Development of Numerical Error Detection Tasks to Analyze the Numerical Capabilities of Language Models

Taku Sakamoto[1], Saku Sugawara[2], Akiko Aizawa[2],
[1]The University of Tokyo, [2]National Institute of Informatics
{t_sakamoto, saku, aizawa}@nii.ac.jp

Abstract

Numbers are used to describe quantities in various scenarios in daily life; therefore, numerical errors can significantly affect the meaning of the entire sentence, and even a single-letter error can be fatal. Detecting numerical errors often requires a high level of commonsense and is difficult even with the recent large language models (LLMs). In this study, we create a benchmark dataset of numerical error detection that uses automatically generated numerical errors. In our analysis, we classify the numerical errors based on the properties of the errors and investigate the ability of the model from several perspectives, including the error class, error size, and passage domain. The experimental results indicate that GPT-3.5, GPT-4, and Llama-3-Instruct (8B) perform well in the numerical error detection task; however, they are not as accurate as humans. We find that the LLMs misidentified correct numbers as errors more frequently than the humans did. In particular, the analysis demonstrates that the current LLMs still need improvement for detecting numerical errors requiring calculations or extensive prior knowledge.

1 Introduction

Numbers provide accurate or approximate quantitative information such as time, size, and monetary value, serving as an essential part of everyday communication (Spithourakis and Riedel, 2018). However, it is difficult for language models (LMs) to deeply understand numbers in text (Sharma et al., 2024; Xu et al., 2024). Previous studies highlighted that LMs that process sentences using embedded representations have difficulty understanding the correspondence between number tokens and their numerical magnitude (Wallace et al., 2019) and understanding the magnitude of numbers that are divided into multiple tokens, such as large numbers and decimals, because they process sentences

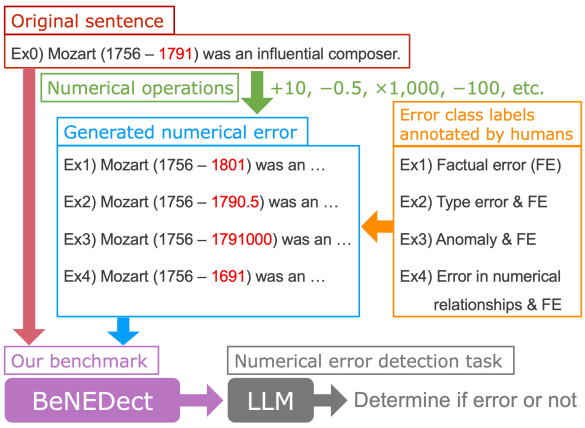

Figure 1: An overview of our approach and examples of annotated numerical errors.

with a finite set of fixed tokens (Singh and Strouse, 2024; Chen et al., 2023; Yuan et al., 2023).

Consider the following numerical error:

(1) Reserve a private room in a restaurant for tomorrow's dinner with **10** people.
(2) * Reserve a private room in a restaurant for tomorrow's dinner with **100** people.

Errors in numbers can significantly affect the meaning of the entire sentence. Therefore, even a single-letter numerical error, as in this example, can cause more damage compared to a textual error (Ng et al., 2014; Kasewa et al., 2018; Bryant et al., 2019). While it is not difficult for humans to notice such a one-digit error and avoid significant damage, LMs would find it difficult to detect numerical errors because it requires a deep contextual understanding. In addition, numerical errors demand a high level of numerical commonsense and diverse skills for detection due to their variety. As an example, consider the case (Ex1–4) shown in Figure 1. In all examples, the number "1791" in the sentence "Mozart (1756–1791) was an influential composer" is replaced by different numbers, where the error type is different. In Ex1, readers would need to

Proceedings of the 31st International Conference on Computational Linguistics, pages 9957–9976
January 19–24, 2025.

know that Mozart died in 1791 (or at least not in 1801) to notice the error (factual error). However, knowing the exact year in which Mozart died is not necessary to notice the errors in Ex2–4. (Naturally, knowing the exact year would be sufficient to notice.) For Ex2, it is sufficient to know that the year number is an integer (type error ∩ factual error). Ex3 can be recognized as an error if readers realize that the magnitude of "1791000" is too large as a year (anomaly ∩ factual error). In Ex4, the error can be detected if readers can compare the magnitudes of 1756 and 1691 (error in numerical relationships ∩ factual error).

In addition, the numerical error detection task has a wide range of applications, including numerical error correction systems and hallucination detection for LLMs. In particular, the hallucination detection is currently an important topic, and it would be greatly beneficial if LMs could solve it, since detecting trivial numerical errors is difficult even for knowledgeable humans. Despite the importance of the numerical error detection task, benchmark datasets for this task have been limited before (Berg-Kirkpatrick and Spokoyny, 2020; Chen et al., 2019). In this study, we developed a **Be**nchmark for **N**umerical **E**rror **De**te**ct**ion task, called **BeNEDect**, which can be used to verify and analyze the ability of LMs to detect numerical errors.[1] Furthermore, we propose to use this task as a measure of the ability of LMs to understand numerical values. By comparing the performances of numerical error detection between LMs and humans based on the diversity of numerical errors, we analyze the differences in the behavior of humans and LMs for numerical values.

First, we classified numerical comprehension skills and numerical commonsense into several categories to analyze the performance of LMs. We then define the classes of numerical errors that require each skill or commonsense knowledge for detection, and propose a classification of numerical errors consisting of four classes. (Details of each class are shown in Section 3.2.3.) Next, we collected example sentences from five domains, including Wikipedia, scientific papers, and news titles, and subsequently, we generated synthetic error data by automatically substituting arbitrary numbers (Figure 1). We subsequently manually annotated them using the aforementioned error classes (Table 2). Finally, the annotated dataset is used to evaluate the knowledge and ability of LLMs, including GPT-3.5 (Winata et al., 2021; Chen et al., 2021; Ouyang et al., 2022), GPT-4 (OpenAI et al., 2024), Llama-3 (Meta, 2024), Flan-T5 (Wei et al., 2022), and T0 (Sanh et al., 2022). The accuracy of these results is analyzed by comparing them with human accuracy.

Although GPT-3.5, GPT-4, and Llama 3 performed well on the numerical error detection task, their accuracy was still not as high as that of humans, indicating room for improvement. Analysis of the performance of numerical error detection by LMs and humans for each passage domain, error class, and error generation method indicated that the performance of these models was similar to that of humans compared to other LMs. However, the difference in accuracy was considerably large for numerical errors that required certain arithmetic operations (in the "error in numerical relationships" class) and for those that required expert-level prior knowledge (in scientific papers). Moreover, we found that the LLMs misidentified correct numbers as errors more often than the humans did. In addition, the results indicated that minor changes in the phrasing of prompts can cause significant variations in detection performance, suggesting that achieving consistent performance and managing prompt-dependent variability remain key challenges for these models. These findings suggest a guideline for future research on the numerical understanding of LMs.

The contributions of this study are as follows:

- We proposed a classification of numerical errors and organized the capabilities needed to detect each error type, linking them to the characteristics of the errors.
- We created a benchmark for the numerical error detection task by introducing synthetic errors into passages collected from a wide range of domains and manually annotating them with their error classes.
- We analyzed the ability of recent LLMs to detect numerical errors from three perspectives: the passage domain, the class of numerical errors, and the operations used to generate numerical errors, comparing their performance with the human performance.

[1] https://github.com/cogma/BeNEDect

2 Related Work

2.1 Numerical Error Detection with LMs

Numerical error detection determines whether target numbers in the input passages are correct (Chen et al., 2019; Berg-Kirkpatrick and Spokoyny, 2020; Spithourakis et al., 2016). They examines the numerical commonsense of architectures, including BiGRU (Cho et al., 2014), CRNN (Kim, 2014; Choi et al., 2016) and BERT (Devlin et al., 2019), by using them to contextually detect anomalous numbers changed by a random scaling factor in a dataset of news titles and market comments. The results confirmed that the LMs used in the experiments detected numerical errors with moderate to high accuracy, and the detection accuracy improved as the error degree increased. In this study, we developed BeNEDect, a benchmark dataset that allows for a more detailed analysis of the ability of LMs to detect numerical errors by generating numerical errors in a wider range of domains using anomaly generation methods that cover more practical errors, such as one-digit errors, sign errors, and copy-paste errors. Subsequently, we employed the numerical error detection task, which, despite its importance, has been infrequently addressed in recent LLMs, as a means of evaluating and analyzing the capabilities of recent LLMs, including GPT-4, Llama 3, and Flan-T5, with a particular focus on error types.

Li et al. (2024) evaluated the performance of LLMs using math word problems. The models were presented with a problem and its solution, which was sometimes incorrect, and were then asked to determine whether the solution contained errors. In their study, the authors defined a classification of erroneous solutions and employed it to analyze the performance of the LMs. Their experiments showed, as we demonstrate, that the latest LLMs are still unable to accurately identify erroneous solutions that require mathematical computation. In our study, we employed the numerical error detection task within passages to evaluate the performance of LLMs. Our analysis encompassed not only their computational capabilities but also their numerical commonsense and memorization abilities regarding numerical data. Furthermore, we offered insights by comparing the performance of LLMs with that of humans.

2.2 LLM Ability on Numerical NLP Tasks

In recent years, LLMs have achieved remarkable results for various NLP tasks (Qin et al., 2024; Huang and Chang, 2023), including numerical NLP tasks (Ahn et al., 2024). LLMs have achieved a certain degree of success in recognizing the magnitude of numbers and in performing accurate arithmetic operations (Yuan et al., 2023; Frieder et al., 2023), which have been difficult with previous LMs such as BERT and RoBERTa (Liu et al., 2019). Consequently, the performance of these LLMs on tasks such as math word problems and numerical reasoning has improved significantly.

Yuan et al. (2023) verified the basic computational capabilities of recent LLMs with the proposed dataset and confirmed that GPT-3.5 and GPT-4 exhibit high performance in arithmetic tasks that were difficult for the previous LMs. Sivakumar and Moosavi (2023); Akhtar et al. (2023); Imani et al. (2023); Mishra et al. (2020) classified skills required in numerical reasoning tasks (e.g., number representation recognition, number sense, and mathematical skills), created a dataset for evaluation based on the classification and analyzed the numerical reasoning abilities of the recent LLMs. Liang et al. (2023); Jie and Lu (2023) demonstrated that, even without using LLMs as solvers, the performance of solver models with a small number of parameters, such as LSTM (Hochreiter and Schmidhuber, 1997) and RoBERTa, can be improved significantly by managing the learning status of the solver models and by creating additional exercises for them with LLM.

This study examined the numerical commonsense of the LLMs regarding quantity and their ability to handle numbers in a simple but practical task setting that is yet to be conducted for LLMs, that is, the numerical error detection task.

3 BeNEDect: A Benchmark for Numerical Error Detection Task

3.1 Task Description

We tested the numerical commonsense and ability of LMs to handle numbers using the numerical error detection task (Figure 1) defined as follows:

Input: Passages containing only one target number (erroneous or not)

Output: Determination of whether the target number is an error

We assume exactly one target number even if multiple numbers appear in an input passage. We

use only one target number because it is rare to have more than one numerical error in a sentence or paragraph, and we aim to enable the models to determine whether the target number is an error by comparing or calculating the target number with other correct numbers in the same passage.

3.2 Dataset Construction

3.2.1 Passage Sources

This study introduced passages from a wide range of domains into BeNEDect by expecting differences in the trends of numbers and the ease of detecting errors by a passage domain. Specifically, we selected passages from the following five datasets with different domains and passage lengths.

- Numeracy-600K (article titles) (Chen et al., 2019)
- ACLsent (scientific papers) (Abekawa and Aizawa, 2016)
- DROP (Wikipedia (history articles and National Football League game summaries)) (Dua et al., 2019)
- QA-text (MCTest (Richardson et al., 2013), RACE (Lai et al., 2017), Project Gutenberg, Open American National Corpus (Ide and Suderman, 2006), ReClor (Yu et al., 2020), Wikipedia (science and arts articles)) (Sugawara et al., 2022)
- FinNum (financial tweets) (Chen et al., 2018)

We collected 1,000 passages containing one or more numbers from each dataset, for a total of 5,000 passages. Statistics of passage lengths and numbers that appear in each dataset are listed in Table 1. First, the passages in Numeracy-600K, ACLsent, and FinNum are short and contain fewer numbers. Therefore, a few contextual clues exist for numerical error detection in these datasets. Numeracy-600K has only 1.5 numbers in a single passage on average, and there are few clues from the other numbers in the passage. In contrast, DROP and QA-text are longer (approximately 200–300 words) and contain more clues in the context. In particular, DROP is a reading comprehension dataset that requires numerical reasoning; therefore, it contains an average of 22.6 numerals per passage. In addition, more than 30% of the numbers in FinNum are decimals, whereas most numbers in Numeracy-600K are integers. These trends in numbers appearing in each dataset are considered to affect the ease of identifying numerical errors.

3.2.2 Generating Numerical Errors

We automatically modified collected passages by randomly selecting one target number in each passage and randomly applying one of the following operations: addition or subtraction, multiplication, and replacement (Figure 1). A total of 5,000 erroneous passages were generated from 5,000 correct passages, resulting in BeNEDect, a 10,000-passage dataset.

Addition or Subtraction We randomly select a size from the following list and add or subtract it from the original number: $[\pm1,000, \pm100, \pm10, \pm1, \pm0.5, \pm0.1]$. Using a random size from the above list allows us to cover a one-digit numerical error (e.g., 2024 $\rightarrow$ 2034), which is considered one of the most common numerical errors in practice.

Multiplication We randomly select a multiplier from the following list and multiply it by the original number: $[*(-1), *0, *0.001, *0.01, *0.1, *0.5, *0.7, *0.9, *1.1, *1.5, *2, *10, *100, *1,000]$. Assuming that decimal point errors and numerical errors owing to misunderstanding of units are common in practice, we randomly selected a multiplier from the aforementioned list.

In both addition/subtraction and multiplication, the more significant the difference or magnification, the higher the detection rate for humans is expected to be. We experimentally verified whether the LMs showed the same tendency as human behavior.

Replacement We replace the target number with a randomly chosen unrelated number in another passage within the same domain. This simulates copy-paste errors. Aforementioned two types of operations do not consider the distribution of numbers within a domain, which can introduce bias into the task. Specifically, they can generate erroneous numbers with a low occurrence rate in the domain, and the rarity and unfamiliarity of such numbers can lead to detection of errors without deep knowledge. Therefore, this operation uses numbers that appear in passages in the same domain to generate numerical errors. We confirmed that the numbers before and after the replacement do not match.

3.2.3 Numerical Error Classification

We categorized numerical errors into four classes according to the ability or numerical commonsense required to detect them: factual error, type error, anomaly, and error in numerical relationships (Table 2) (Lin et al., 2020; Elazar et al., 2019).

Dataset	Numeracy-600K	ACLsent	DROP	QA-text	FinNum
Ave. passage length [words]	10.1	33.5	307.1	193.1	21.9
Ave. #numbers per passage	1.5	5.5	22.6	7.3	2.9
% of integers	97.2 %	79.6 %	85.3 %	81.6 %	67.4 %
Domain	News titles	Scientific papers	Wikipedia	various	Financial tweets
Prior knowledge of annotators	Good	Excellent	Good	Good	Poor

Table 1: Statistics across five different datasets.

Error type	Examples
Factual error	• Spiders have **9** limbs. • Wolfgang Amadeus Mozart (1756-**1991**) was ... • The population of New York City is **14,028** in 2023.
Type error	• March **3.5** • March **-15** • Mike's height is **-3.6** meters.
Anomaly	• Mike's height is **30.6** meters. • The movie I saw yesterday was **2,000** hours long. • **500,000,000** people attended her wedding.
Error in numerical relationships	• Android has more than **6**/5 of the smartphone market. • 1st inning top 1-0, bottom 1-**5**, 2nd inning top 3-1, ... • They split their profits **80**% and 30% between them.

Table 2: Four types of numerical errors.

Factual Error "Factual error" is a class of numerical errors concerning definitions, actual measurements, or years of important historical events (Table 2) (Lin et al., 2020). The numerical error that belongs to this class is determined solely by the properties of the number, and we assume that humans know exactly what the correct number is for the error. Therefore, memorization of basic commonsense is necessary to detect numerical errors in this class. "Factual error" is one type of numerical error, yet many of the capabilities required to detect them are not unique to numbers. It is therefore anticipated that the current LMs will be easier to detect them in comparison to the errors belonging to other classes that require number-specific commonsense to detect, as described below. In this work, we assume that these numbers are considered to be known to high school or college students or found in some textbooks.

Type Error "Type error" is a class of numerical errors that violate number-type constraints such as a decimal number for an integer type or a negative value for a positive number type (Table 2). This class only includes explicit number-type errors, such as a decimal number for a number that should be an integer or a number less than or equal to zero for a number that should be a positive number. Knowledge of the number types allowed for each entity type of numbers is necessary to detect numerical errors in this class. Note that the class does not include numerical errors that are out of bounds but still satisfy the number-type constraint (e.g., March **300**)[2].

Anomaly "Anomaly" is a class of numerical errors for numbers that have certain range or distribution (Elazar et al., 2019). They are not number-type errors, and instead, they can be judged to be anomalies based on commonsense and their contexts (Table 2). Detecting numerical errors in this class requires knowledge about the acceptable range and distribution of numbers for each entity type in context. Moreover, it requires a basic ability to recognize the numerical magnitude of the number words. For example, it is difficult to determine whether "17.94" in the sentence "His height is 17.94 cm" is an error solely based on the co-occurrence probability of the number word with contextual words. Recognizing the magnitude of "17.94" and confirming that it is unnatural by comparing it with the aforementioned distribution is necessary to correctly infer that this is an error. Errors belonging to this class differ from factual errors in that even humans do not know exactly what the correct number is.

Error in Numerical Relationships "Error in numerical relationships" is a class of numerical errors that violate some numerical relationships or arithmetic formulae that must be satisfied among numbers in the same passage, such as ensuring that percentages add up to 100% (Table 2). Knowledge of the constraints among numbers and the mathematical ability to compare the magnitudes of numbers and perform elementary arithmetic operations is necessary to detect errors in this class.

Lastly, errors can belong to more than one class. For example, Ex4 in Figure 1 belongs to the "error in numerical relationships" class, while at the same time it belongs to the "factual error" class because it can be detected even with the knowledge

[2]It is included in the "anomaly" class shown below.

of the exact year. Details on which error classes are compatible are given in Section A.

3.2.4 Annotation of Error Classes

We manually classified the 1,500 numerical errors in BeNEDect (Section 3.2.3). The "undetectable error" class was added to the aforementioned four classes to accommodate numerical errors that are difficult to detect even for humans due to reasons such as slight numerical changes and the need for professional knowledge to detect them, resulting in a total of five classes of annotations.

Careful guidelines for annotation were prepared (details are provided in Appendix A). Crowdsourcing was not used in this study because of the need for careful instructions for annotation, and annotation was performed by eleven annotators, including graduate students and researchers in natural language processing, who annotated 250–500 samples each. Therefore, biases in the prior knowledge of the annotator are expected to exist across domains (Table 1). A majority vote of 1–3 annotators was used to determine the error class for each numerical error. 67.9% of the numerical errors were agreed upon by three annotators, and 94.4% of the errors were agreed upon by more than two annotators.

Figure 2 shows the distribution of the error classes per dataset. According to the results, approximately 30% of errors in the annotated numbers were marked as unrecognizable by the annotators. The most common error class among errors that humans can notice is the "Type error," which is predictable given the error generation design. Further, numerical errors that belonged to the "factual error" class are considerably fewer, thereby confirming that commonsense numbers are rarely mentioned in the actual sentences. The distribution varied widely across datasets, which was affected by passage length, trend of the numbers to appear, and amount of prior knowledge of the annotators.

4 Experiments and Results

4.1 Experiments

We evaluated the performance of the following LMs in the zero-shot and few-shot scenarios: GPT-3.5 Turbo[3] (Brown et al., 2020; Winata et al., 2021; Chen et al., 2021; Ouyang et al., 2022), GPT-4 Turbo[3] (OpenAI et al., 2024), Llama-3-Instruct (8B) (Meta, 2024), Mistral-Instruct-v0.3 (7B) (Jiang et al., 2023), Mathstral-v0.1 (7B) (AI, 2024), FLAN-T5-xl (3B) (Wei et al., 2022; Chung et al., 2022), and T0 (3B) (Sanh et al., 2022). FLAN-T5 has been pre-trained on mathematics-related datasets and is considered skilled in handling numbers. The prompts given in the experiments are described in the Appendix B. The judgment of LMs was determined by applying pattern matching to the output sentences. More than 120 patterns that indicate whether the target number is correct or not were prepared. In the event that neither positive nor negative patterns were present, or that both were present, the output was deemed a generation error.

[3]January 2024 models

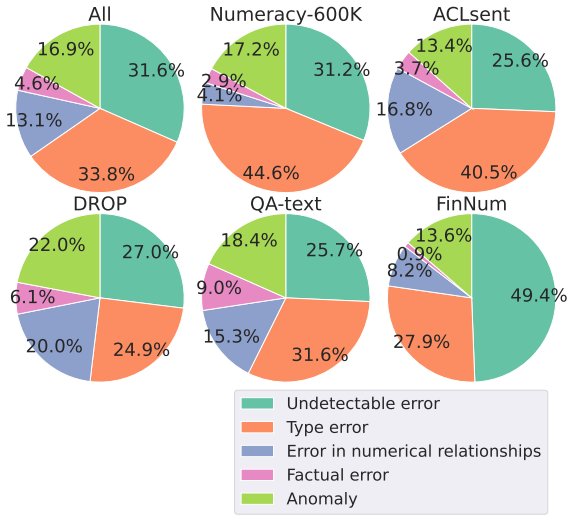

Figure 2: Error class distribution of the annotated numerical errors for each dataset.

4.2 Accuracy for Numerical Error Detection

4.2.1 Zero-shot Accuracy

Table 3 lists the detection accuracies of the LMs in a zero-shot setting for BeNEDect. This showed the highest accuracy for each model among the eight prompts used in the experiments. The results for the other prompts are shown in Appendix C.

In Table 3, "Accuracy" represents the LMs' percentage of correct decisions out of 10,000 cases (5,000 correct and 5,000 wrong numbers). "True positive" and "True negative" represent the percentage of error numbers correctly identified as errors and wrongly identified as correct numbers, respectively. "False positive" and "False negative" indicate the percentage of correct numbers wrongly identified as errors and correctly identified as correct numbers, respectively. "Human" indicates the detection accuracy of the human annotator on 600 randomly selected data from BeNEDect (see Ap-

Model	Accuracy	True positive	True negative	False positive	False negative	Generation error
GPT-3.5	68.0 / 64.6	28.2 / 28.1	21.8 / 16.1	10.2 / 7.1	39.8 / 36.5	0.0 / 9.8
GPT-4	**73.8 / 73.8**	37.0 / 35.8	12.9 / 13.4	13.0 / 10.7	36.8 / 38.0	0.2 / 1.5
Llama-3-8B	69.1 / 68.7	35.4 / 33.0	14.5 / 17.0	16.1 / 14.2	33.7 / 35.7	0.0 / 0.0
Mistral-7B	71.6 / 54.9	29.5 / 9.4	20.5 / 35.9	7.9 / 1.6	42.1 / 45.5	0.0 / 6.1
Mathstral-7B	65.3 / 65.7	31.1 / 21.1	18.9 / 28.9	15.8 / 5.4	34.2 / 44.6	0.0 / 0.0
FLAN-T5-xl	53.5 / 53.8	36.3 / 35.8	13.7 / 14.2	32.8 / 32.0	17.2 / 18.0	0.0 / 0.0
T0	50.1 / 51.6	49.9 / 46.7	0.1 / 3.3	49.8 / 45.1	0.2 / 4.9	0.0 / 0.0
Human	78.3	31.7	18.2	3.5	46.6	0.0

Table 3: Numerical error detection accuracy (%) of the LMs with the zero-shot and few-shot settings for BeNEDect (zero-shot accuracy / few-shot accuracy).

pendix A).

We note that a model with random prediction can still achieve a 50.0% accuracy. Therefore, models other than GPT-3.5, GPT-4, Llama 3, Mistral and Mathstral were not significantly more accurate than the random predictions. Even for FLAN-T5-xl and T0, the improvement in accuracy was only a few points, suggesting that it is still difficult to detect numerical errors, even with recent LLMs. In contrast, GPT-3.5, GPT-4, Llama 3, Mistral and Mathstral demonstrated high performance in the numerical error detection task. Furthermore, GPT-4 detected numerical errors with a high accuracy of 73.8%, even in the zero-shot setting, although the accuracy was not as high as that of humans. Moreover, it was found that the LLMs misidentified correct numbers as errors significantly more frequently than humans.

4.2.2 Few-shot Accuracy

Table 3 also lists the detection accuracy of the LMs for BeNEDect in a few-shot setting. This shows the highest accuracy for each model among the 16 prompts used in our experiments. The results for the other prompts are provided in Appendix C. The models were provided with several passages with and without numerical errors as few-shots (see Appendix B).

Compared with the zero-shot results, most models did not present significant improvements in overall accuracy. One reason for this in some models is the greater number of generation errors in the few-shot setting. The scope for improvement exists in the selection of the few-shots and in the phrasing of the prompts. In contrast, the percentage of correct numbers identified as errors decreased for most models, indicating improvement.

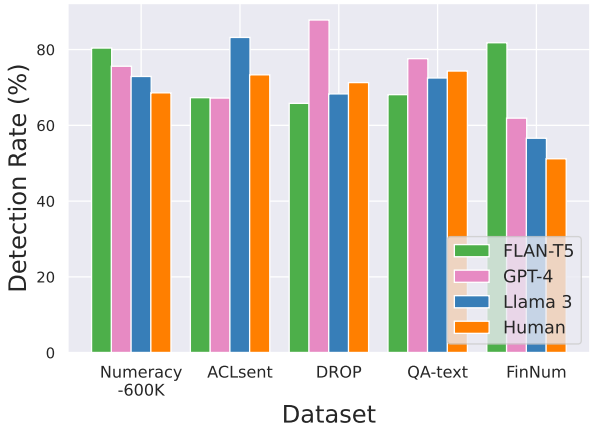

Figure 3: Numerical error detection rates by the source dataset for GPT-4, Llama 3, FLAN-T5, and humans.

4.3 Analysis of Model Performance

We analyzed the performance of GPT-4, Llama-3-8B, and FLAN-T5-xl in the zero-shot setting, which had high accuracy in our experiments, comparing their performance with the human detection accuracy from three perspectives: the passage domain, the class of numerical errors, and the operations used to generate numerical errors. The following analysis is based on the detection accuracy for 5,000 numerical errors, of which 1,500 samples are annotated with error classes (Section 3.2.4).

4.3.1 Model Performance by Passage Domain

Figure 3 illustrates the GPT-4, Llama 3, FLAN-T5, and human numerical error detection rates for each dataset. It shows that while humans had a low detection accuracy in FinNum, humans outperformed GPT-4 in ACLsent. From these results, it can be inferred that human performance is affected by context length and background knowledge (Table 1); however, detailed analysis is a topic for future research. All LMs had higher accuracy in Numeracy-600K, which indicates that they have a reliable knowledge of news and numbers used in

Figure 4: Numerical error detection rates by the error class for GPT-4, Llama 3, FLAN-T5, and humans.

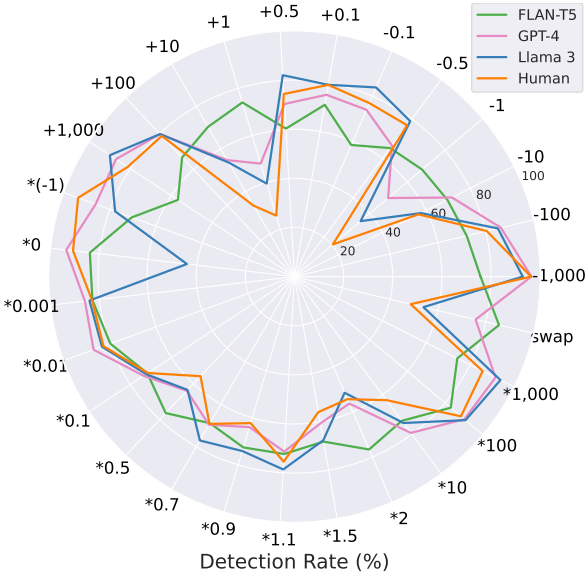

Figure 5: Numerical error detection rates by the numerical error generation method for GPT-4, Llama 3, FLAN-T5, and humans.

news titles. Llama 3 and GPT-4 demonstrated considerably higher accuracy in ACLsent and DROP, respectively, than in the other datasets. This could be due to the possibility that these models have seen the numbers that appear in the evaluation data in some datasets during training. Further investigation of this is provided in the Appendix D.

4.3.2 Model Performance by Error Class

Figure 4 shows the error detection rates for GPT-4, Llama 3, FLAN-T5, and humans according to the error class. Notably, the accuracy of FLAN-T5 did not differ significantly among error classes, whereas the accuracies of GPT-4 and Llama 3 differed significantly depending on the error class. The "undetectable error" class, which comprises errors that cannot be determined to be errors by humans, has a considerably lower detection accuracy than the other classes, not only in the case of humans. Note that the human detection rate for the "undetectable error" class is not 0% because some annotators detected some of the errors in that class. The detection rate of the "error in numerical relationships" class, which requires some type of arithmetic operation or size comparison for detection, is 6–12% lower than that of the other classes. This is consistent with the findings of related work in that even recent LLMs have difficulty in performing accurate arithmetic operations and accurately recognizing the magnitude of numbers (Singh and Strouse, 2024; Yuan et al., 2023; Mishra et al., 2022). Furthermore, it was confirmed that the "factual error" class, which was anticipated to be easier for current LMs to detect because it does not require basically number-specific capabilities for detection, represents the easiest error class to detect for most LMs.

4.3.3 Model Performance by Error Operation

Figure 5 presents the detection rates of GPT-4, Llama 3, FLAN-T5, and humans by the numerical error generation method.

Humans had a high detection rate for large errors caused by operations such as "±1,000," "±100," "*1,000," and "*100." Furthermore, the detection rate was high for numerical errors caused by operations such as "±0.5," "±0.1," "*(-1)," "*0," "*0.001," and "*0.01," which do not change the numerical value significantly but can change the number type from integer to decimal or from a positive number to a negative number or 0. The detection rate of numerical errors caused by the "-1,000" operation, which had a considerable effect on both numerical magnitude and number type, was nearly 100%. Humans were not good at detecting errors caused by operations such as "±1," "±10," "*0.5," "*1.5," and "*2," which had a slight effect on both numerical magnitude and number type. In addition, the errors caused by "swap," a simple random sampling replacement from the same dataset, were difficult for humans to detect.

Similar to that in the previous analysis, GPT-4 and Llama 3 demonstrated greater variation in accuracy per operation than FLAN-T5. Furthermore, they present trends that are considerably close to the human detection rate for each aforementioned operation. As an exception, Llama 3

was significantly less accurate in detecting numerical errors caused by "*0." In contrast, FLAN-T5 neither showed a significant difference in the detection rate between the large and small numerical errors nor it was sensitive to changes in the number type. However, the detection rates for errors caused by operations such as "*0," "*0.001," and "*0.01" were significantly higher than the other operations. This implies a high level of understanding of the numerical magnitude of 0 and whether 0 is allowed in each entity type. We believe that these differences in the ability to recognize numerical magnitudes and the understanding of number types resulted in an overall difference in the detection rates between GPT-4/Llama 3 and FLAN-T5.

5 Conclusion

In this study, we developed a benchmark for numerical error detection, called BeNEDect, using an automatic numerical error generation method and used it to evaluate the numerical error detection capability of the recent LLMs. The results indicate that GPT-3.5, GPT-4, and Llama 3 performed well in the numerical error detection task; however, they could not reach human accuracy, thereby indicating that a scope for performance improvement still exists. We confirmed that the detection accuracies of GPT-4 and Llama 3 are different from those of humans for numerical errors requiring numerical calculations or commonsense knowledge, thereby highlighting problems that LLMs currently face and opening avenues for future research on the numerical understanding of LLMs.

Limitations

We selected several numerical errors with different properties from each class of numerical errors as the few-shot and presented them to LMs in the prompts; however, we did not fully experiment with the best method for selecting the few-shot. Although we experimented with several prompt patterns, a method may provide the model with few-shot examples that can introduce the best performance of the model. The results confirmed that the detection performance of LLMs varies significantly depending on the trivial changes in the phrasing of the prompts, suggesting that achieving consistent performance and managing prompt-dependent variability remain key challenges.

Detailed instructions were required for annotation in this study; therefore, the class annotation of numerical errors was performed by several graduate students and researchers in the field of computer science without crowdsourcing. In addition, the same annotators were evaluated for their accuracy in detecting numerical errors. Therefore, some bias in the prior knowledge of the annotators is expected to exist in some domains. We were unable to conduct error class annotation and evaluate human detection accuracy in a setting that eliminates such biases, and this is a topic for future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments. This work was supported by JSPS KAKENHI Grant Numbers 24K03231.

References

Takeshi Abekawa and Akiko Aizawa. 2016. SideNoter: Scholarly paper browsing system based on PDF restructuring and text annotation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 136–140, Osaka, Japan. The COLING 2016 Organizing Committee.

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 225–237, St. Julian's, Malta. Association for Computational Linguistics.

Mistral AI. 2024. Math∑tral.

Mubashara Akhtar, Abhilash Shankarampeta, Vivek Gupta, Arpit Patil, Oana Cocarascu, and Elena Simperl. 2023. Exploring the numerical reasoning capabilities of language models: A comprehensive analysis on tabular data. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15391–15405, Singapore. Association for Computational Linguistics.

Taylor Berg-Kirkpatrick and Daniel Spokoyny. 2020. An empirical investigation of contextualized number prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4754–4764, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu,

Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Chung-Chi Chen, Hen-Hsen Huang, Yow-Ting Shiue, and Hsin-Hsi Chen. 2018. Numeral understanding in financial tweets for fine-grained crowd-based forecasting. *CoRR*, abs/1809.05356.

Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Numeracy-600K: Learning numeracy for detecting exaggerated information in market comments. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6307–6313, Florence, Italy. Association for Computational Linguistics.

Chung-Chi Chen, Hiroya Takamura, Ichiro Kobayashi, and Yusuke Miyao. 2023. Improving numeracy by input reframing and quantitative pre-finetuning task. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 69–77, Dubrovnik, Croatia. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Keunwoo Choi, George Fazekas, Mark Sandler, and Kyunghyun Cho. 2016. Convolutional recurrent neural networks for music classification. *Preprint*, arXiv:1609.04243.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *Preprint*, arXiv:2210.11416.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Abhijit Mahabal, Deepak Ramachandran, Tania Bedrax-Weiss, and Dan Roth. 2019. How large are lions? inducing distributions over quantitative attributes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3973–3983, Florence, Italy. Association for Computational Linguistics.

Simon Frieder, Luca Pinchetti, Alexis Chevalier, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *Preprint*, arXiv:2301.13867.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Nancy Ide and Keith Suderman. 2006. Integrating linguistic resources: The American national corpus

model. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Shima Imani, Liang Du, and Harsh Shrivastava. 2023. MathPrompter: Mathematical reasoning using large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 37–42, Toronto, Canada. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Zhanming Jie and Wei Lu. 2023. Leveraging training data in few-shot prompting for numerical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10518–10526, Toronto, Canada. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Xiaoyuan Li, Wenjie Wang, Moxin Li, Junrong Guo, Yang Zhang, and Fuli Feng. 2024. Evaluating mathematical reasoning of large language models: A focus on error identification and correction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11316–11360, Bangkok, Thailand. Association for Computational Linguistics.

Zhenwen Liang, Wenhao Yu, Tanmay Rajpurohit, Peter Clark, Xiangliang Zhang, and Ashwin Kalyan. 2023. Let GPT be a math tutor: Teaching math word problem solvers with customized exercise generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14384–14396, Singapore. Association for Computational Linguistics.

Bill Yuchen Lin, Seyeon Lee, Rahul Khanna, and Xiang Ren. 2020. Birds have four legs?! NumerSense: Probing Numerical Commonsense Knowledge of Pre-Trained Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6862–6868, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, and Chitta Baral. 2020. Towards question format independent numerical reasoning: A set of prerequisite tasks. *Preprint*, arXiv:2005.08516.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022. NumGLUE: A suite of fundamental yet challenging mathematical reasoning tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3505–3523, Dublin, Ireland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik

Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Libo Qin, Qiguang Chen, Xiachong Feng, Yang Wu, Yongheng Zhang, Yinghui Li, Min Li, Wanxiang Che, and Philip S. Yu. 2024. Large language models meet nlp: A survey. *Preprint*, arXiv:2405.12819.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. *Preprint*, arXiv:2110.08207.

Mandar Sharma, Rutuja Murlidhar Taware, Pravesh Koirala, Nikhil Muralidhar, and Naren Ramakrishnan. 2024. Laying anchors: Semantically priming numerals in language modeling. *Preprint*, arXiv:2404.01536.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *Preprint*, arXiv:2402.14903.

Jasivan Sivakumar and Nafise Sadat Moosavi. 2023. FERMAT: An alternative to accuracy for numerical reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15026–15043, Toronto, Canada. Association for Computational Linguistics.

Georgios Spithourakis, Isabelle Augenstein, and Sebastian Riedel. 2016. Numerically grounded language models for semantic error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 987–992, Austin, Texas. Association for Computational Linguistics.

Georgios Spithourakis and Sebastian Riedel. 2018. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2104–2115, Melbourne, Australia. Association for Computational Linguistics.

Saku Sugawara, Nikita Nangia, Alex Warstadt, and Samuel Bowman. 2022. What makes reading comprehension questions difficult? In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6951–6971, Dublin, Ireland. Association for Computational Linguistics.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Genta Indra Winata, Andrea Madotto, Zhaojiang Lin, Rosanne Liu, Jason Yosinski, and Pascale Fung. 2021. Language models are few-shot multilingual learners. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 1–15, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ancheng Xu, Minghuan Tan, Lei Wang, Min Yang, and Ruifeng Xu. 2024. NUMCoT: Numerals and units of measurement in chain-of-thought reasoning using large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14268–14290, Bangkok, Thailand. Association for Computational Linguistics.

Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *Preprint*, arXiv:2002.04326.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do large language models perform in arithmetic tasks? *Preprint*, arXiv:2304.02015.

A Numerical Error Classification Annotation and Evaluation of Human Numerical Error Detection Performance

The following are the guidelines given to the annotators for numerical error classification: Annotation was performed in a flow format that allowed multilabeling. Annotators were provided a passage containing only one numerical error while indicating which number was the target numerical error. Internet searches were not allowed because target passages in some datasets can be hit directly. In this annotation, multiple classes may receive the most votes because the annotators are allowed to select multiple labels. In such cases, the numerical error was assumed to belong to all classes that received the most votes.

1. Exclude numerical errors that are not the subject of this study, such as numerical errors in URLs and mathematical formulae.
2. Can you determine that the target number is an error? If no, select "undetectable error" and exit.
3. Can you detect the numerical error because its number type is unnatural? If yes, select "type error."
4. Is the target number considered to be known to high school or college students or found in some textbooks? If yes, select "factual error."
5. Can you detect the numerical error by referring to other numbers in the passage? If yes, select "error in numerical relationships."
6. If "type error" or "factual error" is selected, exit.
7. Can you detect the numerical error without referring to other numbers in the passage? If yes, select "anomaly" and exit. If no, exit.

We created the annotation flow such that "type error" and "factual error" are given priority over "anomaly" in cases where "type error" and "anomaly", or "factual error" and "anomaly" are compatible for numerical errors, respectively. In addition, the annotator was provided with a more detailed description of each numerical error class, its purpose, and five examples of classified numerical errors, as partially presented in the paper.

We evaluated the human detection performance with the same conditions as LMs, including not only passages containing numerical errors but also passages comprising only correct numbers.

B Prompts

Some of the prompts given to the models in the experiments are as follows:

B.1 Zero-shot

```
[zero]
Question: Answer the following yes/no question.
   Is "0.5" in the following sentence an
  error? "0.5 people attended her wedding."
Answer:

-------------------------------------
[zero_correct]
Question: Answer the following yes/no question.
   Is "0.5" in the following passage correct?
   "0.5 people attended her wedding."
Answer:

-------------------------------------
[zero_step]
Question: Answer the following yes/no question.
   Is "0.5" in the following passage an error
  ? "0.5 people attended her wedding."
Answer: Let's think step by step.

-------------------------------------
[zero_YN]
Question: Be sure to answer "Yes" or "No" to
   the following question. Is "0.5" in the
  following passage an error? "0.5 people
  attended her wedding."
Answer:

-------------------------------------
[zero_correct_step_YN]
Question: Be sure to answer "Yes" or "No" to
   the following question. Is "0.5" in the
  following passage correct? "0.5 people
  attended her wedding."
Answer: Let's think step by step.

-------------------------------------
```

B.2 Few-shot

```
-------------------------------------
[few]
Question: Answer the following yes/no question.
   Is "9" in the following passage an error?
   "Spiders have 9 limbs."
Answer: yes

Question: Answer the following yes/no question.
   Is "8" in the following passage an error?
   "Spiders have 8 limbs."
Answer: no

Question: Answer the following yes/no question.
   Is "-3.6" in the following passage an
  error? "Mike's height is -3.6 meters."
Answer: yes

Question: Answer the following yes/no question.
   Is "30.6" in the following passage an
  error? "Mike's height is 30.6 meters."
Answer: yes

Question: Answer the following yes/no question.
   Is "1.8" in the following passage an error
  ? "Mike's height is 1.8 meters."
Answer: no

Question: Answer the following yes/no question.
   Is "2,000" in the following passage an
  error? "The movie I saw yesterday was
  2,000 hours long."
Answer: yes

Question: Answer the following yes/no question.
   Is "8" in the following passage an error?
   "We split the dataset into 8:1:2."
Answer: yes

Question: Answer the following yes/no question.
   Is "7" in the following passage an error?
   "We split the dataset into 7:1:2."
Answer: no

Question: Answer the following yes/no question.
   Is "1691" in the following passage an
  error? "Wolfgang Amadeus Mozart
  (1756-1691) is great."
Answer: yes

Question: Answer the following yes/no question.
   Is "1791" in the following passage an
  error? "Wolfgang Amadeus Mozart
  (1756-1791) is great."
Answer: no

Question: Answer the following yes/no question.
   Is "0.5" in the following passage an error
  ? "0.5 people attended her wedding."
Answer:

-------------------------------------
[few_CoT]
Question: Answer the following yes/no question.
   Is "9" in the following passage an error?
   "Spiders have 9 limbs."
Answer: Spiders have 8 limbs. So the answer
  is yes.

Question: Answer the following yes/no question.
   Is "8" in the following passage an error?
   "Spiders have 8 limbs."
Answer: Spiders have 8 limbs. So the answer
  is no.

Question: Answer the following yes/no question.
   Is "-3.6" in the following passage an
  error? "Mike's height is -3.6 meters."
Answer: The number representing the height
  must be a positive number. So the answer is
   yes.

Question: Answer the following yes/no question.
   Is "30.6" in the following passage an
  error? "Mike's height is 30.6 meters."
Answer: 30.6 meters is too large for human
  height. So the answer is yes.
```

Question: Answer the following yes/no question. Is "1.8" in the following passage an error? "Mike's height is 1.8 meters."
Answer: 1.8 meters is appropriate for human height. So the answer is no.

Question: Answer the following yes/no question. Is "2,000" in the following passage an error? "The movie I saw yesterday was 2,000 hours long."
Answer: 2,000 hours is too long for the length of a movie. So the answer is yes.

Question: Answer the following yes/no question. Is "8" in the following passage an error? "We split the dataset into 8:1:2."
Answer: 8+1+2=11, but the sum of the ratios should be 10. So the answer is yes.

Question: Answer the following yes/no question. Is "7" in the following passage an error? "We split the dataset into 7:1:2."
Answer: 7+1+2=10, which is a fine as a sum of ratios. So the answer is no.

Question: Answer the following yes/no question. Is "1691" in the following passage an error? "Wolfgang Amadeus Mozart (1756-1691) is great."
Answer: The year of death must be later than the year of birth. So the answer is yes.

Question: Answer the following yes/no question. Is "1791" in the following passage an error? "Wolfgang Amadeus Mozart (1756-1791) is great."
Answer: Wolfgang Amadeus Mozart died in 1791. So the answer is no.

Question: Answer the following yes/no question. Is "0.5" in the following passage an error? "0.5 people attended her wedding."
Answer:

[few_correct_step_CoT_YN]
Question: Be sure to answer "Yes" or "No" to the following question. Is "9" in the following passage correct? "Spiders have 9 limbs."
Answer: Let's think step by step. Spiders have 8 limbs. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "8" in the following passage correct? "Spiders have 8 limbs."
Answer: Let's think step by step. Spiders have 8 limbs. So the answer is yes.

Question: Be sure to answer "Yes" or "No" to the following question. Is "-3.6" in the following passage correct? "Mike's height is -3.6 meters."
Answer: Let's think step by step. The number representing the height must be a positive number. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "30.6" in the following passage correct? "Mike's height is 30.6 meters."
Answer: Let's think step by step. 30.6 meters is too large for human height. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "1.8" in the following passage correct? "Mike's height is 1.8 meters."
Answer: Let's think step by step. 1.8 meters is appropriate for human height. So the answer is yes.

Question: Be sure to answer "Yes" or "No" to the following question. Is "2,000" in the following passage correct? "The movie I saw yesterday was 2,000 hours long."
Answer: Let's think step by step. 2,000 hours is too long for the length of a movie. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "8" in the following passage correct? "We split the dataset into 8:1:2."
Answer: Let's think step by step. 8+1+2=11, but the sum of the ratios should be 10. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "7" in the following passage correct? "We split the dataset into 7:1:2."
Answer: Let's think step by step. 7+1+2=10, which is a fine as a sum of ratios. So the answer is yes.

Question: Be sure to answer "Yes" or "No" to the following question. Is "1691" in the following passage correct? "Wolfgang Amadeus Mozart (1756-1691) is great."
Answer: Let's think step by step. The year of death must be later than the year of birth. So the answer is no.

Question: Be sure to answer "Yes" or "No" to the following question. Is "1791" in the following passage correct? "Wolfgang Amadeus Mozart (1756-1791) is great."
Answer: Let's think step by step. Wolfgang Amadeus Mozart died in 1791. So the answer is yes.

Question: Be sure to answer "Yes" or "No" to

```
    the following question. Is "0.5" in the
   following passage correct? "0.5 people
   attended her wedding."
Answer: Let's think step by step.

-------------------------------------
```

C Results for All Prompts

C.1 Zero-shot Accuracy

Table 4 lists the detection accuracy of LMs in the zero-shot setting for all prompts used in our experiments. A slight change in the prompt can significantly change the detection accuracy even for the same model in a zero-shot setting.

C.2 Few-shot Accuracy

Tables 5 and 6 list the detection accuracy of LMs in the few-shot setting for all prompts used in our experiments. A slight change in prompting can significantly change the detection accuracy even for the same model, also in the few-shot setting.

D Detection Accuracy for "Undetectable Error" Class

To further investigate the difference between the LMs and humans, we examined the accuracy of the models for the "undetectable error" class, which encompasses numerical errors that are deemed undetectable by human reviewers. Table 7 lists the results. Llama 3 detected the numerical errors in the "undetectable error" class significantly more accurately in ACLsent than in the other datasets. Similarly, GPT-4 detected errors more accurately in DROP and QA-text. This is consistent with the results in Figure 3 and might indicate that the presence of potential data leakage or shortcuts in these models for these datasets. FLAN-T5 detected the numerical errors in the "undetectable error" class with high accuracy on all datasets, indicating that FLAN-T5 is not capable of basic numerical reasoning.

Model	Prompt	Accuracy	TP	TN	FP	FN	Generation error
GPT-3.5	**zero**	**68.0%**	**28.2%**	**21.8%**	**10.2%**	**39.8%**	**0.0%**
	zero-correct	67.9%	33.5%	16.5%	15.5%	34.4%	0.1%
	zero-step	10.0%	3.2%	2.7%	1.6%	6.8%	80.8%
	zero-correct-step	28.8%	12.6%	7.3%	2.6%	16.2%	57.9%
	zero-YN	56.7%	15.6%	34.4%	8.9%	41.1%	0.0%
	zero-correct-YN	66.0%	40.6%	9.4%	24.6%	25.4%	0.0%
	zero-step-YN	39.0%	36.5%	1.8%	39.3%	2.4%	18.7%
	zero-correct-step-YN	45.2%	4.9%	34.7%	1.7%	40.2%	17.1%
GPT-4	zero-correct	71.9%	39.8%	10.2%	17.9%	32.1%	0.1%
	zero-correct-step-YN	**73.8%**	**37.0%**	**12.9%**	**13.0%**	**36.8%**	**0.2%**
Llama-3-8B -Instruct	zero	66.1%	40.1%	9.9%	24.0%	26.0%	0.0%
	zero-correct	**69.1%**	**35.4%**	**14.5%**	**16.1%**	**33.7%**	**0.0%**
	zero-step	12.2%	4.9%	3.5%	1.5%	7.3%	1.4%
	zero-correct-step	36.6%	24.3%	4.2%	8.9%	12.3%	2.6%
	zero-YN	68.7%	35.7%	14.3%	17.0%	33.0%	0.0%
	zero-correct-YN	67.7%	35.4%	14.6%	17.7%	32.3%	0.0%
	zero-step-YN	29.8%	7.5%	19.2%	4.1%	22.3%	0.4%
	zero-correct-step-YN	34.2%	23.5%	5.0%	13.1%	10.7%	0.2%
Mistral-7B -Instruct-v0.3	zero	68.1%	20.7%	29.2%	2.6%	47.4%	0.1%
	zero-correct	64.6%	32.7%	16.7%	17.2%	31.9%	1.3%
	zero-step	32.9%	5.0%	12.7%	1.1%	27.9%	49.4%
	zero-correct-step	37.9%	18.4%	8.2%	6.7%	19.5%	40.3%
	zero-YN	**71.6%**	**29.5%**	**20.5%**	**7.9%**	**42.1%**	**0.0%**
	zero-correct-YN	66.3%	33.3%	16.7%	16.8%	33.0%	0.1%
	zero-step-YN	35.7%	11.2%	10.9%	6.1%	24.5%	44.4%
	zero-correct-step-YN	41.4%	17.9%	10.7%	5.2%	23.4%	37.2%
Mathstral-7b-v0.1	zero	60.4%	12.5%	37.5%	2.1%	47.9%	0.0%
	zero-correct	62.2%	14.0%	36.0%	1.8%	48.2%	0.0%
	zero-step	41.7%	10.9%	23.4%	4.2%	30.8%	28.9%
	zero-correct-step	34.5%	8.2%	21.4%	4.3%	26.3%	35.3%
	zero-YN	**65.3%**	**31.1%**	**18.9%**	**15.8%**	**34.2%**	**0.0%**
	zero-correct-YN	57.3%	7.7%	42.3%	0.4%	49.6%	0.0%
	zero-step-YN	47.8%	35.9%	8.6%	31.4%	11.9%	11.9%
	zero-correct-step-YN	42.0%	1.7%	38.9%	0.4%	40.4%	17.3%
FLAN-T5-xl	zero	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	zero-correct	50.4%	23.0%	27.0%	22.6%	27.4%	0.0%
	zero-step	49.5%	0.9%	48.4%	0.7%	48.6%	0.0%
	zero-correct-step	51.7%	22.1%	27.9%	20.3%	29.7%	0.0%
	zero-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	zero-correct-YN	51.6%	29.7%	20.3%	28.1%	21.9%	0.0%
	zero-step-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	zero-correct-step-YN	**53.5%**	**36.3%**	**13.7%**	**32.8%**	**17.2%**	**0.0%**
Human		78.3%	31.7%	18.2%	3.5%	46.6%	0.0%

Table 4: Numerical error detection accuracy of LMs with the zero-shot setting for BeNEDect. "TP," "TN," "FP," and "FN" represent true positive, true negative, false positive, and false negative, respectively.

Model	Prompt	Accuracy	TP	TN	FP	FN	Generation error
GPT-3.5	few	44.9%	29.9%	11.0%	26.4%	15.0%	12.1%
	few-correct	50.6%	33.9%	7.9%	25.2%	16.7%	14.0%
	few-step	17.5%	12.3%	5.7%	12.8%	5.2%	51.8%
	few-correct-step	36.2%	10.4%	20.7%	3.7%	25.8%	28.6%
	few-CoT	58.9%	29.2%	12.2%	6.8%	29.7%	19.1%
	few-correct-CoT	63.9%	28.9%	15.3%	8.7%	35.0%	7.3%
	few-step-CoT	18.0%	8.6%	3.4%	1.6%	9.4%	51.9%
	few-correct-step-CoT	32.0%	13.3%	8.2%	3.3%	18.6%	43.6%
	few-YN	43.5%	18.1%	17.0%	11.0%	25.4%	14.9%
	few-correct-YN	46.9%	37.4%	2.9%	31.1%	9.5%	15.6%
	few-step-YN	9.3%	7.2%	2.0%	9.2%	2.0%	59.1%
	few-correct-step-YN	28.1%	13.4%	8.5%	4.9%	14.7%	48.4%
	few-CoT-YN	**64.6%**	**28.1%**	**16.1%**	**7.1%**	**36.5%**	**9.8%**
	few-correct-CoT-YN	62.0%	29.4%	14.2%	10.9%	32.5%	10.2%
	few-step-CoT-YN	16.5%	6.1%	4.0%	1.1%	10.3%	51.3%
GPT-4	few-correct-CoT	64.4%	30.4%	12.7%	4.2%	34.0%	11.3%
	few-correct-step-CoT-YN	**73.8%**	**35.8%**	**13.4%**	**10.7%**	**38.0%**	**1.5%**
Llama-3-8B -Instruct	few	64.7%	31.7%	18.2%	16.9%	33.0%	0.0%
	few-correct	64.0%	19.9%	29.8%	5.6%	44.1%	0.1%
	few-step	15.8%	6.2%	5.5%	0.9%	9.7%	0.7%
	few-correct-step	40.8%	8.1%	17.7%	1.8%	32.6%	0.5%
	few-CoT	55.6%	31.3%	12.1%	17.3%	24.2%	2.3%
	few-correct-CoT	61.7%	28.4%	15.9%	9.7%	33.4%	2.3%
	few-step-CoT	58.8%	26.5%	15.7%	8.1%	32.3%	2.4%
	few-correct-step-CoT	63.2%	25.0%	19.9%	6.6%	38.2%	2.9%
	few-YN	65.4%	37.1%	12.8%	21.7%	28.3%	0.0%
	few-correct-YN	65.5%	21.8%	28.1%	6.3%	43.7%	0.0%
	few-step-YN	64.4%	34.9%	15.1%	20.5%	29.5%	0.0%
	few-correct-step-YN	62.3%	19.9%	30.1%	7.6%	42.4%	0.0%
	few-CoT-YN	54.0%	47.8%	2.2%	43.7%	6.2%	0.0%
	few-correct-CoT-YN	**68.7%**	**33.0%**	**17.0%**	**14.2%**	**35.7%**	**0.0%**
	few-step-CoT-YN	52.3%	35.0%	8.4%	25.7%	17.3%	1.1%
	few-correct-step-CoT-YN	64.0%	22.8%	24.4%	6.8%	41.2%	0.7%

Table 5: Numerical error detection accuracy of LMs (GPT-3.5, GPT-4, and Llama-3-8B-Instruct) with the few-shot setting for BeNEDect. "TP," "TN," "FP," and "FN" represent true positive, true negative, false positive, and false negative, respectively.

Model	Prompt	Accuracy	TP	TN	FP	FN	Generation error
Mistral-7B -Instruct-v0.3	few	49.3%	9.9%	36.5%	9.0%	39.4%	4.3%
	few-correct	47.9%	15.9%	22.3%	6.5%	31.9%	20.9%
	few-step	52.9%	6.7%	37.0%	0.6%	46.2%	7.8%
	few-correct-step	44.1%	15.8%	20.1%	7.7%	28.4%	23.8%
	few-CoT	36.4%	7.2%	21.0%	1.8%	29.1%	36.5%
	few-correct-CoT	41.3%	16.0%	14.6%	4.4%	25.3%	34.6%
	few-step-CoT	36.7%	4.9%	22.9%	1.7%	31.9%	34.8%
	few-correct-step-CoT	33.1%	16.5%	9.2%	6.5%	16.7%	44.4%
	few-YN	52.8%	13.1%	35.2%	9.4%	39.7%	2.1%
	few-correct-YN	51.3%	18.0%	24.5%	9.0%	33.4%	13.6%
	few-step-YN	**54.9%**	**9.4%**	**35.9%**	**1.6%**	**45.5%**	**6.1%**
	few-correct-step-YN	46.5%	16.8%	21.3%	7.9%	29.6%	18.7%
	few-CoT-YN	41.3%	10.4%	21.7%	2.9%	30.9%	29.6%
	few-correct-CoT-YN	44.8%	18.2%	16.6%	6.1%	26.7%	27.5%
	few-step-CoT-YN	40.6%	7.6%	22.6%	2.4%	33.0%	30.0%
	few-correct-step-CoT-YN	36.9%	17.5%	11.9%	7.1%	19.4%	37.0%
Mathstral-7b -v0.1	few	58.6%	9.8%	40.2%	1.1%	48.8%	0.1%
	few-correct	58.3%	9.6%	40.1%	1.0%	48.7%	0.6%
	few-step	59.5%	12.0%	37.8%	2.1%	47.5%	0.6%
	few-correct-step	56.3%	7.0%	42.6%	0.3%	49.3%	0.7%
	few-CoT	51.7%	10.6%	35.6%	3.4%	41.0%	8.7%
	few-correct-CoT	56.5%	21.8%	21.6%	8.7%	34.6%	10.6%
	few-step-CoT	45.6%	11.3%	28.0%	3.0%	34.4%	22.2%
	few-correct-step-CoT	53.9%	25.8%	15.1%	12.7%	28.1%	15.6%
	few-YN	60.5%	13.1%	37.0%	2.5%	47.5%	0.0%
	few-correct-YN	**65.7%**	**21.1%**	**28.9%**	**5.4%**	**44.6%**	**0.0%**
	few-step-YN	59.8%	11.8%	38.2%	2.0%	48.0%	0.0%
	few-correct-step-YN	64.2%	20.0%	30.0%	5.8%	44.1%	0.0%
	few-CoT-YN	59.4%	23.0%	25.1%	11.5%	36.4%	3.4%
	few-correct-CoT-YN	56.9%	10.4%	37.8%	2.2%	46.5%	2.4%
	few-step-CoT-YN	51.7%	14.8%	29.6%	6.4%	37.0%	11.7%
	few-correct-step-CoT-YN	57.2%	17.5%	27.8%	6.0%	39.7%	7.8%
FLAN-T5-xl	few	52.0%	2.7%	47.3%	0.7%	49.3%	0.0%
	few-correct	46.8%	9.7%	40.3%	12.9%	37.1%	0.0%
	few-step	52.0%	6.7%	43.3%	4.6%	45.4%	0.0%
	few-correct-step	49.6%	10.1%	40.0%	10.4%	39.6%	0.0%
	few-CoT	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-CoT	50.3%	18.9%	31.1%	18.6%	31.4%	0.0%
	few-step-CoT	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-step-CoT	52.4%	21.8%	28.2%	19.3%	30.7%	0.0%
	few-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-YN	51.0%	23.9%	26.1%	23.0%	27.0%	0.0%
	few-step-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-step-YN	52.0%	23.6%	26.4%	21.6%	28.4%	0.0%
	few-CoT-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-CoT-YN	52.2%	27.5%	22.5%	25.3%	24.7%	0.0%
	few-step-CoT-YN	50.0%	0.0%	50.0%	0.0%	50.0%	0.0%
	few-correct-step-CoT-YN	**53.8%**	**35.8%**	**14.2%**	**32.0%**	**18.0%**	**0.0%**
Human		78.3%	31.7%	18.2%	3.5%	46.6%	0.0%

Table 6: Numerical error detection accuracy of LMs (Mistral-7B-Instruct-v0.3, Mathstral-7b-v0.1, and FLAN-T5-xl) with the few-shot setting for BeNEDect. "TP," "TN," "FP," and "FN" represent true positive, true negative, false positive, and false negative, respectively.

Model	N600	ACL	DROP	QA	FN
GPT-4	43.9	38.9	54.8	57.1	45.4
Llama-3-8B	37.8	65.6	45.2	37.4	46.6
FLAN-T5-xl	71.4	66.7	60.2	74.7	76.7
Human	5.1	11.1	8.6	15.4	11.0

Table 7: Detection accuracy (%) for numerical errors in the "undetectable error" class. "N600," "ACL," "QA," and "FN" represent Numeracy-600K, ACLsent, QA-text, and FinNum, respectively.