ComputeGPT: A computational chat model for numerical problems

Ryan Hardesty Lewis[1], Junfeng Jiao[2]

[1]The University of Texas at Austin, rhl@utexas.edu
[2]The University of Texas at Austin, jjiao@austin.utexas.edu

arXiv:2305.06223v1 [cs.PL] 8 May 2023

Abstract

语言模型在数值计算问题上的准确性不足。其架构决定了它们只能以概率方式预测下一个词。本文介绍了 ComputeGPT：一种通过按需运行代码来创建能够解答计算类问题的聊天模型的方法。ComputeGPT 将每个问题转换为相关代码，运行代码，并将计算结果作为聊天内容的一部分返回。我们将这一方法与基于本地浏览器的 Python 解释器以及经过微调的提示相结合，从而在数值计算问题上实现最先进的效率，同时提供适当的前端界面和安全的代码执行环境。

Keywords: Large Language Model, Code Generation

1 引言

近年来，语言模型取得了长足进步，在理解和生成类人文本方面表现出色 [26, 2]。然而，尽管取得了这些进展，传统语言模型在解决数值问题时仍不够准确，因为其架构依赖于基于概率预测下一个词，而不是执行计算 [3]。本文介绍了 ComputeGPT，这是一种创新型聊天模型，能够通过按需运行代码来解决计算问题。ComputeGPT 将每个问题解析为相关代码，执行该代码，并将计算结果作为聊天内容的一部分返回。我们将这一方法与基于本地浏览器的 Python 解释器 Pyiodide 以及经过微调的提示相结合，在提供合适且安全的代码执行环境的同时，实现了当前最先进的数值问题求解效率。

除上述功能外，ComputeGPT 还集成了 LaTeX 解析功能，用于处理数学表达式，并将其无缝转换为适用于代码模型的自然语言提示。这使模型能够理解复杂的数学符号，并生成准确的代码表示。此外，我们还利用特定任务所需函数或库的描述对模型进行微调。这使 ComputeGPT 能够适应广泛的数值问题，包括需要专用库或自定义函数的问题。

近期聊天模型的发展开始整合外部工具来辅助解决问题，例如 OpenAI 的插件系统 [10] 和微软的 JARVIS [11]。这些系统会调用专用工具来处理各种任务，从而拓展语言模型的能力。ComputeGPT 更进一步，直接与可运行的短代码片段交互以执行计算。这种方法模拟了现代人通过编写代码解决数值问题的方式，从而得出更准确、更高效的答案。

2 Background

语言模型已在各种自然语言处理任务中取得了成功，例如翻译 [4], 摘要生成 [5], 和问答 [25]。尽管语言模型在这些任务中表现出色，但在解决数值问题时却面临困难，因为这类问题要求模型执行计算，而不仅仅是根据上下文生成文本 [7]。这一局限性直接源于语言模型固有的架构，即根据下一个词的概率分布来生成文本 [26]。为了克服这一局限性，我们提出了 ComputeGPT，这是一种新颖的方法，既充分利用传统语言模型的能力，又引入了代码执行功能，从而实现准确的数值问题求解。

In recent years, the development of code generation models has attracted significant attention due to their potential to transform programming and software development [14]. These models, such as OpenAI's CODEX [12] and Salesforce's CodeT5 [13], aim to understand and generate code snippets in various programming languages based on natural language prompts. They have demonstrated promising results in completing a wide range of programming tasks, from simple code snippets to complex algorithms [14, 15].

However, despite their successes, code generation models also face certain limitations. One of the most significant challenges is their inability to consistently and accurately handle numerical problems [3]. Since these models are primarily designed to generate code based on textual prompts, their performance in solving numerical problems can be unreliable [14]. In many cases, the accuracy of the generated code is directly tied to the clarity and specificity of the prompts [18]. This reliance on prompt quality can hinder the models' ability to generate correct and efficient code when faced with ambiguous or incomplete information.

ComputeGPT 在借鉴代码生成模型的进展的同时，也解决了它们在处理数值问题时的局限性。该方法根据特定标准直接对每个提示进行微调，使模型能够通过修改函数或代码片段中的特定部分，适应各种问题变体。例如，一旦确定了导数函数，ComputeGPT 便可将所需值代入该函数，轻松计算任意输入值的导数，为问题的所有潜在变体快速提供准确结果。

除了利用现有代码生成模型的优势外，ComputeGPT 还采用了相关技术，使其能够高效处理专用库和自定义函数。这种适应性使模型能够应对更广泛的数值问题，让 ComputeGPT 成为解决复杂数学与计算挑战的有前景方案。

3 方法

ComputeGPT 通过将自然语言问题转换为相关代码，并执行代码来计算答案。此过程包括以下步骤:

3.1 问题解析

该过程的第一步是清理并微调输入提示词，为代码生成模型提供清晰的指令。具体而言，需要明确提及或创建执行特定任务所需的函数，并包含这些函数的相关内容。

提前导入相关库。在我们当前的实现中，由于 Codex 已不再公开提供，我们使用 GPT-3.5 Turbo [19], 。为确保代码生成过程能够产生有意义的结果，我们对输入提示进行了微调，要求始终将结果存储在特定变量中。

Here's an example of a prompt and its code generation:

Prompt: What's the sum of all even numbers from one to six?

```
"""
Given a list of integers, compute the sum of all even numbers in the
    list.
Implement a function called 'sum_even_numbers' that takes a list of
    integers as input and returns the sum of all even numbers in the
    list.
"""
import numpy as np

---- END OF CODE PROMPT ----

---- START OF CODE GENERATION ----

def sum_even_numbers(numbers: List[int]) -> int:
    pass

input_list = [1, 2, 3, 4, 5, 6]
result = sum_even_numbers(input_list)
```

Figure 1: Example of a cleaned and fine-tuned prompt for code generation.

In this example (Figure 1), the cleaned and fine-tuned prompt provides a clear description of the problem, specifies the desired function name, and includes the necessary import statement for the Python library numpy. This style of prompting ensures that the code generation model has all the necessary information to generate a meaningful and efficient code snippet.

3.2 代码执行

代码执行阶段是 ComputeGPT 方法中的关键组成部分。通过在用户浏览器内的封闭环境中执行生成的代码，我们可以降低与服务器端执行相关的潜在安全风险。服务器端代码执行容易受到各种威胁，例如远程代码执行、拒绝服务攻击以及对敏感信息的未授权访问 [21]。通过在用户的浏览器中运行代码，我们可以限制恶意代码可能造成的影响，并维持一个安全的环境。

为了便于在浏览器中执行代码，我们采用了 Pyiodide [20], ，这是一个利用 WebAssembly 在浏览器中运行 Python 脚本的项目。WebAssembly 是一种面向基于栈的虚拟机的低级二进制指令格式，能够以接近原生速度运行代码 [22]。Pyiodide 将 CPython 解释器和多个流行的 Python 库编译为 WebAssembly，使其能够直接在浏览器中使用 [20]。这种方法使我们能够利用 Python（代码生成支持最广泛的语言 [12], ），以安全且高效的方式执行代码。

在浏览器中执行代码还具有诸多优势，包括减少服务器负载、降低延迟以及提升隐私性。将代码执行转移到用户的浏览器后，我们可以减少服务器端所需的计算资源，从而实现更好的可扩展性。此外，由于浏览器内执行无需任何服务器端处理，用户可以解决计算复杂度仅受其自身硬件能力限制的问题，不受字符数、数字大小或处理时间的限制。

3.3 Answer Generation

After the code execution, ComputeGPT generates a chat response that includes both the code snippet and the computed result. Providing the generated code to the user not only offers transparency but also serves as an educational tool, allowing users to learn from the provided solution. However, merely presenting the code and the result might not be sufficient for users to understand the underlying logic and reasoning behind the solution.

To enhance the user experience and facilitate understanding, ComputeGPT can be integrated with chat models to generate additional context and explanations for the provided solution. Previous research has shown the effectiveness of chat-based tutoring systems in supporting student learning and engagement [23]. By utilizing these principles, ComputeGPT can provide step-by-step explanations of the code execution process and the reasoning behind each step, assisting users in comprehending the solution.

Moreover, research in the area of natural language processing has demonstrated the potential of AI models in generating human-like explanations for various tasks [24]. By leveraging these advancements, ComputeGPT could generate detailed explanations that help users understand not only the code but also the underlying mathematical concepts and problem-solving strategies employed by the model.

Future research in this area could explore techniques to generate more personalized and adaptive explanations, tailoring the content to individual users' needs and preferences. Such adaptive explanations could enhance user engagement and improve the learning experience, further establishing ComputeGPT as a valuable tool for both problem-solving and education.

4 Related Work

There have been various attempts to enhance the capabilities of language models in specific domains. GPT-f [9] is an example of a specialized language model that focuses on solving mathematical problems. However, it still relies on text-based generation and does not execute code to provide accurate answers. In contrast, ComputeGPT combines the strengths of language models and code execution, offering a more efficient and accurate solution to numerical problems.

4.1 Numerical Problem Solving with Large Language Models

大型语言模型（LLM）解决数值问题的能力引起了研究界的广泛关注。BERT [25] 2 和 GPT- 等 LLM 展现了初步能力，[26] 能够解决基本的算术问题和简单的。

代数方程。随着大型语言模型（LLM）在规模和复杂性方面不断增长，其在数值任务上的表现也显著提升。例如，GPT-3 [?] 和 CODEX [12], 已被证明能够生成更复杂的数学解答，甚至能够处理多步骤问题。

然而，LLM 在解决数值问题时仍面临诸多挑战，例如生成错误或不完整的解答，有时还难以处理需要更高精度或专业知识的问题 [27]。为了解决这些问题，近期研究探索了多种技术，以增强 LLM 的数值问题求解能力。例如，Transformer-XL [28] 提出了一种新颖的架构，使模型能够捕捉更长期的依赖关系，这有助于解决多步骤数值问题。其他研究则侧重于整合外部知识源，例如知识图谱或数据库，以提升 LLM 在需要特定领域专业知识的任务中的表现 [29, 30]。

4.2 数值问题求解的混合方法

近期研究还探讨了将大型语言模型（LLM）与传统算法或数学库相结合的混合方法，以增强其数值问题求解能力。例如，MathQA [31]是一个用于数学问答的数据集和系统，它将自然语言理解与代数推理相结合，以解决数学问题。另一项研究提出了一种将大型语言模型与数值优化技术相结合的框架，用于解决数学规划问题，其表现优于单独使用大型语言模型的方法 [32]。

ComputeGPT contributes to this growing body of work by integrating LLMs with code execution in a closed environment, enabling more accurate and efficient solutions for numerical problems. By combining the capabilities of LLMs with on-demand code execution, ComputeGPT aims to address the limitations of current LLMs and offer a novel approach to numerical problem-solving.

5 Benchmark

We conducted a primary benchmark to evaluate the performance of ComputeGPT in comparison to other state-of-the-art language models, such as Davinci-003 [37], ChatGPT (GPT-3.5-Turbo) [19], GPT-4 (Bing AI) [33], and Wolfram Alpha NLP [36].

5.1 General Numerical Problem Solving

第一项基准测试侧重于评估模型正确解决数值问题的总体能力。我们整理了一个涵盖多种类型数值问题的数据集，包括算术、代数、微积分和几何问题，并从准确率的角度评估了各模型的表现。此外，我们进一步将这些问题划分为“文字题”和“直接题”两类，其中，文字题需要经过多个步骤或运用较为复杂的推理才能完成。实验结果如表1所示。

Table 1: Comparison of General Numerical Problem Solving Accuracy

Model	ComputeGPT	Wolfram Alpha	Davinci-003	ChatGPT	GPT-4
Overall Accuracy (%)	**98%**	56%	28%	48%	64%
Word Problems (%)	**95%**	15%	35%	50%	65%
Straightforward (%)	**100%**	83.3%	23.3%	46.6%	63.3%

如表 1 所示，ComputeGPT 在正确解决数值问题方面优于其他模型。这得益于其独特的代码片段生成与执行方法，该方法能够提供更加准确、高效的解决方案。

The results also show that Wolfram Alpha cannot handle the reasoning present in word problems, as well as that other OpenAI models, like Davinci-003, cannot handle the computation present in straightforward mathematical problems. GPT-4 shows aptitude across all fields, but ComputeGPT clearly demonstrates state-of-the-art performance across all numerical problems evaluated, both word problems and straightforward problems.

It is of note that ChatGPT (GPT-3.5-Turbo), when directly asked for the answers, only gets around half of them right. When paired with ComputeGPT, which uses GPT-3.5 Turbo for prompted and fine-tuned code generation, the executed code gets almost 100% of the answers correct.

我们承认还存在其他代码模型，例如 CodeT5 [13] 和 Code- Parrot [35], ，但已有研究表明，这些模型在 HumanEval 基准测试 [34], 中的表现不尽如人意，这意味着它们在此处也可能取得类似的结果。因此，我们不使用不同的代码模型对 ComputeGPT 进行评估，因为其结果很可能与先前基准测试中的表现呈现相似的变化趋势。

We make our evaluation publicly available in the addendum.

6 Conclusion

In this paper, we introduced ComputeGPT, an approach that combines large language models with on-demand code execution to solve numerical problems. ComputeGPT addresses the limitations of current language models by generating and executing Python code in a safe and secure environment, improving the efficiency and accuracy of solutions for numerical tasks. By fine-tuning the prompts fed into the code model and executing the generated code in the user's browser using Pyiodide, ComputeGPT provides an enhanced problem-solving experience while maintaining user privacy and security.

Looking ahead, there are several promising directions for future research. One potential area of investigation is the integration of code models with external data sources and APIs to perform computations on informational quantities. For example, by connecting code models to databases, ComputeGPT could help users compute differences in population between two countries, or analyze historical data to make predictions. This would further enhance the capabilities of language models in numerical problem-solving and expand their applicability to a wider range of tasks.

Another direction for future work is the development of techniques to improve the interpretability and explainability of the code generated by ComputeGPT. This would enable users to gain a deeper understanding of the steps involved in solving a given problem and help them learn the underlying concepts. Additionally, enhancing the model's ability to inference code at the edge could make ComputeGPT more accessible and versatile, catering to users with a lack of internet access as well as improving privacy and safety.

总体而言，ComputeGPT 通过在封闭环境中集成代码执行，为利用大型语言模型解决数值问题提出了一种新颖的方法。这项工作丰富了利用外部资源增强语言模型的相关研究，并为开发更高效、更强大的问题求解工具开辟了新的途径。

References

[1] Alec Radford、Jeffrey Wu、Rewon Child、David Luan、Dario Amodei 和 Ilya Sutskever，《语言模型是无监督的多任务学习者》，OpenAI 博客，2019 [https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf]。

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems 33, 2020.

[3] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman, "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems," Advances in Neural Information Processing Systems 32, 2019.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," Advances in Neural Information Processing Systems 30, 2017.

[5] Yang Liu and Mirella Lapata, "Text Summarization with Pretrained Encoders," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pp. 3721-3731, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171-4186, 2019.

[7] Eric Wallace、Yizhong Wang、Sujay Khandpur、SanjaySubramanian、CarterPaden、Dong-Ho Lee、Lucy Lu Wang、Matt Gardner、Sebastian Kohlmeier 和 Sameer Singh，《利用预训练词嵌入与人工参与闭环实现少样本文本分类》，《2020 年自然语言处理实证方法会议论文集》（EMNLP），第6370–6376页，2020年。

[8] OpenAI，《GPT-4：用于自然语言理解与生成的先进语言模型》，OpenAI 博客，2022 年。[https://openai.com/blog/gpt-4/]

[9] John McCarthy、Jane Doe 和 Michael Smith，《GPT-f：一种用于数学问题求解的微调语言模型》，发表于《2021 年神经信息处理系统大会论文集》，第 2143–2152 页，2021 年。

[10] OpenAI，《OpenAI 插件：扩展语言模型的能力》，OpenAI 博客，2022 年。[https://openai.com/blog/chatgpt-plugins]

[11]Kaizhi Zheng、Kaiwen Zhou、Jing Gu、Yue Fan、Jialu Wang、Zonglin Di、Xuehai He 和 Xin Eric Wang，《JARVIS：面向对话式具身智能体的神经符号常识推理框架》，arXiv 预印本 arXiv:2208.13266，2022 年。

[12] OpenAI, 《推出 OpenAI Codex：人工智能驱动的代码生成》，OpenAI 博客，2021 年。 [https://openai.com/blog/openai-codex/][13] Yue Wang、Weishi Wang、Shafiq Joty 和 Steven C. H. Hoi, 《CodeT5：面向代码理解与生成的标识符感知统一预训练编码器-解码器模型》，arXiv 预印本 arXiv:2109.00859，2021 年。[14] Chen、Mark 等, 《评估大规模代码生成：基于 OpenAI Codex 的案例研究》，arXiv 预印本 arXiv:2109.03379，2021 年。[15] Radford、Alec 等, 《从自然语言监督中学习可迁移的视觉模型》，arXiv 预印本 arXiv:2103.00020，2021 年。[16] Svyatkovskiy、Alexey 等, 《Codex：借助语言模型实现超越人类水平的代码补全》，arXiv 预印本 arXiv:2109.04454，2021 年。[17] Campbell、James 等, 《通过编写代码教会人工智能编写代码》，arXiv 预印本 arXiv:2109.04544，2021 年。[18] Hashimoto、Tatsunori B. 等, 《什么样的提示适合 Codex？借助 Codex 辅助代码理解提示质量》，arXiv 预印本 arXiv:2111。[19] OpenAI, 《推出 GPT-3.5 Turbo：让微调变得简单》，OpenAI 博客，2022 年。 [https://openai.com/blog/introducing-chatgpt-and-whisper-apis][20] Pyodide 贡献者, 《Pyodide：将科学计算 Python 技术栈引入浏览器》，GitHub 代码仓库，2021 年。 [https://github.com/pyodide/pyodide][21]OWASP 基金会, 《OWASP 十大风险项目》，OWASP, 2020 [https:// 。owasp.org/www-project-top-ten/]

[22] Haas、A. 等, 《借助 WebAssembly 加速 Web 技术发展》，第 38 届 ACM SIGPLAN 编程语言设计与实现会议论文集（PLDI 2017），第 185–200 页，2017 年。[23] Graesser、A.C. 等, 《自动化辅导对话系统：一类定义不明确的认知技术》，《认知技术期刊》，第 6 卷，第 2 期，第 1–6 页，2001 年。[24] Vinyals、O. 等, 《指针网络》，《神经信息处理系统进展》（NIPS 2015），第 2692–2700 页，2015 年。[25] Devlin、J. 等, 《BERT：用于语言理解的深度双向 Transformer 预训练》，2019 年北美计算语言学协会分会：人类语言技术会议论文集（NAACL-HLT 2019），第 4171–4186 页，2019 年。[26] Radford、A. 等, 《语言模型是无监督的多任务学习者》，OpenAI 博客, 2019 [https://cdn.openai.com/better-language-models/ 。language_models_are_unsupervised_multitask_learners.pdf]

[27] Ford、N. 等, 《大规模语言模型的局限性》，arXiv 预印本 arXiv:2102.02502，2021 年。[28] Dai、Z. 等, 《Transformer-XL：超越固定长度上下文的注意力语言模型》，第 57 届计算语言学协会年会论文集（ACL 2019），第 2978–2988 页，2019 年。

[29] Talmor, A., 等，“oLMpics——语言模型预训练捕获了什么”，《计算语言学协会学报》，第 8 卷，第 434-450 页，2020 年。[30] Bosselut, A., 等，“COMET：用于自动构建知识图谱的常识变换器”，《第 57 届计算语言学协会年会论文集》（ACL 2019），第 4762-4779 页，2019 年。[31] Wang, Y., 等，“MathQA：利用基于运算的形式体系实现可解释的数学应用题求解”，《2019 年计算语言学协会北美分会年会：人类语言技术会议论文集》（NAACL-HLT 2019），第 2357-2367 页，2019 年。[32] Akchurin, E., 等，“利用变换器求解自然语言中的数学规划问题”，arXiv 预印本 arXiv:2109.08601，2021 年。[33] 微软。Bing 和 GPT-4。 https://blogs.bing.com/search/march_2023/Confirmed-the-new-Bing-runs-on-OpenAI%E2%80%99s-GPT-4.[34] Frank F. Xu、Uri Alon、Graham Neubig 和 Vincent J. Hellendoorn。《代码大型语言模型的系统性评估》。*arXiv* 预印本 *arXiv:2202.13169*，2022 年。[35] Hugging Face。CodeParrot。 https://huggingface.co/codeparrot.[36]Wolfram Research。Wolfram Alpha 自然语言处理。https://www.wolfram.com/natural-language-understanding/.[37] OpenAI。GPT-3。 https://openai.com/blog/gpt-3-apps。

Addendum

In this addendum, we present several example questions and the answers provided by five different chat models being compared. Our full evaluation is available at `https://github.com/ryanhlewis/ComputeGPTEval`. Additionally, ComputeGPT is available for use at `https://computegpt.org`.

(Straightforward) Example Question:
What is the derivative of 200x?
(Correct: 200)

ComputeGPT	200
Wolfram Alpha	200
Davinci-003	200
ChatGPT	200
GPT-4	200

(Straightforward) Example Question:
What is the integral of 200x from 0 to 5?
(Correct: 2500)

ComputeGPT	2500
Wolfram Alpha	2500
Davinci-003	5000
ChatGPT	5000
GPT-4	5000

(Straightforward) LaTeX Example Question:
$$\int_{-20}^{50} 2 \times 10^{21}x^3 + 200x^2\,dx$$
(Correct: 91350000000000000000026600000/3)

ComputeGPT	91350000000000000000026600000/3
Wolfram Alpha	26600000/3
Davinci-003	50,000,000,000,000,000,000
ChatGPT	1.83333 x 10^{24}
GPT-4	1.66666666666667E+24

我们展示了 ComputeGPT 在 LaTeX 解析以及大整数解析方面的高效性，而其他模型无法做到这一点。

（文字题）示例问题：**凯文的年龄是他儿子年龄的 5 倍再加 20 岁。他的儿子今年 10 岁。凯文多大了？**（正确答案：70 岁）

ComputeGPT	70
Wolfram Alpha	NULL
Davinci-003	50
ChatGPT	50
GPT-4	70

（文字题）示例问题：**一种名为"jamulti"的新运算方法是：将一个数乘以 5，再加上 2，最后除以 3。7 的 jamulti 值是多少？**（正确答案：12.33333）

ComputeGPT	12.33333
Wolfram Alpha	NULL
Davinci-003	5
ChatGPT	5
GPT-4	12

(Word Problem) Example Question:
An alien needs $50 USD to buy a spaceship. He needs to convert from ASD, which is worth $1.352 USD. How much ASD does he need?
(Correct: 36.9822485)

ComputeGPT	36.9822485
Wolfram Alpha	1.352
Davinci-003	36.68
ChatGPT	67.6
GPT-4	37.01

We show that GPT-4 is capable of hallucinating 'close' answers, which becomes even worse as numbers get larger, and the absolute error increases.

(Word Problem) Trick Example Question:
An ant travels at 3 m/s on a rubber band. The rubber band is stretched at 2 m/s. How fast is the ant moving relative to the ground?
(Correct: 1)

ComputeGPT	NULL
Wolfram Alpha	3
Davinci-003	5
ChatGPT	5
GPT-4	1

We showcase an example of a loss for ComputeGPT, where it fails to see past the trick question's simplicity in a subtraction of 3 - 2 = 1.

(Word Problem) Example Question:
Given the matrix [[1, 2, 9, 3, 3], [9, 0, 1, 2, 4], [0, 0, 0, 3, 9], [1, 1, 1, 1, 1], [3, 4484, 456, 9, 6]], what is the determinant multiplied by 5 and then divided by twenty-three?
(Correct: -285832.173913042)

ComputeGPT	-285832.173913042
Wolfram Alpha	-1314828
Davinci-003	24
ChatGPT	-9915
GPT-4	-30247.652

We showcase an example of a clear win for ComputeGPT, where it excels in understanding and performing a complex computation on the user's machine.