, the massive amount of tools makes it difficult to select an appropriate tool for a task. To mitigate this, we use the hierarchical structure of reference materials to organize and select tools. This strategy is also adopted by Du et al. [5], which leverages RapidAPI’s categorization for tool selection.

6 Conclusion

We present REFTOOL, a framework that enhances LLM reasoning through reference-guided tool creation. By leveraging structured materials like textbooks, REFTOOL enables LLMs to generate

accurate tools beyond their internal knowledge, and the hierarchical organization of tools further enables effective tool selection. Experiments across causality, physics, and chemistry domains show consistent improvements over existing tool-creation and domain-specific reasoning methods, while maintaining computational efficiency. REFTOOL demonstrates how grounding tool creation in authoritative references can overcome LLMs’ knowledge limitations, offering a generalizable solution for complex problem-solving.

References

- [1] P. W. Atkins, J. De Paula, and J. Keeler. *Atkins’ physical chemistry*. Oxford university press, 2023.
- [2] T. Cai, X. Wang, T. Ma, X. Chen, and D. Zhou. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*, 2023.
- [3] W. Chen, X. Ma, X. Wang, and W. W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.
- [4] W. Chen, M. Yin, M. Ku, P. Lu, Y. Wan, X. Ma, J. Xu, X. Wang, and T. Xia. Theoremqa: A theorem-driven question answering dataset. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [5] Y. Du, F. Wei, and H. Zhang. Anytool: Self-reflective, hierarchical agents for large-scale api calls. In *International Conference on Machine Learning*, pages 11812–11829. PMLR, 2024.
- [6] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan, et al. The llama 3 herd of models. *ArXiv preprint*, abs/2407.21783, 2024.
- [7] Y. Gu, Y. Shu, H. Yu, X. Liu, Y. Dong, J. Tang, J. Srinivasa, H. Latapie, and Y. Su. Middleware for llms: Tools are instrumental for language agents in complex environments. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7646–7663, 2024.
- [8] A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. Gpt-4o system card. *ArXiv preprint*, abs/2410.21276, 2024.
- [9] A. Jaech, A. Kalai, A. Lerer, A. Richardson, A. El-Kishky, A. Low, A. Helyar, A. Madry, A. Beutel, A. Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.
- [10] M. Li, Y. Zhao, B. Yu, F. Song, H. Li, H. Yu, Z. Li, F. Huang, and Y. Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, 2023.
- [11] S. J. Ling, J. Sanny, W. Moebs, G. Friedman, S. D. Druger, A. Kolakowska, D. Anderson, D. Bowman, D. Demaree, E. Ginsberg, et al. *University Physics*. OpenStax, 2016.
- [12] X. Liu, Z. Peng, X. Yi, X. Xie, L. Xiang, Y. Liu, and D. Xu. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*, 2024.
- [13] X. Liu, Z. Wu, X. Wu, P. Lu, K.-W. Chang, and Y. Feng. Are llms capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9215–9235, 2024.
- [14] P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. N. Wu, S.-C. Zhu, and J. Gao. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems*, 36:43447–43478, 2023.
- [15] P. Lu, B. Chen, S. Liu, R. Thapa, J. Boen, and J. Zou. Octotools: An agentic framework with extensible tools for complex reasoning. *arXiv preprint arXiv:2502.11271*, 2025.
- [16] B. Neal. Introduction to causal inference, 2020.
- [17] OpenAI. Gpt-4 technical report. *ArXiv preprint*, abs/2303.08774, 2023.

- [18] S. Ouyang, Z. Zhang, B. Yan, X. Liu, Y. Choi, J. Han, and L. Qin. Structured chemistry reasoning with large language models. In *International Conference on Machine Learning*, pages 38937–38952. PMLR, 2024.
- [19] X. Pang, R. Hong, Z. Zhou, F. Lv, X. Yang, Z. Liang, B. Han, and C. Zhang. Physics reasoner: Knowledge-augmented reasoning for solving physics problems with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11274–11289, 2025.
- [20] C. Qian, C. Han, Y. Fung, Y. Qin, Z. Liu, and H. Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.
- [21] Y. Qin, S. Liang, Y. Ye, K. Zhu, L. Yan, Y. Lu, Y. Lin, X. Cong, X. Tang, B. Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations*, 2024.
- [22] X. Tang, T. Hu, M. Ye, Y. Shao, X. Yin, S. Ouyang, W. Zhou, P. Lu, Z. Zhang, Y. Zhao, et al. Chemagent: Self-updating library in large language models improves chemical reasoning. *arXiv preprint arXiv:2501.06590*, 2025.
- [23] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, A. Gulati, G. Tanzer, D. Vincent, Z. Pan, S. Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *ArXiv preprint abs/2403.05530*, 2024.
- [24] X. Wang, Z. Hu, P. Lu, Y. Zhu, J. Zhang, S. Subramaniam, A. R. Loomba, S. Zhang, Y. Sun, and W. Wang. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. In *Forty-first International Conference on Machine Learning*, 2024.
- [25] Z. Wang, G. Neubig, and D. Fried. Trove: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *International Conference on Machine Learning*, pages 51177–51191. PMLR, 2024.
- [26] Z. Z. Wang, J. Mao, D. Fried, and G. Neubig. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.
- [27] Z. Z. Wang, A. Gandhi, G. Neubig, and D. Fried. Inducing programmatic skills for agentic tasks. *arXiv preprint arXiv:2504.06821*, 2025.
- [28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [29] L. Wong, J. Mao, P. Sharma, Z. Siegel, J. Feng, N. Korneev, J. B. Tenenbaum, and J. Andreas. Learning adaptive planning representations with natural language guidance. In *The Twelfth International Conference on Learning Representations*, 2024.
- [30] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2022.
- [31] L. Yuan, Y. Chen, X. Wang, Y. R. Fung, H. Peng, and H. Ji. Craft: Customizing llms by creating and retrieving from specialized toolsets. In *12th International Conference on Learning Representations, ICLR 2024*, 2024.
- [32] B. Zhang, K. Zhou, X. Wei, X. Zhao, J. Sha, S. Wang, and J.-R. Wen. Evaluating and improving tool-augmented computation-intensive math reasoning. *Advances in Neural Information Processing Systems*, 36:23570–23589, 2023.
- [33] B. Zheng, M. Y. Fatemi, X. Jin, Z. Z. Wang, A. Gandhi, Y. Song, Y. Gu, J. Srinivasa, G. Liu, G. Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

A Implementation Details

Implementation of the REFTOOL Framework By default, each tool’s demonstration example is included during solution generation. For the causality domain, we omit the example because QRData questions involve data analysis and differ significantly in format from the examples.

The temperature of all models is set to 0. The maximum output tokens are set to 2048 for initial tool generation, refinement, and solution generation, and 512 for hierarchical tool selection. Experiments are conducted on 8 NVIDIA A800 GPUs.

Implementation of Baseline Methods For RAG methods, reference documents are segmented by subsections. Subsections exceeding 1,000 tokens are further divided into 1,000-token chunks. These segments are then processed using the text-embedding-3-large embedding model to generate text embeddings. During inference, we compute the embedding for each question and select the text with the highest similarity score to include in the model’s prompt.

All baselines are evaluated in a zero-shot setting. For Creator, we provide a tool example from a different domain (math) in the tool creation prompt. For ChemAgent, we use GPT-4o for library construction to align with REFTOOL’s setting.

Evaluation Protocol We adopt the original benchmarks’ evaluation code for consistency. The tolerant rate of numerical questions is 3% for QRData, 4% for TheoremQA, and 5% for SciBench.

B Additional Results

B.1 Performance of More Baseline Methods

Table 6: Performance of more baseline methods on GPT-4. Numbers are in percentages (%).

Method	Causality	Accuracy	
		Physics	Chemistry
Direct Reasoning	33.1	14.0	32.8
CoT	41.3	10.5	35.4
PoT	34.2	45.6	51.8
ReAct	50.9	41.2	54.6

Previous works also compare with pure-text baselines like direct reasoning and Chain-of-Thought (CoT) [28], but as our preliminary experiment in Table 6 shows that these methods are much inferior to PoT on GPT-4, we do not add them into the baselines in the main paper. While CoT achieves high performance on the causality domain, this results from educated guessing on multiple-choice questions, with none of the numerical questions being answered correctly. This deviates from QRData’s original goal of conducting data-based quantitative reasoning. Direct reasoning outperforms CoT in physics because, despite the instruction to answer directly, the model still generates intermediate reasoning steps for most questions.

As the code of solving physics and chemistry problems is simpler and successful executes in most cases, multi-turn reasoning is not that necessary in such scenarios, therefore we do not implement the multi-turn settings ReAct and React+REFTOOL. Table 6 also shows that ReAct introduces limited improvement or even negative influence on these domains.

B.2 Sub-dataset Performance of SciBench-chemistry

Table 7 shows the performance of sub-datasets of SciBench-chemistry. While performance varies due to each sub-dataset’s small scale, REFTOOL demonstrates effectiveness in most cases.

B.3 Performance on Reasoning Models

Table 8 shows REFTOOL’s performance on o1-mini, where it improves average accuracy by 4.3% over PoT. This indicates REFTOOL’s compatibility with reasoning models, effectively supplementing their knowledge and capabilities.

Table 7: Performance of sub-datasets of SciBench-chemistry. Numbers are in percentages (%).

Method	Chemmc	Accuracy		
		Matter	Quan	Average
Llama-3.1-70B				
Creator	50.0	34.0	36.4	40.1
StructChem	50.0	21.3	42.4	37.9
ChemAgent	60.5	44.7	39.4	48.2
PoT	65.8	44.7	30.3	46.9
PoT + RAG	63.2	44.7	36.4	48.1
PoT + REFTOOL	63.2	48.9	36.4	49.5
Gemini-1.5-Pro				
Creator	73.7	63.8	42.4	60.0
StructChem	57.9	38.3	54.5	50.2
ChemAgent	78.9	66.0	51.5	65.5
PoT	78.9	59.6	48.5	62.3
PoT + RAG	78.9	63.8	48.5	63.7
PoT + REFTOOL	81.6	66.0	51.5	66.4
GPT-4				
Creator	60.5	46.8	33.3	46.9
StructChem	36.8	19.1	33.3	29.7
ChemAgent	68.4	46.8	42.4	52.5
PoT	68.4	44.7	42.4	51.8
PoT + RAG	65.8	51.1	45.5	54.1
PoT + REFTOOL	71.1	46.8	42.4	53.4
GPT-4o				
Creator	55.3	38.3	36.4	43.3
StructChem	55.3	29.8	36.4	40.5
ChemAgent	78.9	55.3	42.4	58.9
PoT	78.9	55.3	42.4	58.9
PoT + RAG	76.3	51.1	42.4	56.6
PoT + REFTOOL	76.3	53.2	54.5	61.3

Table 8: Performance of o1-mini. Numbers are in percentages (%), with the best performance for each model shown in bold.

Method	Causality	Accuracy	
		Physics	Chemistry
PoT	44.2	56.1	60.5
PoT + REFTOOL	50.2	57.9	65.6

B.4 Performance of Using Gemini-1.5-Pro as the Tool Creation Model

To assess the robustness of REFTOOL’s tool creation module, we experiment with Gemini-1.5-Pro as an alternative LLM. As Table 9 shows, this configuration also achieves superior performance compared to baseline methods.

Compared to GPT-4o-created tools, a relatively lower ratio of Gemini-1.5-Pro-created tools passes the validation. In physics, GPT-4o achieves 82% direct validation success with 8% succeeding after refinement, while Gemini-1.5-Pro achieves 54% direct success with 24% succeeding after refinement. Although refinement helps recover about one-quarter of tools, approximately 20% still get filtered out, potentially leading to incomplete knowledge coverage and slightly lower overall performance compared to GPT-4o-created tools.

Table 9: Performance of REFTOOL using Gemini-1.5-Pro as the tool creation model. Numbers are in percentages (%), with the best performance for each model shown in bold.

Method	Accuracy				
	Llama-3.1-70B	Gemini-1.5-Pro	GPT-4	GPT-4o	Average
Physics					
Physics Reasoner	48.2	50.9	42.1	33.3	43.4
Creator	40.4	57.0	35.1	40.4	43.2
PoT	48.2	57.9	45.6	57.0	52.2
PoT + RAG	44.7	57.0	44.7	57.9	51.1
PoT + REFTOOL (GPT-4o)	53.5	58.8	49.1	57.9	54.8
PoT + REFTOOL (GEMINI)	48.2	58.8	47.4	58.8	53.3
Chemistry					
Creator	40.1	60.0	46.9	43.3	47.6
StructChem	37.9	50.2	29.7	40.5	39.6
ChemAgent	48.2	65.5	52.5	58.9	56.3
PoT	46.9	62.3	51.8	58.9	55.0
PoT + RAG	48.1	63.7	53.7	56.6	55.5
PoT + REFTOOL (GPT-4o)	49.5	66.4	53.4	61.3	57.7
PoT + REFTOOL (GEMINI)	48.3	64.9	52.5	61.6	56.8

Table 10: Performance on another physics dataset: Scibench-fund. Numbers are in percentages (%), with the best performance for each model shown in bold.

Method	Accuracy				
	Llama-3.1-70B	Gemini-1.5-Pro	GPT-4	GPT-4o	Average
Physics Reasoner	56.3	59.2	63.4	36.6	53.9
Creator	56.3	70.4	53.5	57.7	59.5
PoT	53.5	69.0	59.2	73.2	63.7
PoT + RAG	54.9	73.2	59.2	73.2	65.1
PoT + REFTOOL	57.7	73.2	64.8	74.6	67.6
Physics Reasoner (4-shot)	62.0	73.2	63.4	71.8	67.6

B.5 Performance on Another Physics Dataset: SciBench-fund

We evaluated REFTOOL on another physics dataset SciBench-fund [24] with 71 questions, to test tool generalizability.⁶ Table 10 shows that REFTOOL outperforms all zero-shot baselines and matches 4-shot Physics Reasoner’s performance using the same tools in the evaluation of TheoremQA. This demonstrates REFTOOL’s dataset-agnostic nature, where domain-specific tools can be applied across different datasets.

C Human Evaluation Details

In the tool selection process, given a question, annotators are first asked to select at most one chapter from the given book, and none if no chapter is relevant to the question. If they select a chapter, they are then asked to select one tool within the chapter if it is the most useful, select two tools only if they are equally useful, and select none if none of the tools are useful. For each domain, we randomly sample 20 questions where most models (at least 3 out of 4) choose to use tools. All human annotators are fairly paid.

Consistency Metrics For chapter selection, the consistency is computed as:

$$\text{Consistency}_{\text{chapter}} = \frac{|\{\text{human and model select the same chapter}\}|}{|\{\text{both human and model select a chapter}\}|}.$$

⁶We excluded two other SciBench-physics sub-datasets as they require advanced thermodynamics and particle dynamics knowledge beyond our reference textbook’s scope.

Input

Data Description: The data set in flow.csv offers continuous measurements of expression levels of multiple phosphorylated proteins and phospholipid components in human immune system cells ...

Question: Which cause-and-effect relationship is more likely?
A. paks473 causes pmek B. pmek causes paks473
C. No causal relationship exists
Please answer with A, B, or C.

Program of Thoughts (PoT)

```

Solution:
1 def solution():
2     # load data
3     data = pd.read_csv('flow.csv')
4
5     # write code to get the answer
6     # Define the variables
7     X1 = data[['paks473']] # Predictor for A
8     Y1 = data['pmek'] # Response for A
9
10    X2 = data[['pmek']] # Predictor for B
11    Y2 = data['paks473'] # Response for B
12
13    # Fit linear regression models
14    model_A = LinearRegression().fit(X1, Y1)
15    model_B = LinearRegression().fit(X2, Y2)
16
17    # Calculate R-squared values
18    r2_A = r2_score(Y1, model_A.predict(X1))
19    r2_B = r2_score(Y2, model_B.predict(X2))
20
21    # Determine the causal relationship
22    if r2_A > r2_B:
23        return "A"
24    elif r2_B > r2_A:
25        return "B"
26    else:
27        return "C"

```

Answer: A ✘

Reference Book

Chapter.11 Causal Discovery from Observational Data
11.2 Semi-Parametric Causal Discovery

When we fit the data in the causal direction, we get residuals that are independent of the input variable, but when we fit the data in the anti-causal direction, we get residuals that are dependent on the input variable. We depict the regression line \hat{f} we get if we linearly regress Y on X (causal direction) in Figure 11.0a, and we depict the regression line \hat{g} we get if we linearly regress X on Y (anti-causal direction) in Figure 11.0b. Just from these fits, you can see that the forward model (fit in the causal direction) looks more pleasing than the backward model (fit in the anti-causal direction).

Tool Causal Direction Fit

Description: Fit a linear model in the causal direction and compute residuals to test for independence between the input variable and residuals.

Function:

```

1 def causal_direction.fit(X, Y):
2     # Reshape X for sklearn LinearRegression
3     X = np.array(X).reshape(-1, 1)
4     Y = np.array(Y)
5
6     # Fit linear regression model
7     model = LinearRegression()
8     model.fit(X, Y)
9
10    # Compute residuals
11    residuals = Y - model.predict(X)
12
13    # Test for independence between X and residuals
14    # using Pearson correlation
15    corr, _ = pearsonr(X.flatten(), residuals)
16    independence = np.abs(corr) < 0.1 # Threshold
17
18    return {
19        "coefficients": model.coef_,
20        "intercept": model.intercept_,
21        "residuals": residuals,
22        "independence": independence
23    }

```

PoT + RefTool

Selected Chapter: Chapter.11 Causal Discovery from Observational Data
Selected Tool: Causal Direction Fit

Solution:

```

1 def solution():
2     # Load the data
3     data = pd.read_csv('flow.csv')
4
5     # Extract the relevant columns
6     paks473 = data['paks473']
7     pmek = data['pmek']
8
9     # Test causal direction: paks473 -> pmek
10    result_1 = causal_direction.fit(paks473, pmek)
11
12    # Test causal direction: pmek -> paks473
13    result_2 = causal_direction_fit(pmek, paks473)
14
15    # Determine the causal relationship
16    if result_1['independence'] and not result_2['independence']:
17        return "A" # paks473 causes pmek
18    elif result_2['independence'] and not result_1['independence']:
19        return "B" # pmek causes paks473
20    else:
21        return "C" # No causal relationship exists

```

Answer: C ✓

(a) Wrong solution when reasoning with PoT

(b) Correct solution when reasoning with PoT + RefTool

Figure 4: Example case of GPT-4o on a causal problem with (right) and without (left) REFTOOL. This is the detailed version of Figure 3

Input

Question: Point charges $q_1=50\mu\text{C}$ and $q_2=-25\mu\text{C}$ are placed 1.0 m apart. What is the force on a third charge $q_3=20\mu\text{C}$ placed midway between q_1 and q_2 ?

Program of Thoughts (PoT)

```

Solution:
1 def solution():
2     # import libraries if needed
3     import scipy.constants as const
4
5     # write code to get the answer
6     q1 = 50e-6 # C
7     q2 = -25e-6 # C
8     q3 = 20e-6 # C
9     r13 = 0.5 # m
10    r23 = 0.5 # m
11    k = 1/(4*const.pi*const.epsilon_0)
12
13    F13 = k * q1 * q3 / (r13**2)
14    F23 = k * q2 * q3 / (r23**2)
15
16    F_total = F13 + F23
17
18    # return answer
19    return F_total

```

Answer: 18.0 ✘ Did not consider the direction of forces!

Reference Book

Chapter.22 Electric Charges and Fields
5.1 Electric Charge

Coulomb's Law

The magnitude of the electric force (or Coulomb force) between two electrically charged particles is equal to

$$|F_{12}| = \frac{1}{4\pi\epsilon_0} \frac{|q_1 q_2|}{r_{12}^2}$$

The unit vector \hat{r} has a magnitude of 1 and points along the axis as the charges. If the charges have the same sign, the force is in the same direction as showing a repelling force. If the charges have different signs, the force is in the opposite direction of \hat{r} showing an attracting force. (Figure 5.1b).

Tool Electrostatic Force

Description: Calculate the electrostatic force between two charges using Coulomb's law.

Function:

```

1 def electrostatic.force(q1, q2, r):
2     """
3         Parameters:
4             - q1 (float): The charge of the first object in coulombs (C).
5             - q2 (float): The charge of the second object in coulombs (C).
6             - r (float): The distance between the two charges in meters (m).
7
8         Returns:
9             - float: The magnitude of the electrostatic force in newtons (N).
10
11        import math
12
13        # Coulomb's constant (k) in N·m²/C²
14        k = 8.9875517923e9
15
16        # Calculate the force using Coulomb's law
17        force = k * abs(q1 * q2) / (r ** 2)
18
19        return force

```

PoT + RefTool

Selected Chapter: Chapter.22 Electric Charges and Fields
Selected Tool: Electrostatic Force

Solution:

```

1 def solution():
2     # Given values
3     q1 = 50e-6 # 50 μC in coulombs
4     q2 = -25e-6 # -25 μC in coulombs
5     q3 = 20e-6 # 20 μC in coulombs
6     r = 0.5 # distance between q1 and q3, and q2 and q3
7
8     # Calculate the force between q1 and q3
9     force13 = electrostatic.force(q1, q3, r)
10
11    # Calculate the force between q2 and q3
12    force23 = electrostatic.force(q2, q3, r)
13
14    # Since q1 and q3 have opposite signs, the force is attractive (towards q1)
15    # Since q2 and q3 have opposite signs, the force is attractive (towards q2)
16    # Both forces act in the same direction (towards q1)
17    total_force = force13 + force23
18
19    return total_force

```

Answer: 53.9 ✓

(a) Wrong solution when reasoning with PoT

(b) Correct solution when reasoning with PoT + RefTool

Figure 5: Example case of Gemini-1.5-Pro on a physical problem with (right) and without (left) REFTOOL.

And for tool selection, the consistency is computed as

$$\text{Consistency}_{\text{tool}} = \frac{|\{\text{overlap exists between tools selected by human and model}\}|}{|\{\text{both human and model select tools within the same chapter}\}|}.$$

15

Expert Solution

```

1 def solution():
2     # Constants (SI units)
3     h = 6.62607015e-34           # Planck's constant (J·s)
4     c = 299792458e0             # Speed of light (m/s)
5     R_Inf = 10973731.6          # Rydberg constant (m^-1)
6     e = 1.60217662e-19         # Elementary charge (C)
7     m_e = 9.10938356e-31        # Electron mass (kg)
8
9     # Calculate reduced mass of positronium
10    mu = m_e / 2               # Since both particles have mass m_e
11
12    # Calculate modified Rydberg constant for positronium
13    R_N = (mu / m_e) * R_Inf   # This becomes R_Inf / 2
14
15    # Calculate ground state energy (in Joules)
16    E_t = -h * c * R_N
17
18    # Convert to electronvolts
19    E_t_eV = E_t / e
20
21    return E_t_eV

```

Answer: -6.8

Input

Question: The positron has charge +e and mass equal to the electron mass. Calculate in electronvolts the ground-state energy of positronium—an "atom" that consists of a positron and an electron.

Reference Book Chapter.8 Atomic structure and spectra
8.1 Hydrogenic atoms

The ionization energy, I , of an element is the minimum energy required to remove an electron from the ground state, the state of lowest energy, of one of its atoms in the gas phase. Because the ground state of hydrogen is the state with $n = 1$, with energy $E_i = -hcR_H$, and the atom is ionized when the electron has been excited to the level corresponding to $n = \infty$ (see Fig. 8A.5), the energy that must be supplied is

$$I = hc\bar{R}_H \quad (8A.15)$$

Tool Calculate Ionization Energy

Description: Calculate the ionization energy of a hydrogenic atom using the Rydberg constant and Planck's constant.

Function:

```

1 def calculateIonizationEnergy(R_H=109677, h=6.62607e-34,
2                                c=2.997945e10):
3     """
4         Parameters:
5             - R_H (float): The Rydberg constant for hydrogen in cm^-1 (default is 109677 cm^-1).
6             - h (float): Planck's constant in Js (default is 6.62607e-34 Js).
7             - c (float): Speed of light in cm/s (default is 2.997945e10 cm/s).
8
9         Returns:
10            - float: The ionization energy in joules (J).
11    """
12    return h * c * R_H

```

PoT + RefTool

Selected Chapter: Chapter.8 Atomic structure and spectra

Selected Tool: Calculate Ionization Energy

Solution:

```

1 def solution():
2     # Calculate the ionization energy of a hydrogen atom
3     ionizationEnergy = calculateIonizationEnergy()
4
5     # The ground-state energy of positronium is half of
6     # the ionization energy of a hydrogen atom
7     # in absolute value, but negative, and in
8     # electronvolts
9     e = 1.60217662e-19 # Elementary charge in C
10    groundStateEnergy = -ionizationEnergy / (2 * e) # Convert to eV
11
12    return groundStateEnergy

```

Answer: -6.8 ✓

(a) Wrong solution when reasoning with PoT
(b) Correct solution when reasoning with PoT + RefTool

Figure 6: Example case of Llama-3.1-70b on a chemical problem with (right) and without (left) REFTOOL.

D Case Study

Figure 4 provides the detailed causality case discussed in §4.4, while Figures 5 and 6 show physics and chemistry cases. These cases illustrate how REFTOOL helps LLMs solve problems when standard PoT fails. Notably, in Figure 6, while the selected tool doesn't directly solve the question, it provides relevant knowledge for the LLM to solve the question in a roundabout way (through the ionization energy of hydrogen), which is different from the expert solution but also leads to the correct answer.

E Limitations

Task Scope REFTOOL is primarily designed for knowledge-intensive quantitative problem-solving tasks, such as those in causality, physics, and chemistry. It may not be applicable for general code generation or web-agent tasks, where knowledge isn't the primary challenge.

Domain Knowledge Dependency REFTOOL requires the LLM to possess basic domain understanding to interpret reference materials and generate/select tools effectively. If the model lacks foundational knowledge, for instance, being unable to recognize key concepts or follow technical explanations, the generated tools may be incorrect or unusable. Future work could explore integrating lightweight domain adaptation techniques, enabling REFTOOL to handle unfamiliar domains effectively.

F Prompts

Figure 7 - Figure 11 demonstrate the prompts of REFTOOL.

Prompts of the general-reasoning baselines as shown in Figure 12 and Figure 13 are designed with reference to the QRData and SciBench papers. Prompts for ReAct are the same as the QRData paper.

Prompt for Initial Tool Generation

Please extract the skills from the following text. The text is a section from the chapter {chapter} of the book {book}.

Each skill is a python function with comments of parameters and returns, accompanied by a description and a demonstration example of using the skill.

Please limit the number of skills to 2, and organize the skills in a list of json objects.

Please implement the function, and *do not* leave it as a placeholder. Note the indent in code is 4 spaces. All packages used should be imported inside the function. The function should be self-contained.

If the text contains examples, you are encouraged to use the examples in the text, otherwise please design examples by yourself. The answer to the example question is encouraged to be numerical.

NOTE THAT THE SKILL PYTHON CODE SHOULD NOT BE SPECIFIC TO/ONLY APPLIED TO THE CHOSEN EXAMPLE! PLEASE GENERATE GENERAL SKILL CODE.

The output should be in *complete* json structure, starting with '[' and ending with ']'.

Example output:

```
[{
  "description": "Compute the expected return using the Capital Asset Pricing Model (CAPM) formula.",
  "function": """def expected_return(rf, beta, rm):
    """
Parameters:
- rf (float): The risk-free rate.
- beta (float): The beta of the portfolio.
- rm (float): The return on the market.
Returns:
- float: The expected return.
    """
    return rf + beta * (rm - rf)""",
  "example": {
    "question": "Suppose a stock has the following information. It is listed on the London stock exchange and operates throughout Europe. The yield on a UK 10 year treasury is 2.8%. The stock in question will earn 8.6% as per historical data. The Beta for the stock is 1.4, i.e., it is 140% volatile to the changes in the general stock market. What is the expected rate of return?",
    "solution": """def solution():
# Given values.
rf = 0.028 # The yield on a UK 10 year treasury
beta = 1.4 # The stock is 140% volatile to the changes in the general stock market
rm = 0.086 # The stock in question will earn 8.6% as per historical data
# Calculate the expected return .
result = expected_return(rf, beta, rm)
# Return the result.
return result""",
    "answer": 0.109
  }}]
```

Text:

```
{text}
```

Figure 7: Prompt template for initial tool generation.

Prompt for Tool Refinement

Please revise the skill according to the feedback.

The skill is a python function with comments of parameters and returns, accompanied by a description and a demonstration example of using the skill. Please try to keep the original intent of the skill, and modify the description/function/example to address the feedback.

Note the indent in code is 4 spaces. All packages used should be imported inside the function. The function should be self-contained. The answer to the example question is encouraged to be numerical.

NOTE THAT THE SKILL PYTHON CODE SHOULD NOT BE SPECIFIC TO/ONLY APPLIED TO THE CHOSEN EXAMPLE! PLEASE GENERATE GENERAL SKILL CODE.

The output should be in *complete* json structure as the original skill, starting with '{' and ending with '}'.

Original Skill:

```
{skill}
```

Feedback:

```
{feedback}
```

Figure 8: Prompt template for tool refinement.

Prompt for Chapter Selection

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question using the provided data.

The question can be found below. Given the table of content of the book {book}, please select the chapters that you find useful in solving the question.

Please provide an explanation supporting your choice. At the last line of your response, format the number of the chapters with a list, like '[0]'. Limit the number of chapters to at most 1. Output '[]' if none of the chapters are useful. The last line should start with '[' and end with ']'.

Question:

```
{question}
```

Table of Content:

```
{table_of_content}
```

Response:

Figure 9: Prompt template for chapter selection.

Prompt for Tool Selection within Chapter

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question.

The question and the list of skills can be found below. Please select the skills that you find useful in solving the question

Please provide an explanation supporting your choice. At the last line of your response, format the number of the skills with a list, like '[0]'. Limit the number of skills to at most 2. Output '[' if none of the skills are useful. The last line should start with '[' and end with ']'.

Question:

{question}

List of skills:

{tools}

Response:

Figure 10: Prompt template for tool selection within chapter.

Prompt for Solution Generation

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question below. Please write python code to answer the question. Please encase the Python code within triple backticks. You can use any python library you imported. The returned value of the code is supposed to be the answer. The format of the code should be

```
``python
def solution():
# import libraries if needed

# write code to get the answer

# return answer
````
```

Question:

{question}

Please note that we provide you several functions for the above question. If the functions are related to the question, you are encouraged to use the functions to solve the question. The functions will also be provided in execution, so just call them. \*DO NOT\* define the functions again or import the functions.

Functions:

{tools}

Response:

### Tool Template

Function Description:

{description}

Function:

{function}

Example Question:  
{example\_question}

Example Solution:  
{example\_solution}

Figure 11: Prompt template for solution generation. For evaluation of QRData, the data description and ten lines of the shuffled data are also added to the prompt along with the question.

#### Prompt for PoT

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question below. Please write python code to answer the question. Please encase the Python code within triple backticks. You can use any python library you imported. The returned value of the code is supposed to be the answer. The format of the code should be

```
``python
def solution():
import libraries if needed
```

```
write code to get the answer
```

```
return answer
``
```

Question:  
{question}

Response:

Figure 12: Prompt template for PoT. For evaluation of QRData, the data description and ten lines of the shuffled data are also added to the prompt along with the question.

#### Prompt for CoT

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question below. Please provide a clear and step-by-step solution to answer the question. Do not write any code in your answer. Conclude the answer by stating "The answer is therefore \boxed{[ANSWER]}."

Question:  
{question}

Response:

### Prompt for Direct Reasoning

You are a data analyst and good at quantitative reasoning. You are required to respond to a quantitative question below. Directly answer by stating "The answer is therefore \boxed{[ANSWER]}."

Question:  
{question}

Response:

Figure 13: Prompt templates for CoT and direct reasoning. For evaluation of QRData, the content of the data (shuffled and truncated to the first 3500 tokens) is also added to the prompt along with the question. For evaluation of SciBench, the prompt also states “The question will specify the unit of measurement, which should not be included in the answer. Express the final answer as a decimal number with three digits after the decimal point.”