Automatic Generation of Socratic Questions for Learning to Solve Math Word Problems

Kumar Shridhar [*♠] **Jakub Macina** [*♠Φ] **Mennatallah El-Assady** [♠Φ]
Tanmay Sinha [▼] **Manu Kapur** [▼] **Mrinmaya Sachan** [♠]

[♠]Department of Computer Science, ETH Zurich
[Φ]ETH AI Center
[▼]Professorship for Learning Sciences and Higher Education, ETH Zurich

Abstract

Socratic questioning is an educational method that allows students to discover answers to complex problems by asking them a series of thoughtful questions. Generation of didactically sound questions is challenging, requiring an understanding of the reasoning process involved in the problem. We hypothesize that such a questioning strategy can not only enhance human performance but also assist the math word problem (MWP) solvers. In this work, we explore the ability of large language models (LMs) in generating sequential questions for guiding math word problem-solving. We propose various guided question generation schemes based on input conditioning and reinforcement learning. On both automatic and human quality evaluations, we find that LMs constrained with desirable question properties generate superior questions and improve the overall performance of a math word problem solver.

1 Introduction

Questioning can be a valuable way of supporting student thinking. It can be conceived as a scaffold Wood et al. [1976], Quintana et al. [2004], where a more knowledgeable tutor helps a student in solving problems otherwise too difficult. The role of a teacher using questioning is to interject questions that *focus* on the most critical points in an explanation and take the understanding *forward* Anghileri [2006]. Even though question generation (QG) models have been studied for factual SQuAD-like questions [Rajpurkar et al., 2016, Puri et al., 2020], these models fail to generate sequentially-coherent questions [Reddy et al., 2019, Choi et al., 2018]. Furthermore, domain-specific questioning is challenging as the QG model needs to understand the reasoning process required to provide fine-grained responses.

In this work, we explore the use of large language models [Raffel et al., 2019, Radford et al., 2019] to generate guiding sub-questions for math word problems. We define important components of Socratic questioning strategy and demonstrate the effectiveness of inducing the defined questioning properties in large LMs. In particular, we use reinforcement learning (RL) with rewards from various sources including Math question answering (Math QA) models and various forms of input conditioning for generating these questions.

We train and evaluate our models on the recently released GSM8K MathQA dataset Cobbe et al. [2021] of multi-step reasoning MWPs. We illustrate the benefit of our RL-based generation strategy using both automatic and human evaluation metrics. Our evaluation shows that our guided approach makes the generation model ask more *logically relevant* and *structurally correct* questions, which follow the appropriate sequencing of questioning at the *right granularity* level. We further show that

[*] Equal contribution; correspondence at: {shkumar, macinaj}@ethz.ch

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on Math-AI.

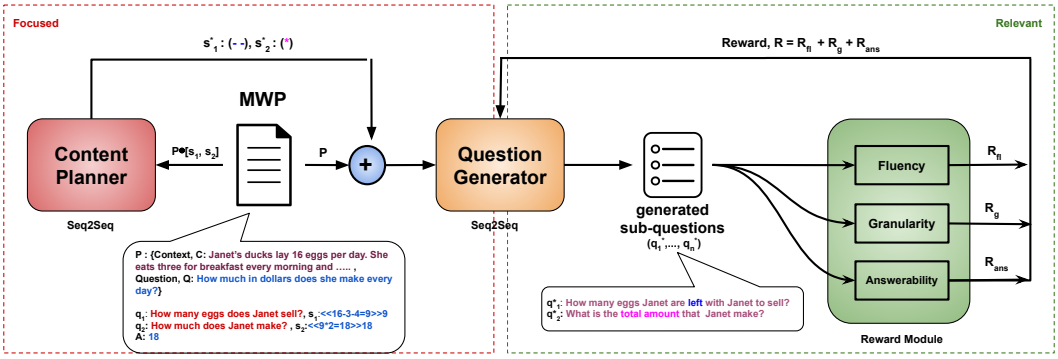

Figure 1: **Our overall methodology:** Two Socratic properties of focused (red dotted box) and relevant (green dotted box) question generation are added to the question generation model. $\oplus$ represents the concatenation operation.

our generated questions, when provided as additional context, can aid a math question answering model and make making intermediate reasoning steps explicit, thereby providing further empirical justification of the value of questioning for math QA model training.

2 Methodology

Inspired by the prior learning sciences work in scaffolding Wood [1994], Anghileri [2006], we hypothesize the most important components of a Socratic questioning strategy as:

(A) **Focused**: An essential property of a good questioning strategy is to ask questions that are directed toward the most critical domain-specific content. Irrelevant questions not only make the process difficult but also force a diversion in the focus and may increase the cognitive load that a student experiences.

(B) **Relevant**: Asking the right sequence of relevant questions that can assist students in reaching the final goal (solving the main question in case of math word problems) is a further important part of good questioning.

We discuss our approach to modeling Socratic questioning using large LMs. We begin by defining our data. We define an MWP dataset $\mathcal{D}$ as a collection of MWPs. Each MWP P in the dataset are accompanied by its solution $\mathbf{S}$ and the numerical answer A. We do not always assume the existence of problem solutions $\mathbf{S}$ and answers A as will also automatically derive them from various MathQA models. Each MWP $P = (C, Q)$ consists of the story context C and the question Q. The problem solution $\mathbf{S}$ consists of n solution steps $\mathbf{S} = (s_1, ..., s_n)$. We define Socratic questioning such that each solution steps s_i can be mapped to a sub-question q_i . We refer to $\mathbf{q}$ as a collection of all Socratic questions $q_1, ..., q_n$ for a given MWP P in our work. An example MWP is present in Figure 1.

Our main module is the Question Generator (QG) module, which is a transformer Vaswani et al. [2017] based encoder-decoder model. The QG model takes the reference Math word problem P and generates the Socratic questions $\mathbf{q}^*$ as close to the true sub-questions $\mathbf{q}$ as possible. The learning objective of the QG module is as:

$$\mathcal{L}_{QG} = -\sum_{i=1}^{n} \log P_{Dec}\left(q_i | q_{:i-1};\ Enc(P)\right) \tag{1}$$

where Enc represents the encoder and Dec represents the decoder for the seq2seq QG model. Note that the sub-question q_i is generated word by word in an auto-regressive manner. Next, we propose to inject the two Socratic questioning properties into our QG model as follows.

2.1 Focused questions

To learn a sequence of disciplined questions focused on specific reasoning steps in the MWP, it is important to ask the right set of questions. We propose a *content planner* ψ that serves as a guiding principle for the QG model to ask the right focused questions. In principle, the content planner

module can extract any relevant information to assist the QG model, but for the task of math word problems, we restrict it to operators (e.g. multiplication) and equations.[2]

We use the same seq2seq architecture for the content planner module as our QG model, with the only difference being that the output comprises a set of equations $s_1^*, .., s_n^*$ or just the operators within the equations (instead of the sub-questions). The generated operators/equations are appended to the input MWP P in the encoder for the QG module and the modified focused learning objective $\mathcal{L}_{QG_f}$ is:

$$\mathcal{L}_{QG_f} = -\sum_{i=1}^{n} \log P_{Dec}\left(q_i | q_{:i-1};\ Enc([P \oplus \text{plan}])\right) \tag{2}$$

Here, `plan` depicts the content planner module's output and $\oplus$ depicts the concatenation operation.

2.2 Relevant questions

An essential element of a good questioning strategy is to ask relevant questions that are not only factually associated with the main problem, but also eventually help in answering the main question. However, there can be any number of relevant questions that can be asked for an MWP. Thus, our goal is to optimize the questioning strategy such that it is relevant, efficient, and rewarding at each step, making sure that the final goal can be achieved with these individual questions. We induce these properties in our QG model using various *rewards* that force the model to stay relevant to the problem.

During training, the QG model samples a set of sub-questions $\mathbf{q}'$, and calculates various rewards based on $\mathbf{q}'$. The parameters of the QG model are updated using the REINFORCE algorithm Williams [2004] as:

$$\mathcal{L}_{\text{RL}} = -\mathbb{E}_{\mathbf{q}' \sim P_{Dec}}[R(\mathbf{q}, \mathbf{q}', P)] = -R(\mathbf{q}, \mathbf{q}', P) \sum_{i=1}^{n} \log P_{Dec}\left(q_i | q_{:i-1};\ Enc(P)\right)$$

The reward function $[R(\mathbf{q}, \mathbf{q}', \mathbf{P})]$ measures the individual rewards for fluency, granularity, and answerability and each individual reward is described in more detail in the Appendix A.2.

2.3 Overall Loss Function

Finally, with the induced Socratic properties in the QG model, the total loss is defined as a combination of the focused learning loss $\mathcal{L}_{QG_f}$ and the rewards loss $\mathcal{L}_{RL}$, as:

$$\mathcal{L} = \alpha\, \mathcal{L}_{QG_f} \ +\ (1-\alpha)\, \mathcal{L}_{RL} \tag{3}$$

where α is a weighting factor.

3 Empirical Analysis

We study the properties of Socratic questioning on the GSM8K dataset Cobbe et al. [2021][3] that consists of 8.5K grade school math word problem with 7.5K training problems and 1K test problems. We used T5 Raffel et al. [2019] as the backbone of both our QG and content planning modules. For the reward-generating QA model, we used GPT-2 Radford et al. [2019] because of resource constraints. However, to improve our performance, a better QA model like GPT-3 Brown et al. [2020] can be used in the future. Both the models are fine-tuned on the GSM8K train set using the Huggingface library Wolf et al. [2019].

We report automatic evaluation using SacreBLEU Post [2018] which is based on exact word overlap, BERT F_1 score Zhang et al. [2019] which is based on DeBERTa He et al. [2020] as the similarity model. We also report $\#Q$, the number of questions generated compared to the number of ground truth reasoning steps (same as *Granularity* reward), and Math QA Solver accuracy (same as the overall *Answerability* reward) to assess if our generated questions helped the QA model reach the final numerical solution.

[2]We also do not consider the step-by-step solutions S in our work, as creating step-by-step textual solution requires a lot of time and effort from teachers and even the largest language models fail to understand MWPs easily [Wei et al., 2022, Chowdhery et al., 2022].

[3]https://github.com/openai/grade-school-math

Strategy	BLEU	BERT F1	#Q	QA Accuracy
Baseline	13.02	0.566	0.056	2.57
No Planning	51.53	0.783	0.428	6.74
+ fluency	52.21	0.784	0.440	-
+ # of questions	51.86	0.784	0.431	-
+ QA	52.22	0.783	0.417	-
+ all weighted	53.39	0.781	0.431	6.75
Operators	54.98	0.788	0.346	7.50
Equations	58.82	0.813	0.807	8.49
+ fluency	59.52	**0.816**	**0.818**	-
+ # of questions	**59.75**	0.814	0.811	-
+ QA	59.37	0.813	0.799	-
+ all weighted	59.62	0.815	0.815	**8.50**

Table 1: **Results Comparison**: QG and QA model performance comparison using different planning strategies and rewards. All experiments are run for at least 3 times and an average score is reported.

Results in Table 1 demonstrate that planning strategies improve the baseline methods by more than 3% on BLEU score with operators as planning, and by more than 7% with equations. Similar to BLEU, we achieve better performance on BERT F_1 scores too. Finally, the number of correct question counts improves with planning and doubles compared to the no-planning variant. However, results show that in all the variants the number of generated sub-questions is less than the number of reasoning steps. This could be improved further by oversampling during the beam search (beam search settings are the same for all variants in this experiment).

Using rewards as a strategy to incentivize the model to generate relevant and rewarding questions improves all the model performance, suggesting the importance of rewards. Moreover, sub-questions with operators and equations as planning improves the QA performance by $1-2\%$. Rewards, although improves the QG quality, have a negligible effect on QA performance. This is mainly because slight improvement in sub-questions quality does not necessarily help in reaching to the final goal.

Human quality evaluation

Next, we perform a human evaluation of the questions generated for 100 randomly selected test MWPs to assess the quality of our model generation (our best model) compared to the baseline (with no planning or reward-based strategies). For this analysis, we divided the questions among 4 annotators with an overlap of 40% of the questions among them to evaluate the generated question quality on the following factors. A 5-point Likert scale ranging from 1 (poor) to 5 (very good) was used for each dimensions of quality assessment: *repetition* - whether questions are repeated, *factuality* - whether all questions can be solved by the information given in the problem, *logical relevance* - if the question is logically related to the MWP, *right sequence* - correct sequence of questions leading to the final answer, *granularity* - questions are granular enough to solve the problem but are still relevant and no retrieval or basic common sense questions are asked, *completeness* - questions are complete with all steps covered to reach to the final answer, and *fluency* - grammatical correctness and fluent in the language.

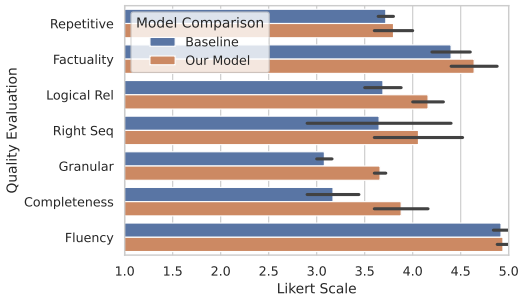

Figure 2: Comparison of baseline versus our model generated sub-questions on several metrics from our human evaluations (showing mean and standard deviation).

Figure 2 presents our findings, clearly demonstrating that our planning and reward strategies lead to superior quality questions on the MWP task. Although both baselines and our model-generated text achieve almost full score (5) on the fluency parameter, our model-generated questions that are more aligned to the MWP, thus leading to a higher score on all the other parameters. We also present a randomly selected sample of generated questions in Appendix subsection A.3.

4 Conclusion

We study the importance of sub-questioning for learning a mathematical concept and explore how LMs may generate these sub-questions. We demonstrate the usefulness of Socratic questioning strategies and propose ways to induce these properties in LMs on a Maths Word Problem dataset. However, we need to be careful in using the questioning strategy in real educational contexts, as improper content can sometimes do more harm than good.

References

Julia Anghileri. Scaffolding practices that enhance mathematics learning. *Journal of Mathematics Teacher Education*, 9(1):33–52, 2006.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. *CoRR*, abs/1908.04942, 2019. URL `http://arxiv.org/abs/1908.04942`.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Bhuwan Dhingra, Manzil Zaheer, Vidhisha Balachandran, Graham Neubig, Ruslan Salakhutdinov, and William W. Cohen. Differentiable reasoning over a virtual knowledge base. *CoRR*, abs/2002.10640, 2020. URL `https://arxiv.org/abs/2002.10640`.

Zhihao Fan, Zhongyu Wei, Piji Li, Yanyan Lan, and Xuanjing Huang. A question type driven framework to diversify visual question generation. In *IJCAI*, pages 4048–4054, 2018.

Heng Gong, Wei Bi, Xiaocheng Feng, Bing Qin, Xiaojiang Liu, and Ting Liu. Enhancing content planning for table-to-text generation with data understanding and verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2905–2914, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.262. URL `https://aclanthology.org/2020.findings-emnlp.262`.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

Qingbao Huang, Mingyi Fu, Linzhang Mo, Yi Cai, Jingyun Xu, Pijian Li, Qing Li, and Ho-fung Leung. Entity guided question generation with contextual structure and sequence information capturing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:13064–13072, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17544`.

Nam Ju Kim, Brian R Belland, and Andrew E Walker. Effectiveness of computer-based scaffolding in the context of problem-based learning for stem education: Bayesian meta-analysis. *Educational Psychology Review*, 30(2):397–429, 2018.

Tassilo Klein and Moin Nabi. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*, 2019.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Liangming Pan, Wenhu Chen, Wenhan Xiong, Min-Yen Kan, and William Yang Wang. Unsupervised multi-hop question answering by question generation. *arXiv preprint arXiv:2010.12623*, 2020.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. Bleu: a method for automatic evaluation of machine translation. 10 2002. doi: 10.3115/1073083.1073135.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Ratish Puduppully and Mirella Lapata. Data-to-text generation with macro planning. *CoRR*, abs/2102.02723, 2021. URL https://arxiv.org/abs/2102.02723.

Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*, 2020.

Chris Quintana, Brian J. Reiser, Elizabeth A. Davis, Joseph Krajcik, Eric Fretz, Ravit Golan Duncan, Eleni Kyza, Daniel Edelson, and Elliot Soloway. A scaffolding design framework for software to support science inquiry. *Journal of the Learning Sciences*, 13(3):337–386, 2004. doi: 10.1207/s15327809jls1303_4. URL https://doi.org/10.1207/s15327809jls1303_4.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

Brian J. Reiser. Scaffolding Complex Learning: The Mechanisms of Structuring and Problematizing Student Work. *Journal of the Learning Sciences*, 13(3):273–304, July 2004. ISSN 1050-8406. doi: 10.1207/s15327809jls1303_2. URL https://doi.org/10.1207/s15327809jls1303_2. Publisher: Routledge _eprint: https://doi.org/10.1207/s15327809jls1303_2.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. Generate & rank: A multi-task framework for math word problems. *arXiv preprint arXiv:2109.03034*, 2021.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. *arXiv preprint arXiv:2004.05483*, 2020.

Katherine Stasaski and Marti A. Hearst. Multiple choice question generation utilizing an ontology. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 303–312, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5034. URL https://aclanthology.org/W17-5034.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. Plan-then-generate: Controlled data-to-text generation via planning. *CoRR*, abs/2108.13740, 2021. URL https://arxiv.org/abs/2108.13740.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ruonan Wang, Yuxi Qian, Fangxiang Feng, Xiaojie Wang, and Huixing Jiang. Co-vqa: Answering by interactive sub question sequence. *arXiv preprint arXiv:2204.00879*, 2022.

Zichao Wang, Andrew S Lan, Weili Nie, Andrew E Waters, Phillip J Grimaldi, and Richard G Baraniuk. Qg-net: a data-driven question generation model for educational content. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, pages 1–10, 2018.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903, 2022. URL https://arxiv.org/abs/2201.11903.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

David Wood, Jerome S Bruner, and Gail Ross. The role of tutoring in problem solving. *Child Psychology & Psychiatry & Allied Disciplines*, 1976.

Terry Wood. Patterns of interaction and the culture of mathematics classrooms. In *Cultural perspectives on the mathematics classroom*, pages 149–168. Springer, 1994.

Zhipeng Xie and Shichao Sun. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305, 2019.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. Graph-to-tree learning for solving math word problems. Association for Computational Linguistics, 2020.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] [section 1]
   (b) Did you describe the limitations of your work? [Yes] [section 2]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] [section 4]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] [subsection A.1]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] [section 2]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] [section 2]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] [subsection A.1]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] [section 2]
   (b) Did you mention the license of the assets? [Yes] [section 2]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] [section 2]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] [section 2]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experiment Details

A.1 Implementation Details

For the training of the models, we used Nvidia Tesla A100 with 40 GB of GPU memory. We ran each experiment for 50 epochs, with a periodical evaluation on the validation set. Training time without using rewards is 10 minutes per epoch. With rewards, the training time per epoch is increased to several hours. We used the T5-large model without modifications for the content planner and question generation module and GPT-2 small as QA solver.

A.2 Definition of Rewards functions

Fluency: It is important that the generated sub-questions are easily understandable and fluent in the meaning they represent. Although the QG training objective ensures the syntax and semantics of the questions generated, rewarding the system to stay fluent is necessary to remove repetitions and illogical questions.

$$R_{fl} = BLEU(\mathbf{q}, \mathbf{q}')$$

where, BLEU(.,.) represents the BLEU score Papineni et al. [2002].

Granularity: As solving an MWP usually involves multiple reasoning steps, asking relevant questions at each step can help in solving the MWP. Moreover, our questioning strategy is based on the fact that the questions are organized, structured, and follow a sequence. With the granularity reward, the model can learn to ask the right number of questions (compared to the number of reasoning steps to solve MWP) in a specific sequence and refrain from unstructured questions.

$$R_g = F(\mathbf{q}, \mathbf{q}')$$

where, $F(\mathbf{q}, \mathbf{q}') = \mathbf{1} - \frac{||\mathbf{q}|-|\mathbf{q}'||}{|\mathbf{q}'|}$, and $|\mathbf{q}|$ and $|\mathbf{q}'|$ denote the number of questions in $\mathbf{q}$ and $\mathbf{q}'$ respectively.

Answerability: For every generated question, it is important to evaluate if the generated questions can be answered given context C and can help in answering the overall MWP. We trained an external QA model that can answer the MWPs by taking help from the sub-questions and evaluating if the generated question can assist in answering the main problem. The answerability reward is provided on both a step-by-step basis (if the QA model can answer a sub-part of the main problem) and overall (if using all sub-questions, whether the final answer was correct or not).

$$R_{ans} = F(A, A')$$

where, $F(A, A') = 1$ if the final answer from the QA model is correct when it is given sub-questions $\mathbf{q}'$ alongside the MWP $\mathbf{P}$, and 0 otherwise. A' denotes the answer from the QA model and A denotes the true answer.

We also evaluated the step-by-step performance of the QA model on the generated sub-questions to check if the QA model can answer the generated sub-questions correctly. This allows us to provide partial rewards at each step to the generation model. The modified sub-step answerability reward is $F(A, A') = \frac{\#a'}{|\mathbf{q}'|}$, where $\#a'$ and $|\mathbf{q}'|$ denote the number of correct answers to the generated sub-questions and total number of generated questions respectively.

A.3 Generated questions and error analysis

Id	Text	Generated question prompts
1	James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week? (540)	How many sprints does James run in a week? How many meters does James run in a week?
2	Carla is downloading a 200 GB file. Normally she can download 2 GB/minute, but 40% of the way through the download, Windows forces a restart to install updates, which takes 20 minutes. Then Carla has to restart the download from the beginning. How long does it take to download the file? (160)	**Missing: time to download 40% of the file** How many minutes does it take to install updates? How many minutes does it take to restart the download? How many minutes does it take to download the file?
3	John buys twice as many red ties as blue ties. The red ties cost 50% more than blue ties. He spent $200 on blue ties that cost $40 each. How much did he spend on ties? (800)	How many blue ties did John buy? How many red ties did John buy? How much did the red ties cost? How much did John spend on ties?
4	Mr. Tan sold his house for $400 000. He paid the transfer fees that amount to 3% of the selling price and also paid a brokerage fee that is 5% of the selling price. If he also paid $250 000 for the remaining loan amount of the house, how much is Mr. Tan's net proceeds from selling the house? (118000)	How much did Mr. Tan pay in transfer fees and brokerage fees? How much did Mr. Tan pay in total? How much is Mr. Tan's net proceeds from selling the house?
5	John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours? (45)	How far did John drive in the first 3 hours? How far did John drive in the remaining 2 hours? How fast did John drive in the next half-hour? How fast did John drive in the remaining 2 hours? **Error: remaining 1 and half hour.** How far did John drive in those 4 hours? How far is John from home at the end of those 4 hours?

Table 2: Some examples of our generated sub-questions

B Further Experiments

B.1 RQ1: Does sub-questioning help in understanding math concepts better?

We hypothesize that high-quality sub-questioning helps Math QA solvers to reach the correct solution, especially when questions are relevant to the concept to be learnt, in the right sequence (ordering) with high granularity in their structure. We use the GPT-2 model as QA solver and fine-tune it on the training set of the GSM8K dataset. Then, we vary properties of the test sub-questions, and examine the performance of the QA Solver. Table 3 demonstrates that Socratic sub-questions significantly improve the model performance from 5.45% to 10.46%. Sub-questioning even helps when only 75% Socratic questions are retained (denoted as $\{q\}^{0.75}$ in the table) or when the order is shuffled (this might be an artifact of the dataset containing minority of examples with strict order). An interesting observation is that when the number of sub-questions is reduced by half or lower (while preserving their order), the model gets confused and performs worse than when it had no sub-questions. Finally, we take the pre-trained T5 model and without fine-tuning it for our task, we take the outputs, and used it alongside the problem P as additional information to solve the problem. The performance goes as low as 2.57%, indicating that non-relevant information degrades the performance.

B.2 Ablation study: Manipulating question properties

Both planning strategies are helpful in generating better questions. To gain a deeper understanding of how content planner ψ affects generated questions, we further analyze the influence of operators as a planning strategy. Here, we randomize operators and their sequence and measure change in performance. Table 4 shows that correct sequence of operators with correct number of operators guide

Variation	QA Accuracy
P	5.45 ($\downarrow$ 45%)
P $\oplus$ $\{q\}$	**10.46**
Granularity	
P $\oplus$ $\{q\}^{0.25}$	3.94 ($\downarrow$ 62%)
P $\oplus$ $\{q\}^{0.5}$	3.35 ($\downarrow$ 67%)
P $\oplus$ $\{q\}^{0.75}$	9.70 ($\downarrow$ 7%)
Order	
P $\oplus$ shuffle($\{q\}$)	8.94 ($\downarrow$ 14%)
Relevance	
P $\oplus$ <base-ques>	2.57 ($\downarrow$ 75%)

Table 3: Comparison of Math QA accuracy (in %) for different variations of experiments with ground truth data. $\{q\}^k$ represents that only $k\%$ of the ground truth sub-questions are used and selected randomly. For e.g., $\{q\}^{0.25}$ represents only 25% of the sub-questions are used. shuffle($\{q\}$) represents all sub-questions, but with shuffled order. Finally, `<base-ques>` are the sub-questions generated from a T5 large model without fine-tuning on our task. ($\downarrow$) represents the drop in the accuracy when compared to the Socratic questions (P $\oplus$ $\{q\}$). $\oplus$ represents the concatenation operation.

Planning	BLEU	BERT F1	#Q
None	51.53	0.783	0.428
Diff op, diff cnt	51.59	0.785	0.415
Diff op, same cnt	54.26	0.786	0.546
Operators (op)	**54.98**	**0.788**	**0.642**

Table 4: Manipulating the planning inputs influences the quality of generated questions and overall QG model performance. **same cnt** has the same number of operators as number of reasoning steps but the types (+-/*) are shuffled, **diff cnt** has both number and type of operator shuffled.

the generation process better than randomized versions. A gap between correct count of operators and random count indicates that having a correct number of operators (of any type) is more valuable than the exact type of operators. We observed that the number of operators guide the model in terms of the number of questions that need to be asked, while type changes the overall quality. Needless to say, for the same number of operators, quality matters.

C Related Work

Socratic questioning approaches have evolved within the learning sciences community into the *theory of scaffolding* Wood et al. [1976], Reiser [2004], which broadly refers to assisting students in problem-solving beyond their zone of proximal development Quintana et al. [2004]. Computer-based scaffolds (e.g., in the form of hints, prompts, feedback) have moderate effects on student learning outcomes Kim et al. [2018], and our work can be used to automatically generate such scaffolds in a form of questioning prompts. For mathematics, Wood [1994] analyzed interactions in math classrooms and proposed two distinct interaction patterns - *funneling*, which functions by guiding students using leading/prompting questions to a predetermined solution procedure, and *focusing*, which functions by drawing student attention to the critical aspects of the problem. We draw inspiration from this strand of work. Our overall question generation approach can be conceived to be similar to funneling, with specific sub-questions focusing on the important domain concepts.

Research on question generation includes visual question generation Fan et al. [2018], Wang et al. [2022], generation of questions for student assessment Stasaski and Hearst [2017], Wang et al. [2018], generation of factual questions based on Wikipedia articles Rajpurkar et al. [2016] or generation of sequential information seeking questions in dialogue-based scenarios Reddy et al. [2019], Choi et al. [2018]. Other work has also explored similar ideas of improving answerability by question-asking Klein and Nabi [2019], Shwartz et al. [2020]. However, factual questions do not usually require much reasoning and mostly boil down to information retrieval from text. In this work, we focus on question generation for reasoning problems.

Prior work on guided and controlled question generation uses either entities as guiding mechanism Huang et al. [2021] or reinforcement learning based graph to sequence approach Chen et al. [2019]. Identification of entities and relationships present in the text often uses rule-based or on-shelf extraction tools, which are hard to extend Dhingra et al. [2020]. Often these single hop questions are

combined to form a multi-hop question that require complex reasoning to solve it Pan et al. [2020]. Controllable text generation has been previously explored in data-to-text generation Puduppully and Lapata [2021], Su et al. [2021]. This is particularly useful for ensuring that the information is correct or the numbers are encapsulated properly Gong et al. [2020]. Our task has similar requirements.

A final strand of related work lies in the ballpark of math problem solvers. Recent work in this area uses specialized architectures such as graph-based encoders Zhang et al. [2020] and tree-based decoders Xie and Sun [2019], and more recently, large pretrained LMs which show state-of-the-art results Cobbe et al. [2021], Shen et al. [2021], Kojima et al. [2022], Wei et al. [2022], Chowdhery et al. [2022]. Application of these approaches to the GSM8K dataset (our data context) still holds considerable room for improvement, primarily in the reasoning capabilities, and latest approaches are still unable to solve third of the problems.