FERMAT: An Alternative to Accuracy for Numerical Reasoning

Jasivan Alex Sivakumar and **Nafise Sadat Moosavi**
Department of Computer Science
University of Sheffield
United Kingdom
{jasivakumar1|n.s.moosavi}@sheffield.ac.uk

Abstract

While pre-trained language models achieve impressive performance on various NLP benchmarks, they still struggle with tasks that require numerical reasoning. Recent advances in improving numerical reasoning are mostly achieved using very large language models that contain billions of parameters and are not accessible to everyone. In addition, numerical reasoning is measured using a single score on existing datasets. As a result, we do not have a clear understanding of the strengths and shortcomings of existing models on different numerical reasoning aspects and therefore, potential ways to improve them apart from scaling them up. Inspired by CheckList (Ribeiro et al., 2020), we introduce a multi-view evaluation set for numerical reasoning in English, called FERMAT. Instead of reporting a single score on a whole dataset, FERMAT evaluates models on various key numerical reasoning aspects such as number understanding, mathematical operations, and training dependency. Apart from providing a comprehensive evaluation of models on different numerical reasoning aspects, FERMAT enables a systematic and automated generation of an arbitrarily large training or evaluation set for each aspect.The datasets and codes are publicly available to generate further multi-view data for ulterior tasks and languages.[1]

1 Introduction

Numerical reasoning is an aspect that is often forgotten despite being an integral part of natural language. It is the ability to interact with numbers using the fundamental mathematical properties and thus model an area of human cognitive thinking (Saxton et al., 2019). Better understanding of numbers in language models would benefit various tasks like fact-checking (Vlachos and Riedel, 2015), text generation (Moosavi et al., 2021; Suadaa et al., 2021), and educational tools (Mandal et al., 2022). Current models' performance are still too weak with respect to numerical accuracy to then be used in downstream tasks like Infotabs (Gupta et al., 2020) which requires identifying numbers in tables and then performing operations to correctly label statements causing factuality errors in such tasks.

Recently, we have observed improved performances on relevant datasets about numerical reasoning using very large language models (Wei et al., 2022b; Lewkowycz et al., 2022; Kojima et al., 2022). However, there are two main limitations to this recent trend. First, as models become larger their access becomes restricted to fewer users, i.e., users with the computational resources of large companies. For example, using one of the best mathematical models, the 540B parameter model Minerva (Lewkowycz et al., 2022), would require over 2212G of memory for inference only. Second, the numerical reasoning capabilities of existing models are measured using a single score, i.e., mostly accuracy on common benchmarks like GSM8K (Cobbe et al., 2021). Therefore, their strengths and shortcomings in different aspects of numerical reasoning compared to other models are not clear. As a result, it is unclear what numerical reasoning aspects should be improved to improve their performance on datasets requiring numerical reasoning.

Motivated by CheckList (Ribeiro et al., 2020), which is a behavioral test set concerning various linguistic aspects of the input language, we propose a unique and open **F**lexible **E**valuation set for **R**epresentating **M**ultiviews of **A**rithmetic **T**ypes,[2] **FERMAT**, for evaluating the numerical reasoning capabilities of models based on multiple key aspects . It evaluates models according to (a) different ranges and representations of numbers, (b) different mathematical operations, and (c) the dependence of models on the fine-tuning data. In

[1] https://github.com/jasivan/FERMAT

[2] We use the terms type, aspect and view interchangeably.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 15026–15043

addition, it contains a tool to automatically generate new instances for each of its aspects. FERMAT enables (a) the identification of the strength and shortcomings of models according to its aspects, and (b) the automatic creation of additional training and evaluation instances using expert written templates that reflect FERMAT's categories.

FERMAT complements the recently proposed LĪLA benchmark (Mishra et al., 2022a) for mathematical reasoning. LĪLA evaluates high-level aspects, e.g. whether performing mathematical reasoning also depends on commonsense knowledge or how the performance changes depending on the difficulty of the input language. However, even the best-performing model on the LĪLA benchmark, i.e., a 2.7B parameter model that is fine-tuned on mathematical datasets, only achieves an accuracy of around 20-30 points when the input is formulated using a simple language and the test data is from a different distribution than that of the training, and it is not clear how to further improve this performance.

FERMAT, on the other hand, takes a deeper look at more fine-grained aspects by diving into the core mathematical abilities of the models and reporting which specific operations a model can or cannot perform and on which numbers. It also provides templates for creating more instances for each aspect, e.g., to generate additional data to further train or evaluate models on certain aspects. FERMAT formulates the evaluation of numerical reasoning using the question answering format, which is commonly used in NLP for evaluating various skills (Tafjord et al., 2019; Dasigi et al., 2019; Jin et al., 2019).

We use FERMAT to highlight that single accuracy scores fail to give a holistic understanding of a model, that template diversity has a high impact in improving performance, and that number encodings play an important part in numerical reasoning. The FERMAT framework could subsequently be adapted for different tasks according to the target application,[3] to give a more targeted approach to improving models. Moreover, while the expert-written templates in FERMAT are written in English, they can easily be translated to be adapted to other languages.

[3]For instance, by automatically converting our QA templates to NLI (Demszky et al., 2018) if NLI is a more suitable format for the downstream task.

2 Related Work

2.1 Datasets

Mathematical datasets focus on exploring different levels of difficulties and areas of maths. Some look at general symbolic maths, where the questions at least involve algebraic notations. A certain group of datasets explores numerical reasoning in context, but the answers may not exclusively be numerical. Unlike FERMAT, all these datasets evaluate models' performances on the whole dataset based on a single score. Moreover, as a result of the availability of many datasets, new benchmarks have also been created based on regrouping the existing datasets according to specific criteria. Such benchmarks are created based on high-level aspects, e.g., how the performance changes when solving maths also depends on commonsense reasoning, when the maths is presented using equations, a simple language, or a complex language, or when the input is presented using a different task format. However, the performance of existing general-purpose models is very low, even on the simplest aspects, e.g., when the maths is presented using a simple language without requiring external knowledge. FERMAT, on the other hand, focuses on a fine-grained analysis of numerical reasoning by aiming to decipher models' ability to understand numbers, operations, and their reliance on the training data.

2.1.1 General maths

Dolphin18K (Huang et al., 2016), DeepMind Mathematics (Saxton et al., 2019) and AQUA (Ling et al., 2017) are datasets that have a focus on solving algebraic problems and therefore use algebraic notation. These datasets are too complex for existing general purpose language models, mainly because they expect multi-hop reasoning.[4] For instance, Wei et al. (2022b) only report an accuracy around 25% for AQUA with a large, 62B parameter, model.

2.1.2 Numerical context

Instead of the algebraic notation, some datasets are worded problems but are formulated as multiple choice questions, e.g. McTaco (Zhou et al., 2019) and AQUA. This multiple choice format simplifies the task into a classification which prevents working with the continuous essence of numbers. Even if these are formatted into generative output tasks they then sometimes expect textual outputs like

[4]E.g. $[(6 \times 8) - (3 \times 6)] \div (6 + 4)$ (Ling et al., 2017).

DROP (Dua et al., 2019). DROP has textual answers that can be extracted from the context which, similarly to the multiple choice questions, are disjoint from the numerical reasoning skill.

2.1.3 Numerical solutions

The only datasets with textual input that solely expect numerical answers are GSM8K (Cobbe et al., 2021), MAWPS (Koncel-Kedziorski et al., 2016), CommonCore (Roy and Roth, 2015) and Illinois (Roy and Roth, 2016). GSM8K provides textual explanation for the solutions which has been effectively used by Wei et al. (2022b). However, similar to AQUA, GSM8K is very difficult for general purpose language models with reported results below 5% accuracy using an 8B parameter model (Wei et al., 2022b). Likewise, MAWPS requires some use of algebra to solve the problems. However, CommonCore and Illinois, which are subsets of MAWPS, are constituted of simpler one or two-hop problems.[5] Since FERMAT is designed to gain better insight by focusing on more accessible problems, CommonCore and Illinois are the ideal datasets.

2.1.4 View-based evaluation sets

Ribeiro et al. (2020) explain the motivation to move away from raw accuracy but towards more informative evaluation sets which give better insight into a given model. They look at different aspects of a test set; the skills needed to correctly solve the problem, in their case, linguistic phenomena like negation in sentiment analysis.

NumGLUE (Mishra et al., 2022b), on the other hand, is a multi-task benchmark that involves numerical reasoning. It combines different tasks like commonsense, domain specific language, quantitative expressions, with arithmetic understanding to create a more challenging benchmark. It also uses different question format such as fill-in-the-blanks, textual entailment, multiple choice questions, span extraction and numerical outputs.

A more mathematically expansive set is the recently introduced LĪLA dataset (Mishra et al., 2022a) where they regroup 20 existing datasets into 23 reasoning tasks including some of NumGLUE. These tasks are split into maths domains (e.g. geometry or arithmetics), language complexity (e.g. only maths, simple language, or long passages involving co-reference), question format (e.g. generative answer or fill in the blank), and background knowledge required (e.g. knowledge of formulae or commonsense). However, as mentioned, existing models struggle even with simple aspects that do not require background knowledge or do not contain complex language or maths. FERMAT complements LĪLA by looking in-depth at more fine-grained numerical reasoning aspects . It also contains expert-written templates associated with each aspect that can be used to generate an arbitrary number of new instances to address the identified shortcomings or generate more evaluation instances. We design FERMAT for arithmetic problems presented using simple language. However, our methodology can be tailored to refine the analysis of LĪLA's other aspects.

2.2 Improving Numerical Reasoning

The literature has two main ways of improving numerical reasoning: (a) by designing task-specific models capable of numerical reasoning (Kumar et al., 2021, 2022; Liang et al., 2022; Dua et al., 2019; Andor et al., 2019; Yang et al., 2021), and (b) by scaling up (Brown et al., 2020; Chowdhery et al., 2022; Chen et al., 2021). Both methods also attempt to further pre-train existing models on maths related data (Geva et al., 2020; Cobbe et al., 2021; Wei et al., 2022b; Lewkowycz et al., 2022; Zhou et al., 2022). Other existing ways include using better number encoding (Muffo et al., 2022) or objective functions (Petrak et al., 2022).

2.2.1 Task-specific models: Maths solvers

Some models have been specifically created to solve maths problems by outputting expressions (Kumar et al., 2021, 2022; Patel et al., 2021) or pseudo-programs (Liang et al., 2022; Dua et al., 2019) which are then evaluated using an external module. Notwithstanding the performance of these models, they can only be used to solve maths problems that, moreover, need to be represented in a closed arithmetic form. This restricts the versatility of these models both in terms of the maths and tasks that they can solve.

Unlike the other maths solvers, GenBERT (Geva et al., 2020) and NT5 (Yang et al., 2021) generate the final output as text, making them more general-purpose. Both are pre-trained on numerical and textual tasks to solve mathematical problems. Both of these models are evaluated on DROP (Dua et al., 2019) which only provides an accuracy score, so their general numerical skill performance is not

[5] An n-hop problem is one with the combination of, at most, n of the basic operations.

well-understood.[6]

2.2.2 Improving maths by scaling

More general-purpose models that perform well with respect to mathematical reasoning are GPT3 (175B) (Brown et al., 2020), PaLM (540B) (Chowdhery et al., 2022) and Codex (175B) (Chen et al., 2021) where their parameter size is given in brackets. GPT3 was fine-tuned by Cobbe et al. (2021) on GSM8K to achieve state of the art results. Similar works using PaLM and Codex investigate prompting (Wei et al., 2022b; Zhou et al., 2022) and extended training (Lewkowycz et al., 2022).

All of these models are general-purpose so are able to do more than solve maths problems but are not well understood. Some ablation studies analyse specific aspects of specific models. For instance, Lewkowycz et al. (2022) conducted a digit study and highlighted that Minerva is unable to perform any multiplication of numbers with more than seven digits. However, their sizes make it impossible for many research and industry communities to utilise them, even just at inference time. We do not have the computation resources or access for running these large models. However, FERMAT, which is publicly available and easily accessible, can be used to perform a more comprehensive analysis of these models to further identify their strengths and shortcomings.

3 Multi-view Evaluation Set: FERMAT

FERMAT gives a holistic view of a model by evaluating fine-detailed aspects of numerical reasoning. It is akin to Ribeiro et al. (2020)'s CheckList, which focuses on linguistic variations for defining its aspects. FERMAT is used to interpret models by evaluating them on three orthogonal views including (a) Number Understanding, (b) Mathematical Operations, and (c) Training Dependency. It also provides an automated method of generating new training or evaluation examples for a given number type or operation.

We collect the initial instances for creating the FERMAT evaluation set using the established Illinois (Roy and Roth, 2016) and CommonCore (Roy and Roth, 2015) datasets. After removing duplicates, we collect 1111 unique instances from these two datasets which we name the *Original* set.[7] We choose instances from CommonCore and Illinois because they perfectly fit with FERMAT's design by providing one or two-hop questions. Moreover, their extensive annotation is supplemented with an alignment between the numbers in the question and the corresponding expression that the solution is calculated from. We leverage these annotations in FERMAT to create different variations of the same problem for different aspects.

3.1 Number Understanding

Each instance of the *Original* set is used to generate 18 different numerical types where the numbers change but the language is fixed. These are categorised as (a) Alternative Representations, and (b) Range of Numbers. Examples of each is given in Table 1.

Number Understanding				Examples
Original				A euro is **5** yens. How much is **25** euros?
Alternate Representation	Same numbers	Number fixed to 1 or 2 decimal place	1	A euro is **5.0** yens. How much is **25.0** euros?
			2	A euro is **5.00** yens. How much is **25.00** euros?
		Worded form	3	A euro is **five** yens. How much is **twenty-five** euros?
		Commuted	4	A euro is **25** yens. How much is **5** euros?
	Same digits	Original as 1 or 2 decimal place numbers	5	A euro is **0.5** yens. How much is **2.5** euros?
			6	A euro is **0.05** yens. How much is **0.25** euros?
		Decimals without leading 0	7	A euro is **.5** yens. How much is **2.5** euros?
			8	A euro is **.05** yens. How much is **.25** euros?
		Original as larger than 1000	9	A euro is **5000** yens. How much is **2500** euros?
	Grouping	Digit separator (commas, spaces)	10	A euro is **323,640** yens. How much is **4,883** euros?
			11	A euro is **323 640** yens. How much is **4 883** euros?
Range of Numbers	Integers	Integers greater than 1000	12	A euro is **323640** yens. How much is **4883** euros?
		Integers less than 1000	13	A euro is **319** yens. How much is **26** euros?
		2, 3 or 4 digit integers	14	A euro is **94** yens. How much is **87** euros?
			15	A euro is **886** yens. How much is **621** euros?
			16	A euro is **2132** yens. How much is **8146** euros?
	Decimals	1 or 2 decimal place random numbers	17	A euro is **73.9** yens. How much is **9.4** euros?
			18	A euro is **0.61** yens. How much is **484.24** euros?

Table 1: Numerical Types with examples.

3.1.1 Alternative Representations

Alternative Representations transforms the numbers into 11 different forms. The first four categories (rows 1 to 4) have the same number as the *Original* set but represented differently whereas the next five categories (rows 5 to 9) use the same digits in the same order but by varying the magnitude of the number. The last two (rows 10 and 11) form the digit grouping subcategory where comma and space separators are used between groups of three digits.[8] This would give insight into the breadth of representations a model can accommodate, independent of the specific digit used, for instance,

[6] Both models report a similar performance (below 2% difference) on DROP, therefore in our work will focus on the smaller one, NT5.

[7] The *Original* set acts as the comparison to existing numerical reasoning benchmarks.

[8] These have different numbers to the original questions because the *Original* set only contains 17 numbers where digit grouping would be visible. For comparison, the numbers are identical to the large integers type from Section 3.1.2.

elucidate whether a model would be able to equally answer "12×34", "34×12" and "1.2×3.4". Note that the commutative category (row 4) refers only to operations that are invariant to operand permutation and thus only has 611 associated questions instead of 1111.

3.1.2 Range of Numbers

The *Original* set has a highly skewed distribution towards smaller integers with 94.89% of numbers being 1 or 2 digit integers. Therefore, a random number generator is used to create 7 sub-categories of a "Range of Numbers" split into integers (rows 12 to 16) with large integers (greater than 1000), small integers (less than 1000) and 2, 3 and 4 digit integers, and decimals (rows 17 and 18) with 1 or 2 decimal place numbers.

3.2 Mathematical Operations

The operations sought by the model plays a vital role in numerical reasoning. A one-hop problem which requires a single operation, to a human, would seem much easier than a two-hop problem where an intermediate calculation would need to be computed first. With regards to this, we consider 9 operation sets generated using basic operations (addition, subtraction, multiplication and division). Their distribution is given in Appendix A.

3.3 Training Dependency Classification

The frequency of the occurrence of a number in pre-training data has a great impact on the performance of the model on those numbers (Razeghi et al., 2022). Motivated by this, FERMAT also includes a view for training dependency, but at the fine-tuning or prompting-level only. Despite the test being unseen, a model could be learning the training data and focalise on seen numbers or seen operations. Therefore, we include a Training Dependency Classification aspect to FERMAT using the following classes based on what was seen during training:[9]

(a) *Exact*: all the numbers and operations are seen with the same operations modulo commutativity, e.g. "$(3 + 2) \times 5$",

(b) *All Numbers*: all the numbers are seen but with different operations, e.g. "$(5 - 2) \div 3$",

(c) *Number & Operation*: at least one number and operation are seen, e.g. "$(5 + 3) \div 4$", the "5" and the addition are at least seen,

(d) *One Number*: at least one number is seen with none of the operations, e.g. "$9 - 5$", the "5" is seen but nor with the "9", nor with subtraction,

(e) *One Operation*: at least one operation is seen without any numbers, e.g. "4+7", the addition is seen but not with these numbers.

It is important to note that all operations from the test set are seen in the training set, therefore according to our classification criteria, the least common class is always *One Operation*. Future work may have more complicated mathematical operations in the test set that are never seen at training time such as powers or trigonometric functions, but we believe these to be too difficult for the models to learn without prior exposure.

3.4 Generating Training Data

In addition to the evaluation set, FERMAT also provides a solution for generating an arbitrary length dataset that targets specific number or operation types.[10] This dataset is generated based on templates that come from three separate sources that are completely independent to the FERMAT evaluation set. The first set comprises of 100 questions written by two professional secondary school mathematics teachers and reviewed by a third one. The distribution of the templates generated reflect a uniform distribution over the operations. The second and third sources are GSM8K and AQUA where 155 and 71 templates were selected respectively. Only the questions that used at most two basic operations were extracted and the numbers were replaced by place holders to transform them into templates. These templates are only used in Section 5.4 to enhance the linguistic and mathematical variety of the templates. The distribution of operations used in the templates alongside some examples are given in Appendix B.

4 Experimental setup

To demonstrate the effectiveness of our evaluation set, FERMAT, we will perform the evaluations in two settings, (a) zero-shot, where we evaluate existing models, and (b) fine-tuned, where we further

[9] All the examples are associated to the test expression, "$5 \times (2 + 3)$".

[10] In this work, it is used for training but it could also be used for evaluation.

train the models on arithmetic data generated using our training data in Section 3.4.

4.1 Zero-shot Evaluation

For zero-shot performance, we evaluate the following models on FERMAT without any training:[11] T0 (3B) (Sanh et al., 2022), FLAN-XL (3B) (Wei et al., 2022a), BHĀSKARA (2.7B) (Mishra et al., 2022a), FLAN-large (770M), FLAN-base (220M), T5-base (220M) (Raffel et al., 2020), BART-base (140M) (Lewis et al., 2020), and NT5 (3M) (Yang et al., 2021), where the size of the models is given in brackets. A zero-shot evaluation is appropriate because these models are intended to be used as off-the-shelf multi-purpose models.

T0, FLAN, BHĀSKARA and NT5 have been trained using prompts, so we also test them with and without prompts. We select the prompts by consulting the original papers and judge which fit closest with our question answering task (see Appendix C for the exact prompts used). From the models we considered, BHĀSKARA, FLAN and NT5 are the ones that have also been trained for maths related datasets. BHĀSKARA is trained on LĪLA and reaches near state of the art performance, thus is a reliable model to compare numerical reasoning capabilities. However, since LĪLA contains lots of existing data, BHĀSKARA has seen 46.89% of the *Original* test set (Mishra et al., 2022a) at training time. It also includes DeepMind Mathematics (Saxton et al., 2019) in its pre-training data. FLAN has also seen DeepMind Mathematics in training. NT5 is pre-trained on synthetic numerical tasks involving non-worded problems with integers up to 20000, decimals, negatives and percentages and textual tasks as described by Geva et al. (2020), and then fine-tuned on DROP.

4.2 Fine-tuned Evaluation

For this setting, we create a training data called *Base* (see Section 4.2.1) on which we fine-tune the following models: FLAN-large, FLAN-base, T5-base , BART-base and NT5 accessed from Huggingface (Wolf et al., 2020). We also use a digit tokeniser as implemented by Petrak et al. (2022) which gives more promising results in fine-tuning experiments compared to using the default tokeniser for numbers.[12] Due to limitations in computational resources, we are unable to use the 3B parameter models for fine-tuning. Moreover, despite BHĀSKARA being advertised as a good starting point for maths related data, it is still too big for us to train.[13]

4.2.1 Training data

The templates described in Section 3.4 were used to generate the *Base* training set of 200K questions with a uniform distribution over four common number types, i.e. integers and decimals with 1 or 2 decimal places all between 0 and 1000, and integers between 1000 and 1000000. This distribution also means that each of these types have 50K questions, so we would suspect that all 1000 integers between 0 to 1000 and most of the 10000 1 decimal place numbers would appear in the training set whereas all 100000 and 999900 respectively from the other two categories cannot be seen. Furthermore, all of the expert templates were used therefore the operation distribution is the same as the one for the template set (see Appendix B). The same methodology was used to create a development set of 1K questions. This was used to decide on hyperparameters which are described in Appendix D.

5 Results

Table 2 illustrates the zero-shot and fine-tuning performance of eight models on FERMAT with green highlighting the stronger performances for a given arithmetic type and red the poorer ones. For models that use prompts (T0, BHĀSKARA, FLAN and NT5), for each type, we report their mean accuracy using all the prompts and no-prompt settings. For these models, the standard deviation between the prompted and non-prompted results is below 1.5%, therefore the reported results are representative (see Appendix E for the full results).

5.1 Zero-shot Evaluation

Firstly, from Table 2's sea of red, we can deduce that most of these models, especially T0 and the base models, tend to perform poorly at arithmetic reasoning, irrespective of size. The best-performing models, BHĀSKARA and FLAN-XL, are ones trained on maths data. But their performance is only respectable for a variant of the *Orig-*

[11] If the output of the examined model contains more than the numerical answer, e.g. the explanation of the answer, we only extract the numerical part from the generated output based on how the model is originally trained. For example, BHĀSKARA gives the answer before an explanation, whereas T0 provides it after.

[12] Note that NT5's tokeniser already separates the digits, so we omit the use of digit tokenisation for this model.

[13] We use NVIDIA V100 GPU nodes with a 32G memory.

	Models (size)	Number Understanding																		Mathematical Operations									
		Alternate Representations											Range of numbers							One-hop				Two-hop					
		Same numbers					Same digits					Grouping		Integers				Decimals											
		Original	Fixed 1dp	Fixed 2dp	Worded	Commuted	Original 1dp	Original 2dp	Original 1dp no 0	Original 2dp no 0	Original 1000+	1000+ comma	1000+ space	1000+ random	Integers 0 to 1000	2 digit	3 digit	4 digit	1dp random	2dp random	a+b	a-b	a*b	a/b	(a+b)-c	a*(b+c)	(a+b)/c	a*(b-c)	(a-b)/c
Zero-shot	T0 (3B)	2.88	1.98	2.79	0.39	3.49	3.99	1.29	1.47	3.33	0.93	0.00	0.09	0.09	0.18	0.75	0.12	0.06	2.04	0.27	1.18	0.66	0.70	1.90	2.14	1.33	2.43	0.37	1.49
	FLAN XL (3B)	22.86	10.44	14.52	20.13	18.28	6.66	3.57	3.69	5.79	5.28	0.00	0.00	0.00	0.45	4.83	0.33	0.00	4.08	0.33	8.61	7.66	11.65	7.92	2.73	3.42	4.25	3.63	4.98
	Bhaskara (2.7B)	23.18	21.60	20.88	18.23	18.49	5.31	3.65	3.87	4.55	4.05	0.00	0.00	0.00	0.18	3.56	0.18	0.18	1.31	0.14	8.84	7.24	8.72	9.61	2.10	5.00	8.55	2.89	8.31
	FLAN large (770M)	11.79	4.71	6.27	10.26	11.24	3.99	1.65	3.51	2.07	2.46	0.00	0.00	0.03	0.12	1.56	0.24	0.03	2.04	0.54	3.88	1.81	6.91	3.83	1.80	1.77	4.51	1.63	2.89
	FLAN base (220M)	4.98	1.95	3.90	3.69	3.98	2.88	1.83	3.48	2.22	0.93	0.00	0.00	0.00	0.18	0.90	0.33	0.00	0.54	0.12	0.88	1.18	1.94	3.10	1.41	1.25	3.17	0.54	2.15
	T5 base (220M)	1.71	2.70	1.62	0.00	2.13	2.34	0.99	2.07	1.62	0.99	0.00	0.00	0.00	0.09	0.36	0.00	0.00	0.81	0.27	0.03	1.23	0.38	1.81	1.23	0.57	1.81	0.35	1.45
	BART base (140M)	2.79	2.79	2.88	0.00	2.46	2.25	0.99	2.88	1.89	0.81	0.00	0.09	0.09	0.00	0.27	0.00	0.00	0.81	0.18	0.03	1.70	0.38	2.35	1.06	0.33	2.93	0.50	2.65
	NT5 (3M)	8.19	8.10	8.10	2.84	5.97	6.71	4.95	4.64	2.48	7.25	0.95	0.00	0.00	6.39	7.52	6.89	6.21	5.00	2.21	11.17	17.85	0.63	2.45	1.19	0.68	1.93	0.31	1.08
Fine-tuned	FLAN large (770M)	28.80	29.79	30.33	8.91	26.02	33.93	29.70	25.20	32.13	18.90	0.00	0.00	5.76	17.01	24.12	15.57	10.98	25.65	13.86	22.85	23.40	23.57	22.60	13.89	14.34	17.76	10.10	20.05
	FLAN base (220M)	26.55	27.63	27.09	6.84	19.64	29.79	27.18	19.44	26.55	15.39	0.00	0.09	6.30	15.48	21.87	15.39	11.43	24.84	15.75	30.22	29.75	18.76	21.47	7.83	5.81	7.54	5.45	19.10
	T5 base (220M)	19.44	21.24	20.34	6.39	16.53	20.88	14.31	10.17	16.02	7.65	0.00	1.17	1.89	7.29	14.76	8.91	4.23	15.84	6.84	12.53	13.27	23.41	14.46	7.13	4.56	3.13	7.35	9.45
	BART base (140M)	18.63	21.24	21.24	0.90	14.89	23.04	18.18	17.28	3.51	10.35	0.00	0.00	5.76	13.68	15.57	12.69	9.18	17.64	10.98	26.00	17.84	13.98	11.67	2.41	2.56	9.92	5.90	11.25
	NT5 (3M)	14.04	15.12	14.49	3.06	12.44	16.11	13.41	13.59	8.73	8.55	0.63	5.04	5.04	13.77	14.85	13.68	8.73	15.03	10.71	34.39	31.24	0.84	1.09	7.13	0.53	0.06	0.72	0.22

Table 2: Zero-shot and fine-tuned performances. Accuracy shown in percentage and all green scores are above the arbitrary threshold of 10% to subduce any false strong performances.

inal set where nearly half of the numbers are single digits.

Secondly, the accuracy level for *Original* is always part of the highest values, expect for NT5, so it is not a representative test set for numerical reasoning despite being derived from existing benchmarks. This could also be due to the poor diversity of the *Original* set as stressed in Section 3.1.2. Contrastingly, NT5 has its highest accuracy for addition and subtraction meaning that it is generally learning operations over specific number types.

Thirdly, even the larger models that are explicitly trained on maths datasets, i.e., BHĀSKARA and FLAN-XL, perform poorly on numbers that contain more than one digit indicating a limitation for their use in real-world tasks where the numbers can be of any range. This is in line with previous studies showing the shortcomings of models on longer digits (Lewkowycz et al., 2022; Muffo et al., 2022).

5.2 Evaluation after Fine-tuning

As expected, with many greener cells, the fine-tuned models are better than their zero-shot counterparts and demonstrate more consistent performance across all the types. FERMAT's training and evaluation set templates, while covering similar aspects, are from completely independent sources. However, we observe that fine-tuning smaller commonly used models on this training data outperforms larger models like BHĀSKARA that are fine-tuned on various maths datasets, for instance BHĀSKARA is trained on over 1.32K distinct questions and programs. This underlines the benefit of creating the training data based on a diverse set of mathematical aspects. The larger FLAN is the only model to consistently improve on the two-hop questions suggesting that more parameters may be required to learn more complex reasoning as observed by Xiong et al. (2021).

Similarly, NT5 only makes significant improvement with addition and subtraction, which it was pre-trained on with synthetic questions. Therefore, as a smaller model, NT5 is only able to better generalise mathematical addition and subtraction but struggles to learn new operations during fine-tuning. However, instead of its size, this could also be due to the complexity of mathematics it has seen at pre-training. In addition, we observe that models' performances on the "Commuted" aspect within the "Same numbers" subset are considerably lower than the other aspects. This indicates a potential for developing better number encodings that learn similar representations for the same number regardless of the position or input representation, e.g., "three" and 3, and 3.0.

5.3 Training dependency of performance

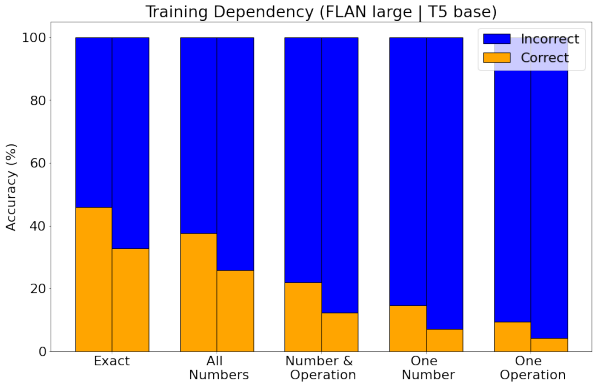

Figure 1: Training and test data overlap separated between correct and incorrect predictions made by FLAN-large (left bars) and T5-base (right bars).

It is important to understand why our fine-tuned models are better across multiple types. For this, we class the expression required to answer the test

sets using the Training Dependency Classification described in Section 3.3. Figure 1 presents the dependency of the training data for the FLAN-large (left bars) and T5-base (right bars) models. For each bar, the ratio of correct (orange) and incorrect (blue) predicted samples are identified (the full results are given in Appendix F).

The bars' monotonic trend suggests that if more of a test expression is seen at training, the model is more likely to answer it correctly. However, even for the exact match category, the performance is only 46%. This is because the language that is used to describe the targeted equation may be different in different instances, e.g. the words "another" and "increases" are only two possible terms suggesting an addition (see Appendix B for their use in context), indicating that the model needs exposure to a variety of different ways maths is expressed and that enriching the training data with higher language diversity can be beneficial.

In addition, the accuracy for *Exact* and *All Numbers* classes are similar for both models highlighting that seeing numbers during training, and therefore having a correct encoding for them, plays an important role in solving their corresponding maths operations, e.g. 89 and 30 appear both in the training set, "*Stacey prints 30 letters to post. The printer was filled with 89 sheets of paper. How many more letters could she print?*", and in the 2 digit test set, "*89 beavers were working on their home. 30 went for a swim. How many beavers are still working on their home?*". This could be seconded by FLAN-large having higher accuracy than T5-base for each class as is has seen more maths at pre-training.

5.4 Impact of training templates

As eluded in Section 5.3, linguistic and mathematical diversity seem to be key to the improvement of numerical reasoning. Therefore, we investigate a model's performance when trained with the different templates, thus diverse language and mathematics. We fix the distribution of the aspects used in all those training instances to equal amounts of "Integers 0 to 1000", "1000+ random", "1dp random" and "2dp random". We use FLAN-base for the experiments of this section as it still has particularly low performances in mainly two-hop aspects according to the results of Table 2, even after fine-tuning. Moreover, it is a small enough model to train on larger datasets.

In this section, we consider the following three training sets to compare the effect of template diversity (see Appendix G for detailed distribution): (1) *Base* is the 200K training data from Section 4.2.1 which only uses the expert templates, (2) *Base Scaled Up* is *Base* with an addition 100K instances from the same distribution of aspects. To make a fair comparison with the next training set, the language and mathematics is fixed as it only uses the expert templates, (3) *Base Diversified* starts with *Base* and also adds 100K instances from the same distribution of aspects. However, unlike all the other training sets which purely use the expert templates, this augments the initial set using templates recovered from GSM8K and AQUA (see Section 3.4) which enhances the language and mathematics seen. We compare FLAN-base fine-tuned on the above training set along with the model's zero-shot baseline performance. Figure 2 illustrates the results of these experiments.

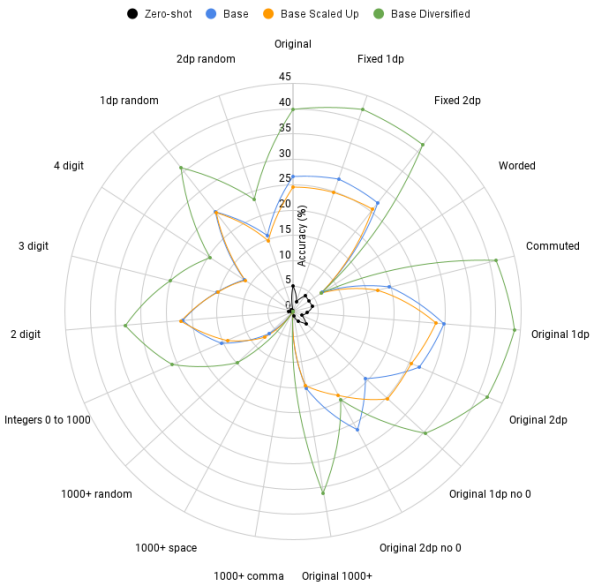

Figure 2: Fine-tuning FLAN-base on the three training sets described in Section 5.4 and the zero-shot results, see Appendix H for table of results.

First, as already established, training on diverse templates over a variety of aspects is beneficial by the shear difference illustrated by Figure 2 between *Zero-shot* (black) and the fine-tuned performance (blue, orange, green). In contrast, when comparing *Base* (blue) and *Base Scaled Up* (orange), we remark that despite seeing 100K more combinations of numbers and operations, the learning stagnates when using the same templates meaning that the model has learnt as much as it could from the breadth of the available templates. Consequently,

either linguistic or mathematical diversity is required to make a sufficient contribution. This phenomenon is, in fact, displayed by the improvement generated by *Base Diversified* (green), in certain aspect by over 21%. The diversity helps the model map the language used to describe particular mathematics better, for instance "share" to mean "division", and possibly observing more variety of this in different context seems to improve the model. Therefore, a diversity in the templates used is important, suggesting that a large variety of language may be required to attempt to further ameliorate the performance. Nevertheless, the mathematical diversity seems to also play a more important role as the diverse templates from GSM8K and AQUA have more two-hop operations (see Appendix B). Relatedly, the mean percentage increase of one-hop operations from *Base* to *Base Diversified* is approximately 95% which is about half the mean percentage increase for two-hop operations, i.e. 187%. This suggests that mathematical variation may be more central than language diversity.

Second, the variance in accuracy between "1dp random" and "2dp random" and analogously "Integers 0 to 1000" and "1000+ random" is also intriguing. Despite having the same number of training instances with these aspects the accuracy is always lower for "2dp random" and "1000+ random" respectively, the reason for this is that these aspects involve harder skill for which either the additional 100K examples or the size of the examined model is not enough to learn this skill.[14] On the other hand, for a simpler aspect like "2 digit" representation, the model's performance improves considerably using the additional training instances. We can conclude that template diversity alone may not improve the models and that work on generalisation over larger sequence of integers (i.e. integers larger than 1000, more than two decimal places) such as tokenisation and representation of numbers is critical.

Third, a noteworthy observation is that *Base Diversified* (green) performs worse than *Base* (blue) only on the "Original 2dp no 0" aspect, e.g., using ".32" instead of "0.32". When further analysing the model's output of this aspect for *Base Diversified*, we note that the model, on top of the 19.8% accuracy, produces an additional 19.7% of outputs containing correct digits but an incorrect magnitude, e.g., the correct answer might be "1.8", but the model predicts "0.18". The model might be disturbed by the decimal place or the absence of zero, implying that number encoding including positioning is vital, and thus, an accurate encoding of numbers is crucial.

[14]This is in line with our preliminary experiments where we observed that using complex maths datasets like GSM8K was not beneficial for general-purpose models to learn basic mathematical reasoning skills.

6 Conclusion

The majority of existing datasets for numerical reasoning evaluate models based on a single score, making it impossible to identify their strengths and shortcomings to further improve them. Multi-view benchmarks are the alternative for a more comprehensive and informative evaluation of models. In this direction, we introduce FERMAT, a multi-view evaluation set that enables a fine-grained analysis of models based on three key aspects including number understanding, mathematical operations, and training dependency. FERMAT's aspects are associated with separate templates for generating instances for both evaluation and training sets, which are collected from completely independent sources and domains.

Our results confirm that comparing a single accuracy score, as with all existing maths datasets, is not representative of the performance on various numerical reasoning aspects as the evaluation dataset may be skewed towards a specific data distribution. Based on our results, a wider language and mathematical variation can improve even smaller models. However, an apparent future direction is to focus on improving number encodings in existing models and understanding how these affect performance.

7 Limitations

Three main limitations with regards to certain aspects of this paper are the comparison against very large models, the distribution of the *Original* set, and the restriction of the output length.

Firstly, due to the lack of computational resources and availability of some models, we were unable to make a rigorous comparison of our fine-tuned models' as described in Section 5.2 against very large models like Minerva (Lewkowycz et al., 2022) or even Codex (Chen et al., 2021). However, these larger models can still be evaluated as FERMAT is made publicly available.

Secondly, another limitation of FERMAT is its use of Illinois and CommonCore which have highly skewed distributions of numbers (see Section 3.1.2)

and their answers are mainly integers which is not representative of the real-world. This undesired effect is mirrored in the number types that use the same numbers as *Original*. However, this was part of our design for FERMAT as the alternative would have been to combined all the ranges of numbers used with the representation, creating too many aspects but mainly conflicting with non-independent analyses between representation and range of numbers. Therefore, we chose to use the same numbers as *Original*, and since the templates will be openly accessible, they can be used to generate more combinations for wider aspects.

Lastly, when generating training questions, despite our best intentions, we had to limit the length of the output to an arbitrary length of 12 digits, therefore some number combination were not possible, for example $1 \div 3 = 0.3333...$. This practical implication could have been avoided with the use of fractions or rounding. But we judged that it would have added an extra layer of difficulty for the models and decided to restrict the output length instead.

Acknowledgements

This work was supported by the Centre for Doctoral Training in Speech and Language Technologies (SLT) and their Applications funded by UK Research and Innovation [grant number EP/S023062/1]. Additional thanks to our mathematics teachers Ana Maria Ocampo Lucumi and Liz Scott for creating and checking the expert templates. A further acknowledgement to Constantinos Karouzos, Mugdha Pandya and Valeria Pastorino for their continued feedback in this research.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. 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 5947–5952, Hong Kong, China. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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. *arXiv preprint arXiv:2107.03374*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A read-

ing comprehension dataset with questions requiring coreferential reasoning. 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 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

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.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany. Association for Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. 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 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *ICML 2022 Workshop on Knowledge Retrieval and Language Models*.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2021. Adversarial examples for evaluating math word problem solvers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2705–2712, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vivek Kumar, Rishabh Maheshwary, and Vikram Pudi. 2022. Practice makes a solver perfect: Data augmentation for math word problem solvers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4194–4206, Seattle, United States. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 997–1009, Seattle, United States. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Sourav Mandal, Swagata Acharya, and Rohini Basak. 2022. Solving arithmetic word problems using natural language processing and rule-based classification. *International Journal of Intelligent Systems and Applications in Engineering*, 10(1):87–97.

Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark, and Ashwin Kalyan. 2022a. LILA: A unified benchmark for mathematical reasoning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5807–5832, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Swaroop Mishra, Arindam Mitra, Neeraj Varshney, Bhavdeep Sachdeva, Peter Clark, Chitta Baral, and Ashwin Kalyan. 2022b. 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.

Nafise Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. Scigen: a dataset for reasoning-aware text generation from scientific tables. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Matteo Muffo, Aldo Cocco, and Enrico Bertino. 2022. Evaluating transformer language models on arithmetic operations using number decomposition. In *Proceedings of the Language Resources and Evaluation Conference*, pages 291–297, Marseille, France. European Language Resources Association.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Dominic Petrak, Nafise Sadat Moosavi, and Iryna Gurevych. 2022. Improving the numerical reasoning skills of pretrained language models. *arXiv preprint arXiv:2205.06733*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 2022. Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 840–854, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2016. Illinois math solver: Math reasoning on the web. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 52–56, San Diego, California. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, 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, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1451–1465, Online. Association for Computational Linguistics.

Oyvind Tafjord, Peter Clark, Matt Gardner, Wen-tau Yih, and Ashish Sabharwal. 2019. Quarel: A dataset and models for answering questions about qualitative relationships. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Andreas Vlachos and Sebastian Riedel. 2015. Identification and verification of simple claims about statistical properties. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2596–2601, Lisbon, Portugal. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Wenhan Xiong, Xiang Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Scott Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oguz. 2021. Answering complex open-domain questions with multi-hop dense retrieval. In *International Conference on Learning Representations*.

Peng-Jian Yang, Ying Ting Chen, Yuechan Chen, and Daniel Cer. 2021. Nt5?! training t5 to perform numerical reasoning. *arXiv preprint arXiv:2104.07307*.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. 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 3363–3369, Hong Kong, China. Association for Computational Linguistics.

Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and Hanie Sedghi. 2022. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*.

Appendix

A Distribution of Mathematical Operations

Table 3 gives the distribution of the various operations that exist in the *Original* set and thus FERMAT's evaluation set.

Hops	Expression	Frequency
One-hop	$a + b$	154
	$a - b$	162
	$a \times b$	113
	$a \div b$	102
Two-hop	$(a + b) - c$	190
	$a \times (b + c)$	100
	$(a + b) \div c$	90
	$a \times (b - c)$	100
	$(a - b) \div c$	100
Total		1111

Table 3: Distribution of the mathematical operations for the *Original* set.

B Templates

The templates' operation distribution is given by Table 4.

Operations	Freq	Operations	Freq
$\mathbf{a + b}$	16	$\mathbf{a - b}$	28
$\mathbf{a \times b}$	28	$\mathbf{a \div b}$	35
$\mathbf{a + b + c}$	9	$\mathbf{a + b - c}$	23
$\mathbf{a \times (b + c)}$	20	$\mathbf{a \times (b - c)}$	13
$\mathbf{(a + b) \div c}$	20	$\mathbf{(a - b) \div c}$	17
$\mathbf{a - b - c}$	3	$(a \div b) + c$	3
$(a \times b) + c$	13	$(a \times b) - c$	5
$(a \times b) \times c$	10	$(a \times b) \div c$	51
$a \div (b + c)$	6	$a \div (b - c)$	8
$a \times (b \div c)$	6	$(a \div b) \times c$	12
Total			326

Table 4: Table of operations present in the training templates with their corresponding frequency. The ones in bold are the ones present in the expert templates.

Exemplar templates from each of three sources are given below where number place holders are in bold:

Expert Template: Britney has **num1** knitting needles. She buys another **num2** . How many needles does she have?

Expert Expression: num1 + num2

GSM8K Template: a trader sells **num1** meters of cloth for $ **num2** . what is the cost price of one metre of cloth ?
GSM8K Expression: (num2 / num1)

AQUA Template: the average weight of **num1** persons increases by **num2** kg when a new person comes in place of one of them weighing **num3** kg . what might be the weight of the new person ?
AQUA Expression: (num3 +(num1*num2))

C Prompts

Examples of the prompts used for the respective models are given below. In the examples, the underlined text is the prompt.
Model: T0
Prompt name: Trivia
Example: Answer the following question. What is 2 plus 3?

Model: T0, FLAN
Prompt name: WebQA
Example: Question: What is 2 plus 3? Answer:

Model: FLAN
Prompt name: Trivia
Example: Please answer this question: What is 2 plus 3?

Model: NT5
Prompt name: NT5 prompt
Example: answer_me: What is 2 plus 3?

D Hyperparameters

The hyperparameters were tested on a smaller set for efficiency. During fine-tuning, we used 100 epochs with an early stopping patience of 10 and threshold of 1.0. The best model was based on accuracy of the evaluation set. All experiments were conducted with a learning rate of 5e-5, weight decay of 0.005, warm-up of 100, float32 and 3 generation beams. The rest of the hyperparameters were as the default setting in Huggingface. The max input length was 512 and max target length, 16 which is above the 12 digit limit we restrained ourselves to for the answers when generating questions. The resource used was an Nvidia Tesla V100 with 32G.

E Zero-shot results with and without prompts

The full results for each model including when prompts were used for all the arithmetic types are given by Table 6.

F Training Dependency Results

The full results for the Training Dependency classification is shown in Table 5.

Model		Exact	All Numbers	Number & Operation	One Number	One Operation
FLAN large	Incorrect	241	522	10696	287	4808
	Correct	204	315	2990	49	497
FLAN base	Incorrect	245	533	11001	268	4913
	Correct	200	304	2685	68	392
T5 base	Incorrect	299	621	12004	312	5079
	Correct	146	261	1682	24	226
BART base	Incorrect	299	605	11689	300	5182
	Correct	146	232	1997	36	123
NT5	Incorrect	382	729	13165	323	5276
	Correct	63	108	521	13	29

Table 5: Training Dependency for all fine-tuned models.

G Distribution of Training sets

Table 7 shows the distribution of the training set created from the templates, with raw numbers of instances generated based on the specific number aspect and mathematical operation design. The bold mathematical operations are the ones present in the expert templates.

H FLAN-base template diversity

Table 8 shows the results of FLAN-base for each numerical reasoning aspects as a zero-shot performance and when fine-tuned on different . Accuracy is given as a percentage. Green cells indicate higher accuracy and red poorer performance.

	Models (size)	Number Understanding																		Mathematical Operations									
		Alternate Representations											Range of numbers							One-hop				Two-hop					
		Same numbers					Same digits					Grouping		Integers				Decimals												
		Original	Fixed 1dp	Fixed 2dp	Worded	Commuted	Original 1dp	Original 2dp	Original 1dp no 0	Original 2dp no 0	Original 1000+	1000+ comma	1000+ space	1000+ random	Integers 0 to 1000	2 digit	3 digit	4 digit	1dp random	2dp random	a+b	a-b	a*b	a/b	(a+b)-c	a*(b+c)	(a+b)/c	a*(b-c)	(a-b)/c
Zero-shot	T0 (3B)	2.79	1.80	3.24	0.45	3.44	3.96	1.17	1.44	3.51	0.99	0.00	0.09	0.09	0.18	0.90	0.18	0.00	1.98	0.18	1.30	0.55	0.51	1.53	2.31	1.42	2.30	0.33	1.56
	T0 (3B) + Trivia prompt	3.06	2.25	2.61	0.63	3.60	4.05	1.26	1.44	3.33	0.90	0.00	0.09	0.09	0.18	0.72	0.09	0.09	1.98	0.27	1.06	0.79	0.79	2.02	2.17	1.47	2.12	0.33	1.22
	T0 (3B) + WebQA prompt	2.79	1.89	2.52	0.09	3.44	3.96	1.44	1.53	3.15	0.90	0.00	0.09	0.09	0.18	0.63	0.09	0.09	2.16	0.36	1.18	0.65	0.80	2.16	1.96	1.10	2.87	0.45	1.70
	T0 (3B) mean	**2.88**	**1.98**	**2.79**	**0.39**	**3.49**	**3.99**	**1.29**	**1.47**	**3.33**	**0.93**	**0.00**	**0.09**	**0.09**	**0.18**	**0.75**	**0.12**	**0.06**	**2.04**	**0.27**	**1.18**	**0.66**	**0.70**	**1.90**	**2.14**	**1.33**	**2.43**	**0.37**	**1.49**
	FLAN XL (3B)	26.01	11.07	17.01	24.03	22.75	6.84	3.33	3.33	5.76	5.67	0.00	0.00	0.00	0.45	4.59	0.27	0.00	3.78	0.27	9.07	7.61	12.02	8.39	2.59	4.27	6.53	5.39	6.28
	FLAN XL (3B) + Trivia prompt	21.78	10.26	13.32	18.81	15.88	6.39	3.33	3.60	5.31	4.95	0.00	0.00	0.00	0.45	5.04	0.27	0.00	4.23	0.36	8.62	7.37	11.54	8.01	2.63	2.82	2.76	2.83	4.67
	FLAN XL (3B) + WebQA prompt	20.79	9.99	13.23	17.55	16.20	6.75	4.05	4.14	6.30	5.22	0.00	0.00	0.00	0.45	4.86	0.45	0.00	4.23	0.36	8.13	7.99	11.40	7.35	2.97	3.16	3.47	2.67	4.00
	FLAN XL (3B) mean	**22.86**	**10.44**	**14.52**	**20.13**	**18.28**	**6.66**	**3.57**	**3.69**	**5.79**	**5.28**	**0.00**	**0.00**	**0.00**	**0.45**	**4.83**	**0.33**	**0.00**	**4.08**	**0.33**	**8.61**	**7.66**	**11.65**	**7.92**	**2.73**	**3.42**	**4.25**	**3.63**	**4.98**
	Bhaskara (2.7B)	19.44	18.90	18.27	16.83	14.89	4.68	3.42	3.87	3.87	4.23	0.00	0.00	0.00	0.18	3.24	0.18	0.18	1.26	0.18	6.91	6.21	6.42	9.42	1.50	5.71	8.46	2.22	8.50
	Bhaskara (2.7B) + prompt	26.91	24.30	23.49	19.62	22.09	5.94	3.87	3.87	5.22	3.87	0.00	0.00	0.00	0.18	3.87	0.18	0.18	1.35	0.09	10.77	8.26	11.02	9.80	2.70	4.29	8.63	3.56	8.11
	Bhaskara (2.7B) mean	**23.18**	**21.60**	**20.88**	**18.23**	**18.49**	**5.31**	**3.65**	**3.87**	**4.55**	**4.05**	**0.00**	**0.00**	**0.00**	**0.18**	**3.56**	**0.18**	**0.18**	**1.31**	**0.14**	**8.84**	**7.24**	**8.72**	**9.61**	**2.10**	**5.00**	**8.55**	**2.89**	**8.31**
	FLAN large (770M)	10.71	4.32	6.30	10.62	11.29	4.41	1.80	3.51	1.98	2.34	0.00	0.00	0.00	0.09	1.53	0.18	0.00	2.16	0.63	4.05	1.82	6.57	3.98	1.33	1.60	5.06	1.72	2.94
	FLAN large (770M) + Trivia prompt	11.61	5.40	5.49	8.19	10.97	3.69	1.44	3.69	2.16	2.70	0.00	0.00	0.00	0.09	1.80	0.36	0.09	1.98	0.54	2.99	1.65	6.72	3.81	2.75	2.08	3.46	1.28	2.61
	FLAN large (770M) + WebQA prompt	13.05	4.41	7.02	11.97	11.46	3.87	1.71	3.33	2.07	2.34	0.00	0.00	0.09	0.18	1.35	0.18	0.00	1.98	0.45	4.60	1.95	7.46	3.70	1.30	1.63	5.01	1.89	3.11
	FLAN large (770M) mean	**11.79**	**4.71**	**6.27**	**10.26**	**11.24**	**3.99**	**1.65**	**3.51**	**2.07**	**2.46**	**0.00**	**0.00**	**0.03**	**0.12**	**1.56**	**0.24**	**0.03**	**2.04**	**0.54**	**3.88**	**1.81**	**6.91**	**3.83**	**1.80**	**1.77**	**4.51**	**1.63**	**2.89**
	FLAN base (220M)	5.31	1.35	3.51	3.69	3.60	2.79	1.89	3.33	2.16	0.99	0.00	0.00	0.00	0.27	0.99	0.36	0.00	0.54	0.09	1.13	1.23	2.06	3.00	1.22	1.18	2.47	0.50	2.22
	FLAN base (220M) + Trivia prompt	4.95	2.61	4.68	2.52	3.93	2.79	1.80	3.51	2.43	0.99	0.00	0.00	0.00	0.00	0.90	0.09	0.00	0.81	0.09	0.38	0.89	1.92	3.00	1.83	1.22	3.59	0.56	2.44
	FLAN base (220M) + WebQA prompt	4.68	1.89	3.51	4.86	4.42	3.06	1.80	3.60	2.07	0.81	0.00	0.00	0.00	0.27	0.81	0.54	0.00	0.27	0.18	1.14	1.41	1.83	3.32	1.17	1.35	3.47	0.56	1.78
	FLAN base (220M) mean	**4.98**	**1.95**	**3.90**	**3.69**	**3.98**	**2.88**	**1.83**	**3.48**	**2.22**	**0.93**	**0.00**	**0.00**	**0.00**	**0.18**	**0.90**	**0.33**	**0.00**	**0.54**	**0.12**	**0.88**	**1.18**	**1.94**	**3.10**	**1.41**	**1.25**	**3.17**	**0.54**	**2.15**
	T5 base (220M)	**1.71**	**2.70**	**1.62**	**0.00**	**2.13**	**2.34**	**0.99**	**2.07**	**1.62**	**0.99**	**0.00**	**0.00**	**0.00**	**0.09**	**0.36**	**0.00**	**0.00**	**0.81**	**0.27**	**0.03**	**1.23**	**0.38**	**1.81**	**1.23**	**0.57**	**1.81**	**0.35**	**1.45**
	BART base (140M)	**2.79**	**2.79**	**2.88**	**0.00**	**2.46**	**2.25**	**0.99**	**2.88**	**1.89**	**0.81**	**0.00**	**0.09**	**0.09**	**0.00**	**0.27**	**0.00**	**0.00**	**0.81**	**0.18**	**0.03**	**1.70**	**0.38**	**2.35**	**1.06**	**0.33**	**2.93**	**0.50**	**2.65**
	NT5 (3M)	6.48	6.57	6.57	3.06	5.56	5.67	2.97	4.68	2.34	5.67	0.18	0.00	0.00	4.68	6.03	5.76	5.04	3.33	1.53	9.58	13.31	0.70	1.96	0.92	0.63	1.87	0.39	0.89
	NT5 (3M) + prompt	9.90	9.63	9.63	2.61	6.38	7.74	5.22	6.30	2.61	8.82	1.71	0.00	0.00	8.10	9.00	8.01	7.38	6.66	2.88	12.76	22.39	0.56	2.94	1.47	0.74	1.99	0.22	1.28
	NT5 (3M) mean	**8.19**	**8.10**	**8.10**	**2.84**	**5.97**	**6.71**	**4.10**	**5.49**	**2.48**	**7.25**	**0.95**	**0.00**	**0.00**	**6.39**	**7.52**	**6.89**	**6.21**	**5.00**	**2.21**	**11.17**	**17.85**	**0.63**	**2.45**	**1.19**	**0.68**	**1.93**	**0.31**	**1.08**

Table 6: Zero-shot results for separate model including different prompts. Accuracy shown in percentage.

Training Templates		Base					Base Scaled Up					Base Diversified				
		1000+ random	Integers 0 to 1000	1dp random	2dp random	Total	1000+ random	Integers 0 to 1000	1dp random	2dp random	Total	1000+ random	Integers 0 to 1000	1dp random	2dp random	Total
Mathematical Operations	**a+b**	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a-b	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a*b	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a/b	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a+b-c	2500	2500	2500	2500	10000	3750	3750	3750	3750	15000	3750	3750	3750	3750	15000
	a*(b+c)	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	(a+b)/c	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	(a-b)/c	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a*(b-c)	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a-b-c	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	a+b+c	2500	2500	2500	2500	10000	3750	3750	3750	3750	15000	3750	3750	3750	3750	15000
	(a*b)+c	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	(a*b)-c	0	0	0	0	0	0	0		0	0	1250	1250	1250	1250	5000
	(a*b)*c	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	(a*b)/c	5000	5000	5000	5000	20000	7500	7500	7500	7500	30000	6250	6250	6250	6250	25000
	a/(b+c)	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	a/(b-c)	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	(a/b)+c	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	(a/b)*c	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
	a*(b/c)	0	0	0	0	0	0	0	0	0	0	1250	1250	1250	1250	5000
Total		50000	50000	50000	50000	200000	75000	75000	75000	75000	300000	75000	75000	75000	75000	300000

Table 7: Distribution of templates for the *Base*, *Base Scaled Up* and *Base Diversified* sets. In bold are the expressions that appear in the expert templates, whereas all expressions appear in the additional GSM8K and AQUA templates.

FLAN base	Number Understanding																		Mathematical Operations									
	Alternate Representations												Range of numbers						One-hop				Two-hop					
	Same numbers					Same digits					Grouping		Integers					Decimals										
	Original	Fixed 1dp	Fixed 2dp	Worded	Commuted	Original 1dp	Original 2dp	Original 1dp no 0	Original 2dp no 0	Original 1000+	1000+ comma	1000+ space	1000+ random	Integers 0 to 1000	2 digit	3 digit	4 digit	1dp random	2dp random	a+b	a-b	a*b	a/b	(a+b)-c	a*(b+c)	(a+b)/c	a*(b-c)	(a-b)/c
Zero-Shot	4.98	1.95	3.90	3.69	3.98	2.88	1.83	3.48	2.22	0.93	0.00	0.00	0.00	0.18	0.90	0.33	0.00	0.54	0.12	0.88	1.18	1.94	3.10	1.41	1.25	3.17	0.54	2.15
Base	26.55	27.63	27.09	6.84	19.64	29.79	27.18	19.44	26.55	15.39	0.00	0.09	6.30	15.48	21.87	15.39	11.43	24.84	15.75	30.22	29.75	18.76	21.47	7.83	5.81	7.54	5.45	19.10
Base Scaled Up	24.48	24.84	25.47	6.66	17.18	28.35	25.38	25.38	18.81	14.85	0.00	0.00	7.56	14.13	22.14	15.30	11.16	24.66	14.76	23.91	34.40	22.64	17.76	8.65	5.87	6.81	5.44	18.56
Base Diversified	39.87	42.12	41.58	6.66	41.24	43.74	41.85	35.55	19.80	36.36	0.00	0.09	14.85	26.01	33.12	25.02	19.53	35.91	23.40	64.94	36.42	50.44	37.25	40.00	17.00	17.78	14.00	27.00

Table 8: Results from fine-tuning FLAN-base on different distribution of templates.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7 - Limitations

☒ A2. Did you discuss any potential risks of your work?
We do not believe our work to have potential risks, instead we aim to reduce environmental impact by looking at alternative to large models.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1 - Introduction

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3 - Multi-view Evaluation Set: FERMAT

☑ B1. Did you cite the creators of artifacts you used?
Section 2 - Related Work

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We aim to provide this when these artifacts are made available in an open repository.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 1 - Introduction, Section 3 - Multi-view Evaluation Set: FERMAT

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We do not use data with sensitive information, all names are randomly generated ones.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Abstract, Section 1 - Introduction, Section 3 - Multi-view Evaluation Set: FERMAT and Appendix

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4 - Experimental Setup, Section 5 - Results and Appendix

C ☑ Did you run computational experiments?

Section 4 - Experimental Setup, Section 5 - Results

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4 - Experimental Setup, Appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 - Experimental Setup, Appendix

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5 - Results, Appendix

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4 - Experimental Setup, Appendix

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.