Backtracking Mathematical Reasoning of Language Models to the Pretraining Data

Yasaman Razeghi[*] ♠ **Hamish Ivison**[*] ♣ **Sameer Singh** ♠ **Yanai Elazar** ◇
♠UC Irvine ♣University of Washington ◇Allen Institute for AI
yasamanrazeghi7@gmail.com

Abstract

In-context learning and chain-of-thought prompting have demonstrated surprising performance improvements on mathematical reasoning benchmarks. Therefore, understanding the underlying factors enabling these capabilities is crucial. However, the specific aspects of pretraining data that equip models with mathematical reasoning capabilities remain largely unexplored and are less studied systematically. In this study, we identify subsets of model pretraining data that contribute to math reasoning ability of the model, and evaluate it on several mathematical operations (e.g. addition, multiplication) and tasks (e.g. the asdiv dataset). We measure the importance of such subsets by continual training of the model on pretraining data subsets, and then we quantify the change in performance on the mathematical benchmark to assess their importance. If a subset results in an improved performance, we conjecture that such subset contributes to a model's overall mathematical ability. Our results unveil that while training on math-only data contributes to simple arithmetic abilities, it does not solely explain performance on more complex reasoning abilities like chain-of-thought reasoning. We also find that code data contributes to chain-of-thought reasoning while reducing the arithmetic performance.

1 Introduction

Recent works have found that pretrained language model performance is greatly improved without any additional math-specific training Kojima et al. [2022], Wei et al. [2022]. These inference time techniques are particularly effective in the realm of mathematical problems, which require more advanced reasoning capabilities. In this work, we aim to identify the source of such math and reasoning capabilities *within the pretraining data* of pretrained language models.

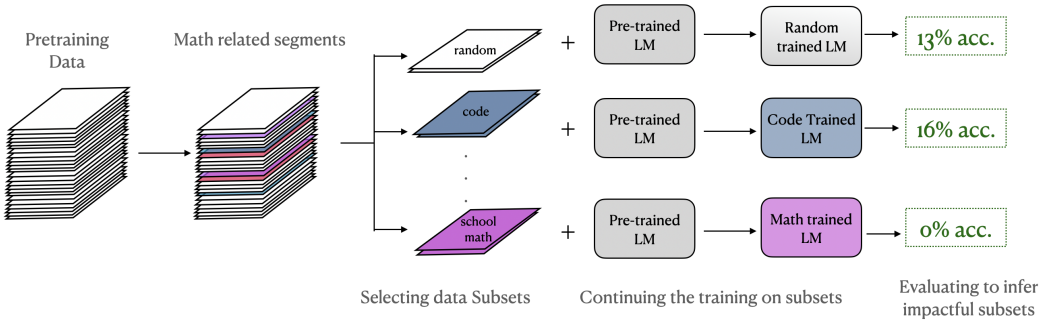
We investigate what specific subsets of pretraining data contribute to language models' mathematical reasoning abilities. Identifying and understanding the pivotal parts of pretraining data that contribute to mathematical reasoning can help to design language models' pretraining data mixtures more effectively. This understanding can also guide efforts in generating and gathering data for pretraining future language models. This task is of particular importance as we enter an era where data availability and quality become critical bottlenecks in advancing language model capabilities.

Our underlying hypothesis for identifying pivotal pretraining data subsets is straightforward: if a subset elevates benchmark performance beyond what a random subset achieves, we infer that this subset contributes to the capabilities measured by that benchmark. Since all data subsets we test are within the pretraining data, continued pretraining serves as a rough approximation of upweighting the datapoints within our chosen subsets. We propose several methods for identifying relevant subsets, based on rules (e.g., regex matches) or similarity (e.g., cosine similarity-based) methods. We also

[*]Equal contribution.

ATTRIB: Workshop on Attributing Model Behavior at Scale at NeurIPS 2023.

Figure 1: Overview of our approach to identify pretraining subsets contributing to model performance. Initially, we select subsets of the pretraining data for continued model training. Subsequently, the evaluation results – in this case, evaluating on asdiv dataset with chain of though prompting – are compared between the model further trained on these subsets and a model trained on a random subset.



employ multiple evaluation tasks, focusing on mathematical reasoning tasks such as simple arithmetic operations or the more complex asdiv [Miao et al., 2020] task with chain-of-thought prompting, to distinguish between levels of mathematical reasoning capabilities.

We illustrate our methodology in Figure 1. As shown in the figure, after segmenting the pretraining data into distinct subsets, we extend the training of the model using these data subsets and evaluate the performance of the resultant models on mathematical reasoning tasks. We then evaluate the further-trained models - in this case, on the asdiv task [Miao et al., 2020] with chain of thought prompting. By examining the relative performance of these models, we can see that additional training on the code subset improves the model's capability in the chain of thought reasoning by roughly 23% over training on random data, while further training on the school math shows no improvement. From this observation, we can conclude that code segments of the pretraining data contribute to math reasoning of language models with natural language explanations.

Our analysis shows that specific subsets, such as school math segments, enhance simple arithmetic skills in language models (e.g., resulting in more than 20% improvement on addition questions). However, they are less effective in improving more complex skills, such as the ability to improve math reasoning through chain-of-thought-based natural language explanations on the asdiv dataset. Interestingly, our findings highlight that training on code-specific subsets improves chain-of-thought mathematical reasoning when code comments are included (3% increase in evaluation benchmarks in comparison). We believe our findings offer insights into language model capabilities by associating specific pretraining data subsets with corresponding abilities. We hope that our methodology and analysis will inspire further research into the fine-grained analysis of how pretraining data mixtures contribute to model's performance on various tasks.

2 Related Work

Tracing model abilities to data Numerous previous studies have examined tracing model abilities to data [Hammoudeh and Lowd, 2023], with work on language models typically focused on the influence of finetuning data [Han et al., 2020, Guo et al., 2021], rather than data seen during pretraining. These techniques can broadly be classified as either retraining-based or gradient-based [Hammoudeh and Lowd, 2023], the former examining model capabilities after (re)training on data subsets, and the latter examining gradients to calculate the influence of training data points. Due to the extreme computational costs of the latter [Grosse et al., 2023], we take a retraining-based approach, continuing the pretraining of a model on carefully-chosen data subsets. However, unlike prior retraining-based work [Jia et al., 2021, Feldman and Zhang, 2020], we do not entirely retrain multiple models from scratch, but rather just continue model pretraining, which is considerably computationally cheaper. However, it comes at the cost of being unable to estimate pointwise influence (since we require multiple datapoints to continue pretraining without the model achieving zero loss trivially). As such, our approach allows us to identify and validate the effect of pretraining data subsets (e.g., data from the same domain), similar to prior studies studying the impact of various higher-level pretraining data statistics (e.g. presence of numbers) on model behavior [Elazar et al., 2023, Razeghi et al., 2022].

Figure 2: Example of text in Pile found by regular expression search

"... Therefore the on-shell mass derived from a short-distance calculation should correlate with the value of the coupling $\alpha_s(\mu)$. This correlation is conspicuously present in the considered here analysis of the sum rules, and can be clearly seen on the plot of Fig. 4. Within the described numerical analysis, one can find that an uncorrelated with α_s mass parameter is $m_b^* = m_b - 0.56, \alpha_s^{\overline{MS}}(\mu)\mu$ for $\mu = 1GeV$, which is determined with a very high statistical accuracy: $m_b^* = 4639 \pm 2\, MeV$, which is due to the fact that the sum rules in the considered range of n are sensitive to dynamics at distances approximately 1 GeV^{-1}...."

3 Methodology

Our objective is to identify and trace subsets of training data responsible for enhancing language models with mathematical reasoning capabilities. We do this by first selecting various data subsets, and then continuing model pretraining on these subsets. We then evaluate these models and compare their performance to the original model. Since this data was already included in the initial pretraining phase, extending the training of the language model on these specific subsets emphasizes their contribution without introducing additional information to the model. Consequently, if our evaluations show improved mathematical abilities, we attribute the math proficiency of the language model to these specific subsets. We also consider a baseline model that is further trained on a random subset of data. Our assumption is that if training on a pretraining data subset results in improved performance in comparison to our random baseline, this subset contributes to the capability of the model tested by the benchmark. We provide an overview of this approach in Figure 1.

3.1 Identifying Relevant Subsets of the Pretraining data

We start our analysis by considering a number of different subsets that could be intuitively considered as 'important':

DM-Math One of the Pile's subsets is "dm-math" Saxton et al. [2019], characterized by its pairing of mathematical questions and answers. Spanning a variety of question types, it operates predominantly at a school-level difficulty. Among the subsets of classified data in the Pile, "dm-math" is the sole segment entirely dedicated to mathematical content, making it a good candidate for enhancing mathematical reasoning.

Code Wei et al. [2022] demonstrate that models trained on code, such as Codex, significantly outperform non-code models (e.g., GPT-3.5) in evaluations of mathematical performance that involve chain-of-thought prompting. Motivated by this observation, we use a subsection from GitHub. We curate two distinct variations of this subset: one with and one without comments. We aim to discern whether the code section of the pretraining data fosters these capabilities, and if so, to ascertain whether these originate from the code itself or the code with the comments. Stripping comments keeps roughly 79% of the original Github data. We detail our comment removal process in Appendix A.

SearchMath Employing regular expression matching, we extract portions of the raw data containing numerals alongside "explanation-like" terms, such as "thus". Since chain-of-thought prompting improves the performance of models on certain math benchmarks, such as asdiv, by encouraging the generation of explanations, our underlying intuition is to identify data segments that provide explanations or solutions to mathematical queries. Our hypothesis is that this portion of data can be potentially beneficial for instilling mathematical chain-of-thought reasoning. In contrast to DM-Math, we aim to also capture problems expressed in natural language and more abstract reasoning examples. An example of this subset of data is in Figure 2.

Similarity Search We compute the cosine similarity metric between GSM8K-training data Cobbe et al. [2021] (as it is similar to our test datasets) and the Pile. We construct this segment by choosing

Figure 3: Example of text in The Pile found by similar search method

> "...the first group bought 8 slices of pizza and 6 soft drinks for \$41.14. The second group bought 5 slices of pizza and 6 soft drinks for \$30.10. How much does one slice of pizza cost? Let x = the price of a slice of pizza Let y = the price of a soft drink 8 slices of pizza and 6 soft drinks cost \$41.14 $8x + 6y = 41.14$. 5 slices of pizza and 6 soft drinks for \$30.10. $5x + 6y = 30.10$. A slice of pizza costs \$3.68 and a soft drink costs \$1.95. * * * 1. A collection of dimes and nickels is made up of 16 coins and is worth \$1.25. How many nickels are in the collection?..."

the highest-ranked sections of Pile based on this cosine similarity. Given the intensive computational demands of this method, we restrict our analysis to a randomly-chosen subset consisting of 5% of the Pile dataset. We split the GSM8K-training data and documents in pile dataset into chunks of 75 tokens and compute the embedding of each chunk using the `all-mpnet-base-v2 model` and the sentence transformers library [Reimers and Gurevych, 2019]. We provide the size of each subset in Appendix C.

3.2 Continued Training with Language Modeling Loss

Given the above data subsets, we continue the training of a trained model, using the optimizer state (of the released model) and the same hyperparameters. We use the same language modeling loss for training the base language model.

3.3 Evaluation

We then evaluate on the following tasks, covering both simple arithmetic (a common test for measuring the numerical reasoning of language models Razeghi et al. [2022], Gao et al. [2021]) and math word problems, which are known to showcase improved accuracy when models employ chain-of-thought reasoning [Wei et al., 2022]:

- **Add-2d**: Addition of two-digit numbers.
- **Add-3d**: Addition of three-digit numbers.
- **Mul-2d**: Multiplication of two-digit numbers.
- **asdiv-COT**: Elementary-level math word problems [Miao et al., 2020].

In the case of simple arithmetic tasks (add-2, add-3d, mul-2d) we evaluate on 200 randomly generated instances. The arithmetic task evaluations are for assessing the models simple math capabilities. We also employ the asdiv dataset, utilizing chain-of-thought prompting, to evaluate the models' step-by-step mathematical reasoning through natural language explanations. For asdiv, we use the existing test split of 2,305 examples. In both cases, we evaluate using 8-shot in-context learning, following the evaluation setting of prior work [Wei et al., 2022], in order to be able to reliably extract answers from non-finetuned language models. For asdiv, we use the chain of thought prompting with the prompt from [Wei et al., 2022].[2] Importantly, we check for exact matches between our evaluation sets and training data segments to mitigate the potential impact of model memorization on our results. We remove any evaluation examples we find from our training data. For example, we find 247 evaluation examples in the DM-Math train set.

3.4 Experimental Details

We use the pythia model [Biderman et al., 2023] as the pretraining data, checkpoints, and code for this model are publicly available and thoroughly documented [Gao et al., 2020]. This makes it easier to apply our continued pretraining following the exact hyperparameters used to initially pretrain the model. We use the 2.8B model as it is the smallest model that provides reasonable performance on

[2] See Appendix B for full prompt.

Table 1: Performance on evaluation tasks after training on different subsets for 2,000 steps. * we trained on the similarity and GSM8K sets for 50 steps only.[4]

Training Subset	add-2d	add-3d	mul-2d	asdiv-COT
Baseline (no training)	62.0%	4.0%	1.0%	13.9%
Random Subset	62.5%	4.5%	2.0%	12.8%
SearchMath	54.5%	3.5%	1.0%	13.1%
DM-Math	89.0%	44.5%	4.0%	0.0 %
Code (w comment)	41.5%	5.0%	1.5%	15.8%
Code (wo comment)	43.0%	9.5%	3.0%	14.7%
Similarity Subset*	74.5%	21.0%	5.5%	13.8 %
GSM8k Train Set*	81.0%	24.0%	9.5%	28.0%

our mathematical reasoning benchmarks, avoiding the cost of large-model pretraining. We follow the hyperparameters used in Biderman et al. [2023].

4 Results

Our baseline evaluations and results after training on different subsets are presented in Table 1.

We first evaluate the original performance of the language model (the Pythia 2.8B model) on mathematical tasks without any further training (**no training**). This serves as our initial reference point. The performance of our baseline model on our evaluation task is in the top row in Table 1. We then continue the training of our chosen language model on a **random subset** of the original training data. The performance after this phase will act as our second baseline. We do this to validate that training on a data mix similar to the overall pretraining mix does not dramatically alter model capabilities. We find that:

Continued pretraining on a random subset does not dramatically change model performance. As expected, continued pretraining on a random subset of pretraining data does not result in significantly changed performance. A comparison between the model trained on a random subset to the original model, as seen in Table 1, reveals a negligible performance difference of less than 1% across all evaluation datasets. This observation validates our hypothesis that the continued pretraining itself does not significantly alter model behaviour.

Math-only data only improves simple arithmetic benchmarks. Training on DM-Math, consisting of only of math question-answer pairs, significantly improves simple arithmetic performance. For example, it improves the performance of 2-digits and 3-digit addition tasks by more than 20% (i.e changing 2-digit addition from 62.5% to 89% and 3-digit addition from 4.5% to 44.5%) . However, the performance in chain-of-thought reasoning (shown by the asdiv task in Table 1) drops to zero. This suggests that math data alone is not enough to solve math word problems using chain-of-thought and that robust language skills from the model are required to connect the steps of reasoning.

Code data improves chain-of-thought reasoning, but not simple arithmetic reasoning. Models trained on code both with and without comments perform largely similarly. Both subsets (especially with comments) appear to moderately improve chain-of-thought performance by increasing the asdiv-COT performance by about 3% in comparison to the random trained model (with 12.8% accuracy), while maintaining or degrading simple arithmetic reasoning performance. These empirical results suggest that code enhances chain-of-thought reasoning abilities, which aid in solving math questions by generating explanations for each step of solving the math question.

Regex-based subsets do not significantly change performance. Training on SearchMath, which contains documents using both numeric and explanatory terms, does not significantly improve

[4]Due to the expensive nature of constructing the similarity subset, we only retrieve 3000 examples for it, and so limit the number of training steps accordingly to avoid overfitting.

performance in either simple arithmetic or more complex math as in asdiv with chain of though reasoning. Examining the data, we find that the regex search may be too broad, capturing a wide range of documents from medical papers to legal proceedings, which may drown out more useful, relevant datapoints. This qualitative examination suggests basic regular expression search might not be the best way to find the most high-quality, relevant sections of the pretraining data.

Similarity-based search yields limited improvements. We find that the data subsets found using similarity-based search improve simple arithmetic tasks, for instance, the model trained on this subset exhibits a 12.5% increase in 2-digit addition task. However, such improvements are not noted in asdiv-COT task, even though training on the GSM8K data itself (which we use for finding relevant pretraining examples) yields improvements in both arithmetic and chain-of-thought tasks. We propose two potential explanations for this observation: First, our similarity search method may be insufficient in detecting all relevant samples of the pretraining data, particularly those associated with more complex skills like chain-of-thought reasoning. Second, the lack of improvement could stem from an incomplete similarity search of the pretraining data. As previously mentioned, our similarity search was conducted on only 5% of the pretraining data, which may contribute to the observed limitations. Further investigation into this matter is reserved for future work.

5 Limitations

We note there are several limitations to our approach. First, our similarity search was confined to small sections of the pretraining data, and we anticipate discovering more substantial and influential parts if we extended our search across more data. Additionally, our method compares evaluations of models further trained on subsets of the relevant data to those trained on a random subset, and so we cannot find contributing subsets of the pretrainin data the do not yield improved performance (e.g., if the subset contributes only when combined with other data, or only contributes to small parts of our evaluation settings). We also note that we did not examine multiple models in this work, and that our results may not hold for larger models or models trained on significantly different data. However, despite such limitations we were able to find subsets that appear to contribute significantly to model performance, providing valuable insights into how different subsets of pretraining data contribute to the model's mathematical reasoning capabilities. Additionally, our method could easily be applied to varying models or benchmarks in future work. This study is still in its early stages, and we anticipate more comprehensive findings as we delve deeper into our approach.

6 Conclusion

In this work, we investigate the parts of the pretaining data that drive the mathematical reasoning of language models. Through evaluating models that we continued training on a diverse set of carefully chosen subsets from the pretraining data, we uncover several key insights into the mathematical and reasoning abilities of pretrained language models. We find that math data alone does not explain reasoning abilities, and that code data contributes to chain-of-thought math reasoning but not arithmetic abilities. Overall, our study provides insights into which parts of the pretraining data contribute to mathematical reasoning capabilities. Future work could examine extending the scope of our approach, either by investigating how to better automatically identify subsets, or examining more models and benchmarks with our approach.

7 Acknowledgment

We would like to thank the members of UCI-NLP, for valuable discussions and feedback on this work. This material is sponsored in part by the DARPA MCS program under Contract No. N660011924033 with the United States Office Of Naval Research, the NSF CAREER award number IIS-2046873 and the NSF award IIS-2008956.

References

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron,

Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023.

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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Albert Danial. cloc: v1.92, December 2021. URL https://doi.org/10.5281/zenodo.5760077.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Amir Feder, Abhilasha Ravichander, Marius Mosbach, Yonatan Belinkov, Hinrich Schütze, and Yoav Goldberg. Measuring causal effects of data statistics on language model's 'factual' predictions, 2023.

Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2881–2891. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1e14bfe2714193e7af5abc64ecbd6b46-Paper.pdf.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. Studying large language model generalization with influence functions, 2023.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.808. URL https://aclanthology.org/2021.emnlp-main.808.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: A survey, 2023.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.492. URL https://aclanthology.org/2020.acl-main.492.

Ruoxi Jia, Fan Wu, Xuehui Sun, Jiacen Xu, David Dao, Bhavya Kailkhura, Ce Zhang, Bo Li, and Dawn Song. Scalability vs. utility: Do we have to sacrifice one for the other in data importance quantification? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8239–8247, June 2021.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 975–984, 2020.

Table 2: Number of documents in each of the data subsets we examine.

Subset Name	# Documents
Random	1999980
DM-Math	9223022
Code (w comments)	28512522
Code (wo. comments)	13938984
SearchMath	344896
Similarity Search	3000

Yasaman Razeghi, Robert L Logan IV, Matt Gardner, and Sameer Singh. 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, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.59. URL `https://aclanthology.org/2022.findings-emnlp.59`.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `https://arxiv.org/abs/1908.10084`.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. *arXiv preprint arXiv:1904.01557*, 2019.

Y. SOMDA. Guesslang. `https://github.com/yoeo/guesslang`, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

A Comment Stripping

We remove comments using cloc [Danial, 2021] and guesslang [SOMDA, 2021]. We pass each document through guesslang, which uses an MLP trained on existing source code to determine which programming language the document contains. We then use cloc's `-strip-comments` command to strip comments. Notably, this removes block comments but keeps inline comments.

B ASDiv Prompt

We use the prompt seen in Figure 4 when evaluating on asdiv, following Wei et al. [2022]

C Data Subset Statistics

We show the number of documents in each data subset we examine in Table C. Note that we do not train for a single epoch for the majority of these subsets.

Figure 4: 8-shot chain of thought prompt used for evaluation on asdiv.

"Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.
Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?
A: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.
Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 - 35 = 39. The answer is 39.
Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 - 12 = 8. The answer is 8.
Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.
Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 * 4 = 20 computers were added. 9 + 20 is 29. The answer is 29.
Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael started with 58 golf balls. After losing 23 on tuesday, he had 58 - 23 = 35. After losing 2 more, he had 35 - 2 = 33 golf balls. The answer is 33.
Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 x 3 = 15 dollars. So she has 23 - 15 dollars left. 23 - 15 is 8. The answer is 8."