Solving Math Word Problems Concerning Systems of Equations with GPT-3

Mingyu Zong, Bhaskar Krishnamachari

USC Viterbi School of Engineering
Los Angeles, California 90089
{mzong, bkrishna}@usc.edu

Abstract

Researchers have been interested in developing AI tools to help students learn various mathematical subjects. One challenging set of tasks for school students is learning to solve math word problems. We explore how recent advances in natural language processing, specifically the rise of powerful transformer based models, can be applied to help math learners with such problems. Concretely, we evaluate the use of GPT-3, a 1.75B parameter transformer model recently released by OpenAI, for three related challenges pertaining to math word problems corresponding to systems of two linear equations. The three challenges are classifying word problems, extracting equations from word problems, and generating word problems. For the first challenge, we define a set of problem classes and find that GPT-3 has generally very high accuracy in classifying word problems (80%-100%), for all but one of these classes. For the second challenge, we find the accuracy for extracting equations improves with number of examples provided to the model, ranging from an accuracy of 31% for zero-shot learning to about 69% using 3-shot learning, which is further improved to a high value of 80% with fine-tuning. For the third challenge, we find that GPT-3 is able to generate problems with accuracy ranging from 33% to 93%, depending on the problem type.

Introduction

A math word problem (MWP) states the properties of entities and their relationships in natural language, along with one or more quantitative questions posed in the end. Students are typically required to convert the natural language expression to one or more algebraic equation and solve them to find the answers to the stated question(s). Word problems have a rich history in mathematics education, going back three to four thousand years (Gerofsky 2004). MWP figure prominently in nearly all school mathematics textbooks (Gerofsky 1996).

Verschaffel *et al.* note that "word problems are among the most difficult kinds of problems that mathematics learners encounter" (Verschaffel et al. 2020). This has led to many decades of research in the mathematics education community, focused on how to help students learn how to solve such problems (Xin and Jitendra 1999; van Garderen 2008; Gooding 2009; Verschaffel et al. 2020).

It is known that one-on-one mentoring can have a significant positive impact on student learning but it is challenging to implement in chronically under-funded public schools due to the high student-teacher ratios. This has motivated research into the use of appropriate information technologies including software systems to enable this kind of personalized learning (Xie et al. 2019). Personalized learning is one of the 14 grand challenges identified by the National Academy of Engineering (NAE 2008).

Because MWPs, by definition, require natural language processing (NLP) and understanding, there has been significant research attention given to the development of appropriate NLP-based approaches to generating, solving and explaining solutions to math word problems. NLP experts' efforts in automatically solving math word problems trace back to the 1960s (Bobrow 1964), and there has been a rich literature in this area (Mukherjee and Garain 2008; Mandal and Naskar 2019; Patel, Bhattamishra, and Goyal 2021). Nevertheless, developing a system that adapts to a wide range of MWPs, particularly, one that gives good performance on more complex problems, remains challenging.

One of the most promising developments in the field of natural language processing has been the emergence of Transformer deep learning models in 2017 (Vaswani et al. 2017). Equipped with a self-attention mechanism, a transformer is able to capture longer-range dependency within a text to better understand the context for each word in a given problem. Transformer models have superseded other deep learning models such as recurrent neural networks, as they are able to handle longer inputs, can be trained faster, and because they show better results on a variety of tasks. These tasks include text classification, translation, question answering and even modeling human language processing (Merkx and Frank 2020).

GPT-3 is the latest publicly available transformer released by OpenAI, with an impressive few-shot learning capability (Brown et al. 2020). This autoregressive transformer has 175 billion parameters and was pre-trained on over 400 billion high-quality tokens. Among a series of evaluation tasks that OpenAI developers performed on GPT-3, the model displayed noticeable capability in completing a) common sense reasoning tasks, b) contextual comprehension tasks, and c) mathematical reasoning tasks with additional training (Brown et al. 2020; Cobbe et al. 2021). Altogether, it has the potential to make significant contributions to a natural language question-answering system. This motivates us to dive deeper into its utility for math word problems.

In this work, we focus on exploring GPT-3's performance on various tasks pertaining to one type of MWP with an intermediate level of difficulty, namely, word problems that can be expressed as a system of two linear equations in two unknown variables. We made this choice for several reasons. First, it helps to make the study somewhat more well-defined, clearly scoped, and potentially easier to replicate, given the vast number of different types of MWP that are possible. Second, restricting the scope to this class of problems allows us to write a simple verifier program to test automatically whether a given output set of equations extracted by GPT-3 are correct. Third, this class of problems is not too easy — for example, some consider the class of simpler problems involving just one unknown variable to be already "solved" (Patel, Bhattamishra, and Goyal 2021). Fourth, at the same time, it is not an overly complicated or esoteric class of problems, being routinely encountered by middle school or high school students in introductory algebra classes as well as on standardized tests such as the SAT.

We address three main questions in this project:

Q1 How good is GPT-3 at classifying problems into different themes?

Q2 How good is GPT-3 at extracting a system of linear equations directly from problem descriptions?

Q3 How good is GPT-3 at creatively generating valid problems?

One reason to explore GPT-3's ability to classify problems (Q1) is that this is actually one of the key instructional methods that math educators use to help students approach the solution of MWPs (van Garderen 2008). Our investigation of GPT-3's ability to extract the system of equations correctly from the MWP description (Q2) is motivated by the fact that there already exist symbolic mathematical solvers such as WolframAlpha (Research 2009) that can not only solve the extracted equations but can also show the step by step process by which the equations can be solved. However, as of today, such tools are generally far from capable of extracting the equations to solve an arbitrary word problem. If a system like GPT-3 turns out to be good at extracting the correct equations from a MWP, then it can be connected with a tool such as WolframAlpha in a pipeline that allows the correct answer for the problem to be worked out and explained in a step-by-step manner. Another related task that could be explored in this context is that of explaining how the equations are obtained given the MWP textual description. We believe this is a much more challenging task and defer the exploration of this question to future work. Finally, we explore GPT-3's ability to generate new word problems (Q3) because this could be helpful in auto-generating a large bank of questions for students to learn from or be assessed on.

We created several datasets based on 200 questions obtained by scraping the web for such problems (these datasets are made publicly available for the benefit of other researchers, online at https://github.com/anrgusc/MWP2L). We present a number of experiments and quantitative results addressing each of these three questions. Concerning Q1, we consider five themes (see Table 1), and find that GPT-3 classifies problems with high accuracy for every theme but one. Concerning Q2, we adopted zero-shot, one-shot, and few-shot learning, as well as the fine-tuning method, to determine if the model learns from additional examples and which one is optimal for our goal. Since OpenAI developers have found in their research that the number of examples given in the prompt affects the model's performance on the same task (Brown et al. 2020), we hypothesized that, given more examples for it to learn from, the model would perform better at equation extraction. Our results confirm that the more examples provided, the better it performs. In general, fine-tuned model outperforms the original GPT-3; while for few-shot learning, three-shot ¿ two-shot ¿ one-shot ¿ zero-shot. Overall, the fine-tuned model achieved 80% accuracy on the testing set covering all themes. As for Q3, i.e., problem generation, we find that GPT-3 is effective at creating new questions with substituted numbers and/or subjects, but it has difficulty when it is required to write questions about a different topic.

Related Work

Prior Applications of NLP to MWP

An AI-based natural language question-answering system for MWPs was first proposed in 1964 (Bobrow 1964). Early stage models which simulate human understanding of this type of question rely heavily on operation rules, a pre-designed database, or a comprehensive knowledge base to obtain a desirable success rate in restricted domains, with a predictable lack of ability to deal with out-of-scope inputs (Mukherjee and Garain 2008). Researchers then made attempts to create a more scalable framework. With well-designed categories and corresponding templates for equation solutions, ontology can be used to correctly map problems to their belonging class (Morton and Qu 2013). However, the key element of categorizing problems has noticeable complexity.

In recent years, researchers adopted supervised learning, semantic parsing, as well as reasoning approaches in their systems and gained remarkable answer correctness from various test sets (Kushman et al. 2014; Shi et al. 2015). Following the rise of deep learning algorithms, the recurrent neural network (RNN) was introduced to further improve models' performance. An RNN-based sequence-to-sequence model was developed to translate textual descriptions to mathematical expressions (Wang, Liu, and Shi 2017). Subsequent research mainly focused on combining subsidiary tree structures to improve the performance of such a translator (Wang et al. 2019; Liu et al. 2019). Even though the success of a sequence-to-sequence network is no longer built on complicated feature engineering, scientists have noticed its deficiency in capturing relations between quantities. To remedy this, an alternative graph-to-tree deep learning architecture, with a graph encoder to relate quantities to attributes and a tree decoder to form expressions, was proposed (Zhang et al. 2020). To compensate existing models for the lack of ability to take common-sense knowledge into account while processing the problems, others have implemented a novel sequence-to-tree network. The innovative knowledge graph, extracted from an external knowledge base according to key words within an MWP, is the key for the tree-decoder to utilize global information (Wu et al. 2020). In brief, hand-crafted features and elaborate systems are two essential components in research projects regarding MWP solutions; researchers had not yet been able to exclude them both from their methodology, until the emergence of Transformer models.

Prior Applications of Transformers to MWP

Like humans building a knowledge system on all the events they have encountered, a neural network gains comprehensive problem-solving ability by learning from examples it has seen. But RNN models are hard to be trained on large-scaled data because of slow computation. Besides, evidence show that they are solving the problems by matching patterns in training data instead of learning and understanding, resulting in overestimated performance (Patel, Bhattamishra, and Goyal 2021). Thus, after “attention”, a mechanism to highlight most relevant parts in input, was used to develop the faster learner, transformer, RNN models lost their dominant place in natural language processing. Regarding MWP solving, Griffith *et al.* created multiple versions of transformer to output prefix, infix, and post-fix representations of expressions. The post-fix model outstood with absolute accuracy between 82.5% to 100% on four different test sets. In fact, pre-processing procedures boost accuracy by up to 11% (Griffith and Kalita 2021). Kim *et al.* modified the classic transformer architecture so as to apply the operand-pointer mechanism, which helps model to recognize relations between entities (Kim et al. 2020). Feature-engineering-based or architecture-design-based models are accurate and effective on a number of test sets, but their ability fail to extend to a larger subset of MWP, which means re-engineering is in demand whenever researchers want to expand the scope.

Top tech companies have invested significant amount of resources in building pre-trained transformers during these years, most advanced model would usually be replaced by a even larger architecture in a year or so. Besides an increased size constructed by more layers and attention heads, newer model are often trained on an expanded dataset. One earlier model succeeded in a wide range of NLP tasks is Google’s product, BERT. A recent assessment shows that, given a dataset containing MWP with simple arithmetic, BERT reached almost 80% correctness in converting the problem to expression (Tan et al. 2021). The most intriguing finding for us is that this achievement required no complementary configurations. At the time we conduct research about language models, BERT is no longer the most sophisticated transformer. Serving as the latest pre-trained transformer open to the public, the complete model of GPT-3 beats its predecessors with 96 layers and 96 attention heads in each layer. It’s trained on more than 400 billion high quality tokens (Brown et al. 2020), and the size of word embeddings is increased by a factor of 8 compared to GPT-2. Therefore, we assess GPT-3’s capability in MWP understanding to see if it is positively related to the model’s complexity. Following the prior work on pre-trained language model, all completions of tasks are independent from manipulation of input data or modification of network structure. That is, we demand the model to create solution equations using problems in their original form and spend minimal effort in prompt engineering. We are able to confirm its capability with these ordinary settings.

Scope and Problem Formulations

Architecture for an AI-based Tutor for Word Problems

An intelligent tutoring system for MWP could be expected to 1) identify the type of problem, 2) output step-by-step instructions, 3) extract the correct system of linear equations, 4) successfully solve the equations, and 5) generate similar problems for users to practice. This paper’s primary focus is on part 3 (see Q2 below), along with some attention on parts 1 (see Q1 below) and 5 (see Q3 below). We excluded part 4 from this project because there are existing online resources such as WolframAlpha that can solve a given system of equations correctly, and we leave part 2 to future work as a more challenging task.

Q1. Categorizing Word Problems

Based on a manual review of the problems in our dataset, we grouped all word problems pertaining to two linear equations in two variables into five different categories: a) sum and difference (S&D), b) item and property (I&P), c) perimeter of rectangles (POR), d) motion (MO), and e) mixture (MI); an example of each category is provided in Table 1. We expect the model to output the name of one group given the word problem’s text as input, and assess its classification ability accordingly.

Q2. Extracting Equations

Given a word problem, the task for GPT-3 is to extract two linear equations that can be used to derive the correct answer. We standardize on the variables x and y to simplify the task. Together with problem description (and additional examples for one-shot and few-shot learning), we offer instructions at the beginning of the prompt to make sure the model knows what to do and what are the symbols to use. In Table 2, we show an illustration of expected and unacceptable answers for a given question. Accuracy is obtained by comparing the right answer with generated text.

Q3. Generating Word Problems

Various topics or subjects appeared in the word problems in our datasets, including moving objects, liquids, household items, money, numbers, weights, geometry, and rectangular objects. We tested GPT-3’s creativity by giving one example and asking it to write a similar one (within or cross topic). In Table 3, we give one example of within-topic generation and one from cross-topic generation. One thing to notice is that outcome of a cross-topic generation is still expected to fall

Sum and Difference (S&D)	The sum of half of a number, x, and another number, y, is -28. The difference of x and y is 7. Find x and y.
Item and Property (I&P)	Three apples and four bananas cost $4.85. Three apples and ten bananas cost $8.75. Find the cost of an apple.
Perimeter of Rectangle (PoR)	The length of a rectangle is 3 cm less than double the width, and the perimeter is 53 cm. Find its dimensions.
Motion (MO)	Joey and Natasha start from the same point and walk in opposite directions. Joey walks 4 km/h faster than Natasha. After 2 hours, they are 31 kilometers apart. How fast did each walk?
Mixture (MI)	Twelve gallons of a 31% acid mixture is obtained by mixing a 23% solution with a 48% solution. How much of each must be used?

Table 1: Instance problem from each category.

Problem	How many gallons of a 20% antifreeze solution and a 10% antifreeze solution must be mixed to obtain 40 gallons of a 16% antifreeze solution?
Valid Response	0.2*x+0.1*y=0.16*(x+y);x+y=40
Invalid Response	20x+10y=16*40 (only one equation is derived)
Invalid Response	20x+10(40-x)=16(40) (failed to use required variable y)
Invalid Response	2*x+1*y=40;0.2*x+0.1*y=0.16 (incorrect interpretation)

Table 2: Extraction task example.

under the same category. We don't consider cross-category generation in this project.

Experimental Methodology

Dataset

We prepared datasets of six different sizes to meet the growing needs of the project: 20 question (L20), 30 questions (L30), 50 questions (L50), 100 questions (L100), 200 questions (L200), and 1000 questions (L1000). These are summarized in Table 4. L100 is expanded from L20 for testing; L1000 is constructed from L200 in the same manner for fine-tuning. By "expanded from A" we mean each problem-answer pair in set A is used to make four copies with distinct combination of numbers. Our Python script read in one pair of problem and answer at a time and randomly generated new number for every original value based on data type

problem given in prompt	The larger of two numbers is 10 more than twice the smaller. If the smaller is subtracted from the larger, the result is 26. Find the numbers.
within-topic generation outcome	The larger of two numbers is 15 more than twice the smaller. If the smaller is subtracted from the larger, the result is 33. Find the numbers.
cross-topic generation outcome	The larger of two angles is 10 more than twice the smaller. If the smaller is subtracted from the larger, the result is 26. Find the angles.

Table 3: Problem generation.

	expanded from	**# of questions from web**	**# of questions from GPT-3**
L20	N/A	20	-
L30	N/A	30	-
L50	N/A	30	20
L100	L20	20	-
L200	N/A	200	-
L1000	L200	200	-

Table 4: Datasets.

(float or integer). We recorded all existing combinations of numerical values and added a new question to our dataset only if it has a distinct sequence of numbers and relations. This ensures that no two questions in the larger dataset are identical. Moreover, a "$\pm$ 0.7" range was used for float generation and a "$\pm$ 2" range was used for getting random integers. Zeros were allowed, but negative values were not accepted. Every set has its specific usage as discussed below. There is no intersection between any training sets and test sets, thus we properly avoided data contamination. Moreover, problems in L20, L50, L100, L200, and L1000 have an equal distribution of the five themes.

For Q1

We asked GPT-3 to classify every question within the L50 dataset. Each query was formulated as a multiple choice problem. One instance was provided at the beginning, followed by the question: "Which type of question does the above one belong to?" and the five categories were given below as the only options in a multiple choice format.

For Q2

Our project began with the L30 dataset. It was split into training, cross-validation (cv), and testing set with a ratio of 5:12:13.

Prompt for zero-shot learning was composed of a two-sentence instruction: "Extract a system of two linear equations in terms of x and y from the question. Separate the equations with semicolon.\n" followed by the problem statement. Zero-shot learning was performed only once on the entire dataset as it needs no separate training/cross-validation process.

Few-shot learning methods require us to feed in examples. Their prompts shared a different beginning: "Extract a system of two linear equations in terms of x and y from the question. Separate the equations with semicolon like given in the example:\n", followed by numbered example(s).

We performed K-shot learning, for $K = 1 \cdot 5$ (i.e., one-shot, two-shot, up to five-shot learning) on the cross-validation (cv) set with all possible combinations of K training examples possible from the 5 training examples. Among all of them, those resulted in the highest accuracy with respect to learning method were picked out and used on the testing set to ensure that our reported extraction accuracy would be a fair representation of the model's performance.

Another embedded variable that we can manipulate and should take into account is the temperature of the GPT-3 model. Its value decides how much randomness is involved in the model's generation of response. A lower value of temperature indicates that the responses would be more deterministic, otherwise they are more random and likely to be changed in another run. Taking into account its impact on text completion, we let the model experience the same procedures five times with temperature 0.1, 0.3, 0.5, 0.7, and 0.9 to explore whether a specific value is the best fit for our task[1].

To find the upper limit of the model, we further introduced fine-tuning into our research. OpenAI recommends training the model on at least 500 instances, so we created L1000 as previously described. OpenAI allows users to further fine-tune a previously fine-tuned model and will preserve models at each training stage. We further divided the process into three stages. We fed in 100 examples during the first run, then another 400, and finally the rest of 500 questions. All sample training sets have an equal number of questions from each class. As for testing, L20 was used to record extraction accuracy at each stage; L100 was used as well to compare whether the results are representative. Having the same number of questions from each theme in these datasets allows us to analyze the model's performance by category at the same time.

For Q3

In order to effectively create new questions, we provide one example at the beginning of each query, followed by a one-sentence instruction: "Given the above question, use different values to write a similar question about {topic}." The placeholder "{topic}" was replaced by a specific topic subject during generation.

Regarding within-topic generation, three instances from each group were randomly selected. Each was tested ten times to compute successful generation rate. We report rates by category using mean of means.

For cross-topics generation, we considered every reasonable pair of categories and topics. We used one random example from each pair, asked the model to generate questions belonging to a different combination for ten times, and counted the number of successes. We report these values in the Experimental Results section.

[1]Unless otherwise specified, the completion tasks always use 0.3 as temperature.

Experimental Results

Classification

GPT-3 achieved over 80% accuracy for all groups but the "item and property" class. For the "mixture" and "perimeter of rectangle" groups, it successfully recognized all the belonging questions (Table 5). One interesting finding is that all ten questions belonging to "item and property" were classified as "mixture", which in a way could be justified, as a typical "item and property" concerns combinations of two types of items.

Category	**Accuracy**
Sum and Difference (S&D)	0.80
Item and Property (I&P)	0.00
Motion (MO)	0.90
Mixture (MI)	1.00
Perimeter of Rectangle (POR)	1.00

Table 5: Classification accuracy.

Few-shot Learning

	0.1	**0.3**	**0.5**	**0.7**	**0.9**
1-shot	(3) (4)	(3)	(3) (4)	(4)	(4)
2-shot	(1,4)	(1,4)	(1,4)	(1,2) (2,4) (3,4)	(2,3) (1,4) (2,4) (4,5)
3-shot	(2,3,4) (1,3,4) (3,4,5) (1,3,5) (1,2,3)	(1,3,4) (1,3,5)	(2,3,4) (1,3,4)	(2,3,4)	(1,2,4) (2,3,5)
4-shot	(1,2,3,4) (1,2,3,5) (1,3,4,5) (2,3,4,5)	(2,3,4,5)	(1,2,4,5) (1,2,3,4)	(1,2,4,5) (2,3,4,5)	(1,2,3,5) (1,2,3,4)

Table 6: Examples with best performance on the cv set for different temperature parameters from 0.1 to 0.9.

	0.1	**0.3**	**0.5**	**0.7**	**0.9**
0-shot	0.2308	0.1539	0.1539	**0.3077**	0.1539
1-shot	0.3846	0.3846	**0.5385**	0.3077	0.4615
2-shot	0.5385	0.5385	0.4615	**0.6154**	0.5385
3-shot	0.6154	**0.6923**	0.5385	0.4615	0.3846
4-shot	0.6154	0.6154	0.6154	**0.6923**	**0.6923**
5-shot	0.6154	0.5833	**0.6923**	0.5385	0.5385

Table 7: Highest accuracy on test set.

Five examples in the training set are numbered from 1 to 5. After running all possible combinations of training examples on the cross-validation set, we picked out the prompts that performed the best for each temperature, which are shown in Table 6.

Using these prompts, we ran GPT-3 on the test set and measured the corresponding success rates. We present model's best performance with respect to learning method and temperature in Table 7. It is clear that two-shot, three-shot, four-shot, and five-shot learning are comparable methods in most cases. Summarizing by learning methods, zero-shot learning achieved 0.3077 with temperature 0.7, one-shot achieved 0.5385 with temperature 0.5, two-shot achieved 0.6154 with temperature 0.7, and the rest three reached 0.6923 with different temperatures. In conclusion, upper limit of GPT-3 as a few-shot learner is 0.6923 on this testing set; there is no clear pattern for a relationship between temperature and extraction accuracy; and we would expect lower limit to be lifted as more examples are provided to the model. One might notice success rate is in inverse proportion to temperature for three-shot learning, but this is because we left the "prompt" variable uncontrolled while picking out the best values. When we use the same prompt and only manipulate the temperature, the trend vanished.

Fine-tuning

Model	**Accuracy**
3-shot learning	0.40
Fine-tuned w/ 100 examples	0.75
Fine-tuned w/ 500 examples	0.75
Fine-tuned w/ 1000 examples	0.80

Table 8: Results from L20.

Our fine-tuned model was tested on L20 set at each stage. The corresponding results are shown in Table 8.

Running on the 100 instances (L100), fine-tuned GPT-3 obtained an extraction accuracy of 78.00% while three-shot learning achieved 48.00%, indicating that generally a fine-tuned model exceeds few-shot learning by 30%-40% accuracy in extracting system of linear equations.

By Category

Model	**S&D**	**I&P**	**POR**	**MO**	**MI**
3-shot w/o example	0.5	0.7	0.2	0.0	0.5
3-shot w/ example	0.2	0.7	0.4	0.4	0.7
5-shot w/ example	0.3	0.7	0.4	0.4	0.8

Table 9: Accuracy by category on L50.

Model	**S&D**	**I&P**	**POR**	**MO**	**MI**
3-shot learning	0.05	0.70	0.50	0.40	0.75
Fine-tuned GPT-3	0.45	0.75	0.95	0.75	1.00

Table 10: Accuracy by category on L100.

We hypothesized that offering one or more examples from the same category would enhance or at least stabilize model's performance on a group of problems. We used L50 and few-shot learning methods to investigate this hypothesis. The results support the hypothesis for all but one category, the "sum and difference" category (Table 9). Next, we tested on the L100 dataset for more proof. We chose the best-performed three-shot learning model (with one problem belonging to the "mixture" group and two from the "item and property" group) to compare with our final version of fine-tuned GPT-3 (Table 10). Consequently, we obtained 0.75 and 0.70 accuracy, respectively, for those two categories, while getting considerably lower success rates for the other three. With the model fine-tuned on 1000 examples, all values increased. Four of the five groups yielded acceptable rates of at least 75%.

We conclude that this transformer model struggles the most with the "sum and difference" group, considering even fine-tuning couldn't boost the accuracy to above 50%. On the contrary, it learns fast to deal with questions classified to either "perimeter of rectangle", "motion", or "mixture" group. Due to its constantly stable performance with respect to the "item and property" category, we infer that the model naturally has ability to work with this type of MWP but has limited capability for improvement. Despite the unresolved mystery of "sum and difference" group, in general, our results indicate that relevant same-group examples are helpful as prompts in few-shot learning.

Simulated Question Generation

We set temperature to 0.7 in order to take advantage of the model's creativity in writing problems. GPT-3 was instructed to performed a zero-shot learning. Using a prompt that contains one within-category example followed by "given the above example, write a similar problem with different subjects and different numbers" obtained various generation accuracy. We accepted questions with at least one substituted number or subject, but didn't give credit if the provided instance is returned without modification. By taking average of three success rates, the POR category resulted in the highest value: 0.93±0.14, while the MO category has the lowest: 0.33±0.45 (Table 11).

	S&D	**I&P**	**POR**	**MO**	**MI**
avg	0.43	0.60	0.93	0.33	0.70
sd	0.51	0.20	0.12	0.40	0.52

Table 11: Results of within-topic generation.

Category and topic	**Success rate on different topics**
MI + liquids	items: 0/10; money: 0/10
MI + items	liquids: 8/10; money: 0/10
MI + money	liquids: 0/10; items: 0/10
S&D + numbers	geometry: 5/10; weights: 10/10
S&D + geometry	numbers: 0/10; weights: 10/10
S&D + weights	numbers: 0/10; geometry: 2/10
I&P + items	money: 3/10
I&P + money	items: 0/10

Table 12: Results of cross-topic generation.

Not all categories span multiple topics. Specifically, the "motion" category concerns only object movements, and problems in the "perimeter of rectangle" are all about rectangular objects. Therefore, we were left with the rest three categories to explore cross-topics generation. Only a few pairs appeared to have an acceptable rate above 70% (Table 12).

Discussion

Classification results suggests an alternative set of classes needs to be proposed in order to improve GPT-3's performance on this task. Combining "item and property" and "mixture" into a larger category could be a short-term solution. However, for the long-term goal of creating a well-functioning tutoring system, it has to be able to distinguish these confusing problems.

Our experiment on extraction accuracy reached 80% along with 1000 training problems. By reviewing results on L100, we noticed that fine-tuned GPT-3 generally would fail on the duplicates if it couldn't get the original question right; on the other hand, it hardly make mistakes on duplicates if it succeeded on the original piece. In Table 13 we show a number of problems that our fine-tuned model had difficulty in processing. One finding is that directly applying commonsense knowledge to problem solving remains an issue even though the transformer has seen millions of texts. Additionally, inconsistency and random errors also exists in output equations.

Q:	Pi High School ordered 40 science books. The next week, the school ordered 30 algebra books. The bill for the first order was \$360 greater than the bill for the second order. The two bills together totaled \$3960. Find the price of an algebra book.
A:	40*x+30*y=3960;x+360=y
Q:	A boat traveled 24 miles downstream in 2 hours. The return trip took twice as long. What is the speed of the boat in still water?
A:	2*(x+y)=24;2*(x-y)=24
Q:	The length of a rectangle is 1 cm more than six times the width. If the perimeter of the rectangle is 82 cm, what are its dimensions?
A:	2*x+2*y-82=1;x-1=6*y
Q:	In a two digit number. The units digit is thrice the tens digit. If 36 is added to the number, the digits interchange their place. Find the number.
A:	x+y=10;3*x+y=36

Table 13: Invalid output.

As for problem generation, our simple prompt yielded satisfying outcomes with certain categories. The bimodal distribution of results for cross-topics generation suggests that there might exist better prompts to boost these lower values. We acknowledge that some topics are too broad, like "items", casting additional difficulty to the task. Prompt engineering and design of classes are two paths for future work.

Conclusions

This study has explored the usage of GPT-3 in three key aspects of linear math word problem tutoring: classifying problems, converting description to a system of two linear equations, and generating similar problems using one given example. Our study shows promising results for all three problems, particularly the 80% success rate for extracting systems of equations with fine-tuning suggests that it is practically useful for this purpose. In the future, we plan to explore prompts and techniques that can guide GPT-3 to explain how the answers are derived.

References

Bobrow, D. G. 1964. Natural language input for a computer problem solving system.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Hilton, J.; Nakano, R.; Hesse, C.; and Schulman, J. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Gerofsky, S. 1996. A linguistic and narrative view of word problems in mathematics education. *For the learning of mathematics*, 16(2): 36–45.

Gerofsky, S. 2004. *A man left Albuquerque heading east: Word problems as genre in mathematics education*, volume 5. Peter Lang.

Gooding, S. 2009. Children's difficulties with mathematical word problems. *Proceedings of the British Society for Research into Learning Mathematics*, 29(3): 31–36.

Griffith, K.; and Kalita, J. 2021. Solving Arithmetic Word Problems with Transformers and Preprocessing of Problem Text. *arXiv preprint arXiv:2106.00893*.

Kim, B.; Ki, K. S.; Lee, D.; and Gweon, G. 2020. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 3768–3779.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 271–281.

Liu, Q.; Guan, W.; Li, S.; and Kawahara, D. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, 2370–2379.

Mandal, S.; and Naskar, S. K. 2019. Solving arithmetic mathematical word problems: A review and recent advancements. *Information Technology and Applied Mathematics*, 95–114.

Merkx, D.; and Frank, S. L. 2020. Human Sentence Processing: Recurrence or Attention? *arXiv preprint arXiv:2005.09471*.

Morton, K.; and Qu, Y. 2013. A novel framework for math word problem solving. *International Journal of Information and Education Technology*, 3(1): 88.

Mukherjee, A.; and Garain, U. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2): 93–122.

NAE. 2008. 14 Grand Challenges for Engineering.

Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models really able to Solve Simple Math Word Problems? *arXiv preprint arXiv:2103.07191*.

Research, W. 2009. Wolfram—Alpha.

Shi, S.; Wang, Y.; Lin, C.-Y.; Liu, X.; and Rui, Y. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, 1132–1142.

Tan, M.; Wang, L.; Jiang, L.; and Jiang, J. 2021. Investigating math word problems using pretrained multilingual language models. *arXiv preprint arXiv:2105.08928*.

van Garderen, D. 2008. Middle school special education teachers' instructional practices for solving mathematical word problems: An exploratory study. *Teacher Education and Special Education*, 31(2): 132–144.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 6000–6010.

Verschaffel, L.; Schukajlow, S.; Star, J.; and Van Dooren, W. 2020. Word problems in mathematics education: A survey. *ZDM*, 52(1): 1–16.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B. T.; and Shen, H. T. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7144–7151.

Wang, Y.; Liu, X.; and Shi, S. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 845–854.

Wu, Q.; Zhang, Q.; Fu, J.; and Huang, X.-J. 2020. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 7137–7146.

Xie, H.; Chu, H.-C.; Hwang, G.-J.; and Wang, C.-C. 2019. Trends and development in technology-enhanced adaptive/personalized learning: A systematic review of journal publications from 2007 to 2017. *Computers & Education*, 140: 103599.

Xin, Y. P.; and Jitendra, A. K. 1999. The effects of instruction in solving mathematical word problems for students with learning problems: A meta-analysis. *The Journal of Special Education*, 32(4): 207–225.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020. Graph-to-tree learning for solving math word problems. Association for Computational Linguistics.