Mathematics, word problems, common sense, and artificial intelligence

Ernest Davis
Dept. of Computer Science
New York University
New York, NY 10012
davise@cs.nyu.edu

December 5, 2023

Abstract

The paper discusses the capacities and limitations of current artificial intelligence (AI) technology to solve word problems that combine elementary knowledge with commonsense reasoning. No existing AI systems can solve these reliably. We review three approaches that have been developed, using AI natural language technology: outputting the answer directly, outputting a computer program that solves the problem, and outputting a formalized representation that can be input to an automated theorem verifier. We review some benchmarks that have been developed to evaluate these systems and some experimental studies. We discuss the limitations of the existing technology at solving these kinds of problems. We argue that it is not clear whether these kinds of limitations will be important in developing AI technology for pure mathematical research, but that they will be important in applications of mathematics, and may well be important in developing programs capable of reading and understanding mathematical content written by humans.

1 Introduction

A central aspect of the mathematics learned in grade school is understanding how math applies in real-world situations and how mathematical analysis can be used to answer questions about real things. Consider, for instance, the following two elementary problems:

> **Problem 1:** George has seven pennies, a dime, and three quarters. Harriet has four pennies and four quarters. First, George gives Harriet thirty-one cents in exact change; then Harriet gives him back exactly half of her pennies. How much money does George now have?
>
> **Problem 2:** You have an empty cylindrical open container whose inside has a diameter of 8 centimeters and a height of 20 centimeters. and a pitcher with 200 ccs of water. You first empty the pitcher into the cylinder, then put a solid rock cube, 4 cm on a side, into the container so that it is sitting flush against the bottom of the container. What is the height of the water in the container?

In recent years, artificial intelligence (AI) technology has made extraordinary progress in a wide range of domains, such as playing recreational games, tagging and generating images, transcribing speech,

and translating between languages. In particular, in recent years AI has achieved extraordinary success at certain kinds of language tasks, and, as discussed throughout this volume, noteworthy successes at a range of mathematical tasks.

One might reasonably suppose, therefore, that problems as simple as problems 1 and 2 above must be well within the capacities of the current AI technology. However, that is not at all the case. *As of May 2023, there does not exist any AI that can reliably solve elementary mathematical word problems.* The goal of this article is to elaborate on that observation; to describe at a very high level how the current technology works and what it can and cannot do. The paper concludes (section 4) with a brief and speculative discussion of the relevance of these limitations to the prospect for using AI as a tool for research mathematics.[1]

2 Math word problems and world knowledge

Solving word problems like (1) and (2) above requires knowledge of three kinds. First, and most obviously, there is the elementary math involved: elementary arithmetic and solid geometry, respectively. Second, one has to know the language; if these questions, as printed above in English, are presented to a speaker who knows only Polish, then they will not be able to answer, no matter how well they know the math. Third, and most easily overlooked, each of these requires basic knowledge about the world, and an understanding of how the world is characterized mathematically. In problem 1, you have to know the value of U.S. coins, but more fundamentally, you have to understand the dynamics of possession and giving: If A gives X to B, then the result is that B now has X and A no longer has X. In problem 2, you have to understand the basic physics of liquids and solids: solids maintain their shape over time, liquids maintain their volume but not their shape, solids and liquids do not overlap and so on.

In many simple problems, a significant part of the real-world understanding required is purely "common sense" — the basic understanding of the realities of daily human existence that is shared by all people past early childhood.[2] When people work on problems of this kind, the commonsense knowledge and reasoning involved is often so obvious that it goes entirely unnoticed; what is difficult is finding the mapping to mathematics and carrying out the mathematics.

Other forms of world knowledge can also enter into solving math word problems. *Common knowledge*, such as the value of U.S. coins in problem 1, can be culturally dependent and is sometimes taught explicitly in schools, but is universal among adults in the society under discussion. *Encyclopedic knowledge* is specific facts about particular entities, such as the knowledge needed to answer the question, "How much older, in days, was George Washington than Abraham Lincoln?" *Expert knowledge* is knowledge held by experts but not by lay people. Obviously these categories are vague and there is no point in trying to delimit them with any precision.

A problem that calls on the combination of mathematics with world knowledge will generally involve other cognitive capacities as well. If it is posed in language, then it requires comprehension of the

[1] Footnote added December 2023: This paper was written in May 2023. The state of the art has advanced extremely rapidly, and though the general discussion here is still valid, some of the specific examples and assertions are out of date. It did not seem worthwhile to rewrite the paper, as the details in the new version would likewise again be outdated in six months. However, on the suggestion of the editors, let me here list a few important works that have appeared since. Experiments testing the capacity of systems that integrate ChatGPT with Python or with Wolfram Alpha, as discussed in sections 3.4.1 and 3.4.3 are reported in [13, 29]. A systematic study of the inability of large language models including GPT-4 to solve word problems involving simple linear orderings of a small number of objects is reported in [2]. For some interesting recent results on autoformalization techology (section 3.4.2), see [1, 20].

[2] This characterization of common sense has obvious significant problems and limitations, but it will suffice for our discussion here. An in-depth discussion will be found in [12].

language. If the problem statement includes mathematical notation or other technical notation (musical, chemical, etc.). then that notation must be understood. If an embodied agent must solve a problem in the physical world, then understanding it requires perception, most commonly vision. If the problem involves a diagram or graph, then it requires vision plus an understanding of the graphical conventions. If an embodied agent must carry out a task in the real world, then that requires perception and manipulation.

It will be convenient to distinguish a number of categories of math problems. A *symbolic mathematics problem* is one posed in mathematical notation with minimal use of natural language (i.e. English, Russian, etc.), e.g. "Solve $x^3 - 6x^2 + 11x - 6 = 0$". A *word problem* is a mathematical problem with more than minimal use of natural language. A *purely mathematical word problem* is a word problem that makes minimal reference to any non-mathematical concepts, e.g. "Find a prime number p such that $p + 36$ is a square number." "What is the volume of a regular icosahedron with diameter 5?" A *real-world word problem* is a word problem whose solution requires the use of non-mathematical knowledge of some kind. A *commonsense word problem* (CSW) is a word problem that involves significant use of commonsense knowledge and perhaps also common knowledge, but not encyclopedic or expert knowledge; this is the most important category for our purposes. Finally, an *elementary commonsense word problem* is a CSW that requires only elementary math, however one chooses to draw that line (generally elementary school math or high school math). Again, obviously, the lines delimiting these categories are vague.

Real-world word problems generally are common in grade-school math classes, rarer in introductory college courses, and extremely rare in advanced college math courses. They remain common in advanced courses in other topics that use mathematics for applications; there, generally, the knowledge involved is primarily expert knowledge, though commonsense knowledge often forms an underlying and largely unobserved foundation.

3 Artificial intelligence for word problems

The challenge of developing AI systems that solve mathematical word problems has been studied since Daniel Brobrow's STUDENT system of 1964 [3]. Skipping over the history of the problem over the subsequent 50 years, we will focus here on recent systems that apply AI technology to math word problems.

Progress in AI over the last twenty years has been almost entirely driven by *machine learning* (ML). Broadly speaking, in machine learning, a general-purpose computational architecture, with little or no built-in knowledge of the domain or the task, is trained on a collection of relevant data, and it finds underlying patterns in the data that it can use[3] to carry out the task with some degree of accuracy. Almost all current AI programs are built around machine learning mechanisms. In many cases, the AI program consists of a single module with a generic ML architecture. formatting at the front and back ends. A notable example was the AI that were used in 2021 to suggest a new theorem in knot theory, as described0 in [8] and [10]. The AI used in that project was an off-the-shelf deep-learning architecture; what was difficult in that project was preparing the training data and interpreting the results. Other AI systems require a complex, hand-crafted architecture that integrates multiple modules, some trained with ML, some built with traditional approaches to programming. Much recent work in robotics is in this category. Another recent example was the Cicero system, which plays the game Diplomacy [5].

[3] I am including supervised learning, unsupervised learning. and reinforcement learning in this general category.

3.1 Large Language Models

As discussed above, solving CWPs requires combining commonsense reasoning, language abilities, and mathematics. For the last five or six years, both natural language processing (NLP) and (more surprisingly) commonsense reasoning in AI has been entirely dominated by one particular technology known as large language models (LLMs) (in current AI parlance "model" means "computer program."). Almost all recent work on math word problems has an LLM at the core.[4]

Large language models are the most recent development of thirty years of study of how neural network AI technology (now often called "deep learning") can be applied to language-oriented tasks. We will not attempt a technical description of their workings here (see [30] for an introduction and [16] for more detail), but we will describe the characteristics most important for our discussion here.

The user of an LLM inputs a text, called the prompt; the LLM's task is to generate a plausible continuation of the prompt. The LLM generates its response word by word; or, more precisely, token by token. (A token is an element of text: a word, a fragment of a word, a punctuation mark, a mathematical symbol, etc.) That is: A front-end program (tokenizer) divides the user's input into a sequence of token $w_1 \dots w_n$. The LLM first computes the most probable (or, depending on the settings, a reasonably probable) next token w_{n+1}. It next computes the token w_{n+2} that is most likely to follow $w_1 \dots w_{n+1}$. And so on; it repeatedly computes the most probable next token, until it reaches a halting state. (Chatbots such as ChatGPT have some additional mechanisms so that they engage in dialogue rather than monologues.) The problem of generating a probable next token after a given string is known as the "language modeling" task.

The generation of text is thus driven by the conditional probability of producing token w_n following $w_1 \dots w_{n-1}$, $P(w_n \,|\, w_1 \dots w_{n-1})$. The AI computes this in terms of a function $f_{\vec{\alpha}}(w_1 \dots w_n)$ which is implicitly encoded in the network structure of the AI. This function $f_{\vec{\alpha}}$ is controlled by a parameter $\vec{\alpha}$, a real-valued vector. In current systems, the dimension of $\vec{\alpha}$ ranges between about 10^9 and 10^{12}.

The training set for the LLM is a vast corpus of text ($10^{10} - 10^{13}$ tokens) downloaded from the Web. Detailed accounts of the content of these corpora have not, in general, been published;[5] but it is safe to say that the majority consists of English language documents of various kinds, but that the training corpora also include substantial quantities of texts in other languages, of software written in popular computer languages, of mathematics in mathematical notation, of images, and other kinds of data.

Training the LLM consists in finding the value of the parameter $\vec{\alpha}$ such that the probability function $P(w_n \mid w_1 \dots w_{n-1}) = f_{\vec{\alpha}}(w_1 \dots w_n)$ matches the training set T as closely as possible. That is, there is an error function $E_T(\vec{\alpha})$ that measures the discrepancy between the system's predictions and the actual sequences in T when the AI uses $f_{\vec{\alpha}}$ to generate predictions. The value of $\vec{\alpha}$ is chosen to (approximately) minimize $E_T(\vec{\alpha})$. This minimization uses a gradient descent algorithm. The training process requires enormous amounts of computation — months of computation time on large networks of computers — as well as a substantial amount of expert human labor. It is feasible only for large AI labs, not for individual academic researchers or small companies.

This training procedure for an LLM is carried out only once, when the LLM is created. However, it is possible to further "fine-tune" an LLM to a particular task by further training on texts specifically relevant to that task.

The underlying architecture of the LLM, which determines the function $f_{\vec{\alpha}}$, and the training procedure are both very general in structure and are built to carry out prediction for essentially any kind of input string with some kinds of regularities. They do not reflect *any* knowledge of the

[4] An exception is Wolfram Alpha, discussed in section 3.4.3 below.

[5] The BigScience large language is an important and admirable exception. `https://bigscience.huggingface.co/blog/model-training-launched`

characteristics of natural language generally, of any particular language, of the external world that language describes, of the various uses of language, or of any task other than string prediction. *All* the knowledge of language, its meaning, and its content that the system possesses is in terms of how best to carry out prediction over the training set.

As with all AI systems based on neural networks/deep learning, the function $f_{\vec{\alpha}}$ is *opaque*, in the sense that it is extremely difficult to find a relation between the vector $\vec{\alpha}$ and the behavior of the system, either on a particular example or in general. It is therefore generally impossible to incorporate any of what is known about language, the world, or mathematics into the system other than through training. It is also impossible to debug errors as is done in "conventional" computer programming. All that can be done is to retrain the system from scratch (or, in rare cases, to add a hand-crafted patch at the back-end to deal with the problem.)

In earlier NLP research, different tasks — question answering, chatbots, summarization, information extraction, translation and so on — were each handled separately by systems built specifically for that purpose. However, in the last three or four years, it has turned out that language modeling tasks can serve as a basis for all of these; indeed for any task where the input and output are in language [4]. For many language-oriented tasks, LLMs can be as or more effective as software systems carefully designed for the task.

It has also turned out, quite unexpectedly, that the quality of answers generated by the very large LLMs can often be significantly improved by including a handful of examples of the kind of output desired in the prompt, or even by including general directives such as "Let's think about this step by step". Thus, if you want it to translate from German to English, it helps to include a few examples of translation in the prompt; if you want it to answer mathematical questions of a particular kind, it helps to include a few examples of that kind of problem in the prompt. This is known as "few-shot prompting"; it was first observed in the LLM GPT-3, released in 2020 [6]. It has given rise to a new area of study: "prompt engineering", the creation of prompts that guide LLMs to correct outputs of the proper form.

LLMs are prone to so-called *hallucinations*: since they have no sense of underlying reality, they often generate text that reads smoothly and is presented in a completely confident tone but which is actually incoherent or false. Depending on circumstances, a human reader can find this amusing, frightening, annoying, or misleading. It can therefore be very dangerous to trust the text produced by an LLM without checking its veracity

3.2 A fallacious proof generated by GPT-4

Because large language models generate output one word at a time, they have no ability either to plan ahead what they are going to say overall or to go back and correct errors. Thus a slight misstep at one point can lead them down a wrong path. When trying to generate a mathematical proof, this can be catastrophic.

Table 1 shows a fallacious proof of Fermat's little theorem generated by Bing Chat, a chatbot that is publicly accessible via the Microsoft Bing search engine. As of May 2023, Bing Chat is powered by the LLM GPT-4.

This "proof" is clearly a misremembering of a correct proof by induction; one can easily prove the theorem in the form $a^p = a \bmod p$ by induction on a using the binomial theorem.[6] GPT-4 gets it almost right, but, since it tries to prove the theorem in the form $a^{p-1} = 1 \bmod p$ directly, it is led into increasing absurdity; first claiming that p divides $p - 1$ (and, implicitly, that all the binomial coefficients are divisible by $p - 1$) and then failing to notice that the conclusion it thinks it has

[6]Thanks to Aravind Srinivasan for pointing this out.

proved is different from the premise needed for the induction.

In a neutral context, GPT-4 would certainly not confuse the formula $a^{p-1} = 2 \bmod p$ with the formula $a^{p-1} = 1 \bmod p$ or accept the claim that $p - 1$ is divisible by p. But it does not consider those to be impossible propositions, merely unlikely sequences of tokens; and apparently, given the high probability it assigns to the idea that it can complete the proof in a way that resembles the proofs it has seen, it is willing to accept these steps despite their inherent low probability.

3.3 LLMs' abilities at common sense, math, and language

Outside a mathematical context, LLMs achieve significant levels of success at commonsense reasoning, though they are certainly not reliable, and they have been steadily improving over time. Unquestionably LLM-based AIs are currently the most powerful and general, publicly available[7] technology for commonsense reasoning that have been built.

LLMs also have displayed some mathematical abilities. GPT-3, when originally released in 2020 [6], was tested on simple integer arithmetic problems. When prompted with a few correct examples, it achieved 100% accuracy on two-digit addition (e.g. "What is 35 plus 72?") with gradually diminishing accuracy as the number of digits increases, though only 21.3% accuracy on problems involving two arithmetic operations and one-digit numbers (e.g. "What is (2 + 4) * 6?"). More recent systems have achieved significantly higher levels of success [26] but the general pattern remains. LLMs also can often, though unreliably, generate correct mathematics at a higher level as illustrated by the "proof" of Fermat's little theorem above in table 1. Though the proof overall is wrong, many of the individual components — the form of a proof by induction, the binomial coefficients, to the extent that they are given — are correct.

Finally, LLMs are fundamentally language technology, and in many respects their purely linguistic abilities are extraordinary and flawless. The current generation of LLMs can produce essays several paragraphs long that, in their language use, are often indistinguishable from capable human writers. I have only once seen them make a grammatical error in writing modern English (a failure of subject-verb number agreement in a tricky case).

LLMs thus have at least some facility at the three components of CSW: common sense, math, and language. However, putting these together is a different matter.

3.4 Using an LLM to solve a word problem

There are three major approaches to using an LLM to solve mathematical word problems. They all take the problem, possibly supplemented with a prompt as input to the LLM; they differ in the nature of the output.

The most straightforward approach is simply to ask the LLM to produce the answer to the problem. This can work well with simple mathematics, though it breaks down when more complicated math is required.

[7] The CYC project [19]. begun in 1985 and still ongoing, aims at encoding commonsense knowledge in a quasi-logical symbolic system and using automated inference to do commonsense reasoning. It is certainly large: it contains some millions of "concepts" and tens of millions of propositions, and it represents several hundred person-years of expert human labor. However, the contents are proprietary: no systematic description, evaluation, or benchmark test has ever been published, and the scientific literature on its workings is very small. Consequently, it is impossible for outsiders to know what its capabilities are.

User: Can you give me a proof of Fermat's little theorem?

BingChat: Sure! There are two main ways to prove Fermat's Little Theorem; one is by using the **induction method** and the other is by using **group theory**. Which one would you like?

[The group theory proof generated by Bing Chat is based on the theorem that the size of a finite group G is divisible by the order of any element $a \in G$. The proof as presented by Bing Chat is flawed but on the right track]

User: The induction method.

BingChat: Sure! Here's a proof of Fermat's Little Theorem using the induction method.

Fermats Little Theorem states that if p is a prime number and a is an integer not divisible by p, then a^{p-1} is congruent to 1 modulo p.

Proof by Induction:

Let's assume that p is a prime number and a is an integer not divisible by p.

We will prove that a^{p-1} is congruent to 1 modulo p by induction on a.

Base case: When $a = 1$, we have $1^{p-1} = 1$, which is congruent to 1 modulo p.

So the base case holds.

Inductive step: Assume that the statement holds for some integer k, i.e., k^{p-1} is congruent to 1 modulo p. We need to show that the statement also holds for $k + 1$, i.e., $(k + 1)^{p-1}$ is congruent to 1 modulo p.

Expanding $(k + 1)^{p-1}$ using the binomial theorem, we get:

$(k + 1)^{p-1} = k^{p-1} + (p - 1) \cdot k^{p-2} + ... + (p - 1) \cdot k + 1$

Since k^{p-1} is congruent to 1 modulo p by the induction hypothesis, we can replace it with 1 in the above expression:

$(k + 1)^{p-1} = 1 + (p - 1) * k^{p-2} + ... + (p - 1) * k + 1$

Now, all the terms in the above expression except for the first and last terms are divisible by p, since $p - 1$ is divisible by p. So we can simplify the above expression as follows:

$(k + 1)^{p-1} = 2 \mod p$
Thus, $(k + 1)^{p-1}$ is congruent to 2 modulo p. This completes the inductive step.

By induction, we have shown that for any integer a not divisible by p, a^{p-1} is congruent to 2 modulo p. This completes the proof of Fermat's Little Theorem using the induction method.

Table 1: A false proof generated by Bing Chat, powered by GPT-4. Experiment by the author, 5/20/2023.

3.4.1 Translating a word problem into executable code

A second approach is to have the LLM produce computer code that can then be executed to compute the answer. Generating code is, in fact, the most successful practical applications of LLMs to date. AI programs like OpenAI's Codex [7] and GitHub CoPilot [24] are used by professional programmers as assistants to help write code; some experts have enthusiastically reported that they save significant time and effort. These AIs are particularly effective at generating code of standard but non-obvious form, such as finding the names and arguments to library functions based on a verbal description; they thus save the programmer a tedious search through documentation. They are much less reliable in generating programs of significant length of complex structure.

Drori et al. [14] used Codex to generate Python code that would compute the answers to a collection of problems drawn from undergraduate math courses at MIT and Columbia. Their system involved the following steps. First, the problem was modified by a hand-crafted automated front end into a more standardized natural language form; generally this required only adding a few stock phrases such as "Use sympy". Second, the modified problem was then input to Codex, which output Python code. Finally, if the Python code did not give the correct answer, the system searched for similar examples, with solutions, in the training data. These were used to create a few-shot prompt, and the problem was attempted again with the new prompt.

Drori et al. claimed a success rate of 81%; however, for a number of reasons, that figure is highly misleading [11]. Two successful examples, chosen as illustrations in [14], are shown in table 2.

A few points about these examples should be noted. In both examples, the system ended up altering the original specification: in the first, it did not use the definition of the derivative; in the second, presumably the author of the problem did not intend that simulation should be used. In the first example, Codex is carrying out an almost trivial transformation from the language expression to a hand-constructed sympy library function. In the second example, the transformation from language to code is not at all trivial, but it is safe to say that Codex is here closely imitating some code that was in its training set.

An inherent limitation of this approach is that it carries out the translation in strict order: It first generates the code, then executes it. Presumably all the numerical calculations are done in the execution part. If so, then the code must work for all numerical values of the parameters. This can end up turning an easy problem into a well-nigh impossible one (or even an absolutely impossible one). Many commonsense problems are easy for some of the numerical parameters involved but extremely difficult for other values. If the problem is stated with values for which the problem is easy, then generating the code for the difficult cases is pointless, but may well be impossible to avoid in this approach. As one example of many, consider the problem of determining whether an object of complex shape fits through a cylindrical hole. If the object is small enough (e.g. its diameter is less than the diameter of the cross section of the hole) , then trivially, it can fit through. If it is too large (e.g. it contains a sphere whose diameter is larger than the diameter of the cross section of the hole) then, trivially, it cannot fit through. Otherwise the calculation is more difficult, and for tight fits, it can be *much* more complicated. Certainly the output code can be written so that, in simple cases, the code for difficult cases is not *executed*, but the point is, if the problem at hand is an easy case, it should not be necessary to even *generate* the code.

3.4.2 Translating word problems into logical specifications

A third approach to solving word problems is built around *autoformalization*, the process of automatically translating mathematical specifications given in natural language into a formal specification that then can be given to a formal verification system. Wu et al [31] used two LLMs, PaLM and Codex, to translate word problems into the formal notation used by the interactive proof assis-

Question: Find the derivative of the function using the definition of a derivative.
`f(x) = (x**2-1)/(2*x-3).`

Input: Using sympy find the derivative of the function using the definition of a derivative.
`f(x) = (x**2-1)/(2*x-3).`

Program:

```
import smpy as sp

x = sp.Symbols('x')

f = (x**2-1)/(2*x-3)

print(sp.diff(f,x))
```

Answer: `2*x/(2*x-3)-2*(x**2-1)/(2*x-3)**2`

Problem: One generates a number x from a uniform distribution on the interval [0,θ]. One decides to test H0: $\theta = 2$ against HA: $\theta \neq 2$ by rejecting H0 if x$\leq$0.1or x$\geq$1.9. Compute the probability of a type 1 error.

Input: One generates a number x from a uniform distribution on the interval [0,θ]. One decides to test H0: $\theta = 2$ against HA: $\theta \neq 2$ by rejecting H0 if x$\leq$0.1or x$\geq$1.9. Using simulation compute the probability of a type 1 error.

Program

```
import numpy as np

def generate_x(theta)
    return np.random.uniform(0,theta)

def test_hypothesis(x)
    if x <= 0.1 or x >= 1.9:
        return TRUE
    else:
        return FALSE


def simulate_type_1_error(theta,n)
    count = 0
    for i in range(n)
        x = generate_x(theta)
        if test_hypothesis(x):
            count += 1
    return count / n

print(simulate_type_1_error(2,100000))
```

Answer: 0.1

Table 2: Translating word problems into Python code. From [14].

Question Natural language version: "When all the girls at Madeline's school line up in rows of eight, there are seven left over. If instead they line up in rows of four, how many are left over? The final answer is 3." Translate the natural language version to an Isabelle version

PaLM output:

```
theorem
  fixes n::nat
  assumes "n mod 8 = 7"
  shows "n mod 4 = 3"
```

Question Natural language version: "A map of the town that Annie, Barbara, and Charlie live in can be represented by the Cartesian plane. Annie is located at (6, -20) and Barbara is located at (1, 14). They agree to meet at the closest point that is equidistant from their current locations and walk upwards together to get to Charlies location at ($\frac{7}{2}$, 2). How many units upward do Annie and Barbara walk together to get to Charlie? The final answer is 5." Translate the natural language version to an Isabelle version:

Codex Output:

```
theorem
  fixes a b c :: "real \<times> real"
  assumes h0 : "a = (6, -20)
    and h1 : "b = (1, 14)"
    and h2 : "c = (7 / 2, 2)"
  shows "dist a c + dist b c = 5"
```

Table 3: Examples of autoformalization from [31]. The translation in the second example is incorrect, though it is marked in [31] (p. 23) as "correct".

tant Isabelle. Since the training set for the LLMs presumably contains only a comparatively small quantity of Isabelle, they provided long few-shot prompts (different prompts for different areas of mathematics) with ten examples. They tested their systems on examples taken from benchmark collections MATH and MiniF2F, both of which contain mathematical competition problems. They found that the system created a perfect translation 25% of the time. Problems that require finding a specific value as an answer were reworded by adding, "The final answer is A" where A is the correct answer at the end of the natural language formulation. (This is reasonable for testing the adequacy of the translation of α into a formal notation; it would not, of course, be reasonable for testing the capacity of an AI to *solve* the problem.) Table 3 shows two examples of two problems translated into Isabelle.

3.4.3 Wolfram Alpha

Wolfram Alpha[8] is a manually constructed system with a wealth of mathematical algorithms and encyclopedic and expert knowledge from many scientific and sociological. It takes input posed in natural language; however, its natural language abilities are comparatively limited and it has little commonsense reasoning ability.

At least two systems have been built that use GPT4 to translate a user query in to a form that

[8] https://www.wolframalpha.com/

Successes

Problem: A point $\mathbf{p}$ is chosen at random within the 100-dimensional box $\mathbf{B} = [0, 100]^{100}$ following a uniform distribution. What is the probability that the Euclidean distance from $\mathbf{p}$ to the boundary of $\mathbf{B}$ is less than 1?
Answer: $1 - 0.98^{100} \approx 1 - 1/e^2 = 0.8647$

Problem: What is the probability that a randomly-chosen $100 \cdot 100$ matrix, over the finite field F_2, is invertible?
Answer

$$\prod_{i=1}^{100} \frac{2^j - 1}{2^j} \approx 0.289$$

Failures

Problem: Viewed from Vega, what is the angle between Sirius and the Sun?
Correct Answer: 0.0972 radians = 5.6°. **GPT4+Wolfram Alpha:** .005060 degrees.
Problem: Joe says that he lives 10 miles from Lake Michigan, that Beth lives 10 miles from Lake Michigan, and that he and Beth live 100 miles apart. Is it possible that Joe is telling the truth? Answer 'Yes' or 'No'."

Correct Answer: Yes. **GPT4+Wolfram Alpha:** No.

Table 4: Examples of successes and failures on test set problems, from [13]

Wolfram Alpha can accept [13, 29].[9] Each of these has been tested over benchmark collection of math and physics problems across a range of difficulty. As one would expect, the hybrid system works very well when the translation from English to mathematics is straightforward, but on problems where the translation itself requires significant reasoning, the systems (table 4). Problems involving spatial reasoning or reasoning about sequences of events were particularly difficult. There is currently (December 2023) no AI system that works reliably on problems comparable to problems 1 and 2 of section 1.

3.5 Benchmarks

Standard collection of elementary math problems serve as benchmarks to compare the capacities of different AI systems with one another and with humans, to measure progress in the technology, and to serve as targets for research.

Numerous benchmark collections of word problems have been assembled [22], [25] . The mathematical difficulty ranges from elementary school to International Mathematical Olympiad and introductory college math courses. The language use ranges from non-linguistic to moderately complex. Some require no world knowledge; others requires some combination of commonsense, common, expert, and encyclopedic knowledge. Question formats include open form question answering, fill-in-the-blank, multiple choice, and proof construction.

One high-quality collection of 1000 elementary CSPs is SVAMP (Simple Variations on Arithmetic Math Problem) dataset [25]. The problems in SVAMP involve only commonsense and common knowledge; the language is simple; the math required is either one or two arithmetic operations.

[9] This paragraph was entirely rewritten in December 2023, reflecting changes in the state of the art.

Allan brought two balloons and Jake brought four balloons to the park. How many balloons did Allan and Jake have in the park?

Allan brought two balloons and Jake brought four balloons to the park. How many more balloons did Jake have than Allan in the park?

Jack had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now?

Dorothy had 142 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Dorothy have now?

Jack had 142 pencils. Dorothy had 50 pencils. Jack gave 31 pencils to Dorothy. How many pencils does Jack have now?

Table 5: Examples from SVAMP dataset

[User] What is the second letter of the sixth word of the sentence, "Nothing is told of him but that once, abruptly, he asked a question and received an answer"?

[Bing Chat] The second letter of the sixth word of the sentence, "Nothing is told of him but that once, abruptly, he asked a question and received an answer" is "h".

[User] How many words are in the sentence, "Nothing is told of him but that once, abruptly, he asked a question and received an answer"?

[Bing Chat] There are 16 words are in the sentence, "Nothing is told of him but that once, abruptly, he asked a question and received an answer"?

[User] Harry, Ann, Carol, and Ted were playing marbles. Harry had 3 red marbles and 2 blue marbles. Ann had 1 red marble and a white marble. Ted had two white marbles and a green marble. Carol had a blue marble and a green marble. How many marbles were there in total?

[Bing Chat] In total there were 11 marbles: Harry had 5 marbles (3 red and 2 blue), Ann had two marbles (1 red and 1 white), Ted has 3 marbles (2 white and 1 green), and Carol had 2 marbles (1 blue and 1 green).

Table 6: Some tests carried out by the author on Bing Chat, powered by GPT-4, May 19, 2023

The numbers involved are positive integers between 1 and 999. Each example is given in a number of paraphrases and variants, in order to avoid the kinds of artifacts discussed below in section 3.7. Table 5 shows some example problems in SVAMP.

Many of the benchmark collections used to test AI math abilities are taken from tests developed for human students — math course problems, IQ tests, SAT tests, Math Olympiad tests, and so on. However, these generally take for granted that the test-takers have certain basic abilities; this is a safe assumption for humans but not for AIs, whose abilities are very different [9]. Other benchmarks have been developed specifically for AIs, but these tend to focus on some abilities and omit others. Consequently, many categories of problems can go untested.

For example, apparently none of the benchmark sets for mathematical ability tests an AI's ability to count. Surely, an AI system that can solve (some) advanced math problems can count up to six? Not necessarily, as determined in an informal test (table 6).[10] Likewise, many forms of commonsense mathematical inference that draw on a basic of time, space, and physical reasoning are not tested in any existing AI benchmark.

[10]The technology is rapidly improving; GPT3.5, released in December 2022, failed on much simpler examples than those in table 6.

3.6 General caveat

It cannot be assumed that, because an AI does well on a class of problems, it can do well on a seemingly easier problem. As we have seen, the powerful AI GPT-4, which does reasonably well on various simple math tasks, cannot reliably count to six.

Likewise, it cannot be assumed that an AI that can carry out each component of a given task separately can carry them out when they are combined. For example, LLMs in general do much worse on problems that involve two arithmetic operations than those that require one, both in word problems and in purely mathematical problems. As remarked above, in the proof of Fermat's little theorem, GPT-4 both asserted that p divides $p-1$ and confused the statement $a^{p-1} = 2 \bmod p$ with the statement $a^{p-1} = 1 \bmod p$; it would not make either of these mistakes in isolation. The capacities and incapacities of these kinds of programs do not at all resemble those of human beings.

3.7 Artifacts

A persistent problem in ML, in all its applications, is that the patterns that the AI finds do not actually reflect the fundamental characteristics of the problem, but rather superficial regularities in the training data, known as "artifacts". For example, AIs trained over corpora of medical images have "learned" to associate diagnoses with features of the different imaging devices used at different hospitals rather than the actual content of the image. Artifacts are particularly likely to arise if a large data corpus is partitioned into a training set, used to train the ML system, and a test set, used to evaluate it; any superficial regularity in the corpus as a whole can be learned from the training set and then applied to the test set. In general, avoiding artifacts in AI systems demands great care in constructing corpora and carrying out tests.

These kinds of problems have, in fact, arisen in AI for math word problems. For example, in one experiment [25], an AI trained on the benchmark ASDiv-A seems to have learned[11] that if the word "every" appears in a problem, it should multiply the two numbers involved, whereas if the word "each" appears, it should divide them. The AI therefore got the right answer multiply the two numbers involved, whereas if the word "each" appears, it should divide them. The AI therefore got the right answer on questions like "John delivered 3 letters at every house. If he delivered for 8 houses, how many letters did John deliver?" and "Sam made 8 dollars mowing lawns over the Summer. He charged 2 bucks for each lawn. How many lawns did he mow?", but the wrong answer on questions like, "John delivered 3 letters at every house. He delivered 24 letters in all. How many houses did John visit to deliver letters?" and "Sam mowed 4 lawns over the Summer. If he charged 2 bucks for each lawn, how much did he earn?." Apparently problems conforming to the rule were more common in the dataset than those violating the rule. Thus, if an AI is trained on a training set collected from this corpus, and then tested on a separate group of problems from the same corpus, it may well be able to use these kinds of regularities to achieve a fairly high success rate without at all understanding the actual meaning of the problem or its relation to the mathematical operations.

4 Relevance to mathematical activities

As we have seen, current AI technology has serious limitations in its ability to engage with even very simple real-world mathematical problems posed in natural language. However, it is far from clear how relevant those limitations to an AI's ability to do advanced mathematics. One might naturally

[11] AI systems that used "deep learning" and similar learning techniques are opaque; characterizing what they have learned in terms of these kinds of rules is always approximate and relies on indirect evidence.

suppose that a technology that cannot reliably count up to six is not about to have much impact on groundbreaking mathematical research, but that is not a safe assumption.

I doubt, for instance, that the limitations discussed here are very relevant to the prospects of building AIs that can construct formal proof of statements in pure mathematics given formal specifications. Human mathematicians may well draw on their commonsense knowledge in understanding advanced mathematics and searching for proofs, though the degree to which they do this is unknown. (It probably varies widely across individuals.) But that does not at all imply that AIs would need to do likewise. The best chess-playing programs cannot recognize and name concepts like a "fork" or a piece being "pinned" in chess, and cannot read and discuss an article in a chess journal, but nonetheless they play very good chess.

However, there are important aspects of mathematical practice where I expect that these limitations will be important. One is applications of many kinds. The mathematization of real-world domains and tasks generally conceals the commonsense reasoning involved and gives the illusion of a rigorously formal process, but it does not actually eliminate the need for commonsense understanding. This is true even in physics, at least on the experimental side. Understanding how the LIGO gravitational wave detector works involves a commonsense understanding of the components in addition to a lot of technical understanding; you cannot prove the correctness of the experimental device from first principles. In areas such as biology, medicine, cognitive science, and, still more, the social sciences, commonsense reasoning is even more unavoidable. Understanding which parts of a mathematical model correspond to reality and which are mathematical simplifications, and, consequently, which inferences from the model may be valid and which are artifacts of the model is hardly possible without a commonsense understanding of the situation.

Mathematical informal exposition for a non-expert audience often draws on commonsense reasoning. Looking through some recent mathematical articles in *Quanta Magazine*: An article on Ramsey numbers [28] uses parties, buckets, and bound books as analogies. An article on tilings [17] includes numerous pictures of tilings and calls on the reader's intuitive understanding of how these can or cannot be extended to infinity and are or are not invariant under certain kinds of translation. An article on knot theory [27] unsurprisingly draws on intuitions about physical loops and surfaces. An article on non-transitive dice [18] calls on the reader's experience playing "knot-paper-scissors". Certainly, these are not representative of mathematical research broadly; they have been chosen for articles in *Quanta* in part precisely because they connect with a reader's general understanding. Nonetheless they indicate that connections between commonsense understanding and at least some aspects of mathematical research persist.

Likewise, teaching mathematics, particularly at elementary levels, is often most effective if the instructor can connect the theory being taught to the students' understanding of the real world.

I would also conjecture, with much less confidence, that the limitations discussed here would raise challenges to building an AI that can read and understand (human-written) mathematical articles. I do not know of any systematic analysis of the cognitive processes required to read an extended proof, but it seems likely that they draw significantly on basic commonsense understanding. Again, no doubt it varies by field and by proof style. It seems likely that proofs that draw strongly on one's visual sense and spatial, such as (at the elementary level) those in [23] or the beautiful proof of Fermat's little theorem in [15] will be particularly difficult. *Writing* mathematical proofs that are human readers find intelligible and well-written may well be easier than reading them; it is often found, in AI, that generation is easier than comprehension.

Acknowledgements

Thanks for helpful feedback to Scott Aaronson, Yuling Gu, Doug Hofstadter, and the editors, particularly Maia Fraser and Michael Harris.

References

[1] Z. Azerbayev, B. Piotrowski, H. Schoelkopf, E.W. Ayers, D. Radev, J. Avigad. *ProofNet: Autoformalizing and Formally Proving Undergraduate-Level Mathematics* arXiv preprint:2302.12433 https://arxiv.org/abs/2302.12433

[2] Y. Benchekroun, M. Dervishi, M. Ibrahim, J-B. Gaya, X. Martinet, G. Mialon. *WorldSense: A Synthetic Benchmark for Grounded Reasoning in Large Language Models.* arXiv preprint arXiv:2311.15930 https://arxiv.org/abs/2311.15930

[3] D.G Bobrow, *A question-answering system for high school algebra word problems.* Proceedings of the fall joint computer conference, part I October 27-29, 1964, 591-614).

[4] R. Bommasani et al. *On the Opportunities and Risks of Foundation Models.* arXiv preprint arXiv:2108.07258. (2021) https://arxiv.org/abs/2108.07258

[5] A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried et al. *Human-level play in the game of Diplomacy by combining language models with strategic reasoning.i Science* **378**, no. 6624 (2022): 1067-1074.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan et al. *Language models are few-shot learners.* Advances in Neural Information Processing Systems **33** (2020) 1877-1901. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html

[7] M. Chen et al. *Evaluating large language models trained on code.* arXiv preprint arXiv:2107.03374. (2021) https://arxiv.org/abs/2107.03374

[8] A. Davies, P. Velikovi, L. Buesing, S. Blackwell, D. Zheng, N. Tomaev, R. Tanburn et al. *Advancing mathematics by guiding human intuition with AI.* Nature **600**, no. 7887 (2021) 70-74.

[9] E. Davis *Using human skills taxonomies and tests as measures of AI.* In Stuart Elliott (ed.) AI and the Future of Skills, Volume 1: Capabilities and Assessments, OECD Publishing, 2021.

[10] E. Davis. *Deep Learning and Mathematical Intuition: A Review of (Davies et al. 2021).* arXiv preprint arXiv:2112.04324. 2021. https://arxiv.org/abs/2112.04324

[11] E. Davis. *Limits of an AI program for solving college math problems.* arXiv preprint arXiv:2208.06906. 2021. https://arxiv.org/abs/2107.03374

[12] E. Davis. *Benchmarks for Automated Commonsense Reasoning: A Survey.* ACM Computing Surveys, to appear. (2024). https://dl.acm.org/doi/abs/10.1145/3615355

[13] E. Davis and S. Aaronson. *Testing GPT-4 with Wolfram Alpha and Code Interpreter plug-ins on math and science problems.* arXiv preprint arXiv:2308.05713. http://arxiv.org/abs/2308.05713

[14] I. Drori et al. *A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level.* Proceedings of the National Academy of Sciences (PNAS) **119** (2021) No. 32, p.e2123433119. https://www.pnas.org/doi/abs/10.1073/pnas.2123433119

[15] J. Ellenberg. *Shape: The Hidden Geometry of Information, Biology, Strategy, Democracy, and Everything Else.* Penguin Press, (2021).

[16] A. Gillioz, J. Casas, E. Mugellini, and O.A. Khaled. *Overview of the Transformer-based Models for NLP Tasks.* In 2020 15th Conference on Computer Science and Information Systems (FedCSIS), pp. 179-183. IEEE, (2020). https://ieeexplore.ieee.org/abstract/document/9222960

[17] P. Honner. *Patterns That Go On Forever but Never Repeat.* Quanta Magazine, May 23, 2023. https://www.quantamagazine.org/math-that-goes-on-forever-but-never-repeats-20230523/

[18] E. Klarreich. *Mathematicians Roll Dice and Get Rock-Paper-Scissors,*' Quanta Magazine, January 19, 2023. https://www.quantamagazine.org/mathematicians-roll-dice-and-get-rock-paper-scissors-20230119/

[19] D.B. Lenat, M. Prakash, and M. Shepherd. *CYC: Using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks.* AI Magazine **6** (1985) no. 4, 65-65.

[20] J. Meadows and A. Freitas. *Introduction to Mathematical Language Processing: Informal Proofs, Word Problems, and Supporting Tasks.* Trans. Assoc. Computational Linguistics. **11** 2023 1162-1184. https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00594/117587

[21] S. Miao, C.C. Liang, and K.Y. Su. *A diverse corpus for evaluating and developing English math word problem solvers.* Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020. 975984, https://doi.org/10.18653/v1/2020.acl-main.92

[22] S. Mishra, M. Finlayson, P. Lu, L. Tang, S. Welleck, C. Baral, T. Rajpurohit et al. LILA*: A unified benchmark for mathematical reasoning.* arXiv preprint arXiv:2210.17517 (2022). https://arxiv.org/abs/2210.17517

[23] R.B. Nelsen, *Proofs Without Words: Exercises in Visual Thinking.* Mathematical Association of America (1993).

[24] N. Nguyen and S. Nadi. *An empirical evaluation of GitHub copilot's code suggestions.* Proceedings of the 19th International Conference on Mining Software Repositories. 2022. 1-5.

[25] A. Patel, S. Bhattamishra, and N. Goyal. *Are NLP Models really Able to Solve Simple Math Word Problems?* arXiv preprint arXiv:2103.07191 (2021). https://arxiv.org/abs/2103.07191

[26] J. Qian, H. Wang, Z. Li, S. Li, and X. Yan. *Limitations of Language Models in Arithmetic and Symbolic Induction.* arXiv preprint arXiv:2208.05051 (2022). https://arxiv.org/abs/2208.05051

[27] L. Sloman, *Mathematicians Eliminate Long-Standing Threat to Knot Conjecture.* Quanta Magazine, February 2, 2023.

[28] L. Sloman, 2023. *A Very Big Small Leap Forward in Graph Theory.* Quanta Magazine, May 2, 2023.
https://www.quantamagazine.org/after-nearly-a-century-a-new-limit-for-patterns-in-graphs-20230502/

[29] X. Wang, ZHu, 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.* arXiv preprint arXiv:2307.10635 (2023). https://arxiv.org/abs/2307.10635

[30] S. Wolfram. *ChatGPT gets its 'Wolfram Superpowers!'.* (2023) https://writings.stephenwolfram.com/2023/03/chatgpt-gets-its-wolfram-superpowers/

[31] Y. Wu, A.Q. Jiang, W. Li, M.N. Rabe, C. Staats, M. Jamnik, and C. Szegedy. *Autoformalization with large language models.* i arXiv preprint arXiv:2205.12615 (2022). https://arxiv.org/abs/2205.12615