Towards Development of Models that Learn New Tasks from Instructions

by

Swaroop Mishra

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

Approved February 2023 by the
Graduate Supervisory Committee:

Chitta Baral, Chair
Yezhou Yang
Eduardo Blanco
Arindam Mitra

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

Humans have the remarkable ability to solve different tasks by simply reading textual instructions that define the tasks and looking at a few examples. Natural Language Processing (NLP) models built with the conventional machine learning paradigm, however, often struggle to generalize across tasks (e.g., a question-answering system cannot solve classification tasks) despite training with lots of examples. A long-standing challenge in Artificial Intelligence (AI) is to build a model that learns a new task by understanding the human-readable instructions that define it. To study this, I led the development of NATURAL INSTRUCTIONS and SUPERNATURAL INSTRUCTIONS, large-scale datasets of diverse tasks, their human-authored instructions, and instances. I adopt generative pre-trained language models to encode task-specific instructions along with input and generate task output. Empirical results in my experiments indicate that the instruction-tuning helps models achieve cross-task generalization. This leads to the question: how to write good instructions? Backed by extensive empirical analysis on large language models, I observe important attributes for successful instructional prompts and propose several reframing techniques for model designers to create such prompts. Empirical results in my experiments show that reframing notably improves few-shot learning performance; this is particularly important on large language models, such as GPT3 where tuning models or prompts on large datasets is expensive. In another experiment, I observe that representing a chain of thought instruction of mathematical reasoning questions as a program improves model performance significantly. This observation leads to the development of a large scale mathematical reasoning model BHASKAR and a unified benchmark LILA. In case of program synthesis tasks, however, summarizing a question (instead of expanding as in chain of thought) helps models significantly. This thesis also contains the study of instruction-example equivalence, power of decomposition instruction to replace the need for new models and origination of dataset bias from crowdsourcing instructions to better understand the advantages and disadvantages of instruction paradigm. Finally, I apply the instruction paradigm to match real user needs and introduce a new prompting technique HELP ME THINK to help humans perform various tasks by asking questions.

Dedicated to the service of the Almighty

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Dr. Chitta Baral for his invaluable guidance. He has helped me in exploring many research problems, experimenting with new ideas and collaborating with a diverse set of people. I could pursue my ideas and research visions with full support from Dr. Baral. I am grateful to him for being a constant source of encouragement.

I am also thankful to my thesis committee members Dr. Eduardo Blanco, Dr. Yezhou Yang, and Dr. Arindam Mitra for their guidance and suggestions in my research. Their expertise in the field, insightful comments, and unwavering belief in my abilities were instrumental in helping me complete my research.

I would also like to extend my thanks to my mentors: Dr. Hanna Hajishirzi (Allen Institute for AI), Dr. Daniel Khashabi (Allen Institute for AI), Dr. Ashwin Kalyan (Allen Institute for AI), Dr. Peter Clark (Allen Institute for AI), Dr. Elnaz Nouri (Microsoft Research), Dr. Kuzman Ganchev (Google Research) and Chris Alberti (Google Research) for their invaluable feedback, encouragement, and support.

I would like to acknowledge the financial support provided by ASU, DARPA and the resources made available by ASU HPC that enabled me to conduct my research and complete this thesis.

I extend my appreciation to my colleagues and friends: Anjana Arunkumar, Yizhong Wang, Man Luo, Pulkit Verma, Bhavdeep Sachdeva, Pan Lu, Pratyay Banerjee, Tejas Gokhale, Kuntal Pal, Neeraj Varshney, Mihir Parmar, Shailaja Sampat, Yiran Luo, Ravsehaj Singh Puri, Kirby Kuznia, Himanshu Gupta, Maitreya Patel, Pruthvi Patel, Ankita Sharma, Neha Vadnere, other members of the Cognition and Intelligence Lab and Tsu (pet) who provided me with emotional support and encouragement during this journey. They were a constant source of strength and motivation.

I am blessed to have a good number of awesome friends: Amulya, Kiran, Sandip, Dipu, Varun, Makara, Subhasish, Jyoti, Subrat, Suhan, Nikunj, Sumeet, Rakesh. Their support and confidence in me pushed me further in research.

Finally, I would like to express my deep love and gratitude to my parents, sister, mama, and all my family members for unwavering support and encouragement. Their love and support were the foundation that sustained me throughout this journey. Thank you all for your unconditional love and support.

TABLE OF CONTENTS

Page

CHAPTER

APPENDIX

LIST OF TABLES

Table Page

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

The traditional approach in Machine Learning (ML) has been to learn a map between input and output pairs. With the evolution of ML, various tasks have been defined e.g. NLP tasks: NLI, Numerical reasoning QA, Co-reference resolution etc. Can we leverage the task information we have available to help ML models learn a task quickly with a very few data samples? The recent advancement in language models to understand language paves a way for utilizing the task information in ML. Our proposal is motivated from humans who often learn a task quickly and comprehensively from task instructions, e.g. there is evidence by the work of ANDERSON and MAGILL (2021) that humans understand an exercise quickly when they are provided with instructions that describe the exercise, than when they are asked to learn directly from few examples that demonstrate the exercise. A possible explanation behind this is that instructions provide a holistic view of the task, whereas there is a risk of a specific type of bias while learning just from examples. An advantage of the proposed instruction learning paradigm is that often the instructions are easily available, e.g. the high quality crowdsourcing instructions which typically helps crowdworkers in creating consistent and large scale datasets.

A long-standing challenge in AI is to build a model that learns a new task by understanding the human-readable instructions that define it. To study this, we build NATURAL INSTRUCTIONS and SUPERNATURAL INSTRUCTIONS, large-scale datasets of diverse tasks, their human-authored instructions, and instances. We adopt generative pre-trained language models to encode task-specific instructions along with input and generate task output. Our results indicate that the instruction-tuning helps models achieve cross-task generalization. This leads to the question: how to write good instructions? Backed by extensive empirical analysis of large language models, we observe important attributes for successful instructional prompts and propose several reframing techniques for model designers to create such prompts. Our results show that reframing notably improves few-shot learning performance; this is particularly important on large language models, such as GPT3 where tuning models or prompts on large datasets is expensive. We also observe that representing a chain of thought instruction of mathematical reasoning questions as a program improves model performance significantly. We leverage our observation to build a large scale mathematical reasoning model BHASKAR and a unified benchmark LILA. In case of program synthesis tasks, we observe that summarizing a question (instead of expanding as in chain of thought) helps models significantly.

In order to better understand the advantages and disadvantages of the instruction-learning paradigm, we study the extremities of the instruction learning paradigm. We investigate the power of question decomposition as an alternative to the development of new models since building new models may not be an ideal option owing to the cost, time and environmental impact associated with it. We explore an alternative route: can we modify data by expressing it in terms of the model's strengths, so that a question becomes easier for models to answer? We empirically observe that decomposition improves model performance significantly. We also observe that instruction examples contain the risk of propagating dataset bias. On experimenting across a range of datasets, we observe that instruction examples used to create NLU benchmarks often exhibit clear patterns that are propagated by annotators to the collected data. In addition, we investigate the effect of instruction bias on model performance, showing that instruction patterns can lead to overestimation of model performance as well as limit the ability of models to generalize to other task examples. Finally, we apply the instruction paradigm to match real user needs and introduce a new prompting technique

HELP ME THINK to help humans perform various tasks by asking questions.

To quantify the benefits of instruction-learning over conventional machine learning, we investigate: how many data samples is an additional instruction worth? We observe that instruction learning is very effective in a low data regime: an additional instruction can be equivalent to around 200 data samples on average across tasks. We also observe that Instruction tuned models are quick learners, as further instruction tuning an instruction-tuned model on downstream tasks surpasses the SOTA model with just 25% of training samples in both single task and multitask settings. We also propose methods to learn from less data, e.g. our data selection method enables RoBERTA to achieve near-equal performance on 2% data of SNLI. Finally, we build several new benchmarks to help learn tasks that are relatively harder for language models to learn, e.g. we propose NumGLUE, a multi-task benchmark that evaluates the performance of AI systems on eight different tasks, that at their core require simple arithmetic understanding.

We focus on the development and experiments associated with Natural Instructions in chapter 3. We have witnessed great progress in solving many NLP datasets through fine-tuning pre-trained language models (LMs) (Peters *et al.*, 2018; Brown *et al.*, 2020). More recent studies show tremendous promise in generalization *within* the set of observed tasks through multi-task training and unified encoding (Khashabi *et al.*, 2020; Aghajanyan *et al.*, 2021). However, cross-task generalization – *generalization* to *unseen* tasks – has generally remained under-explored. For example, can we supervise a model with instances of grammar checking or question answering tasks, yet expect it to solve a different task like question typing (Fig.1.1). Evidently, humans are capable of such generalizations; an average human can follow natural language *instructions* to solve a variety of problems, as evident by the success of crowdsourcing platforms (also argued in (Efrat and Levy, 2020)).

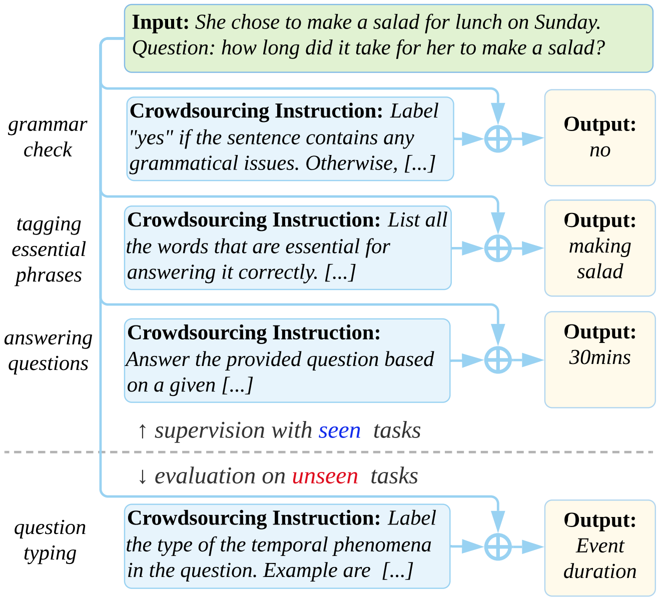

Input: She chose to make a salad for lunch on Sunday. Question: how long did it take for her to make a salad?
grammar check
Crowdsourcing Instruction: Label "yes" if the sentence contains any grammatical issues. Otherwise, [...]
Output: no
tagging essential phrases
Crowdsourcing Instruction: List all the words that are essential for answering it correctly. [...]
Output: making salad
answering questions
Crowdsourcing Instruction: Answer the provided question based on a given [...]
Output: 30mins
↑ supervision with seen tasks
↓ evaluation on unseen tasks
question typing
Crowdsourcing Instruction: Label the type of the temporal phenomena in the question. Example are [...]
Output: Event duration

Figure 1.1: We Construct the Natural Instructions Dataset from Crowdsourcing Instructions and Instances of Different Nlp Datasets. We Study If Models Can Learn from Seen Tasks and Generalize to Unseen Tasks given Their Natural Crowdsourcing Instructions.

In this thesis, we study if models can generalize to *unseen* tasks given their crowdsourcing instructions (Fig.10.1).

We build Natural Instructions, a dataset consisting of *natural* crowdsourcing instructions for various tasks and their instances. Training on *seen* tasks seen in our dataset, we build a model that learns to follow natural instructions that define a task and perform tasks (i.e., mapping input to output). Testing on *unseen* tasks unseen, we evaluate if the model can perform *unseen* tasks solely from their instructions and without any task-specific labeled data.

We compile Natural Instructions from task instructions written by researchers for crowd-

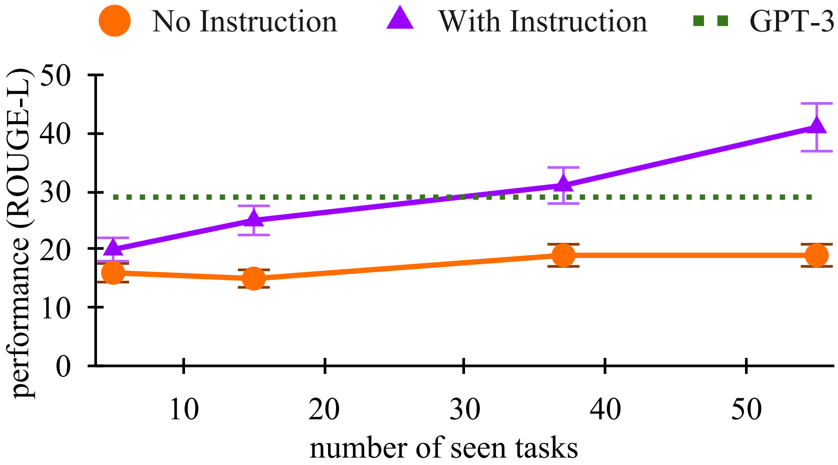


Figure 1.2: BART Evaluation on *Unseen* Tasks (y-axis Is Perf. On Unseen) When Supervised with *Seen* Tasks (x-axis Is |Seen|). A Model Using Instructions (I_t) Consistently Improves with More Observed Tasks. In Contrast, Models with No Access to the Instructions Show No Sign of Improved Generalization.

sourcing existing NLP datasets. Such crowdsourcing instructions often elaborate a variety of details about how a task should (and should not) be done. To provide a systematic study of various elements of crowdsourcing instructions, we map them to a unified *schema* to cover the most important elements of task descriptions — such as definition, constraints, positive and negative examples. We collect tasks in Natural Instructions as minimal stand-alone steps provided to crowdworkers to complete a downstream NLP task. For example, tasks collected from QASC (Khot *et al.*, 2020) include sub-tasks about generating topic words or combining facts, as well as answering multi-hop questions. Therefore our dataset not only contains typical downstream tasks in NLP, but also the intermediate subtasks that are not well-represented in the common benchmarks. The unified schema and the collection of minimal subtasks enable training LMs that can generalize across different tasks by learning from instructions. In total, our dataset consists of 61 distinct NLP tasks and $193k$ instances.

Our experimental results indicate that LMs learn to leverage natural language instructions as they show improved generalization to new tasks. For example, a BART (Lewis *et al.*, 2019) achieves a 19% gain in terms of cross-task generalization compared to a model not using instructions. Importantly, LMs can generalize better to unseen tasks if they observe more tasks in training (Figure 1). This upward trajectory suggests the potential for stronger cross-task generalizable models upon scaling up the diversity of tasks represented in a meta-dataset of task instructions. Despite the benefits of instructions, we observe a sizable gap between models' generalization and their estimated upperbounds, encouraging the community to work on this challenging problem.

In chapter 4, the focus is on applying the multitask learning framework (with and without instructions) to numerical reasoning tasks. Given the ubiquitous nature of numbers in text, reasoning with numbers to perform simple calculations is an important skill of AI systems. While many datasets and models have been developed to this end, state-of-the-art AI systems are brittle; failing to perform the underlying mathematical reasoning when they appear in a slightly different scenario. Drawing inspiration from GLUE that was proposed in the context of natural language understanding, we propose NumGLUE, a multi-task benchmark that evaluates the performance of AI systems on eight different tasks, that at their core require simple arithmetic understanding. We show that this benchmark is far from being solved with neural models including state-of-the-art large-scale language models performing significantly worse than humans (lower by 46.4%). Further, NumGLUE promotes sharing knowledge across tasks, especially those with limited training data as evidenced by the superior performance (average gain of 3.4% on each task) when a model is jointly trained on all the tasks as opposed to task-specific modeling. Finally, we hope that NumGLUE will encourage systems that perform robust and general arithmetic reasoning within language, a first step towards being able to perform more complex mathematical reasoning.

What kinds of instructional prompts are easier to follow for Language Models (LMs)? In chapter 5, we study this question by conducting extensive empirical analysis that sheds light on important features of successful instructional prompts. We propose several reframing techniques for model designers to manually create more effective prompts. Some examples include decomposing a complex task instruction into multiple simpler tasks or itemizing instructions into sequential steps. Our experiments compare the zero-shot and few-shot performance of LMs prompted with reframed instructions on 12 NLP tasks across 6 categories. Compared with original instructions, our reframed instructions lead to significant improvements across LMs with different sizes, underscoring the cross-model generality of these guidelines. For example, the same reframed prompts boost few-shot performance of GPT3-series and GPT2-series by 12.5% and 6.7% respectively averaged over all tasks. Furthermore, reframed instructions reduce the number of examples require to prompt LMs in the few-shot setting. We hope these empirically-driven techniques will pave the way for more effective ways to prompt LMs in the future.

In chapter 6, the focus is on applying the instruction learning framework (with and without instructions) to numerical reasoning tasks. Mathematical reasoning skills are essential for general-purpose intelligent systems to perform tasks from grocery shopping to climate modeling. Towards evaluating and improving AI systems in this domain, we propose Līla, a unified mathematical reasoning benchmark consisting of 23 diverse tasks along four dimensions: (i) mathematical abilities, e.g. arithmetic, calculus (ii) language format e.g. question-answering, fill-in-the-blanks (iii) language diversity e.g. no language, simple language (iv) external knowledge e.g. commonsense, physics. We construct our benchmark by extending 20 datasets benchmark by collecting task instructions and solutions in the form of Python programs, thereby obtaining explainable solutions in addition to the correct answer. We additionally introduce two evaluation datasets to measure out-of-distribution performance and robustness to language perturbation. Finally, we introduce Bhāskara, a general-purpose mathematical reasoning model trained on Līla. Importantly, we find that multi-tasking leads to significant improvements (average relative improvement of 21.83% F1 score single-task models), while the best performing model only obtains 60.40%, indicating the room for improvement in general mathematical reasoning and understanding.

In chapter 7, we contribute to the development of a prompting technique for program synthesis tasks. Despite the success of large pre-trained language models (LMs) such as Codex, they show below-par performance on the larger and more complicated programming related questions. We show that LMs benefit from the summarized version of complicated questions. Our findings show that superfluous information often present in problem description such as human characters, background stories, and names (which are included to help humans in understanding a task) does not help models in understanding a task. To this extent, we create a meta-dataset from the frequently used APPS dataset and the newly created CodeContests dataset for the program synthesis task. Our meta-dataset consists of human and synthesized summaries of the long and complicated programming questions. Experimental results on Codex show that our proposed approach outperforms baseline by 8.13% on the APPS dataset and 11.88% on the CodeContests dataset on average in terms of strict accuracy. Our analysis shows that summaries significantly improve performance for introductory (9.86%) and interview (11.48%) programming questions. However, it shows improvement by a small margin ($\sim 2\%$) for competitive programming questions, implying scope for future research in this direction.

In chapter 8, we focus on an alternate route to the development of Large Language Models. Large Language Models (LMs) have achieved state-of-the-art performance on many Natural Language Processing (NLP) benchmarks. With the growing number of new benchmarks, we build bigger and more complex LMs. However, building new LMs may not be an ideal option owing to the cost, time and environmental impact associated with it. We explore an alternative route: can we modify data by expressing it in terms of the model's strengths, so that a question becomes easier for models to answer? We investigate if humans can decompose a hard question into a set of simpler questions that are relatively easier for models to solve. We analyze a range of datasets involving various forms of reasoning and find that it is indeed possible to significantly improve model performance (24% for GPT3 and 29% for RoBERTa-SQuAD along with a symbolic calculator) via decomposition. Our approach provides a viable option to involve people in NLP research in a meaningful way. Our findings indicate that Human-in-the-loop Question Decomposition (HQD) can potentially provide an alternate path to building large LMs.

In chapter 9, our focus is to study instruction-example equivalence. Recently introduced *instruction-paradigm* empowers non-expert users to leverage NLP resources by defining a new task in natural language. Instruction-tuned models have significantly outperformed multitask learning models (without instruction); however they are far from state-of-the-art task specific models. Conventional approaches to improve model performance via creating datasets with large number of task instances or architectural changes in model may not be feasible for non-expert users. However, they can write alternate instructions to represent an instruction task. *Is Instruction-augmentation helpful?* We augment a subset of tasks in the expanded version of Natural Instructions with additional instructions and find that it significantly improves model performance (up to 35%), especially in the low-data regime. Our results indicate that an additional instruction can be equivalent to $\sim$200 data samples on average across tasks.

In chapter 10, we propose a technique to help humans think. Controlling the text generated by language models and customizing the content has been a long-standing challenge. Existing prompting techniques proposed in pursuit of providing control are task-specific and lack generality; this provides overwhelming choices for non-expert users to find a suitable method for their task. The effort associated with those techniques, such as in writing examples, explanations, instructions, etc. further limits their adoption among non-expert users. In this chapter, we propose a simple prompting strategy Help me Think where we encourage GPT3 to help non-expert users by asking a set of relevant questions and leveraging user answers to execute the task. We demonstrate the efficacy of our technique Help me Think on a variety of tasks. Specifically, we focus on tasks that are hard for average humans and require significant thinking to perform. We hope our work will encourage the development of unconventional ways to harness the power of large language models.

Chapter 2

BACKGROUND AND RELATED WORK

In this chapter, we discuss the background and various related works of the instruction learning paradigm.

Instructions in NLP applications. Prior work has studied "instructions" in various niches, such as robotic instructions (Shridhar *et al.*, 2020; Stepputtis *et al.*, 2020), databases (Kim *et al.*, 2020), programming (Lin *et al.*, 2018; Shao and Nakashole, 2020), *inter alia*. Such instructions are inherently different from ours, as they are intended to be mapped to pre-defined symbolic forms (e.g., SQL commands). Conversely, our instructions describe general NLP tasks (no underlying grammar) for measuring task-level generalization.

Learning from instructions. There is recent literature on the extent to which models follow language instructions (Hase and Bansal, 2021; Ye and Ren, 2021; Gupta *et al.*, 2021b; Zhong *et al.*, 2021). For example, Efrat and Levy (2020) examine if language models can follow crowdsourcing instructions with no further training. On the contrary, our work is pursuing a fundamentally different goal: creating a dataset of crowdsourcing instructions and task instances and formulating cross-task generalization by training models on seen tasks and measuring generalization to the remaining unseen ones. Weller *et al.* (2020) construct a crowdsourced dataset with short question-like task descriptions. Compared to this work, our instructions are longer, more complex and natural since they were used to collect datasets through crowdsourcing.

PromptSource and FLAN (Wei *et al.*, 2022a; Sanh *et al.*, 2022) are two concurrent works that pursue a similar goal as ours. A key difference between our work to these works is in terms of data collection strategy. Our work uses natural instructions created by

NLP researchers before the dataset instances were created by crowd workers, and hence it contains the complete definition of each task (definition, things to avoid, negative examples, etc.). On the other hand, instructions in the concurrent work are collected retroactively based on the already-available task instances. Our *natural* instructions enable evaluating models on how they learn tasks given different elements of task descriptions. (See §3.5.5 for further comparisons.) Nevertheless, we believe that all these approaches to constructing instructions and task categories are complementary and the community will benefit from considering both towards solving the challenging problem of cross-task generalization.

Prompt engineering. Constructing effective discrete prompts for language models to perform NLP tasks is an active area of research (Schick and Schütze, 2021; Reynolds and McDonell, 2021; Liu *et al.*, 2021b). Such prompts are often extremely short and may not include a complete definition of complex tasks. In contrast, our instructions encode detailed instructions as they were used to collect the datasets. Moreover, the goals are different: Most prompt-engineering approaches seek prompts with higher performance on a particular task, typically through assumptions about their target task which make them non-trivial to generalize to any other task. However, our introduced meta dataset enables the measurement of generalization to unseen tasks.

Beyond standard multi-task learning. Multi-task learning is a long-standing goal for AI (Caruana, 1997) and has led to successful models that can support a wider range of tasks (McCann *et al.*, 2018; Raffel *et al.*, 2020; Khashabi *et al.*, 2020; Aghajanyan *et al.*, 2021; Ye *et al.*, 2021). Most of the conventional setups in the multi-tasking literature evaluate on instances that belong to the tasks that are seen, i.e., their labeled instances were observed during training. We augment this setup by introducing natural language instructions which enable our models to bridge to tasks that were not seen during training.

Datasets for Numerical reasoning. Quantitative reasoning has been a challenging problem for a long time. Small question answering datasets were proposed to understand the quantitative aspect of natural language such as the template-based dataset which solved questions with equations as parameters (Kushman *et al.*, 2014), addition-subtraction dataset (Hosseini *et al.*, 2014) and arithmetic problems dataset (Koncel-Kedziorski *et al.*, 2015). Difficulty of questions were increased in subsequent datasets (Upadhyay *et al.*, 2016). Later, larger datasets were created to facilitate deep learning research (Ling *et al.*, 2017; Dua *et al.*, 2019b). Several other maths datasets have been proposed to improve explainability (Amini *et al.*, 2019), diversity (Miao *et al.*, 2020), scale information in language embeddings (Zhang *et al.*, 2020) and hardness of math questions (Hendrycks *et al.*, 2021b). One of the motivations behind creating our benchmark NumGLUE is to test for simple arithmetic reasoning independent of the context or the presentation style of the problem. Further, To the best of our knowledge, our work is the first to consider multiple tasks in the numerical reasoning space.

Multi-Task Benchmarks. With increased success of deep learning based models on individual tasks, there has been a significant push both in the NLP community and in the broader AI community towards general purpose models that excel at multiple tasks. Naturally, various benchmarks and challenges that test for such understanding have been proposed. For instance, the BAbI dataset (Weston *et al.*, 2015), GLUE (Wang *et al.*, 2019) and the subsequent harder SuperGLUE (Wang *et al.*, 2019) were proposed to both evaluate and drive progress in language understanding via shared linguistic knowledge across tasks. McCann *et al.* (2018) build a multi-task dataset via a novel approach – formatting each task as that of question-answering. In the more restricted setting of reading comprehension, Dua *et al.* (2019a) build a meta-dataset that spans multiple domains and reasoning skills.

Multi-task Models. With the growing interest towards models that go beyond specific datasets, various neural models that can perform mutliple tasks have been proposed. When the underlying reasoning is similar – eg. commonsense reasoning, problem decomposition or linguistic understanding – it has been found that training on multi-task datasets yields more robust and accurate models. For instance, the Multi-task Question Answering Network (McCann *et al.*, 2018), T5 (Raffel *et al.*, 2020), GPT3 (Brown *et al.*, 2020) and GPT3-Instruct models aim to build general purpose language models that are capable of transferring linguistic understanding across tasks. A similar approach is taken by Khashabi *et al.* (2020) in the setting of question-answering and Lourie *et al.* (2021) in the scope of commonsense reasoning. Further, Muppet (Aghajanyan *et al.*, 2021) adds an additional step of pre-finetuning between pretraining and finetuning that improves generalization to multiple tasks.

Discrete Prompts Constructing effective discrete prompts for language models to perform NLP tasks is an active area of research (Schick and Schütze, 2021; Tam *et al.*, 2021; Logan IV *et al.*, 2021; Reynolds and McDonell, 2021). Most such works focus on light-weight changes to the original prompt (Liu *et al.*, 2021b). Unlike the earlier literature, we focus on framings of complex instructions, which often lead to reframed prompts that are often very different from the original raw instructions. While our proposed prompt-reframing is not quite algorithmic, the principles behind them are relatively simple, which can hopefully motivate algorithmic solutions in future. Our goal in reframing is fundamentally different from the meta-training with instructions (Mishra *et al.*, 2022f; Sanh *et al.*, 2022; Wei *et al.*, 2022a). Such approaches depend on labeled data (language prompts for thousands of tasks) which can be costly to collect. Additionally, they require fine-tuning models which can be costly for larger LMs. Exploring effective framings of language instructions can provide alternative ways of utilizing LMs.

Continuous Prompts Tuning continuous prompts leads to the making of space-efficient models compared to fine-tuning model parameters (Liu *et al.*, 2021c; Lester *et al.*, 2021). Despite being algorithmic, these models require propagating gradient information across the whole architecture, leading to high computational costs, which is a key bottleneck when it comes to large LMs such as GPT3. While our proposal to reframe instructional prompts requires human intervention, it provides model designers with several relatively easy rules-of-thumb to come up with language prompts that work effectively with large LMs.

Role of Input Framing There have been repeated observation of models' sensitivity to the input representations. For example encoding numbers in text (Nogueira *et al.*, 2021)). More recently, in the prompting literature, we have seen the importance of selecting of informative examples and how they are ordered (Zhao *et al.*, 2021; Lu *et al.*, 2021d; Liu *et al.*, 2021a). Motivated by the role of input reframing, we develop reframing as a general method that provides various ways for model designers to reframe a diverse category of tasks.

Mathematical Reasoning Datasets. Our work in Līla builds on an existing body of mathematical reasoning literature. Early work in this areas focuses on small-scale datasets testing addition-subtraction (Hosseini *et al.*, 2014), templated questions with equations as parameters (Kushman *et al.*, 2014) and other forms of arithmetic reasoning (Koncel-Kedziorski *et al.*, 2015; Upadhyay *et al.*, 2016; Roy and Roth, 2017, 2018; Ling *et al.*, 2017). Later datasets increase in complexity and scale, incorporating reading comprehension (Dua *et al.*, 2019b), algebra (Saxton *et al.*, 2019), and multi-modal contexts (Lu *et al.*, 2021a, 2022b). Other numerical reasoning datasets focus on diversity (Miao *et al.*, 2020) with multiple categories of numerical reasoning tasks (e.g., Amini *et al.*, 2019). Most recently, new datasets have focused on increasing difficulty, e.g., olympiad problems (Hendrycks *et al.*, 2021b) and adversarial problems (Patel *et al.*, 2021), as well as increasing the knowledge requirements to solve tasks, with a growing focus on commonsense reasoning (Zhou *et al.*, 2019; Zhang *et al.*, 2020; Lu *et al.*, 2021b; Mishra *et al.*, 2022g). A separate line of work in mathematical reasoning includes datasets testing mathematical theorem proving (e.g., Li *et al.*, 2021; Wu *et al.*, 2021; Welleck *et al.*, 2021; Zheng *et al.*, 2021; Han *et al.*, 2021). We do not, however, consider theorem proving in our work, choosing instead to focus on numerical reasoning.

Task Hierarchy and Multi-tasking in Numerical Reasoning. We take inspiration from the success of multi-task learning in NLP (Weston *et al.*, 2015), including benchmarks (e.g., Wang *et al.*, 2018, 2019; Dua *et al.*, 2019a) and multitasking models (e.g., McCann *et al.*, 2018; Khashabi *et al.*, 2020; Lourie *et al.*, 2021; Aghajanyan *et al.*, 2021). NumGLUE (Mishra *et al.*, 2022g) has been proposed as a multi-tasking numerical reasoning benchmark that contains 8 different tasks. Līla expands NumGLUE to provide wider coverage of mathematical abilities, along with evaluation that captures out-of-domain, robustness, and instruction-following performance. Our introduction of mathematical reasoning categories and the evaluation setup in Līla is inspired by task hierarchies in other domains such as vision (Zamir *et al.*, 2018) and NLP (Rogers *et al.*, 2021) which appear in large scale benchmarks (e.g., Srivastava *et al.*, 2022; Wang *et al.*, 2022d).

Instructions for Program Synthesis In the past, there are several methods including semantic parsing (Ge and Mooney, 2005), deductive approaches, enumerative and stochastic search, and constraint solving which have gained attention for program synthesis (Gulwani *et al.*, 2017). With the advent of machine/deep learning, Balog *et al.* (2016) introduced a neural network based model for solving programming competition-style problems. Devlin *et al.* (2017) used sequence-to-sequence approach to do program synthesis. Furthermore, Hendrycks *et al.* (2021a) introduced the APPS dataset for testing the accuracy of large LMs on program synthesis. Hendrycks *et al.* (2021a) leveraged the *GPT-Neo* model (Black *et al.*, 2021) which they fine-tune for this task using APPS dataset. CodeT5 model (Wang *et al.*, 2021) utilizes many different training objectives. Recently, Austin *et al.* (2021) explore limitations of large language models and propose two new benchmarks, MBPP and MathQA-Python. The Codex model (Chen *et al.*, 2021a) is an advanced code generation model that powers GitHub's Copilot. The state of the art model for program synthesis was introduced by Deepmind called AlphaCode (Li *et al.*, 2022). They released their dataset CodeContests, which was used to fine-tune and test their model, and was used in our work (Kuznia *et al.*, 2022). Our approach suggesting smaller instructions compliments other approaches in improving model performance in instruction paradigm (Mishra *et al.*, 2022f; Wei *et al.*, 2022b; Parmar *et al.*, 2022b; Nye *et al.*, 2021; Puri *et al.*, 2022; Luo *et al.*, 2022; Wei *et al.*, 2022a; Sanh *et al.*, 2022)

Question Decomposition A recent methodology to reason over multiple sentences in reading comprehension datasets is to decompose the question into single-hop questions (Talmor and Berant, 2018; Min *et al.*, 2019). Min *et al.* (2019) decompose questions from HotpotQA using span predictions based on reasoning types and picks the best decomposition using a decomposition scorer. Khot *et al.* (2021) generate decompositions by training a BART model on question generation task by providing context, answers and hints. Wolfson *et al.* (2020) crowd-sourced annotations for decompositions of questions. Perez *et al.* (2020), on the other hand, uses the unsupervised mechanism of generating decomposition by mapping a hard question to a set of candidate sub-questions from a question corpus. Iyyer *et al.* (2017) answer a question sequentially using a neural semantic parsing framework over crowdsourced decompositions for questions from WikiTableQuestions. Decomposition using text-to-SQL query conversion has also been studied (Guo *et al.*, 2019). Also, knowledge graphs are combined with neural networks to generate decompositions (Gupta and Lewis, 2018). Recently, Xie *et al.* (2022) presented another use case where decompositions can be used to probe models to create explanations for their reasoning.

Chapter 3

CROSS-TASK GENERALIZATION BY LEARNING FROM INSTRUCTIONS

3.1 Introducing Cross-Task Generalization

Let $\mathcal{T}_{seen} = \{t_1, \ldots, t_n\}$ be a set of seen tasks, and $\mathcal{T}_{unseen} = \{t'_1, \ldots, t'_m\}$ be a set of unseen tasks. Let $\mathcal{T} = \mathcal{T}_{seen} \cup \mathcal{T}_{unseen}$. For each task $t \in \mathcal{T}$, let I_t denote the instruction (given in natural language) for that task, and $Train_t$ and $Test_t$ consist of the training and test set of that task. Both $Train_t$ and $Test_t$ consist of pairs (x, y), where x denotes an input to the task and y is the output of the task. Let $Train_{\mathcal{T}}$ denote $\bigcup_{t \in \mathcal{T}} Train_t$, and $Test_{\mathcal{T}}$ denote $\bigcup_{t \in \mathcal{T}} Test_t$.

Standard supervised learning algorithms learn task specific models. I.e., for a task t, they learn a model M_t which gets trained on $Train_t$ and gets evaluated on $Test_t$. Unified multi-task models learn a model $M_{\mathcal{T}}$ that gets trained on $Train_{\mathcal{T}}$ and gets evaluated on $Test_{\mathcal{T}}$. Both do not use the instruction corresponding to the tasks.

Our goal is in learning a model M that learns from $\mathcal{T}_{seen}$ and uses the instructions during learning and that can now be used on unseen tasks. I.e., it is evaluated on $\mathcal{T}_{unseen}$. During training (fine-tuning), for all tasks $t \in \mathcal{T}_{seen}$, if $(x, y) \in Train_t$ then I_t and x are given as input and y is the expected output. During our initial (zero-shot) evaluation, for all tasks $t' \in \mathcal{T}_{unseen}$, if $(x', y') \in Test_{t'}$ then $I_{t'}$ and x' are given as input to M and $M(I_{t'}, x')$ is compared with the gold standard output y'. This has been discussed further in our work (Mishra *et al.*, 2022f).

3.2 Natural Instructions

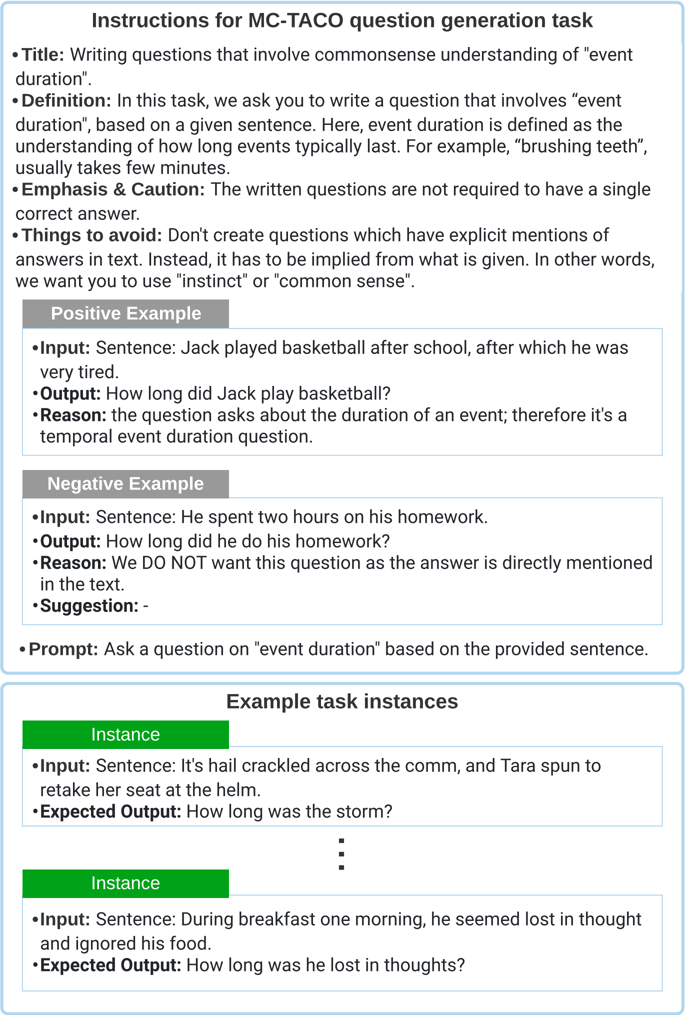
Natural Instructions consists of instructions that describe a task (e.g., question answering) and instances of that task (e.g., answers extracted for a given question). Fig.3.1 shows an example instruction for the task of ‘generating questions that require an understanding of event duration’ accompanied with positive and negative examples that contextualize the task. Here we introduce a schema for representing instructions (§3.2.1) and then describe how existing datasets (their crowdsourcing templates) are mapped into our schema (§3.2.2).

Since following natural language instructions is a challenge for state of the art models (Efrat and Levy, 2020), we limit source datasets in Natural Instructions to Question Answering datasets. Incorporation of NLI and other NLP tasks may make the task of following natural language instructions even harder. Hence Natural Instructions contains various tasks required to construct QA datasets.

3.2.1 Instruction Schema

We reuse existing crowdsourcing templates and crowdworker annotation on these templates as our main goal is to construct a dataset of *Natural* Instructions and crowdsourcing templates are mostly natural. Also, they are mostly curated by data creators and so quality is of less concern here in contrast to the quality concerns that exist in setting up a new crowdsourcing setup for data creation.

Instructions used in crowdsourcing various datasets, are written by distinct authors for different purposes, and they are different in a variety of ways. We introduce a unified schema (Fig.3.2) to consistently represent these diverse forms of instructions. Our instruction schema is the result of our pilot study conducted on a subset of datasets. Below we describe the



Instructions for MC-TACO question generation task

- **Title:** Writing questions that involve commonsense understanding of "event duration".
- **Definition:** In this task, we ask you to write a question that involves "event duration", based on a given sentence. Here, event duration is defined as the understanding of how long events typically last. For example, "brushing teeth", usually takes few minutes.
- **Emphasis & Caution:** The written questions are not required to have a single correct answer.
- **Things to avoid:** Don't create questions which have explicit mentions of answers in text. Instead, it has to be implied from what is given. In other words, we want you to use "instinct" or "common sense".

Positive Example

- **Input:** Sentence: Jack played basketball after school, after which he was very tired.
- **Output:** How long did Jack play basketball?
- **Reason:** the question asks about the duration of an event; therefore it's a temporal event duration question.

Negative Example

- **Input:** Sentence: He spent two hours on his homework.
- **Output:** How long did he do his homework?
- **Reason:** We DO NOT want this question as the answer is directly mentioned in the text.
- **Suggestion:** -

- **Prompt:** Ask a question on "event duration" based on the provided sentence.

Example task instances

Instance

- **Input:** Sentence: It's hail crackled across the comm, and Tara spun to retake her seat at the helm.
- **Expected Output:** How long was the storm?

⋮

Instance

- **Input:** Sentence: During breakfast one morning, he seemed lost in thought and ignored his food.
- **Expected Output:** How long was he lost in thoughts?

Figure 3.1: An Example from Our Dataset. Note That It Follows the Schema Provided In Fig.3.2. See Fig .3.11 for More Examples.

ingredients of this schema:

- TITLE provides a high-level description of a task and its associated skill (such as question generation, answer generation).
- PROMPT is a single sentence command that often appears before the input instance and connects it to the instructions.
- DEFINITION provides the core detailed instructions for a task.
- THINGS TO AVOID contain instructions regarding undesirable annotations that must be

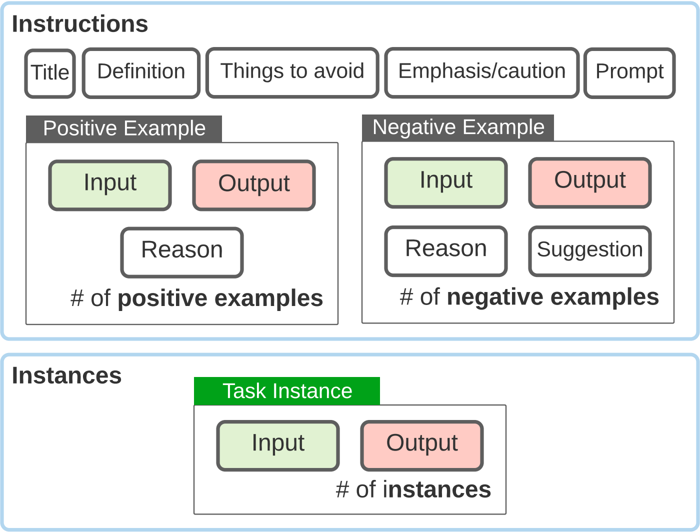


Figure 3.2: The Schema Used for Representing Instruction In Natural Instructions (§3.2.1), Shown in Plate Notation.

avoided. These help to define the scope of a task and the space of acceptable responses.

- EMPHASIS AND CAUTION are short, but important statements highlighted in the crowdsourcing templates which were intended to be emphasized or warned against.
- POSITIVE EXAMPLES contain inputs/outputs similar to the input given to a worker/system and its expected output, helping crowdworkers better understand a task (Ali, 1981).
- NEGATIVE EXAMPLES contain inputs/outputs to emphasize THINGS TO AVOID by providing examples that must not be produced.
- REASON provides explanations behind why an example is positive or negative.
- SUGGESTION contains suggestions on how a negative example could be modified to turn it into a positive example.

The next section describes the process of mapping the raw instructions (designed for crowdworkers) to our instruction schema.

Natural human-readable instructions. The collected instructions in natural language are designed for crowd workers and hence understandable to laypeople in a way that they can perform the intended task seamlessly after reading the given instructions.

Diversity. The collected instructions cover various intricacies of language for a wide variety of reasoning skills since they are naturally written by creators of datasets. Importantly, we avoid templating our instructions (Clark *et al.*, 2020).

Minimal descriptions. While the instructions written for humans often contain repetition (for emphasis), it is not clear whether such repetition is necessary for machines. Hence, we prefer the instructions that are concise while conveying their purpose.

Minimal tasks. Unlike most of the crowdsourcing templates that involve sequences of annotation steps, we would like each subtasks to be minimal, that is, Decomposition of those into subtasks should be done in such a way that each of the subtasks represents the simplest possible step.

Negative instructions. The conventional learning paradigms in NLP mostly rely on the inductive bias produced by *positive examples*. However, for humans, *negative instructions* (describing *undesirable behaviors*; (Lin *et al.*, 2003; Jindal and Roth, 2011)) are effective means to communicate a given task's scope. For humans, concise *negative examples* can be as informative as many positive examples. Our collected instructions include negative instructions originally provided to crowd workers to add constraints for data generation.

3.2.2 Constructing Natural Instructions

In this section, we describe various considerations in the process of mapping raw instructions (designed for crowdworkers) to our proposed schema (§3.2.1), while adhering to our desiderata. Here, we first describe our data source collected from existing datasets (§3.2.2)) and the process of mapping the raw instructions (designed for crowdworkers) to our instruction schema (§3.2.1).

Collecting Data

Collecting raw instructions and instances. We use existing, widely adopted NLP benchmarks that are collected via crowdsourcing platforms and hence, come with crowdsourcing templates. In the first step, we identified several datasets and engaged with their authors to get their crowdsourcing templates and raw data. This yields the following datasets: CosmosQA (Huang *et al.*, 2019), DROP (Dua *et al.*, 2019b), Essential-Terms (Khashabi *et al.*, 2017), MCTACO (Zhou *et al.*, 2019), MultiRC (Khashabi *et al.*, 2018), QASC (Khot *et al.*, 2020), Quoref (Dasigi *et al.*, 2019), ROPES (Lin *et al.*, 2019) and Winogrande (Sakaguchi *et al.*, 2020). [1]

Splitting crowdsourcing instructions into minimal tasks. Almost all the crowdworking instructions include sequences of steps to guide crowdworkers in creating task instances. For example, QASC and MCTACO include 7 and 19 steps in the data creation process, respectively. We divide crowdsourcing instructions into their underlying steps and generate multiple subtasks that are minimal and standalone. [2] Table 3.1 shows subtasks extracted for Quoref and QASC. For example, the main task in Quoref is to answer a question given a context paragraph, but the crowdsourcing template consists of two sub-tasks of *question generation* and *answer generation* with their separate instructions. This process results in a more consistent definition of tasks, enabling a successful mapping of instructions into our schema, in contrast to the work of Efrat and Levy (2020) that uses crowdsourcing

[1] We only focus on textual instructions and avoid datasets that involve visual or auditory steps, mostly focusing on QA datasets that were available to the authors.

[2] We eliminate tasks that involve model-in-the-loop.

source dataset	task
Quoref (Dasigi *et al.*, 2019)	question generation answer generation
QASC (Khot *et al.*, 2020)	topic word generation fact generation combining facts question generation answer generation incorrect answer generation

Table 3.1: Examples of the Datasets and the Tasks Formed From Them. The Extracted Tasks Are Independent Annotation Assignments in the Crowdsourcing Templates of the Datasets.

instructions as-is.

In total, there are 61 tasks, which are categorized into 6 semantic categories (Table 3.2). We assigned these broad categories to the tasks to understand their collective behavior in the experiments. It is noteworthy that, despite the apparent resemblance of the tasks included in the same category, any pair of tasks is distinct. For example, while *question generation* is part of Quoref, CosmosQA, and QASC, each has its own separate variant of the question generation task.

Mapping Raw Instructions to Schema

We manually fill in the fields of our instruction schema with the content from the crowdsourcing instructions. For instance, parts of the raw instructions that are highlighted for emphasis are incorporated as part of our *emphasis/caution* field. The modifications suggested in this step were applied by one author and were verified by another author (On average, the

category	# of tasks	# of instances
question generation	13	$38k$
answer generation	16	$53k$
classification	12	$36k$
incorrect answer generation	8	$18k$
minimal modification	10	$39k$
verification	2	$9k$
Total	61	$193k$

Table 3.2: Task Categories and Their Statistics.

process of data curation for each task takes around 5 hrs-34 hrs.

Improving description quality and consistency. We edit raw instructions to ensure their quality. Particularly, we fix writing issues (typos, ambiguities, etc.) and redact repetitions. While repetition often helps in augmenting human understanding, short and concise instructions are often more effective for computers due to their limited attention span (Beltagy *et al.*, 2020).

Augmenting examples and reasons. There is a large variance in the number of examples provided in the raw instructions. Instructions often include more positive examples, or some instructions do not include any negative examples (e.g., QASC). Whenever possible, we add negative examples such that each task has at least two negative examples. Furthermore, not all raw instructions contain REASONS or SUGGESTIONS for each of their examples. For example, positive examples are usually not accompanied by explanations, and most datasets do not include suggestions. We add them, wherever such information is missing in the instructions.

Collecting input/output instances for subtasks. Most of our tasks are the intermediate steps in the crowdsourcing process. Therefore, to extract input/output instances for each task, we need to parse the raw annotations of crowdworkers for every step. Since each dataset stores its annotations in a slightly different format, extracting and unifying such intermediate annotations can be non-trivial.

Verification. An annotator verified the quality of the resulting data in consultation with dataset authors. The annotator iterated on the authors' feedback (avg of 3 iters) until they were satisfied.

Quality assessment. We ask independent human annotators to answer 240 random instances (20 instances from 12 random tasks, used later for our evaluation §3.3.1). The subsequent evaluation of the human-generated responses results in more than 96% accuracy, which indicates that humans can effortlessly understand and execute our instructions.

Natural Instructions Statistics

In summary, Natural Instructions consists of subtasks each with a set of instructions and input/output instances (Fig.3.1 and 3.2). In total, the dataset includes 61 tasks and 193k instances. Table 3.2 shows data statistics for each task category. [3] On average, instructions contain 4.9 positive examples and 2.2 negative examples. The longest element of instructions is usually DEFINITIONS with 65.5 tokens and the shortest is TITLE with 8.3 tokens (more statistics in Table 3.3).

3.3 Problem Setup and Models

Here we define different cross-task generalization settings (§3.3.1) and the models (§3.3.2).

[3] We limit the number of instances in each task to $6.5k$ to avoid massive instance imbalance.

statistic	value
“title” length	8.3 tokens
“prompt” length	12.6 tokens
“definition” length	65.5 tokens
“things to avoid” length	24.1 tokens
“emphasis/caution” length	45.0 tokens
“reason” length	24.9 tokens
“suggestion” length	19.6 tokens
avg. length of “input in examples” (tokens)	50.2
avg. length of “output in examples’ (tokens)	6.6
num of positive examples	4.9
num of negative examples	2.2

Table 3.3: Statistics of Natural Instructions

3.3.1 Task Splits and Generalizations Types

To teach a model we supervise it with the set of tasks in train and evaluate them in eval.

Random split. This setup follows the common practice in benchmarking NLP models with random data splits. Here, two tasks from each task category (Table 3.2) in Natural Instructions are randomly selected for evaluation, and the rest of the tasks are used for training. This leads to 12 tasks in unseen and 49 tasks in seen. [4]

Leave-one-out generalization. To better understand the nature of cross-task generalization, we study more restrictive settings of dividing training and evaluation tasks.

[4] Those tasks that do not accept a relatively reliable automatic evaluation are excluded from unseen.

leave-one-category: evaluates how well a model generalizes to a task category if it is trained on others – no task of that category is in seen.

leave-one-dataset: evaluates how well a model can generalize to all tasks in a particular dataset if it is trained on all other tasks – no task of that dataset is in seen. This split prevents any leakage across tasks that belong to the same source datasets.

leave-one-task: evaluates how well a model can learn a single task by training on all other tasks.

3.3.2 Models

We build models using pre-trained LMs with encoder-decoder architectures BART (Lewis *et al.*, 2019) for fine-tuning and GPT3 (Brown *et al.*, 2020) for few-shot experiments.

Encoding instructions and instances. For every problem setup, we map a given instruction I_t and an input instance x into a textual format and decode an output y and obtain $enc(I_t, x)$. This encoding function is then fed to an encoder-decoder model to predict y: $M : enc(I_t, x) \to y$.

Encoding instances follows a standard NLP paradigm of mapping an input instance to text. Each instruction I_t consists of multiple elements as described in our instruction schema (§3.2.1). Here, we map each element of the instruction to a textual format and append it before the input instance. Fig.3.3 shows how we encode the full instruction.

According to our schema (§3.2.1), each instruction I_t for the t-th task is a set that contains the following fields:

$$I_t = ttitle, tdef., tavoid, temph., tprompt, tpos.ex., tneg.ex.$$

To feed the instances to LMs, we first encoder them into plain text. Let $enc(I, x)$ define

$\texttt{Prompt} : tprompt$

$\texttt{Definition} : tDefinition$

$\texttt{Things to Avoid} : tavoid.$

$\texttt{Emphasis\&Caution} : temph.$

$\texttt{NegativeExample1}-$

  $\texttt{input} : tpos.ex., \texttt{output} : tpos.ex., \texttt{reason} : tpos.ex.$

$\texttt{PositiveExample1}-$

  $\texttt{input} : tpos.ex., \texttt{output} : tpos.ex.\texttt{reason} : tpos.ex.$

$\texttt{input} : x, \texttt{output} :''$

Figure 3.3: Encoding Instruction I_t, Where tc Refers to the Text of a Component c in the Instruction Schema.

a function that maps a given instruction I and input instance x to plain text. Evidently, there are many choices for this function. In our study, we consider the following encodings:

NO-INSTRUCTIONS encoding. This encoding is the conventional paradigm where no instructions exist:

$$\begin{aligned} enc(I_t, x) :=& \texttt{input} : x \\ & \texttt{output} :'' \end{aligned} \tag{3.1}$$

PROMPT encoding. In this encoding, we append the prompt message before the input:

$$\begin{aligned} enc(I_t, x) :=& \texttt{Prompt} : tprompt \\ & \texttt{input} : x \\ & \texttt{output} :" \end{aligned} \tag{3.2}$$

PROMPT + DEFINITION encoding. In this encoding, the prompt message and the task definition appear before the input:

$$\begin{aligned} enc(I_t, x) :=& \text{``}\texttt{Definition} : tdef. \\ & \texttt{Prompt} : tprompt \\ & \texttt{input} : x \\ & \texttt{output} :" \end{aligned} \tag{3.3}$$

Intuitively, this encoding is more informative and more complex than “prompt” encoding.

Full Instructions encoding. This encoding contains all the instruction content:

$$
\begin{aligned}
enc(I_t, x) :=\text{``}&\texttt{Definition} : tdef. \\
&\texttt{Prompt} : tprompt \\
&\texttt{Things to Avoid} : tavoid. \\
&\texttt{Emphasis\&Caution} : temph. \\
&\text{``}\texttt{NegativeExample1}- \\
&\quad \texttt{input} : tpos.ex.(\texttt{input}) \\
&\quad \texttt{output} : tpos.ex.(\texttt{output}) \\
&\quad \texttt{reason} : tpos.ex.(\texttt{reason}) \\
&\texttt{NegativeExample2}- \\
&\quad \ldots \\
&\text{``}\texttt{PositiveExample1}- \\
&\quad \texttt{input} : tpos.ex.(\texttt{input}) \\
&\quad \texttt{output} : tpos.ex.(\texttt{output}) \\
&\quad \texttt{reason} : tpos.ex.(\texttt{reason}) \\
&\texttt{PositiveExample2}- \\
&\quad \ldots \\
&\texttt{input} : x \\
&\texttt{output} :\text{''}
\end{aligned}
\tag{3.4}
$$

where $enc_{\text{ex}}(I_t)$ is an alternating encoding positive and negative examples. We include as many examples as possible, before exceeding the input limit.

Positive Examples encoding. This encoding contains only positive examples of the subtask (no task description, etc).

model ↓	evaluation set unseen →	random split of tasks	leave-one-category (QG)	leave-one-dataset (QASC)	leave-one-task (QASC QG)
BART (fine-Tuned)	Multitask Learning	13	6	37	20
	Instruction Tuning	**32**	**17**	**51**	**56**
GPT3 (not fine-tuned)	Full instructions	24	33	22	33
BART (fine-Tuned)	Supervised Upperbound	67	59	60	66

Table 3.4: Cross-task Generalization of BART under Various Splits. Fine-tuned BART Shows Improved Performance When Provided with Instructions. It also Achieves Better Performance than GPT3, Despite Being over $1k$ times Smaller. All Numbers are ROUGE-L.

$$\begin{aligned} enc(I_t, x) := \quad & \texttt{input} : tpos.ex.(\texttt{input}) \\ & \texttt{output} : tpos.ex.(\texttt{output}) \\ & \dots \\ \texttt{input} & : x \\ \texttt{output} & :" \end{aligned} \tag{3.5}$$

Such example-only models have been used in several recent studies in the field (Zhao *et al.*, 2021).

To study the impact of each instruction element for cross-task generalization, we compare these encodings: (1) PROMPT, (2) POS. EXAMPLES, (3) PROMPT + DEFINITION, (4) PROMPT + THINGS TO AVOID, (5) PROMPT + EMPHASIS , (6) PROMPT + POS. EXAMPLES, (7) PROMPT + DEFINITION + POS. EXAMPLES, and (8) FULL INSTRUCTION. Each of these (e.g., PROMPT and POS. EXAMPLES) correspond to prompting setups in the recent literature (Le Scao and Rush, 2021; Lu *et al.*, 2021d).

BART. We use BART (base) (Lewis *et al.*, 2019) which allows us to fine-tune its model parameters. This is an encoder-decoder architecture with $140m$ parameters. For each setup,

model ↓	task category →	QG	AG	CF	IAG	MM	VF	avg
BART (fine-tuned)	NO INSTRUCTION	26	6	0	21	33	7	13
	PROMPT	27	22	7	22	34	**9**	20
	+DEFINITION	35	24	50	25	36	7	30↑ (+50)
	+THINGS TO AVOID	33	24	4	24	**58**	**9**	25↑ (+25)
	+EMPHASIS	38	23	16	**26**	49	3	26↑ (+30)
	+POS. EXAMPLES	53	22	14	25	17	7	23↑ (+15)
	+DEFINITION+POS. EXAMPLES	51	23	**56**	25	37	6	33↑ (+65)
	POS. EXAMP.	**55**	6	18	25	8	6	20
	FULL INSTRUCTION	46	**25**	52	25	35	7	32↑ (+60)
GPT3 (not fine-tuned)	FULL INSTRUCTION	33	18	8	12	60	11	24 (+11)

Table 3.5: Cross-task Generalization under Random Split. Models Show Improved Results When Provided with Instructions. The Numbers in Parenthesis Indicate Absolute Gains Compared to 'no Instructions' Baseline. Fine-tuned BART Achieves Better Performance than GPT3, Despite Being over $1k$ times Smaller. Category Names: QG: Question Generation, AG: Answer Generation, CF: Classification, IAG: Incorrect Answer Generation, MM: Minimal Text Modification, VF: Verification. All Numbers are ROUGE-L (in Percentage).

the input is encoded using different instruction elements, trained on all seen tasks, and evaluated on unseen (§3.3.1).

GPT3. As a comparison, we evaluate GPT3 (Brown *et al.*, 2020) which is a $175B$ parameter autoregressive LM ($\times 1.2k$ larger than BART) and has shown promising results in mimicking demonstrations provided in its prompt. We cannot fine-tune the parameters of this massive model and use it as-is under its default setting on the evaluation tasks in unseen (§3.3.1) using the encoding introduced earlier.

3.4 Experiments

Evaluation metrics. We treat all of our tasks as text generation problems and evaluate them with automated evaluation metrics for text generation. In particular, we use ROUGE-L (Lin, 2004) to automatically evaluate the generated outputs. [5]

Implementation details. For BART, our models are trained for 3 epochs with a learning rate of 5e-5 for a given training split and input encoding. For GPT3, we use the `davinci-instruct` engine and produce outputs with greedy decoding, generating up to a maximum number of tokens of 16 (the default value). We use the default stop condition which is 2 newline tokens. [6]

3.4.1 Generalization Under Various Task Splits

Table 3.4 reports the results of the BART model train and evaluated with various task splits (§3.3.1). For comparison, we evaluate GPT3 which uses no fine-tuning, unlike BART that is fine-tuned with the seen tasks.

The first column corresponds to random split of tasks, while the remaining columns report cross-task generalization results of the BART model under leave-one-x splits (§3.3.1). For $x = \underline{\text{category}}$, the tasks in *question-generation* category are held out during training. For $x = \underline{\text{dataset}}$, the tasks that were extracted from the *QASC* dataset were excluded from training. For $x = \underline{\text{task}}$, we train a model on all tasks, except *QASC question generation* task which is used for evaluation.

[5] Our experiments show that other metrics, e.g. BLEURT (Sellam *et al.*, 2020) are also correlated with ROUGE-L, which has also been used in generative QA tasks.

[6] The relevant code is available at: https://github.com/allenai/natural-instructions-v1

Instructions benefit cross-task generalization. The results indicate that BART benefits from instructions in generalizing to new tasks, regardless of task splits. For example, under random split, the model using FULL INSTRUCTIONS results in +19% gains over a model that is not using instructions. This is particularly interesting for leave-one-category-out split since the trained model can generalize to the tasks of a particular semantic category, without being exposed to it. In comparison to GPT3, the fine-tuned BART model that utilizes instructions achieves a stronger performance despite being $\times 1k$ smaller than GPT3. For example, a BART model using FULL INSTRUCTIONS achieves 8% higher performance than GPT3 under random split of tasks.

Note that the absolute values in leave-one-category are lower due to the difficulty of this setup compared to, for example, the random split setup. While all settings involve evaluating on tasks not seen during training, the leave-one-category setting enforces more dissimilarity among training and evaluation tasks.

3.4.2 Generalization Under Instruction Encoding and Task Categories

Table 3.5 reports the results of the BART model per encodings of different instruction elements (§3.3.2) and for different task categories. The table shows that encoding more elements of the instructions generally achieves better results than just using PROMPT or POSITIVE EXAMPLES. It additionally shows that the benefit of the instruction elements seems to depend on the target task category. We observe that the *question-generation* (QG) tasks benefit the most from POSITIVE EXAMPLES, whereas in *classification* (CF), POSITIVE EXAMPLES are of little help. We hypothesis this is because it is easier to mimic question-generation based on a few examples, whereas it is difficult to define classes via a few examples, where DEFINITION can be more helpful. The models show little improvement in *verification* (VF). We hypothesize these tasks are inherently more difficult, partially because of their distinctness from the rest of the tasks in the dataset. We hope future work on this line will study a wider variety of tasks and will improve our understanding of such failure cases.

3.4.3 Generalization vs. Number of Seen Tasks

Fig.1 compares the impact of the number of seen tasks for cross-task generalization. For supervision, we randomly sample a few tasks as seen and evaluate on 6 tasks (one from each category). (each point in the figure is averaged over 5 random subsamples.) The results show that with NO-INSTRUCTION encoding there is no tangible value in observing more tasks. In contrast, the generalization of the models that encode instructions improves with observing more tasks. This is an exciting observation since it suggests that scaling up our dataset to more tasks may lead to stronger instruction-following systems.

3.4.4 Analyses

Upperbound: Task-specific Models. For each task, we obtain a task-specific model (§ 3.1) by training BART separately on each task's annotated training data. We evaluate these task-specific models to obtain a loose estimate of *upperbounds* for each task.

On average, task-specific models score 66% which is considerably higher than our models' best generalization (32%; Table 3.4). This indicates that there is considerable room for improving generalization-based models that use instructions.

3.4.5 Error Analysis

Table 5.7 shows the breakdown of the most common error types for the QASC question generation task by analyzing 30 errors (more error analyses can be found in Table 3.7).

Model ↓	Split ↓	w/ neg. examples	w/o neg. examples
BART	random	32	**35**
	leave-one-x		
	↳ x = category (AG)	19	**21**
	↳ x = dataset (Quoref)	37	37
	↳ x = task (QASC QG)	56	**57**
GPT3	-	24	**44**

Table 3.6: Effect of Excluding Negative Examples from Full Instruction Encoding. Negative Instructions Are Surprisingly Difficult for the Models to Learn From.

error type	BART
Generates a nonsensical/vague question	47
Generate an invalid question	8
Generates a yes/no question	4
Copies the given fact or a subset of it	3
Generates unanswerable questions	3

Table 3.7: Percentage of Errors on QASC QG Task. The Numbers Do Not Sum to 100 since the Error Types Are Not Mutually Exclusive.

Category	Helpful Fields	Explanation
Question Generation (QG)	1. DEFINITION	- Provides a holistic picture of the task.
	2. EMPHASIS & CAUTION	- Provides key information for solving the task.
	3. POSITIVE EXAMPLES	- This gives an idea of what is expected in the output.
	4. NEGATIVE EXAMPLES	- Good to know the common mistakes people do.
Answer Generation (AG)	1. PROMPT	- It limits the exploration space to question spans.
	2. DEFINITION	- Provides a general understanding of the task.
	3. POSITIVE EXAMPLES	- Reason field is very helpful.
Classification (CF)	1. DEFINITION	- The task is unclear without this field.
Incorrect Answer Generation (IAG)	1. DEFINITION	- Helps understand the utility of such a task.
	2. EMPHASIS & CAUTION	- Source of some useful shortcuts.
	3. POSITIVE EXAMPLES	- Helps in understanding the type of questions asked.
Minimal Text Modification (MM)	1. THINGS TO AVOID	- Provides critical information.
Verification (VF)	1. DEFINITION	- Makes the task easy to understand.
	2. THINGS TO AVOID	- Contains useful tips required for this task.
	3. POSITIVE EXAMPLES	- Exemplifies task understanding.
	4. NEGATIVE EXAMPLES	- Helps avoid potential mistakes.

Table 3.8: Results of Humans' Perceived Importance of Instruction Elements. Our Annotators, for Example, Find Definition and Things to Avoid to Be Helpful for Classification and Minimal Text Modification Tasks, Respectively.

Impact of Negative Examples. Crowdsourcing instructions often include negative examples to exemplify undesirable responses. We study how negative examples in instructions affect cross-task generalization. Our case study (Table 3.6) indicates that the models work better *without* (w/o) negative examples, contrary to the previously-observed benefits of other instructional elements (e.g., definition, positive examples). This is aligned with the previous studies (Xuan *et al.*, 2020; Lin *et al.*, 2003) that discuss the challenges of learning from negative examples. Interestingly, GPT3's drop (44 vs 24) is more significant than BART (35 vs 32), showing that BART can partly recover through the training step.

Error Analysis. We randomly sample 30 erroneous predictions of our fine-tuned BART on 3 distinct tasks (Winogrande answer generation; QASC question generation; MC-TACO incorrect answer generation). We categorize the errors into common patterns (Table 3.7).

Unlike GPT3 which generally suffers from generating redundant content irrelevant to instructions, our BART model provides more control, however, it can fail to generate proper output. Here are several erroneous predictions from our model:

Does not follow instruction and generate an invalid question

GPT3 input:[...]input: Fact: Populations of viruses do not grow through division because they are parasites. output:

GPT3 output: Populations of viruses do not grow through division?

Generates a nonsensical/vague question

BART input:[...]input: Fact: graduations are used for measuring the volume of substances. output: ?

BART output: How many substances are there in volume?

Copies the given fact or a subset of it

BART input:[...] input: Fact: If someone is not breathing, begin with ventilation. output: ?

BART output: If someone is breathing, begin with ventilation.

Explains the question after generating it

GPT3 input:[...] input: Fact: Marmots in underground burrows will be protected from wild fire. output:

GPT3 output: What will be protected? reason: It's a good question because it is...

Generates a yes/no question

BART input:[...] input: Fact: Lack of saliva hinders the break down of food into components the body can absorb. output: ?

BART output: Does saliva hinders the break down of food into components the body can absorb?

Generates candidate answers as output

GPT3 input:[...] input: Fact: Marmots in underground burrows will be protected from wild fire. Output:"

GPT3 output: a) Burrows will protect marmots from a wild fire?

Generates questions whose answer does not exist

GPT3 input:[...]input: Fact: electrical energy is converted to heat energy to cook bread. output:

GPT3 output: How is electrical energy converted to heat energy to cook bread?

Generates a Generic Question Independent of Context

GPT3 input:[...] input: Fact: Populations of viruses do not grow through division because they are parasites. output:

GPT3 output: What does the given fact mean?

Perceived Impact of Instruction Elements. We survey human annotators to find out the value of instruction elements to humans. Except for the negative examples which were shown to be difficult for models, we observe similar trends between humans' perceived value of those elements (Table 3.8) and their contributions to the model performance (Table 3.5). For example, humans viewed DEFINITION and THINGS TO AVOID as necessary fields for *classification* and *minimal text modification* categories, respectively, which is compatible with our empirical observations (e.g., PROMPT + DEFINITION has the highest score on the CF category in Table 3.5).

3.5 Analysis of Datasets and their Templates

3.5.1 Division of Crowdsourcing Instructions into Minimal Tasks

Fig. 3.7 shows an example of how a task is divided into multiple subtasks for the MC-TACO dataset. MC-TACO has five categories (Event Duration, Event Frequency etc.). Each category contributes to 2 subtasks one for question generation and one for answer generation.

Number of tasks in each dataset. Fig. 3.4 illustrates how the number of steps in the data creation process varies across the 6 datasets. QASC and MC-TACO contain a relatively higher number of steps in the data creation process in comparison to DROP, Quoref, CosmosQA, and Winogrande.

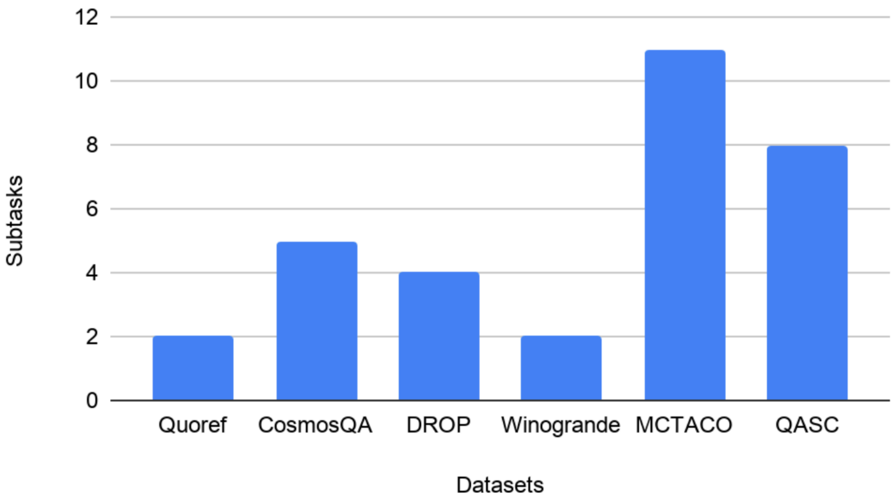


Figure 3.4: Variations in the Number of Subtasks

3.5.2 Analysis of Crowdsourcing Templates

We analyzed crowdsourcing templates of 6 datasets: CosmosQA Huang *et al.* (2019), DROP Dua *et al.* (2019b), MC-TACO Zhou *et al.* (2019), QASC Khot *et al.* (2020), Quoref Dasigi *et al.* (2019), and Winogrande Sakaguchi *et al.* (2020). Our intention behind the analysis is to identify similarities and differences across templates and subsequently decide regarding the collection of more templates.

Size of the instructions. We observe significant variation in size across the 6 datasets (Fig. 3.6). In the case of QASC, the instruction size associated with each step of the data creation process is very high, whereas for Winogrande, it is exactly the opposite– instruction size associated with each step of the data creation process is very low. Instead, the size of the common instruction (i.e., the instruction preceding the first step of the data creation process) is high in Winogrande; this is also seen for DROP. The major mode of instruction varies across datasets. Examples and instructions associated with each step of data creation respectively take up the majority of space in Quoref and CosmosQA. MC-TACO relies on examples to explain the crowdsourcing task, while Winogrande and QASC depend mostly on common instructions and instructions associated with each step of the data creation process respectively, to explain the task to the crowdworker.

The number of positive/negative examples. Variation in the occurrence of POSITIVE and NEGATIVE Examples across datasets has been illustrated in Fig. 3.5. Only Winogrande provides an equal number of POSITIVE and NEGATIVE Examples. QASC instructions do not contain any NEGATIVE Examples. Overall, DROP instructions consist of a relatively higher number of examples than other datasets.

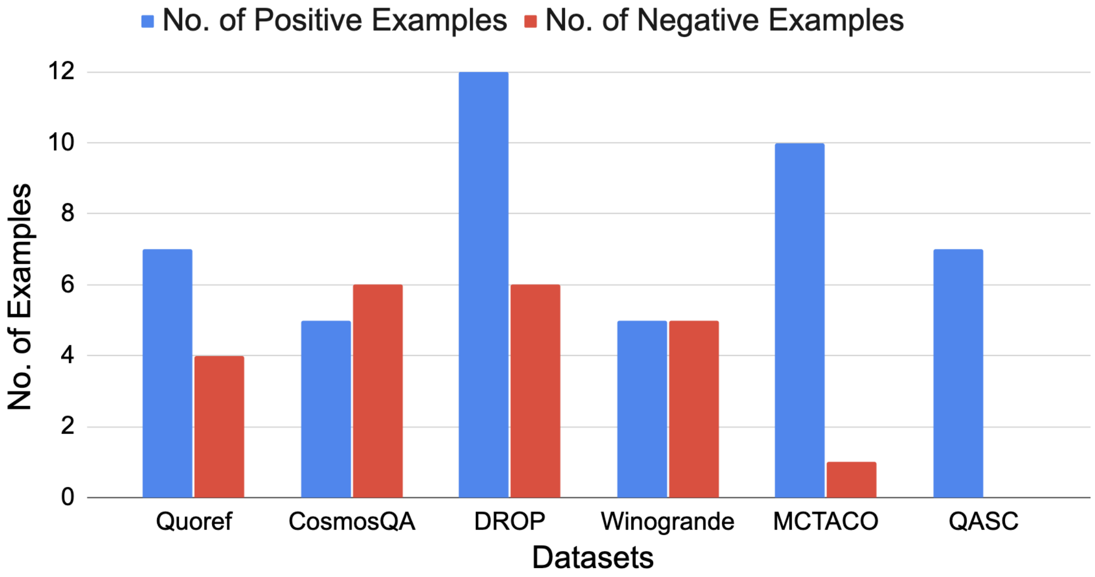


Figure 3.5: Variation in the Number of Positive and Negative Examples

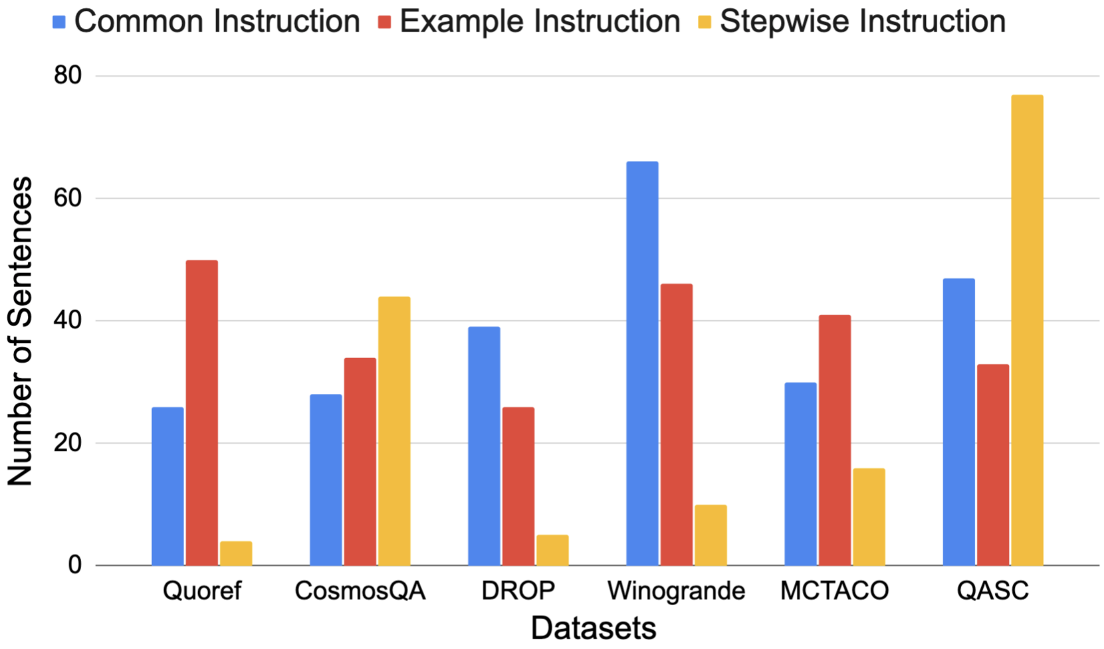
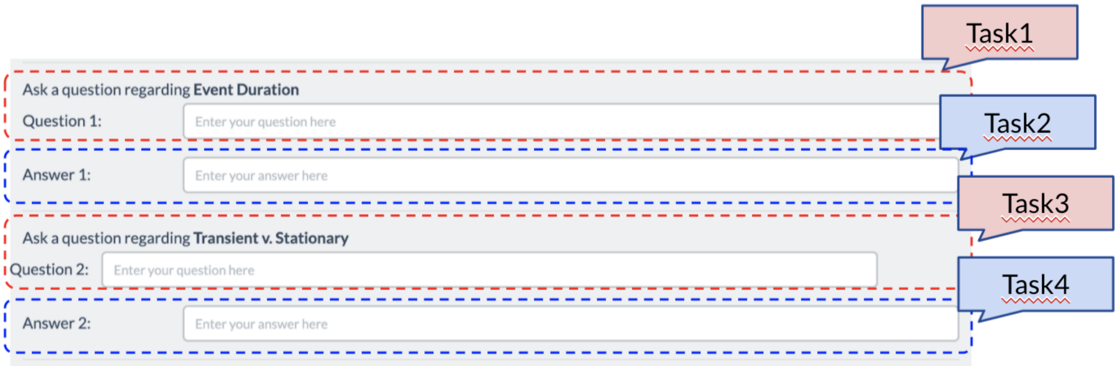


Figure 3.6: Variation in the Number of Sentences in the Crowdsourcing Instructions Across Datasets

Presence of reasons/suggestions in examples. All datasets except QASC contain both POSITIVE and NEGATIVE Examples. However, Quoref is the only dataset to provide REASONS for all the POSITIVE and NEGATIVE Examples. There are explanations associated with each of the NEGATIVE Examples, but the presence of explanations associated with POSITIVE Examples varies across datasets. Finally, Quoref is the only dataset to provide SUGGESTIONS along with the REASONS associated with the NEGATIVE Examples.

3.5.3 Qualitative Analysis

Writing Style. There is significant variation in writing style across the datasets, even among those datasets that have a common a objective (e.g., DROP, Quoref and QASC). DROP instructions say *"There is an AI running in the background which will also try to answer the question. You won't be able to submit the question if the AI gives the same*



Ask a question regarding **Event Duration**

Question 1: Enter your question here

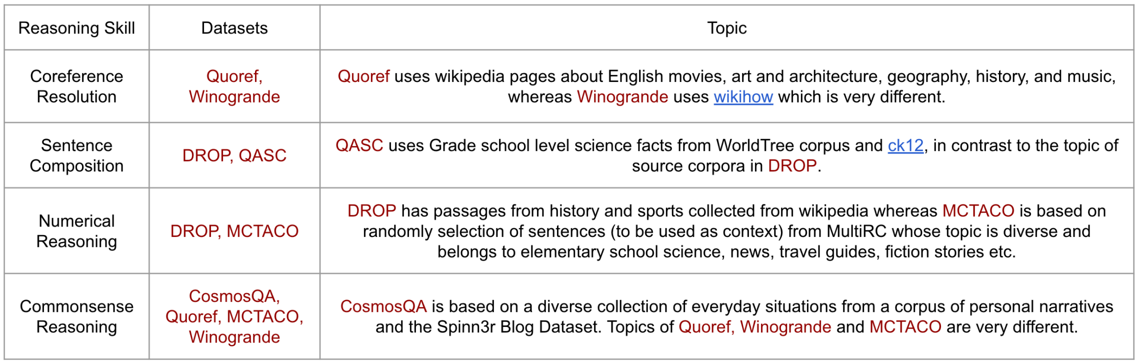
Answer 1: Enter your answer here

Ask a question regarding **Transient v. Stationary**

Question 2: Enter your question here

Answer 2: Enter your answer here

Figure 3.7: Dividing a Data Creation Task into Multiple Subtasks for the MC-TACO dataset.

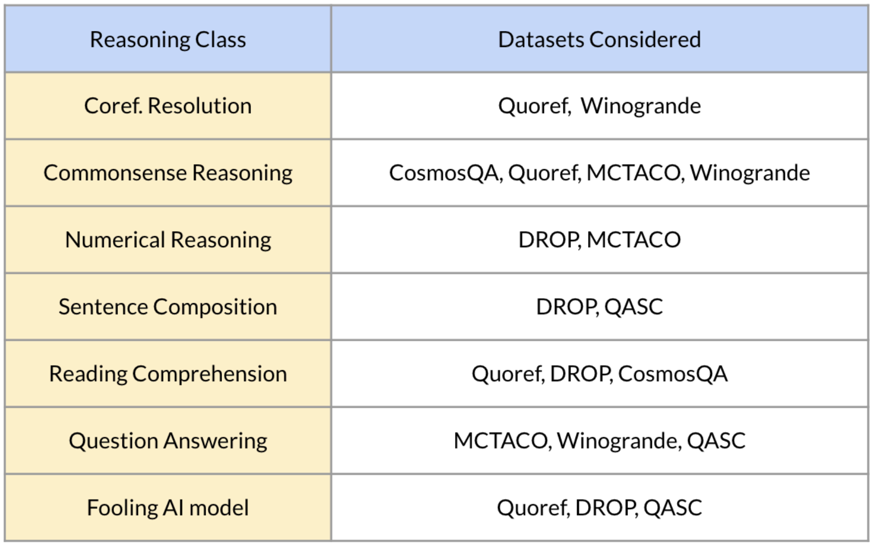


Reasoning Skill	Datasets	Topic
Coreference Resolution	Quoref, Winogrande	Quoref uses wikipedia pages about English movies, art and architecture, geography, history, and music, whereas Winogrande uses wikihow which is very different.
Sentence Composition	DROP, QASC	QASC uses Grade school level science facts from WorldTree corpus and ck12, in contrast to the topic of source corpora in DROP.
Numerical Reasoning	DROP, MCTACO	DROP has passages from history and sports collected from wikipedia whereas MCTACO is based on randomly selection of sentences (to be used as context) from MultiRC whose topic is diverse and belongs to elementary school science, news, travel guides, fiction stories etc.
Commonsense Reasoning	CosmosQA, Quoref, MCTACO, Winogrande	CosmosQA is based on a diverse collection of everyday situations from a corpus of personal narratives and the Spinn3r Blog Dataset. Topics of Quoref, Winogrande and MCTACO are very different.

Figure 3.8: Variation in Topics

response." The writing style in Quoref however is different: *"We also want you to avoid questions that can be answered correctly by someone without actually understanding the paragraph. ..."* In QASC, variations are as follows: *"Two AI systems will try to answer your question. Make sure you fool at least one AI with an incorrect answer. If you fool both AIs, you will receive a bonus of $0.25."*

Information. We observe that sometimes instructions of a dataset contain information that is relevant to several other datasets, which do not contain similar instruction information. For example, Quoref, DROP and CosmosQA are datasets that are all based on reading comprehension tasks. CosmosQA contains a step in the data creation process asking users to skip passages containing inappropriate or offensive content. This information is also



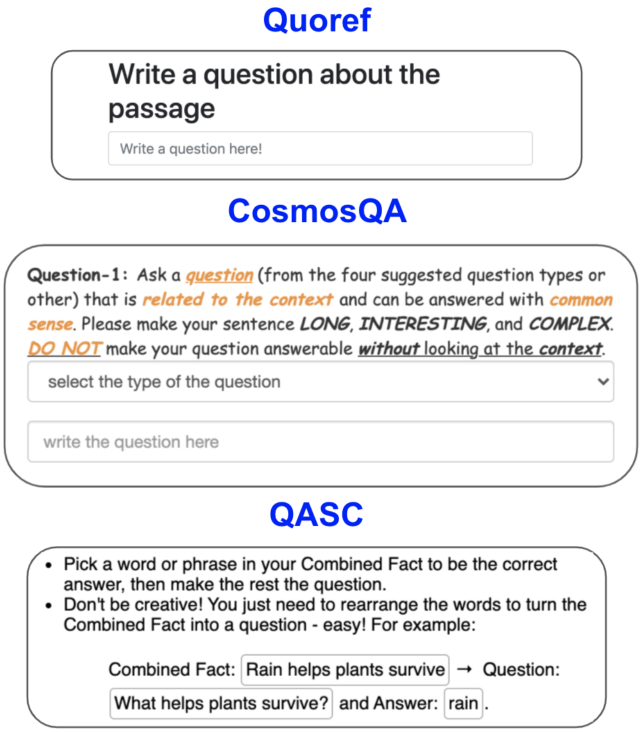
Reasoning Class	Datasets Considered
Coref. Resolution	Quoref, Winogrande
Commonsense Reasoning	CosmosQA, Quoref, MCTACO, Winogrande
Numerical Reasoning	DROP, MCTACO
Sentence Composition	DROP, QASC
Reading Comprehension	Quoref, DROP, CosmosQA
Question Answering	MCTACO, Winogrande, QASC
Fooling AI model	Quoref, DROP, QASC

Figure 3.9: Variation in Reasoning Skills

relevant to Quoref and DROP, but is not mentioned in their respective instructions.

Topic. Fig. 3.8 illustrates some examples where the reasoning skill associated with the datasets is the same, but the topic varies. The experience gained creating data for one topic may help with understanding instructions and creating data for another dataset with the same underlying reasoning skill.

Hardness. In a typical crowdsourcing task, certain tasks may be harder than the others, often these are the core tasks, e.g.: question generation, adversarial data creation, etc. Additional information, especially in the form of tips is always helpful in solving these hard tasks. Figure 3.10 illustrates that the task of question generation is stated differently in Quoref, CosmosQA, and QASC. QASC mentions an easy and detailed way to create questions, whereas CosmosQA mentions several different attributes of a good quality question. Knowing about the CosmosQA and QASC question generation processes may help with data creation for Quoref and other such question generation tasks, where less additional information is provided regarding question creation.



Quoref

Write a question about the passage

CosmosQA

Question-1: Ask a *question* (from the four suggested question types or other) that is *related to the context* and can be answered with *common sense*. Please make your sentence ***LONG***, ***INTERESTING***, and ***COMPLEX***. DO NOT make your question answerable ***without*** looking at the ***context***.

QASC

- Pick a word or phrase in your Combined Fact to be the correct answer, then make the rest the question.
- Don't be creative! You just need to rearrange the words to turn the Combined Fact into a question - easy! For example:

Combined Fact: Rain helps plants survive → Question: What helps plants survive? and Answer: rain.

Figure 3.10: Variation in Task Specification: Quoref Contains a Single Line Instruction Whereas the CosmosQA Contains a Detailed Instruction. QASC on the Other Hand, Contains Examples along with Instruction.

Associated reasoning skill. Finally, there are similarities among datasets in terms of their underlying skill requirements. Fig. 3.9 illustrates datasets clustered based on similarity in their associated reasoning class.

3.5.4 Data Curation Effort

Table 3.9 shows the effort distribution in the data curation process of Natural Instructions. Step-8 which involves parsing instances is the main bottleneck in the data curation process. Table 3.10-3.12 shows the detailed structure of tasks in Natural Instructions. Fig. 3.11 shows examples of four different tasks in Natural Instructions.

step	task	time per task
1	Identify crowdsourced dataset and engage with their authors.	20-30 mins
2	Go through the template and understand the task.	10-15 mins
3	Manually fill fields in the schema with content from the template.	30-45 mins
4	Iterate over the instructions to ensure their clarity while eliminating the repeated content. Fix writing issues in examples, also typos etc.	2-3 hrs
5	Create negative examples if not present. Add the missing explanations to the examples.	1-2 hrs
6	Extract the input/output instances from raw crowdsourcing annotations.	0.5-24 hrs
7	Final inspections of the data to verify the data quality	0.25- 2hrs
	Overall	6-34 hrs

Table 3.9: Steps Taken to Curate Each Task in Natural Instructions and Their Estimated Times.

3.5.5 Qualitative Comparison to PromptSource

We provide a comparison between our proposed dataset and PromptSource (Sanh *et al.*, 2022). PromptSource tasks are mainly focused on the common NLP downstream tasks (such as question-answering, coreference, NLI, etc). However, since we create tasks from various steps (including the intermediate steps) in a data creation process, our instructions contain a

task id	title	source dataset	task category
1	task001_quoref_question_generation	Quoref	Question Generation
2	task002_quoref_answer_generation	Quoref	Answer Generation
3	task003_mctaco_question_generation_event_duration	MC-TACO	Question Generation
4	task004_mctaco_answer_generation_event_duration	MC-TACO	Answer Generation
5	task005_mctaco_wrong_answer_generation_event_duration	MC-TACO	Incorrect Answer Generation
6	task006_mctaco_question_generation_transient_stationary	MC-TACO	Question Generation
7	task007_mctaco_answer_generation_transient_stationary	MC-TACO	Answer Generation
8	task008_mctaco_wrong_answer_generation_transient_stationary	MC-TACO	Incorrect Answer Generation
9	task009_mctaco_question_generation_event_ordering	MC-TACO	Question Generation
10	task010_mctaco_answer_generation_event_ordering	MC-TACO	Answer Generation
11	task011_mctaco_wrong_answer_generation_event_ordering	MC-TACO	Incorrect Answer Generation
12	task012_mctaco_question_generation_absolute_timepoint	MC-TACO	Question Generation
13	task013_mctaco_answer_generation_absolute_timepoint	MC-TACO	Answer Generation
14	task014_mctaco_wrong_answer_generation_absolute_timepoint	MC-TACO	Incorrect Answer Generation
15	task015_mctaco_question_generation_frequency	MC-TACO	Question Generation
16	task016_mctaco_answer_generation_frequency	MC-TACO	Answer Generation
17	task017_mctaco_wrong_answer_generation_frequency	MC-TACO	Incorrect Answer Generation
18	task018_mctaco_temporal_reasoning_presence	MC-TACO	Classification
19	task019_mctaco_temporal_reasoning_category	MC-TACO	Classification
20	task020_mctaco_span_based_question	MC-TACO	Classification
21	task021_mctaco_grammatical_logical	MC-TACO	Classification

Table 3.10: Detailed Set of Tasks Included in Natural Instructions

task id	title	source dataset	task category
22	task022_cosmosqa_passage_inappropriate_binary	Cosmosqa	Classification
23	task023_cosmosqa_question_generation	Cosmosqa	Question Generation
24	task024_cosmosqa_answer_generation	Cosmosqa	Answer Generation
25	task025_cosmosqa_incorrect_answer_generation	Cosmosqa	Incorrect Answer Generation
26	task026_drop_question_generation	DROP	Question Generation
27	task027_drop_answer_type_generation	DROP	Classification
28	task028_drop_answer_generation	DROP	Answer Generation
29	task029_winogrande_full_object	Winogrande	Minimal Text Modification
30	task030_winogrande_full_person	Winogrande	Minimal Text Modification
31	task031_winogrande_question_generation_object	Winogrande	Question Generation
32	task032_winogrande_question_generation_person	Winogrande	Question Generation
33	task033_winogrande_answer_generation	Winogrande	Answer Generation
34	task034_winogrande_question_modification_object	Winogrande	Minimal Text Modification
35	task035_winogrande_question_modification_person	Winogrande	Minimal Text Modification
36	task036_qasc_topic_word_to_generate_related_fact	QASC	Minimal Text Modification
37	task037_qasc_generate_related_fact	QASC	Minimal Text Modification
38	task038_qasc_combined_fact	QASC	Minimal Text Modification
39	task039_qasc_find_overlapping_words	QASC	Verification
40	task040_qasc_question_generation	QASC	Question Generation
41	task041_qasc_answer_generation	QASC	Answer Generation
42	task042_qasc_incorrect_option_generation	QASC	Incorrect Answer Generation

Table 3.11: Detailed Set of Tasks Included in Natural Instructions

task id	title	source dataset	task category
43	task043_essential_terms_answering_incomplete_questions	Essential Terms	Answer Generation
44	task044_essential_terms_identifying_essential_words	Essential Terms	Verification
45	task045_miscellaneous_sentence_paraphrasing	Miscellaneous	Minimal Text Modification
46	task046_miscellaenous_question_typing	Miscellaenous	Classification
47	task047_miscellaenous_answering_science_questions	Miscellaenous	Answer Generation
48	task048_multirc_question_generation	MultiRC	Question Generation
49	task049_multirc_questions_needed_to_answer	MultiRC	Classification
50	task050_multirc_answerability	MultiRC	Classification
51	task051_multirc_correct_answer_single_sentence	MultiRC	Answer Generation
52	task052_multirc_identify_bad_question	MultiRC	Classification
53	task053_multirc_correct_bad_question	MultiRC	Minimal Text Modification
54	task054_multirc_write_correct_answer	MultiRC	Answer Generation
55	task055_multirc_write_incorrect_answer	MultiRC	Incorrect Answer Generation
56	task056_multirc_classify_correct_answer	MultiRC	Classification
57	task057_multirc_classify_incorrect_answer	MultiRC	Classification
58	task058_multirc_question_answering	MultiRC	Answer Generation
59	task059_ropes_story_generation	ROPES	Minimal Text Modification
60	task060_ropes_question_generation	ROPES	Question Generation
61	task061_ropes_answer_generation	ROPES	Answer Generation

Table 3.12: Detailed Set of Tasks Included in Natural Instructions

broader variety of tasks. For example, tasks for chaining facts (task 38; Table 3.11), question typing (task 27; Table 3.11) or detecting inappropriate content (task 22; Table 3.11) are unique additions in Natural Instructions. Additionally, since our instructions were originally written by various researchers and targeted at crowdworkers, they are elaborate and contain the complete definition of each task. This is somewhat evident from observation that GPT3 leads to higher performance on our instructions (Table 3.13). Last but not least, since we represent the instructions in a structured format, we are able to ablate various elements of the instructions (definition, negative/positive examples, etc.) and empirically quantify their contributions (§3.4).

Task	Model	PromptSource	Natural Instructions
Quoref QA (002)	GPT3-Instruct	43	**47**
	GPT3	2	**13**
DROP QA (028)	GPT3-Instruct	6	**10**
	GPT3	2	**3**

Table 3.13: Comparing Zero-shot Performance of Gpt3 on Our Instructions Vs. Promptsource. The Instructions Curated in This Work, Despite Being Lengthier, Lead to Higher Performance.

3.6 Additional Analysis of Results

3.6.1 Comparison to Raw Instructions

We seek to understand the value of breaking the tasks into sub-tasks and mapping them into our proposed schema (§3.2.2). We compute performance of raw instructions (first sub-task of four datasets), in the same vein as Efrat and Levy (2020)'s setup. We compare this to our FULL INSTRUCTION - NEG EXAMPLES encoding. We observe that GPT3 leads to notably higher performance with our encoding compared to raw instructions. Weak performance of LMs on raw instructions aligns with (Efrat and Levy, 2020)'s finding that "language model performs poorly".

This might be partly due to the verbose language of the raw instructions: the average length of the raw instructions is $2.5k$ tokens, in comparison to 950 tokens for our encoding. While repetition often helps human understanding, concise instructions seem to be more effective for computers.

task	Natural Instructions	PromptSource (Sanh et al. 2021)

MC-TACO (question answering)	* Definition: In this task we ask you to write answer to a question that involves “absolute timepoint" of events, which is defined as understanding of when events usually happen. For example, "going to school" usually happens during the day (not at 2 A.M).
	* Emphasis: Note that a lot of the questions could have more than one correct answers. We only need a single most-likely answer. Please try to keep your "answer" as simple as possible. Concise and simple "answer" is preferred over those complex and verbose ones.
	* Prompt: Answer the given question on "absolute timepoint" of events.
	Sentence: {{ sentence }}
	Question: {{ question }}
		Given the context,
		{{sentence}}
		observe the following QA pair and check if the answer is plausible:
		Question: {{question}}
		Answer: {{answer}}

Quoref (question answering)	* Definition: In this task, you're expected to write answers to questions involving multiple refences to the same entity.
	Emphasis: The answer to the question should be unambiguous and a phrase in the paragraph. Most questions can have only one correct answer.
	* Prompt: Answer the given question. Your answer must be a single span in the passage.
	Passage: {{ passage }}
	Question: {{ question }}
		Given the following context:
		{{context}}
		answer the following question:
		{{question}}

CosmosQA (question answering)	* Definition: Craft one correct answer to the question given in input. To make it more interesting, try to use non-stereotypical language if possible. Make sure your correct answer is reasonably long, consistent with the context, and requires common sense (instead of explicit extraction from the context.)
	* Emphasis: 1. In your answer, use as few words as possible from the given context. 2. Use a response that is uncommon/non-stereotypical, so that it is less predictable. 3. To be less repetitive, please vary your language for each question.
	* Prompt: Craft one correct answer to the question given in input.
	Context: {{ context }}
	Question: {{ question }}
		{{ context }}
		According to the above context, choose the best option to answer the following question.
		Question: {{ question }}
		Options: {{answer_choices}}

DROP (question answering)	* Definition: This task involves creating answers to complex questions, from a given passage. Answering these questions, typically involve understanding multiple sentences. Make sure that your answer has the same type as the "answer type" mentioned in input. The provided "answer type" can be of any of the following types: "span", "date", "number". A "span" answer is a continuous phrase taken directly from the passage or question. You can directly copy-paste the text from the passage or the question for span type answers. If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words. A "number" type answer can include a digit specifying an actual value. For "date" type answers, use DD MM YYYY format e.g. 11 Jan 1992. If full date is not available in the passage you can write partial date such as 1992 or Jan 1992.
	* Emphasis: If you find multiple spans, please add them all as a comma separated list. Please restrict each span to five words.
	* Prompt: Write an answer to the given question, such that the answer matches the "anwer type" in the input.
	Passage: {{ passage }}
	Question: {{ question }}
		Context: {{passage}}
		I am trying to figure out the answer to the question from the above context. Can you tell me the answer?
		Question: {{question}}
		Answer:

Table 3.14: Qualitative Comparison of the Task Instructions for Several Shared Tasks among Natural Instructions and Promptsource.

3.6.2 T0pp baseline for Natural Instructions

We evaluate T0pp (Sanh *et al.*, 2022) on our evaluation tasks of the random split (§3.3.1). Tasks on which T0pp was trained on were excluded from evaluation. Table 3.15 illustrates summary of our results. T0pp shows improved results when provided with detailed instructions compared to the PROMPT encoding. Most performance gain comes in the Question Generation category whereas the only performance drop comes from the Minimal Text Modification category. We attribute this behavior to the distribution of training tasks in T0pp; if a task has adequate number of similar tasks seen during training, we probably do not need detailed instructions and prompt only is enough; whereas for other tasks, detailed instructions in the form of definition etc. helps.

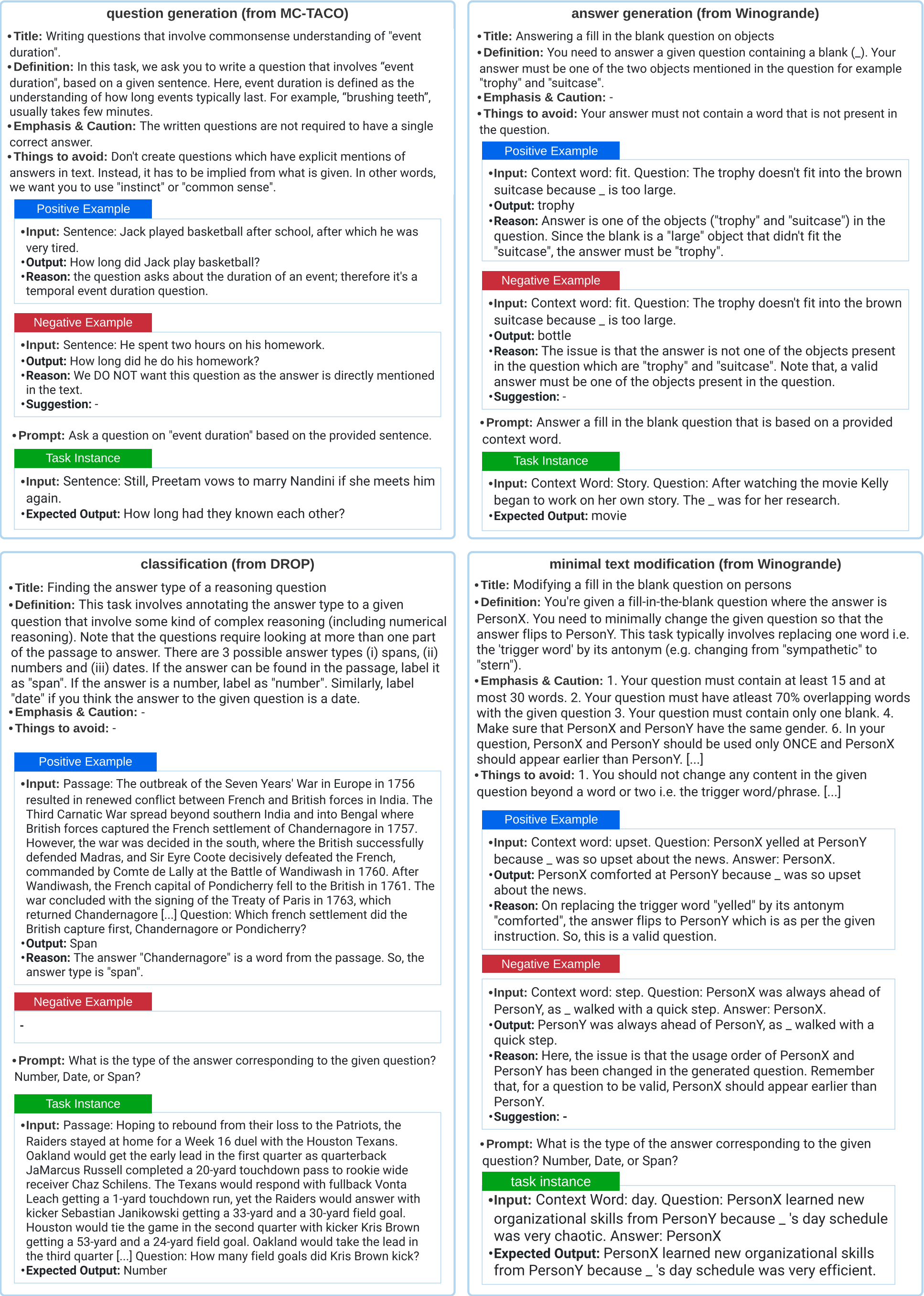
model ↓	task category →	QG	AG	CF	IAG	MM	VF	avg
T0pp	PROMPT	17	37	-	21	**69**	2	29
	FULL INSTRUCTION	**22**	**39**	-	**22**	64	2	**30**

Table 3.15: T0pp Performance with Prompt and Detailed Instructions for Different Task Categories under Random Split. Tasks on Which T0pp Were Trained on Were Excluded from Evaluation, since They Are No More Unseen Tasks. Models Show Improved Results When Provided with Detailed Instructions Compared to the PROMPT Encoding.) Category Names: QG: Question Generation, AG: Answer Generation, CF: Classification, IAG: Incorrect Answer Generation, MM: Minimal Text Modification, VF: Verification. All Numbers Are ROUGE-L.

3.7 Conclusion

We studied the goal of building models that generalize to new tasks by encoding and understanding crowdsourcing instructions. We introduced Natural Instructions [7] , which is built based on existing crowdsourced datasets, that enables building such models and systematically evaluating them. To the best of our knowledge, this is the first work to show the benefit of instructions towards improved cross-task generalization. Additionally, we observe that our proposed task has a large room for improvement, which we believe will bring more attention to building stronger models that can generalize to a wider range of tasks.

[7] https://instructions.apps.allenai.org/



question generation (from MC-TACO)

- **Title:** Writing questions that involve commonsense understanding of "event duration".
- **Definition:** In this task, we ask you to write a question that involves "event duration", based on a given sentence. Here, event duration is defined as the understanding of how long events typically last. For example, "brushing teeth", usually takes few minutes.
- **Emphasis & Caution:** The written questions are not required to have a single correct answer.
- **Things to avoid:** Don't create questions which have explicit mentions of answers in text. Instead, it has to be implied from what is given. In other words, we want you to use "instinct" or "common sense".

Positive Example

- **Input:** Sentence: Jack played basketball after school, after which he was very tired.
- **Output:** How long did Jack play basketball?
- **Reason:** the question asks about the duration of an event; therefore it's a temporal event duration question.

Negative Example

- **Input:** Sentence: He spent two hours on his homework.
- **Output:** How long did he do his homework?
- **Reason:** We DO NOT want this question as the answer is directly mentioned in the text.
- **Suggestion:** -

- **Prompt:** Ask a question on "event duration" based on the provided sentence.

Task Instance

- **Input:** Sentence: Still, Preetam vows to marry Nandini if she meets him again.
- **Expected Output:** How long had they known each other?

answer generation (from Winogrande)

- **Title:** Answering a fill in the blank question on objects
- **Definition:** You need to answer a given question containing a blank (_). Your answer must be one of the two objects mentioned in the question for example "trophy" and "suitcase".
- **Emphasis & Caution:** -
- **Things to avoid:** Your answer must not contain a word that is not present in the question.

Positive Example

- **Input:** Context word: fit. Question: The trophy doesn't fit into the brown suitcase because _ is too large.
- **Output:** trophy
- **Reason:** Answer is one of the objects ("trophy" and "suitcase") in the question. Since the blank is a "large" object that didn't fit the "suitcase", the answer must be "trophy".

Negative Example

- **Input:** Context word: fit. Question: The trophy doesn't fit into the brown suitcase because _ is too large.
- **Output:** bottle
- **Reason:** The issue is that the answer is not one of the objects present in the question which are "trophy" and "suitcase". Note that, a valid answer must be one of the objects present in the question.
- **Suggestion:** -

- **Prompt:** Answer a fill in the blank question that is based on a provided context word.

Task Instance

- **Input:** Context Word: Story. Question: After watching the movie Kelly began to work on her own story. The _ was for her research.
- **Expected Output:** movie

classification (from DROP)

- **Title:** Finding the answer type of a reasoning question
- **Definition:** This task involves annotating the answer type to a given question that involve some kind of complex reasoning (including numerical reasoning). Note that the questions require looking at more than one part of the passage to answer. There are 3 possible answer types (i) spans, (ii) numbers and (iii) dates. If the answer can be found in the passage, label it as "span". If the answer is a number, label as "number". Similarly, label "date" if you think the answer to the given question is a date.
- **Emphasis & Caution:** -
- **Things to avoid:** -

Positive Example

- **Input:** Passage: The outbreak of the Seven Years' War in Europe in 1756 resulted in renewed conflict between French and British forces in India. The Third Carnatic War spread beyond southern India and into Bengal where British forces captured the French settlement of Chandernagore in 1757. However, the war was decided in the south, where the British successfully defended Madras, and Sir Eyre Coote decisively defeated the French, commanded by Comte de Lally at the Battle of Wandiwash in 1760. After Wandiwash, the French capital of Pondicherry fell to the British in 1761. The war concluded with the signing of the Treaty of Paris in 1763, which returned Chandernagore [...] Question: Which French settlement did the British capture first, Chandernagore or Pondicherry?
- **Output:** Span
- **Reason:** The answer "Chandernagore" is a word from the passage. So, the answer type is "span".

Negative Example

-

- **Prompt:** What is the type of the answer corresponding to the given question? Number, Date, or Span?

Task Instance

- **Input:** Passage: Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week 16 duel with the Houston Texans. Oakland would get the early lead in the first quarter as quarterback JaMarcus Russell completed a 20-yard touchdown pass to rookie wide receiver Chaz Schilens. The Texans would respond with fullback Vonta Leach getting a 1-yard touchdown run, yet the Raiders would answer with kicker Sebastian Janikowski getting a 33-yard and a 30-yard field goal. Houston would tie the game in the second quarter with kicker Kris Brown getting a 53-yard and a 24-yard field goal. Oakland would take the lead in the third quarter [...] Question: How many field goals did Kris Brown kick?
- **Expected Output:** Number

minimal text modification (from Winogrande)

- **Title:** Modifying a fill in the blank question on persons
- **Definition:** You're given a fill-in-the-blank question where the answer is PersonX. You need to minimally change the given question so that the answer flips to PersonY. This task typically involves replacing one word i.e. the 'trigger word' by its antonym (e.g. changing from "sympathetic" to "stern").
- **Emphasis & Caution:** 1. Your question must contain at least 15 and at most 30 words. 2. Your question must have atleast 70% overlapping words with the given question 3. Your question must contain only one blank. 4. Make sure that PersonX and PersonY have the same gender. 6. In your question, PersonX and PersonY should be used only ONCE and PersonX should appear earlier than PersonY. [...]
- **Things to avoid:** 1. You should not change any content in the given question beyond a word or two i.e. the trigger word/phrase. [...]

Positive Example

- **Input:** Context word: upset. Question: PersonX yelled at PersonY because _ was so upset about the news. Answer: PersonX.
- **Output:** PersonX comforted at PersonY because _ was so upset about the news.
- **Reason:** On replacing the trigger word "yelled" by its antonym "comforted", the answer flips to PersonY which is as per the given instruction. So, this is a valid question.

Negative Example

- **Input:** Context word: step. Question: PersonX was always ahead of PersonY, as _ walked with a quick step. Answer: PersonX.
- **Output:** PersonY was always ahead of PersonY, as _ walked with a quick step.
- **Reason:** Here, the issue is that the usage order of PersonX and PersonY has been changed in the generated question. Remember that, for a question to be valid, PersonX should appear earlier than PersonY.
- **Suggestion:** -

- **Prompt:** What is the type of the answer corresponding to the given question? Number, Date, or Span?

task instance

- **Input:** Context Word: day. Question: PersonX learned new organizational skills from PersonY because _ 's day schedule was very chaotic. Answer: PersonX
- **Expected Output:** PersonX learned new organizational skills from PersonY because _ 's day schedule was very efficient.

Figure 3.11: Examples from Natural Instructions. Each Task Follows the Schema Provided In Fig. 3.2.

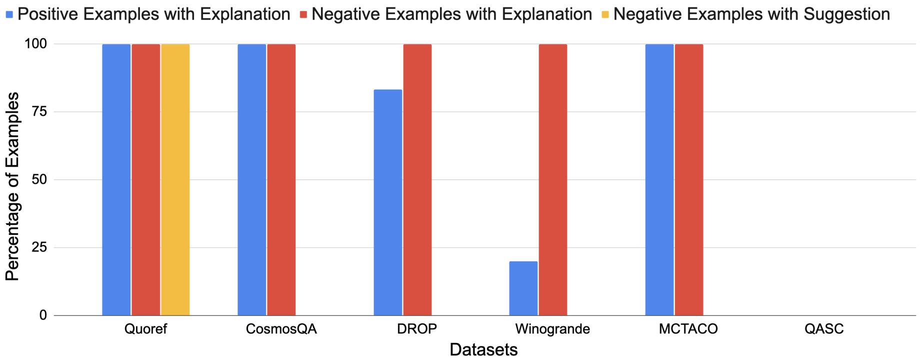


Figure 3.12: Variation in Reasons and Suggestions Associated with Examples

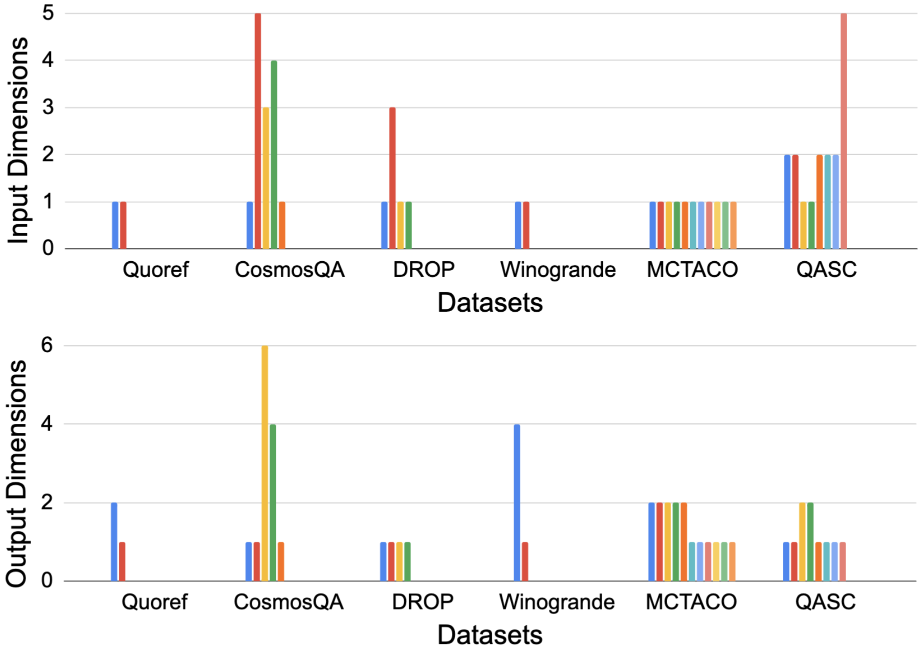


Figure 3.13: Variation in Number of Dimensions

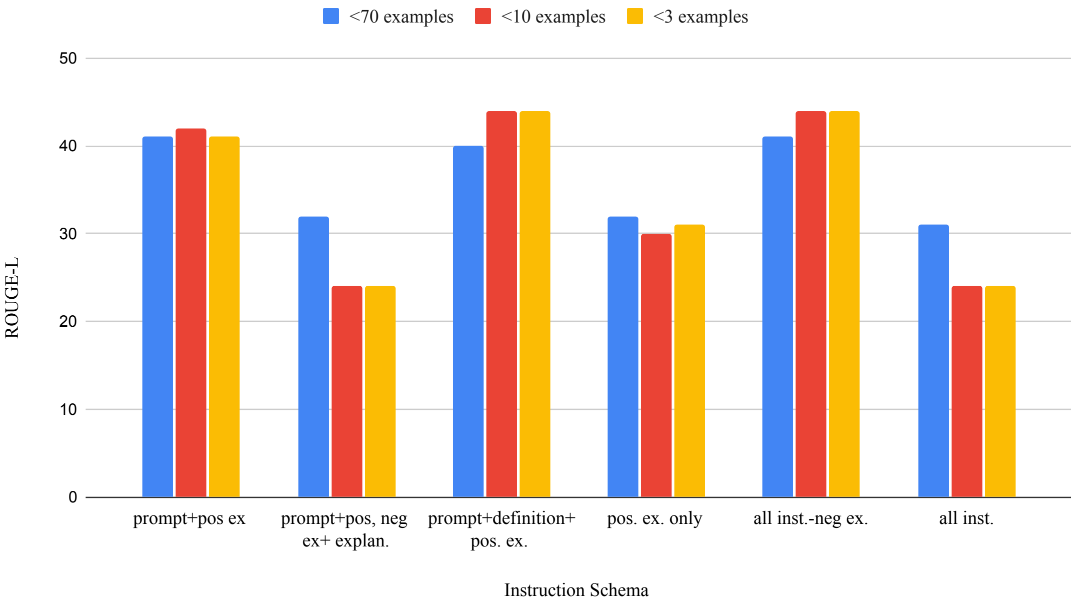


Figure 3.14: GPT3 Performance as a Function of the Number of Examples in Its Encoding. The Number of Examples Is Limited by Three Upperbounds: 3, 10 and 70. This Shows That Addition of Examples Is Not Helping GPT3.

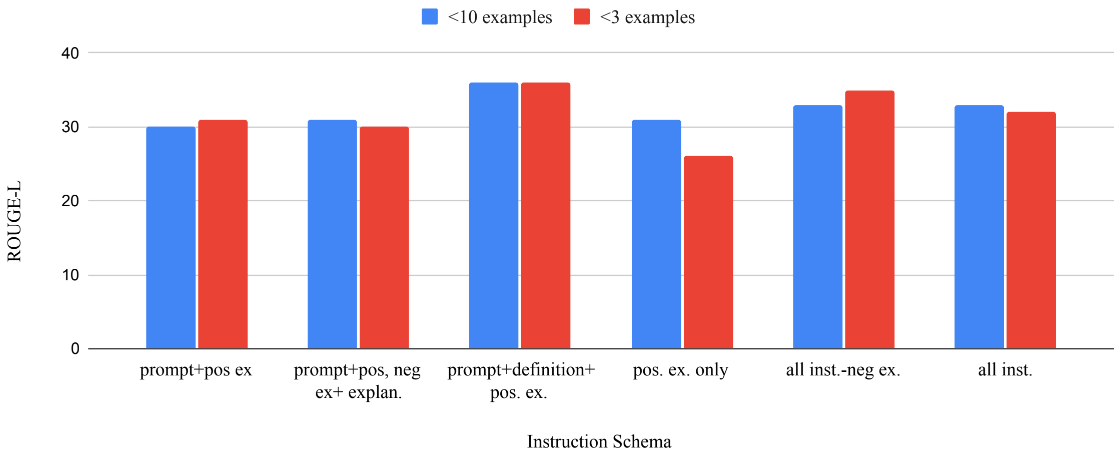


Figure 3.15: BART Performance as a Function of the Number of Examples in Its Encoding. The Number of Examples Is Limited by Two Upperbounds: 3 and 10. This Shows That Addition of Examples Is Not Helping Bart. Since Bart's Maximum Token Size Is 1024, It Can Not Fit a Lot Examples Unlike GPT3, so We Did Not Experiment Further with Larger Number of Examples.

Chapter 4

NUMGLUE: MULTITASKING IN NUMERICAL REASONING

4.1 Introduction

Reasoning with numbers is an important skill that occurs in various day-to-day scenarios and not surprisingly, numbers are ubiquitous in textual data. To train AI reasoning systems that can perform simple mathematical reasoning, many tasks have been proposed (Dua *et al.*, 2019b; Ravichander *et al.*, 2019; Koncel-Kedziorski *et al.*, 2016). Despite these efforts, current state-of-the-art AI systems are brittle and fail when problems involving similar mathematical reasoning are posed in a slightly different manner. For instance, presenting a word problem in a different manner as shown in 4.1, while hardly affecting human performance, is sufficient to confuse state-of-the-art AI systems (The recently released GPT3-Instruct, a fine-tuned model with 175B parameters produces inconsistent answers for these questions). This brittleness in reasoning indicates that the models latch on to spurious signals in the specific dataset resulting in "solving" the dataset while not truly understanding the underlying reasoning skill of simple arithmetic.

Further, we believe that building AI systems that can truly understand and apply simple arithmetic reasoning is a mandatory first step towards successfully tackling complex mathematical reasoning skills (Saxton *et al.*, 2019; Hendrycks *et al.*, 2020a, 2021b). This has been discussed further in our work (Mishra *et al.*, 2022g).

NumGLUE. To this end, we propose NumGLUE, a multi-task benchmark consisting of eight different tasks that at their core test for arithmetic reasoning skills. For example, as discussed in 4.1, tasks can involve word problems presented in a slightly different manner or

Original Word Problem

John had 5 apples. He gave 3 to Peter. How many apples does John have now?

Fill In The Blanks Format

John had 5 apples. He gave 3 to Peter. John has ____ apples now.

NLI Format

Premise: John had 5 apples. He gave 3 apples to Peter. Hypothesis: John has 2 apples now. Does the hypothesis entail, contradict or is neutral to the premise?

Comparison Format

John had 5 apples. He gave 3 to Peter. Who has more apples?

Figure 4.1: A System That Can Robustly Perform Numeric Reasoning over Language Should Be Able to Solve Problems Such as the above, Regardless of How the Problem Is Posed. However, We Observe Existing Systems Are Brittle; Producing Inconsistent Solutions to Such Minor Stylistic Variations.

can involve additional reasoning strategies like commonsense reasoning or reading comprehension to be combined with the core skill of simple arithmetic. Our benchmark consists of four new tasks in addition to four existing ones; with $\sim 100K$ problems spread across eight different tasks.

The motivation behind NumGLUE is similar to GLUE (Wang *et al.*, 2018, 2019), a multi-task benchmark that aimed at models that demonstrated superior language understanding by learning the underlying linguistic features. NumGLUE is designed with goal of progressing towards AI systems that are capable of performing arithmetic reasoning in a general setting; achieving superior performance on our benchmark requires the ability to correctly identify and perform the underlying arithmetic reasoning without relying on task or dataset-specific signals. Finally, we hope that NumGLUE will encourage systems that perform robust and general numeric reasoning within language, a first step towards being able to perform more complex mathematical reasoning.

4.2 NumGLUE

As mentioned previously, our NumGLUE benchmark consists of both new and already existing arithmetic reasoning tasks. We first begin by introducing the novel datasets curated by us before providing a brief overview of existing tasks that are part of NumGLUE. Finally, in this section, we provide an analysis of the datasets demonstrating that it contains interesting and diverse linguistic and mathematical properties.

NumGLUE Benchmark. Our proposed NumGLUE benchmark is a collection of eight different tasks that together include $\sim 100K$ questions. The tasks may either be self-contained or require additional background knowledge (commonsense reasoning) to arrive at the final solution; however, all the tasks, at their core, involve arithmetic reasoning. 4.1 shows an example question belonging to each task along with indicating the total number of data points associated with each task. It is important to note that tasks are imbalanced with only ~ 400 examples for Task 1 and nearly $50K$ questions under Task 5. While we could have under-sampled the questions to create a balanced suite, we retain the imbalanced dataset in order to mimic the real world – for instance, arithmetic word problems are more abundant as opposed to word problems that may require commonsense reasoning in addition to arithmetic reasoning.

Data Partition and Evaluation. We randomly partition data in each task into training

Task	Question Setting	Size	Example
TASK 1	Commonsense + Arithmetic	404	Question: A man can lift one box in each of his hands. How many boxes can a group of 5 people hold in total? Answer: 10
TASK 2	Domain specific + Arithmetic	1620	Question: How many units of H_2 are required to react with 2 units of C_2H_4 to form 2 units of C_2H_6? Answer: 2
TASK 3	Commonsense + Quantitative	807	Question: A person wants to get shopping done quickly. They know that they can get through the check-out at big store in 5 minutes whereas it can take 20 minutes at small store. The store they go to finish quickly is? (A) big store (B) small store? Answer: big store
TASK 4	Fill-in-the-blanks	1100	Question: Joan found 70 seashells on the beach. She gave Sam some of her seashells. She has 27 seasshells left. She gave _____ seashells to Sam? Answer: 43
TASK 5	RC + Explicit NR	54212	Passage: <>. Question: How many counties were added in 1887? Answer: 2
TASK 6	RC + Implicit NR	32724	Passage: <>. Question: Which player kicked the shortest field goal? Answer: David Akers
TASK 7	Quantitative NLI	9702	Statement 1: James took a 3 - hour bike ride, Statement 2: James took a more than 1 - hour bike ride, Options: Entailment or contradiction or neutral?, Answer: Entailment
TASK 8	Arithmetic word problems	1266	Question: Joe had 50 toy cars. If he gives away 12 cars, how many cars will he have remaining?, Answer: 38

Table 4.1: Size and Example of Each Task in the NumGLUE Benchmark. RC: Reading Comprehension, NR: Numerical Reasoning

(70%), development (10%) and test (20%) sets . In the case of reading comprehension tasks (Task 5 and 6), we assign all questions corresponding to a passage to the same split – we do this in order to discourage any data leakage and thereby, allowing models to potentially rely on memorization to arrive at the correct answer.

For each task, we report the F1 measure and as an aggregate measure of performance on the NumGLUE benchmark similar to Dua *et al.* (2019b), we report the (unweighted) average of the F1 scores corresponding to each task.

4.2.1 Novel Datasets

The novel tasks proposed as part of NumGLUE are a combination of both freshly collected data and intelligent modifications of already existing datasets. The four novel arithmetic reasoning tasks introduced are as follows [1] :

Task 1: Commonsense + Arithmetic Reasoning. Consider the following question – *How many faces do 10 dice have?* Answering this not only requires simple arithmetic multiplying the number of faces in a die by ten but also requires knowing that a standard die has six faces. We collect this dataset by first asking the annotator to write down a numerical commonsense fact (a human has 2 hands, a day has 24 hours) and then use frame a question that requires using this numerical fact as part of a simple arithmetic calculation.

Task 2: Domain Specific + Arithmetic Reasoning. *How many units of hydrogen are required to produce 10 units of water?* This question, similar to the previously introduced task of arithmetic reasoning questions, requires additional domain-specific knowledge –

[1] We annotate the datasets manually.

specifically, that each unit of water contains two units of hydrogen. We curate a dataset of such questions that require both domain-specific knowledge and arithmetic reasoning motivated by the finding that QA systems perform poorly on the ARC dataset (Clark *et al.*, 2018) consisting of grade-school level science questions. Specifically, the dataset collected by us requires understanding of a small set of chemistry (conservation of mass in chemical reactions) and physics principles ($speed = distancetime$).

Task 3: Commonsense + Quantitative Comparison. *A golf ball weighs 40g and a baseball weighs 150g. Which has a higher gravitational force?* Answering this question requires both knowing that mass is directly proportional to gravitational force and a numerical comparison via subtraction. We collect such quantitative comparison questions by using the QuaRel dataset (Tafjord *et al.*, 2019) containing questions from diverse fields such as physics and economics as the starting point. The annotator chooses a subset of these questions that involve numerically comparable quantities (for instance, in this example, mass of the objects involved) to create the required task of quantitative comparison questions.

Task 4: Fill-in-the-blanks Format. Unlike the previously proposed tasks that require external information (commonsense knowledge) in addition to simple arithmetic reasoning, this task is self-contained but a stylistic variant of existing math word problems. We source word problems from the Arithmetic Word Problem repository (Roy and Roth, 2015, 2017, 2018) and convert them into the fill-in-the-blanks format. For an example of such a conversion, refer to 4.1.

4.2.2 Existing Datasets

We now review existing datasets while discussing any modifications made when including them in NumGLUE. In general, for all the datasets included, we perform a filtering step to clean and control for the quality of the data points being included. This step includes – a) discarding questions that do not have answer annotations b) eliminating questions with high lexical overlap with the remainder of the dataset and c) fixing any type mismatches present in the data (“7.0 students” → “7 students”).

Task 5: Reading Comprehension (RC) + Explicit Numerical Reasoning. We select a subset from the DROP (Dua *et al.*, 2019b) dataset to create this task. Specifically, the selected questions involve reading comprehension and numerical reasoning but importantly, the required answer is also a number.

Task 6: Reading Comprehension (RC) + Implicit Numerical Reasoning. Consider the following question based on a relevant passage – *Which state has the highest income tax rate?* Here, while the final answer is a name, arriving at it requires performing comparison (subtraction). We classify such questions in the DROP dataset as a separate task in NumGLUE.

Task 7: Quantitative NLI EQUATE (Ravichander *et al.*, 2019) introduces quantitative NLI questions that require simple arithmetic calculations to be performed in order to accurately classify the relationship between the provided premise and the hypothesis. As noted in 4.1, many word problems can also be easily converted to this format and is therefore, a diverse and interesting task for evaluating arithmetic reasoning skills of AI systems.

Task 8: Arithmetic Word Problems Finally, we arrive at one of the earliest and extensively studied class of arithmetic reasoning problems word problems. The specific dataset included as part of our NumGLUE benchmark is a combination of multiple datasets proposed by Koncel-Kedziorski *et al.* (2016), (Koncel-Kedziorski *et al.*, 2015) and Kushman *et al.* (2014).

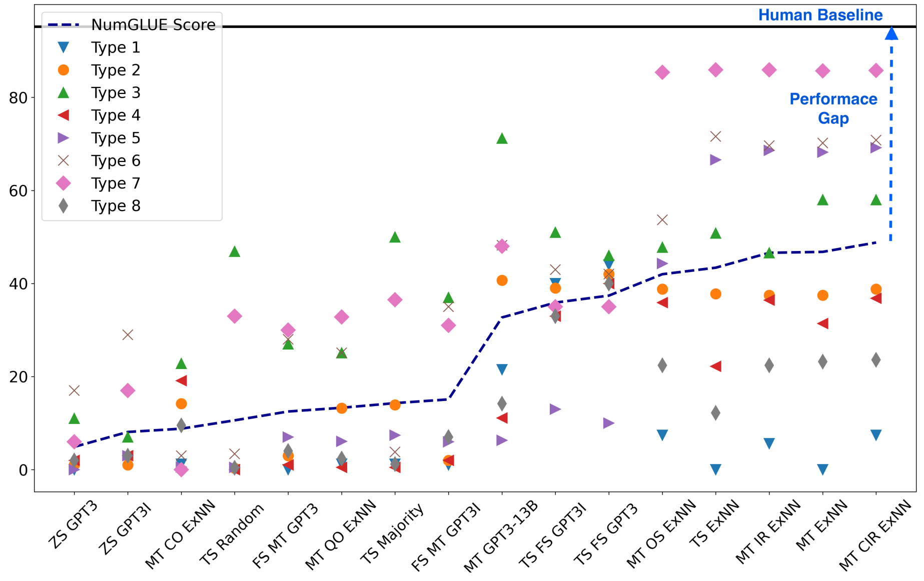


Figure 4.2: Performance of Zeroshot, Fewshot and Finetuning Baselines Across NumGLUE. There Is a Significant Gap Between the Highest Performing Model and the Human Baseline. ZS: Zeroshot, Gpt3I: Gpt3-instruct, MT: Multi-task, TS: Task-specific, QO: Question Only, CO: Context Only, EXNN: Ex-numnet,FS: Few-shot, OS: Oversampling, IR: Information Retrieval, CIR: Conditional Information Retrieval.

Further, to ensure that the benchmark as a whole is diverse, we eliminate questions that have a high sentence similarity with questions from the fill-in-the-blanks task.

4.2.3 Data Quality Analysis:

In order to ensure a high-quality test set, three independent annotators evaluate each question in the test set across all tasks. A tiny portion of the data marked as invalid or with disagreement between the annotators was excluded, resulting in a verified, high-quality NumGLUE evaluation suite. We also perform a variety of analyses and find that the novel question tasks we created (task 1-4) have higher quality than the existing question tasks

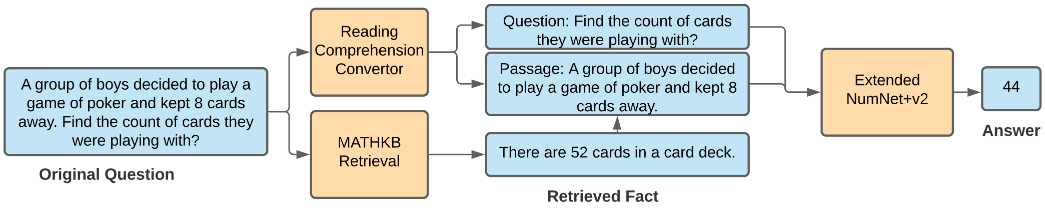

A group of boys decided to play a game of poker and kept 8 cards away. Find the count of cards they were playing with?
Original Question
Reading Comprehension Convertor
MATHKB Retrieval
Question: Find the count of cards they were playing with?
Passage: A group of boys decided to play a game of poker and kept 8 cards away.
There are 52 cards in a card deck.
Retrieved Fact
Extended NumNet+v2
44
Answer

Figure 4.3: Our Proposed Memory-augmented Model That Detects the Type of Task (1-8), Uses Information Retrieval from *Math KB* and Append the Information That Gets Fed to Ex-NumNet

since they have higher average vocabulary (number of unique words per number of samples), higher number of unique nouns, verbs and other POS tags and have less semantic textual similarity among each other (indicating lower repetition).

4.3 Experiments

In this section, we establish multiple baselines on our benchmark and discuss their performance.

4.3.1 Baselines

We evaluate several baselines on our benchmark – (i) Heuristic, (ii) Zero-shot, (iii) Few-shot, (iv) Fine-tuning and (v) Human. We use two kinds of model architectures (i) Neuro-symbolic, a memory augmented *novel* architecture that extends Numnet+v2 (Ran *et al.*, 2019) and (ii) End-to-end, GPT3 (Brown *et al.*, 2020).

Architectures. In the multi-task setting where the same model is trained on all the NumGLUE tasks, we use Reading Comprehension (RC) as the common format – converting each task to RC format via a set of hand-coded rules. In addition to being capable

Learning	Baseline	Baseline	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	NumGLUE
	category	name									Score
Heuristic	Task-specific	Random	0	0.3	46.9	0	0.5	3.4	33	0.4	10.6
	Task-specific	Majority	1.2	13.9	50	0.5	7.4	3.8	36.5	1.2	14.3
Zero-Shot	-	GPT3	0	1	11	2	0	17	6	2	4.9
	-	GPT3-Instruct	2	1	7	3	3	29	17	3	8.1
Few-Shot	Task-specific	GPT3	**44**	**42**	46	40	10	42	35	40	37.4
	Task-specific	GPT3-Instruct	40	39	51	33	13	43	35	33	35.9
	Multi-task	GPT3	0	3	27	1	7	28	30	4	12.5
	Multi-task	GPT3-Instruct	1	2	37	2	6	35	31	7	15.1
Fine-tuning	Multi-task	GPT3-13B	21.5	40.7	**71.2**	11.1	6.3	48.2	48.0	14.2	32.7
Fine-tuning	Multi-task (Q-only)	Ex-NumNet	1.2	13.2	25.1	0.5	6.1	25.1	32.8	2.4	13.3
	Multi-task (C-only)	Ex-NumNet	1.2	14.2	22.8	19.1	0.6	3	0	9.5	8.8
	Single-task	Ex-NumNet	0	37.8	50.8	22.2	66.6	**71.6**	85.9	12.2	43.4
	Multi-task	Ex-NumNet	0	37.5	58	31.4	68.2	70.2	85.7	23.2	46.8
	Multi-task + IR	Ex-NumNet	5.6	37.5	46.6	36.4	68.6	69.6	**85.9**	22.4	46.6
	Multi-task + CIR	Ex-NumNet	7.4	38.8	58	**36.8**	**69.2**	70.8	85.8	**23.6**	**48.8**
	Multi-task + OS	Ex-NumNet	7.4	38.8	47.8	35.9	44.3	53.7	85.4	22.4	42.0
-	-	Human	94.4	94.5	97.8	95	94.7	96.1	96.5	92.8	95.2

Table 4.2: F1 Performance of Various Baselines on the NumGLUE Test Set Across Various Tasks 1-8. Human Performance Was Calculated on 100 Samples of Each Task (81 of Task 1) [*IR = Information Retrieval, CIR=conditional Information Retrieval, OS=oversampling, Q. Only: Question Only, C. Only: Context Only].

of faithfully representing all the constituent tasks, the RC format also allows us to inject additional context in the IR setting without affecting the rest of the pipeline [2] . On the other hand, GPT3 being a generative model does not require such modifications. Importantly, note that both models are inputted the exact same information for the multi-task experiments.

Heuristic Baselines with Task Oracle. For this baseline, we assume a task oracle that

[2] Henceforth we will be calling our extension to Numnet+v2 as Ex-NumNet

knows the task a particular question belongs (in a multi-task setting) – we use this to make our heuristic baselines more competitive. The first heuristic baseline is *random*: we randomly select one of the options in case the question has multiple options (task 3 and 7), a number between 0 to 100 for questions having a numerical answer and a random entity present in the passage for questions having a text segment from the passage as the answer. In the *majority* baseline, we select the most frequent answer for each task such as "Entailment" for NLI questions and similarly, the most frequent number for questions having numerical answer and the major entity present in the passage for questions having span based answer. As the task information is known, we include these baselines under task-specific baselines when discussing results.

Zeroshot and Fewshot Baselines. We use GPT3 (Brown *et al.*, 2020) and the more recent GPT3-Instruct [3] . We have two types of few shot baseline (i) task specific and (ii) multi task. In case of task specific fewshot baseline, instances of the same task are used as in-context examples (Brown *et al.*, 2020) whereas in case of multitask few shot baseline, instances from all tasks are used to condition the model. Multitask fewshot is naturally a harder setting as it is task-agnostic. We use default parameters in GPT3 and GPT3-Instruct. In few-shot setting, we experiment after feeding as many examples as it can fit within the tokensize. For few shot experiments, we randomly select examples and averaged the results over 5 runs.

Fine-tuning Baselines. We first consider variations of the fine-tuning baselines in the context of our neuro-symbolic model, Ex-NumNet. We use it as bias-checking baseline – to ensure that solving the benchmark correctly requires considering all of the information presented to it. To this end, we evaluate the performance of our model when finetuned only on the question (Q-only) or the context (C-only). Next, we present task-specific and

[3] newly released by OpenAI as part of the GPT3 finetuned series

multi-task baselines where Ex-NumNet is fine-tuned on individual tasks and the entire NumGLUE benchmark respectively. With the goal of addressing the data imbalance across the tasks, we include an oversampling baseline that oversamples data from tasks with limited data so as to ensure that the model sees the same number of examples from each constituent task.

In addition, we propose a new architectural modification to Ex-NumNet. Noting that our baseline model Ex-NumNet does not take into account external knowledge, we create a new enhanced architecture in the form of a memory-augmented model that does Information Retrieval (IR) (Khot *et al.*, 2019) with respect to a knowledge base we create, *MATH KB* to identify the needed knowledge. This is inspired by the observation that formula book and mathematical knowledge make the task easier for humans while solving math questions of various types. We then use this knowledge in the Ex-NumNet setting. Figure 4.3 illustrates our approach which leverages our newly created knowledge base *MATH KB*. Conditional IR model is different from the regular IR model in the sense that, IR is performed only for questions of task 1 , 2 and 4, since they require external knowledge to get answered.

Finally, we discuss fine-tuning baselines in the context of end-to-end models, specifically GPT3. We finetune the GPT3-13B model (for which the finetuning capability has been recently provided by OpenAI [4]) in the multi-task setting i.e. the desired setting of the NumGLUE benchmark.

Human Baseline. Human baseline was calculated on 100 test set samples of each task (81 of Task 1) by averaging the scores of four annotators.

[4] https://beta.openai.com/docs/guides/fine-tuning

4.4 Results and Discussion

Table 4.2 shows the performance of various baseline models on the test set of our benchmark. Note that the performance of all baseline models is significantly lesser than the human baseline (Figure 4.2). We now discuss various insights based on these results.

Does the benchmark contain bias that a model can exploit? A challenging dataset requires the model to ideally consider all the information provided to it before arriving at an answer. To ensure that this is indeed the case, we perform ablations where only one portion of the input is provided i.e. either the question or the context. Both these "bias-checking" baselines perform poorly even in task-specific settings – indicating that both the benchmark and constituent tasks are challenging.

Which Tasks are Hard to Solve? Our results show that task 1 which requires numerical commonsense knowledge, is the hardest task to solve. Similarly, tasks 2, 4 and 8 appear to be comparatively harder from the rest. One pattern among these tasks is that all of them expect the answer to be numeric. Numeric answer requires accurate calculation. So, models might have difficulty in learning the task directly from data. This hypothesis is also justified from the *slight* drop in human performance in these tasks..

On the other hand, task 7 has the best performance among all. Further, we see that performance on task 6 is slightly better than task 5 – although both tasks are sourced from the same dataset, we observe that models answer span based questions better as compared to numeric answers. Relatively higher performance for task 3 suggests that models find it easier to answer in an MCQ setting.

Does IR Help? Results show that knowledge help in improving performance of tasks

1, 2 and 4 – where indeed, external knowledge like commonsense or domain-specific knowledge is needed in addition to arithmetic reasoning to arrive at the correct answer. However, task 3 is an exception to this trend and in fact registers a drop in the score when provided with (unnecessary) additional information; we find that this shortcoming is fixed when using conditional information retrieval (CIR) which in fact leads to the strongest baseline presented in this work.

Does Oversampling help overcome data imbalance across tasks? Even though oversampling results in higher performance in certain tasks (in comparison with the multitask baseline), specifically the ones with smaller training data, it results in significant drop in performance in the other extreme, i.e tasks with bigger training data. Also, it never performs better than the Conditional IR module in multitask setting.

4.4.1 Error Analysis

We now present an analysis of the errors made by our baselines to indicate potential avenues for future research.

We analyze errors associated with 50 samples in each of the 8 tasks and find that there are mainly 4 categories of error models make: (1) producing invalid output (e.g. answering text where the answer is supposed to be a number, answering a text different from the classes allowed in a classification problem), (2) copying a number from the question instead of calculating the answer, (3) incorrect calculation – this can be due to multiple reasons including (i) using an incorrect operation e.g. subtraction in place of addition, (ii) incorrect parsing of numbers or (iii) incorrect knowledge of numerical commonsense facts. (4) producing redundant text after producing correct answer. Based on error distribution in Table 4.3, we observe that the majority of errors come from incorrect calculation. Further, GPT3

Error	Ex-NumNet	GPT3
Invalid output	16 %	7%
Copy number	5 %	3%
Incorrect calculation	71 %	56%
Redundant text	8 %	34%

Table 4.3: Error Analysis for the Best Ex-NumNet Multitask+CIR and GPT3 Task-specific Model

is better than Ex NumNet+v2 in producing valid outputs, but it produces more redundant text.

Future Directions: Bigger model, more data or . . . ? Table 4.2 shows that fine-tuned GPT3-13B outperforms other baselines on task 1, 2 and 3. Recall that these tasks require external knowledge and perhaps, this is the reason why GPT3, already pre-trained on a diverse web-scale text corpus has an edge over other baselines on these tasks. In case of the smaller Ex-NumNet, it is interesting that multitask baselines are higher than the single task baselines by 3.4% on average and that information retrieval helps in tasks that require external knowledge. Also notice that, GPT-3 is better on smaller datasets and NumNet is better on large datasets. This may indicate that GPT-3 is a better few-shot learner but not necessarily a better many-shot learner. This non-overlapping performance of GPT-3 and Ex-numnet, end-to-end and neuro-symbolic models respectively, indicates that a potential future direction for research is to combine the best of both the models.

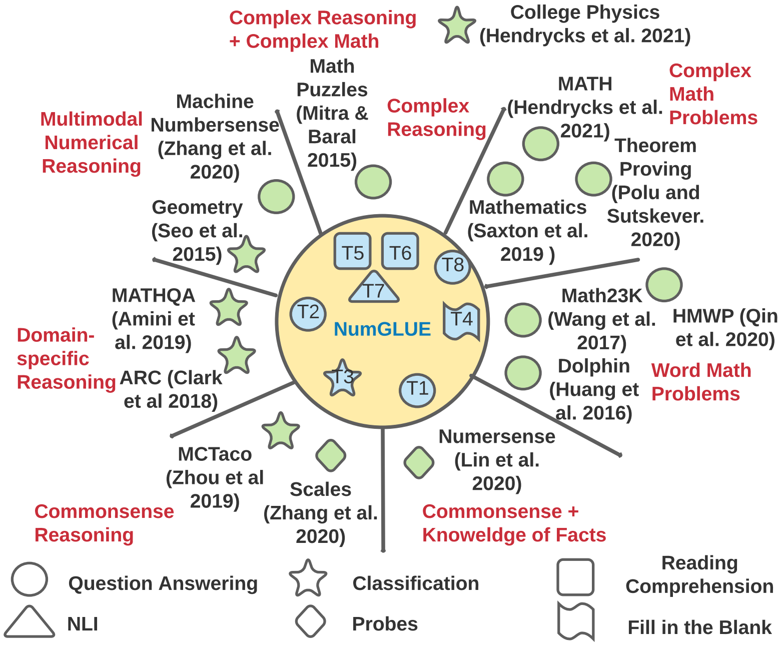

Complex Reasoning + Complex Math
College Physics (Hendrycks et al. 2021)
Complex Math Problems
MATH (Hendrycks et al. 2021)
Math Puzzles (Mitra & Baral 2015)
Complex Reasoning
Multimodal Numerical Reasoning
Machine Numbersense (Zhang et al. 2020)
Theorem Proving (Polu and Sutskever. 2020)
Mathematics (Saxton et al. 2019)
Geometry (Seo et al. 2015)
T5
T6
T8
T7
T2
NumGLUE
T4
T3
T1
MATHQA (Amini et al. 2019)
Math23K (Wang et al. 2017)
HMWP (Qin et al. 2020)
Domain-specific Reasoning
ARC (Clark et al 2018)
Dolphin (Huang et al. 2016)
Word Math Problems
MCTaco (Zhou et al 2019)
Numersense (Lin et al. 2020)
Scales (Zhang et al. 2020)
Commonsense Reasoning
Commonsense + Knowelдge of Facts
Question Answering
Classification
Reading Comprehension
NLI
Probes
Fill in the Blank

Figure 4.4: Our Dataset NumGLUE (Center in the Yellow Circle) Has Been Positioned with Respect to Existing Datasets. T1-T8 Represents 8 Tasks. Note That, NumGLUE contains the Feature of Being Format Invariant Unlike Other Datasets. Position of Datasets Within Clusters Is Done Based on Their Semantic Category, for Example T1 Numerical Commonsense Qa Is Closer to the Cluster of Commonsense Reasoning + Knowledge of Facts; Its Position Reflects the Same

4.5 Additional Analysis

4.5.1 NumGLUE vs Other Datasets:

As figure 4.4 shows, we select each task from one of the clusters of numerical reasoning datasets (except the multi-model reasoning cluster since we wanted to limit our dataset to text only).

4.5.2 Construction of NumGLUE :

Figures 4.5 and 4.6 illustrate detailed data creation process for task 1, task 2, task 3 and task 4 questions with the help of an example for each task. We follow the same procedure for creating other examples within the task.

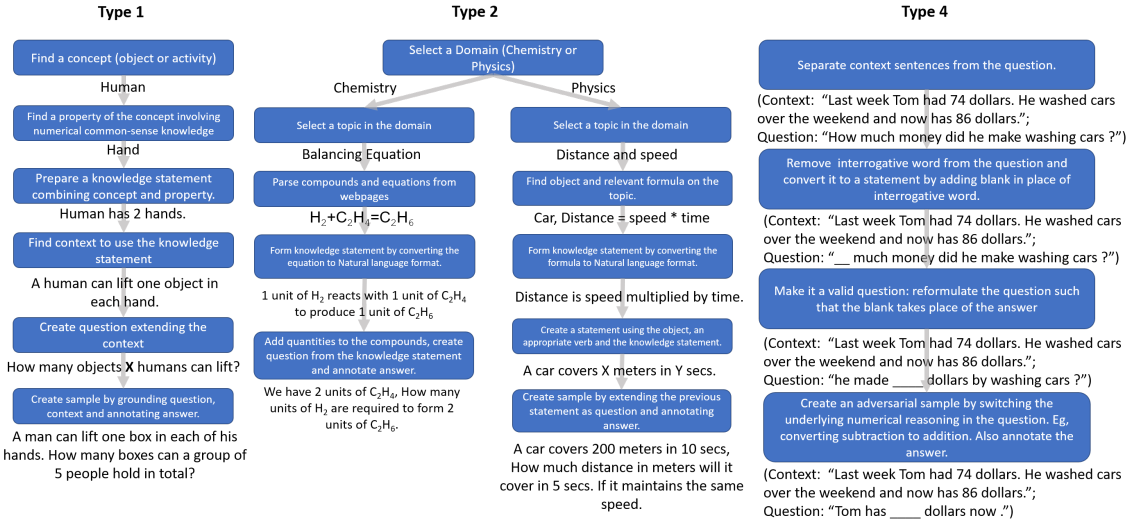

Type 1
Find a concept (object or activity)
Human
Find a property of the concept involving numerical common-sense knowledge
Hand
Prepare a knowledge statement combining concept and property.
Human has 2 hands.
Find context to use the knowledge statement
A human can lift one object in each hand.
Create question extending the context
How many objects X humans can lift?
Create sample by grounding question, context and annotating answer.
A man can lift one box in each of his hands. How many boxes can a group of 5 people hold in total?
Type 2
Select a Domain (Chemistry or Physics)
Chemistry
Physics
Select a topic in the domain
Balancing Equation
Parse compounds and equations from webpages
$H_2+C_2H_4=C_2H_6$
Form knowledge statement by converting the equation to Natural language format.
1 unit of H_2 reacts with 1 unit of C_2H_4 to produce 1 unit of C_2H_6
Add quantities to the compounds, create question from the knowledge statement and annotate answer.
We have 2 units of C_2H_4, How many units of H_2 are required to form 2 units of C_2H_6.
Select a topic in the domain
Distance and speed
Find object and relevant formula on the topic.
Car, Distance = speed * time
Form knowledge statement by converting the formula to Natural language format.
Distance is speed multiplied by time.
Create a statement using the object, an appropriate verb and the knowledge statement.
A car covers X meters in Y secs.
Create sample by extending the previous statement as question and annotating answer.
A car covers 200 meters in 10 secs, How much distance in meters will it cover in 5 secs. If it maintains the same speed.
Type 4
Separate context sentences from the question.
(Context: "Last week Tom had 74 dollars. He washed cars over the weekend and now has 86 dollars."; Question: "How much money did he make washing cars ?")
Remove interrogative word from the question and convert it to a statement by adding blank in place of interrogative word.
(Context: "Last week Tom had 74 dollars. He washed cars over the weekend and now has 86 dollars."; Question: "__ much money did he make washing cars ?")
Make it a valid question: reformulate the question such that the blank takes place of the answer
(Context: "Last week Tom had 74 dollars. He washed cars over the weekend and now has 86 dollars."; Question: "he made ____ dollars by washing cars ?")
Create an adversarial sample by switching the underlying numerical reasoning in the question. Eg, converting subtraction to addition. Also annotate the answer.
(Context: "Last week Tom had 74 dollars. He washed cars over the weekend and now has 86 dollars."; Question: "Tom has ____ dollars now .")

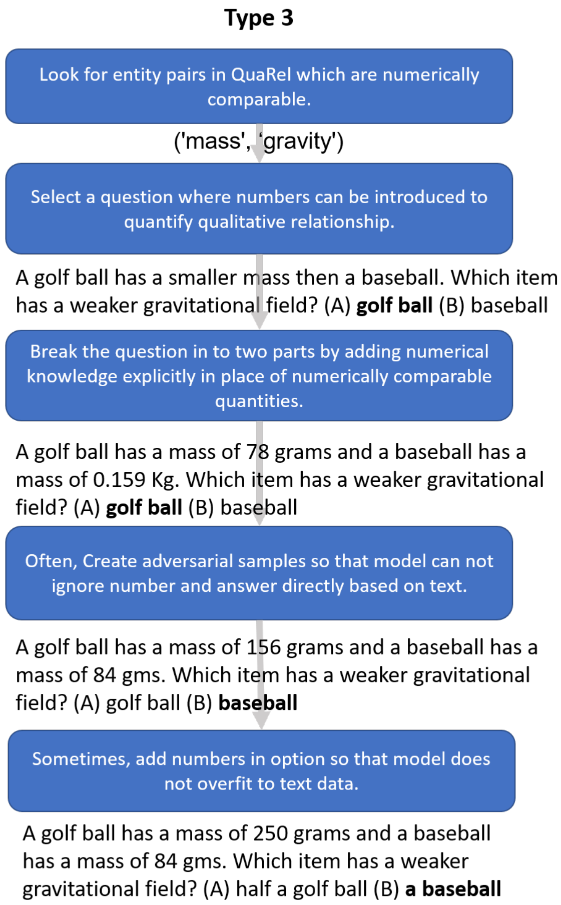
Figure 4.5: Step by Step Data Creation Process for Task 1, 2 and 4 Questions

4.5.3 GPT3-Instruct's Response

We used GPT3-Instruct on various forms of a simple arithmetic question. An expert did tuning of various parameteres such as temperature, stop condition, presence penalty, engine, maximum token size. However, GPT3-Instruct still could not solve the basic aritmetic questions reliably.

4.5.4 Data Quality Analysis of NumGLUE

In this section, we discuss various linguistic and statistical properties of our benchmark; ones that we believe result in the quality, diversity and challenging nature (Gururangan *et al.*, 2018; Mishra *et al.*, 2020b; Mishra and Sachdeva, 2020; Swayamdipta *et al.*, 2020; Mishra



Look for entity pairs in QuaRel which are numerically comparable.

Select a question where numbers can be introduced to quantify qualitative relationship.

A golf ball has a smaller mass then a baseball. Which item has a weaker gravitational field? (A) **golf ball** (B) baseball

Break the question in to two parts by adding numerical knowledge explicitly in place of numerically comparable quantities.

A golf ball has a mass of 78 grams and a baseball has a mass of 0.159 Kg. Which item has a weaker gravitational field? (A) **golf ball** (B) baseball

Often, Create adversarial samples so that model can not ignore number and answer directly based on text.

A golf ball has a mass of 156 grams and a baseball has a mass of 84 gms. Which item has a weaker gravitational field? (A) golf ball (B) **baseball**

Sometimes, add numbers in option so that model does not overfit to text data.

A golf ball has a mass of 250 grams and a baseball has a mass of 84 gms. Which item has a weaker gravitational field? (A) half a golf ball (B) **a baseball**

Figure 4.6: Step by Step Data Creation Process for Task 3 Questions

et al., 2020a; Arunkumar *et al.*, 2023) of the proposed NumGLUE benchmark.

Vocabulary Size. First, we calculate vocabulary size of each task by finding the number of unique words across all questions. Since our dataset is unbalanced in terms of question task, we find the average vocabulary size by dividing vocabulary size with number of data in that task.

Which Data has Higher Average Vocabulary? As illustrated in Figure 4.7a, most of the tasks belonging to the novel dataset category have relatively better average vocabulary size. This implies questions in those tasks have less repetitiveness. Furthermore, we expand our vocabulary analysis to understand Figure 4.7a better. We dive deep to analyze different parts of speech. Figure 4.7b summarises our analysis. Most of the novel datasets have more average number of nouns, verbs and adjectives implying there are more varieties of entities, actions and attributes. This further means that datasets belonging to the novel category are more diverse in nature.

Sentence Similarity Analysis We extend our analysis to reinforce our inference from the word vocabulary analysis. We find Semantic Textual Similarity (STS) of a sentence with every other sentence.

Which Data Consists of Most Dissimilar Sentences? As depicted by Figure 4.7c-4.7f, most questions in QuaRel have high similarity value with other questions indicating the repetitiveness of data. Same is true for majority of EQUATE data. DROP also has high similarity among questions. However, similarity among questions in our dataset is significantly less. Some similarity boxes can be seen in the chart. They are mostly due to task 2 data, and partly due to task 3 data. Lesser similarity implies that our dataset is far less repetitive than others. Also, the repetition in our dataset is sparse and is not equally distributed among the whole dataset unlike others. This way, our dataset is more diverse.

Note that question in Task 2 have lower vocabulary and further, a higher similarity as well. As a small set of chemistry and physics principles are used to generate questions, the result is a fairly templated or uniform-looking dataset – leading to the observed reversal of trends in this particular task.

4.5.5 Ex-NumNet

Figure 4.8 illustrates our baseline model: Ex-NumNet. This contains a Reading Comprehension Converter module which converts each task of question to reading comprehension format. Figure 4.9 illustrates various examples of how each task of questions gets converted to the reading comprehension format. We add a task converter module to detect task of a question. We design task converter heuristically based on the features associated with questions (e.g. NLI contains "Sentence 1" and "Sentence 2" whereas completion contains a blank). We convert each of the tasks to RC format. For NLI questions, we use the premise sentence as passage, hypothesis as the question and append the string "Entailment, contradiction or neutral?" to the question so that it has a span based answer. For other questions, we tokenize the question string into its constituent sentences and use a heuristic approach to split the question string into passage and question. Furthermore, for option based questions, we append all the options at the end of the question.

4.5.6 Proposed Memory-Augmented Model

Figure 4.8 illustrates our baseline model Ex-NumNet. We add an IR mechanism as described in Algorithm 1 and illustrated in Figure 3. As mentioned in the 'Baselines' subsection (Experiments section), we convert each task to RC format in our baseline and append the knowledge retrieved using IR from *MATH KB* at the end of the passage. In our experiments, we use the following hyperparameters in the IR process: $Z = 50$, $v = 10$, $th = 0.75$ and $b = 0.1$.

Formalization Let D represents dataset, s represents sample, K represent the *MATH KB*, v represents the number of knowledge statements retrieved for each sample, th is the cut off STS (Semantic Textual Similarity) value above which knowledge statements are treated redundant and removed, b is the reduction we do iteratively on th until v statements remain.

We create a knowledge base, *MATH KB* by accumulating all tasks of external knowledge which are needed to solve questions of various tasks (e.g. human has 2 hands, cow has 4 legs, there are 24 hours in a day .). We also add math formulae required to solve questions in our benchmark (e.g. the formula of speed in terms of distance and time). We add all these in the form of plain text separated by new line. We use Elasticsearch to retrieve relevant knowledge sentences. We further filter them using a heuristic threshold of relevance. We append this knowledge at the beginning of the passage so that continuity is not broken between passage and question. Figure 3 illustrates our approach.

4.5.7 Hyper Parameters Used

All the experiments were run with the following hyper parameters, batch size was kept at 16 where as the eval batch size was 5. The maximum number of epoch ran for the experiments were 5 with the warm-up kept at 0.06. The learning rate used was 1.5e-5 and the weight decay was 0.01.

All above hyper parameters were selected using a grid search; we kept rest of the hyper parameters unaltered. All the experiments were performed on "TeslaV100-SXM2-16GB", with which the model takes 24hrs to train on nearly 100k samples.

4.5.8 Additional Examples

We provide additional examples of task 1, 2, 3 and 4 questions here to better illustrate the novel datasets we have created as part of our NumGLUE.

Algorithm 1: Our Information Retrieval Approach

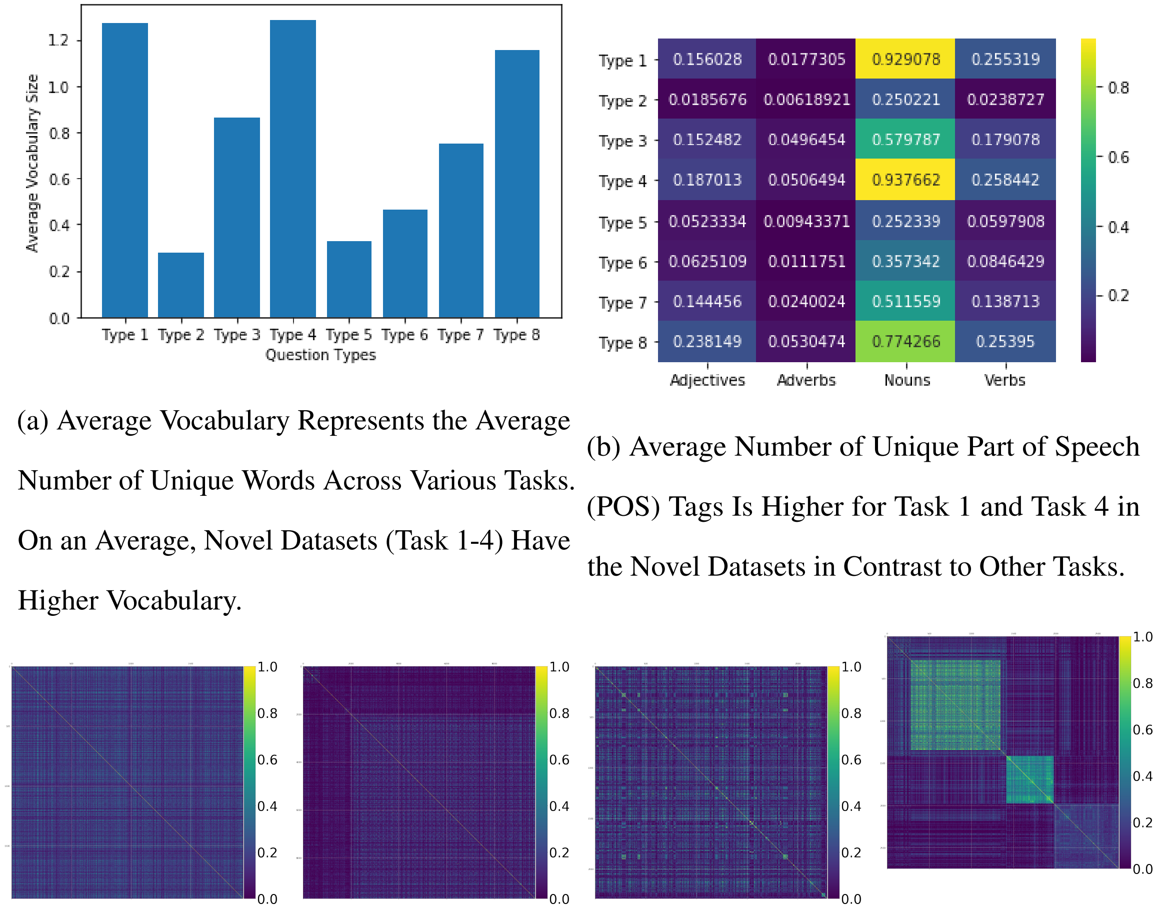
```
Input: Dataset $D$, MATH KB $K$ Hyper-Parameters: $Z$, $v$, $th$, $b$
Output: $v$ Knowledge sentences
forall $s \in D$ do
    Concat Question and Answer ;
    Generate Query by retaining only verbs, adjectives and adverbs;
    forall $j \in K$ do
        Create Index using Elastic Search ;
        Retrieve top Z sentences from MATH KB.
    end
    while size(Z)>v do
        forall $k \in Z$ do
            forall $u \in k - 1$ do
                if STS(Z(u),Z(k))>th then
                    Delete k;
                end
            end
        end
        th=th-b;
    end
end
```

4.6 Conclusion

We propose NumGLUE [5] , a multi-task benchmark to test for arithmetic understanding. Our benchmark consists of eight tasks including four new ones. While some of the tasks require external knowledge like commonsense or domain-specific information in addition

[5] https://allenai.org/data/numglue

to arithmetic reasoning, some are self-contained e.g. arithmetic word problems. Further, we demonstrate that our benchmark is far from being solved – with state-of-the-art large scale models achieving considerably lower performance than humans. This indicates that current AI systems are incapable of performing simple arithmetic reasoning in a general setting – indicating a fundamental hurdle towards AI systems that understand complex mathematical concepts like differential equations or combinatorics. Finally, we present various baselines including a novel architecture (memory augmented Ex-NumNet) that demonstrate the advantages of various modeling choices (e.g. end-to-end vs neuro-symbolic models). Specifically, we show that training in the multi-task setting leads to meaningful sharing of knowledge across tasks as evidenced by an average gain of 3.4% on tasks compared to task-specific modeling. Finally, we hope that our benchmark not only leads to AI systems that are capable of performing simple arithmetic reasoning in a fairly general setting but also results in progress towards more complex mathematical reasoning capability.



(a) Average Vocabulary Represents the Average Number of Unique Words Across Various Tasks. On an Average, Novel Datasets (Task 1-4) Have Higher Vocabulary.

(b) Average Number of Unique Part of Speech (POS) Tags Is Higher for Task 1 and Task 4 in the Novel Datasets in Contrast to Other Tasks.

(c) STS Plot for the Quarel Dataset Shows Significant Repetition Across Samples

(d) STS Plot for the Equate Dataset Shows Considerable Repetition Across Samples.

(e) STS Plot for the Drop Dataset Shows Repetitions for Most Part of the Data.

(f) STS Plot for the Novel Datasets Show Relatively Lower Repetition than Other Datasets

Figure 4.7: Data Quality Analysis of NumGLUE across Various Tasks of Data. On an Average, Novel Datasets Have Higher Quality than the Others since They Have Higher Average Vocabulary, Higher Average Pos Tag Numbers and Lower Semantic Textual Similarity (Sts) among Each Other. X-axis and Y-axis Represents Samples Ordered in the Same Way, an Ideal High Quality Dataset Would Have a Bright Line in the Diagonal and For the Rest of the Places It Should Be Dark Signifying Lower Repetition Across Instances.

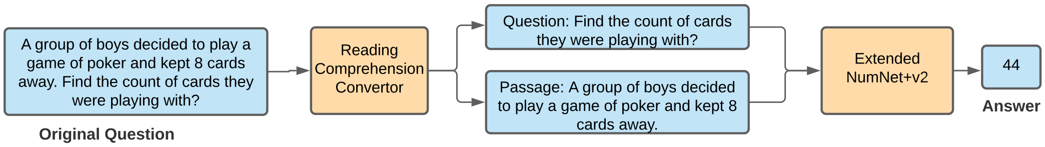
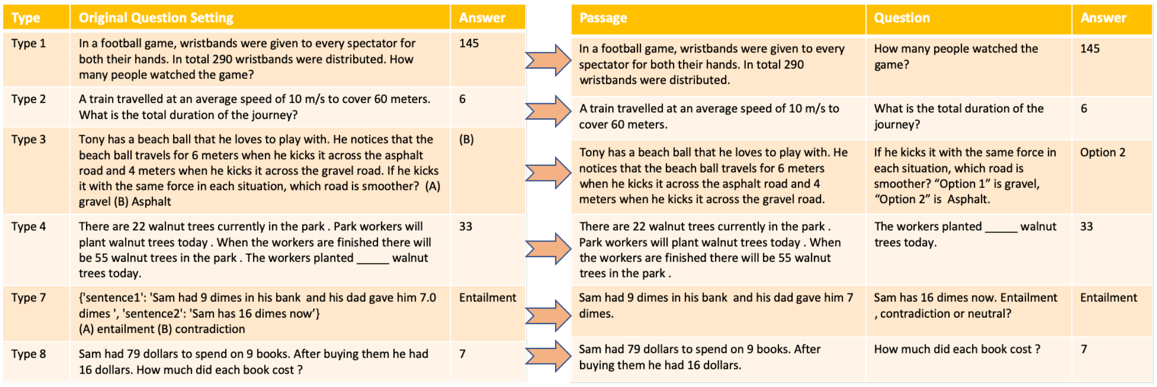

A group of boys decided to play a game of poker and kept 8 cards away. Find the count of cards they were playing with?
Original Question
Reading Comprehension Convertor
Question: Find the count of cards they were playing with?
Passage: A group of boys decided to play a game of poker and kept 8 cards away.
Extended NumNet+v2
44
Answer

Figure 4.8: Architecture of Ex-NumNet



Type	Original Question Setting	Answer
Type 1	In a football game, wristbands were given to every spectator for both their hands. In total 290 wristbands were distributed. How many people watched the game?	145
Type 2	A train travelled at an average speed of 10 m/s to cover 60 meters. What is the total duration of the journey?	6
Type 3	Tony has a beach ball that he loves to play with. He notices that the beach ball travels for 6 meters when he kicks it across the asphalt road and 4 meters when he kicks it across the gravel road. If he kicks it with the same force in each situation, which road is smoother? (A) gravel (B) Asphalt	(B)
Type 4	There are 22 walnut trees currently in the park . Park workers will plant walnut trees today . When the workers are finished there will be 55 walnut trees in the park . The workers planted _____ walnut trees today.	33
Type 7	{'sentence1': 'Sam had 9 dimes in his bank and his dad gave him 7.0 dimes ', 'sentence2': 'Sam has 16 dimes now'} (A) entailment (B) contradiction	Entailment
Type 8	Sam had 79 dollars to spend on 9 books. After buying them he had 16 dollars. How much did each book cost ?	7

Passage	Question	Answer
In a football game, wristbands were given to every spectator for both their hands. In total 290 wristbands were distributed.	How many people watched the game?	145
A train travelled at an average speed of 10 m/s to cover 60 meters.	What is the total duration of the journey?	6
Tony has a beach ball that he loves to play with. He notices that the beach ball travels for 6 meters when he kicks it across the asphalt road and 4 meters when he kicks it across the gravel road.	If he kicks it with the same force in each situation, which road is smoother? "Option 1" is gravel, "Option 2" is Asphalt.	Option 2
There are 22 walnut trees currently in the park . Park workers will plant walnut trees today . When the workers are finished there will be 55 walnut trees in the park .	The workers planted _____ walnut trees today.	33
Sam had 9 dimes in his bank and his dad gave him 7 dimes.	Sam has 16 dimes now. Entailment , contradiction or neutral?	Entailment
Sam had 79 dollars to spend on 9 books. After buying them he had 16 dollars.	How much did each book cost ?	7

Figure 4.9: Conversion of Various Tasks to Reading Comprehension Format

Question	**Knowledge Required**	**Answer**
Ella and Lily are playing a game that requires 10 die. Find out the total number of faces in 10 die.	A die has 6 faces	60
Jacob and Lillian are running a km long race. Jacob finished the race when Lillian was 190 meters from the finish line. How many meters did Lillian cover till that time?	1000 meters make a km	810
A man can lift one box in each of his hands. How many boxes can a group of 5 people hold in total?	A human being has 2 hands	10

Table 4.4: Example Questions Where Numerical Knowledge Required to Answer Is Not Explicitly Provided in the Question.

Question	Knowledge Required	Answer
Find the mass percentage of H in C6H6	Mass of C is 12 units and mass of H is 1 unit	7.69
How many units of H2 are required to react with 2 units of C2H4 to form 2 units of C2H6	H2 + C2H4 = C2H6	2
A car covers 912 meters in 19 seconds. If bike's speed is one fourth of the car. Find the distance covered by the bike in 4 seconds.	distance travelled = speed * time	48

Table 4.5: Example Questions Where Domain Knowledge Is Required to Answer a Question.

QuaRel Question	**Transformed Question**
A person wants to get shopping done quickly. They know that they can get through the check-out at big store faster than they can at small store. The store they go to to finish quickly is (A) **big store** (B) small store	A person wants to get shopping done quickly. They know that they can get through the check-out at big store in 5 minutes whereas it can take 20 mintues at small store. The store they go to to finish quickly is (A) **big store** (B) small store
Tina is racing her two dogs. Her greyhound is slim, her rottweiler is heavy. The dog that gets faster more quickly is the (A) rottweiler (B) **greyhound**	Tina is racing her two dogs. Her greyhound weighs 88 lbs and her rottweiler weighs 79 lbs. The dog that gets faster more quickly is the (A) **rottweiler** (B) greyhound
A golf ball has a smaller mass then a baseball. Which item has a weaker gravitational field? (A) **golf ball** (B) baseball	A golf ball has a mass of 78 grams and a baseball has a mass of 0.159 Kg. Which item has a weaker gravitational field? (A) **golf ball** (B) baseball

Table 4.6: Examples Showing Conversion of Quarel Questions to Quantitative Comparison Questions

Arithmetic Word Problem	**Transformed Question**
Joan found 70 seashells on the beach. She gave Sam some of her seashells. She has 27 seashell left. How many seashells did she give to Sam ? **43**	Joan found 70 seashells on the beach . She gave Sam some of her seashells . She has 27 seashells left. She gave _____ seashells to Sam. **43**
Last week Tom had 74 dollars. He washed cars over the weekend and now has 86 dollars. How much money did he make washing cars ? **12**	Last week Tom had 74 dollars. He washed cars over the weekend and made another 86 dollars. Tom has _____ dollars now . **160**

Table 4.7: Examples Showing MAWPS Questions and Corresponding Questions in Completion Format

Chapter 5

REFRAMING INSTRUCTIONAL PROMPTS

5.1 Introduction

Prompting language models (LMs) (Liu *et al.*, 2021b) has made NLP modules accessible to non-expert users through plain text instructions (We focus on instructional prompts (Efrat and Levy, 2020) as opposed to exemplar prompts which are already well-studied (Brown

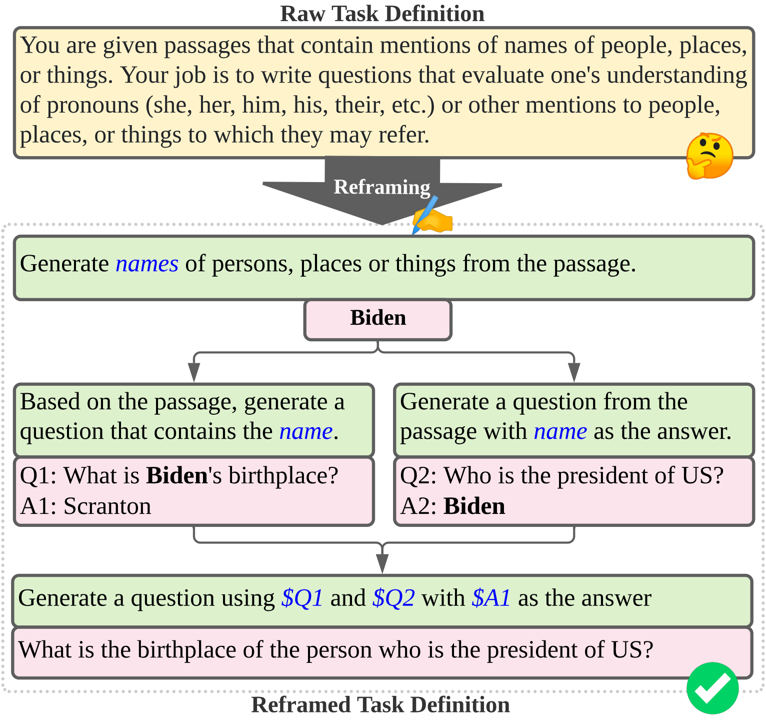

Raw Task Definition
You are given passages that contain mentions of names of people, places, or things. Your job is to write questions that evaluate one's understanding of pronouns (she, her, him, his, their, etc.) or other mentions to people, places, or things to which they may refer.
Reframing
Generate names of persons, places or things from the passage.
Biden
Based on the passage, generate a question that contains the name.
Q1: What is Biden's birthplace?
A1: Scranton
Generate a question from the passage with name as the answer.
Q2: Who is the president of US?
A2: Biden
Generate a question using $Q1 and $Q2 with $A1 as the answer
What is the birthplace of the person who is the president of US?
Reframed Task Definition

Figure 5.1: GPT3 Has Difficulty in Writing Questions That Require Entity Coreference Resolutions Based on a Single Lengthy Prompt (Top, in Yellow), However, It Succeeds in Solving a Manually Reframed Task That Has Four Simpler Sub-steps (Bottom, in Green).

et al., 2020; Lu *et al.*, 2021d) of NLP tasks). Such task instructions written by non-expert users are often long and contain abstract descriptions which are not easy to follow for LMs, as evidenced by their low performance (Efrat and Levy, 2020; Mishra *et al.*, 2022f). However, it is not quite clear whether this is due to the inherent difficulty of the target tasks or an artifact of the complex phrasing of their language instructions.

In this analysis, we aim to understand the sensitivity of LMs to the framing of instructional prompts. In particular, we study several *reframing* techniques to frame instructional prompts differently so that LMs achieve better understanding of the task. These reframing techniques are motivated by various empirical intuitions such as ease of understanding concise and concrete instructions and those that contain little abstract statements about human commonsense or their background knowledge. For example, Fig.5.1 shows a reframing example which involves decomposing a task into multiple sub-tasks. The intended task here is writing questions that require entity coreference (Dasigi *et al.*, 2019). While GPT3 fails in solving the original task instruction (the yellow box at the top), it succeeds when the task is decomposed to four simpler and easier sub-tasks.

We provide analysis for five diverse reframing techniques [1] . These include incorporating low-level patterns about the target task, decomposing and itemizing instructions, stating the task constraints, and providing specialized instructions (examples in Table 5.1-5.3).

We analyze reframed instructions over 12 tasks from Natural Instructions (Mishra *et al.*, 2022f), which contains a variety of NLP tasks and their instructions. Empirically, we compare the quality of LMs (GPT2/3 Radford *et al.* 2019; Brown *et al.* 2020) in two settings: raw vs reframed instructions. In particular, we observe that the reframed prompts have notable performance gains over raw instructions (the gap between the red and blue

[1] https://github.com/allenai/reframing

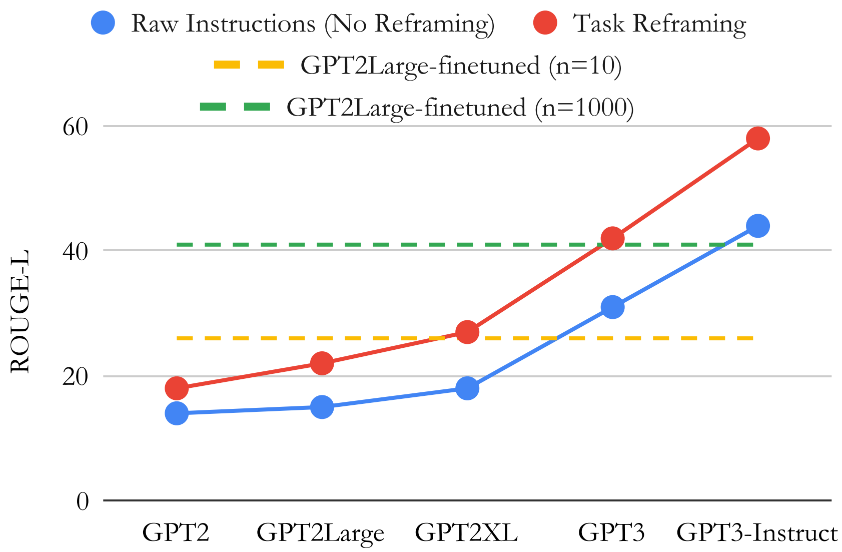


Figure 5.2: Across a Variety of Model Sizes, Reframed Prompts Consistently Show Considerable Performance Gain over Blue Raw Task Instructions (No Reframing)in a Few-shot Learning Setup. Since Fine-tuning GPT3 Is Prohibitively Expensive, We Show the Performance of Fine-tuning Smaller Models (Horizontal Lines). This Results Indicate That *Evaluating* Reframed Prompts on a Large Model like GPT3-instruct (Red Line) Might Be More Effective That *Fine-tuning* a Smaller Model like GPT2large (Green Line) with 200× More Data. Details of the Experiments In §5.3.

trends in Fig.5.2) with an average of 14% and 17% gains when using GPT3-instruct in the few-shot and zero-shot setups, respectively. Furthermore, the average gains across tasks remain consistent across different models hinting at consistency of reframed prompts on various architectures. This is in contrast to the widely-used fine-tuning approaches which need to be performed separately for each model. Reframing prompts by model designers can be particularly effective when evaluated on large LMs, where fine-tuning can be prohibitively expensive (such as GPT3). In particular, we observe that, reframed prompts on GPT3-instruct score roughly 17% higher than GPT2Large that is supervised with $1k$ instances (i.e., 200× more data).

While reframing instructions are not algorithmic, nonetheless, we view this systemic analysis as a preliminary stepping stone in this direction. We hope that this study will lead to the development of algorithmic better few-shot learning methods that generalize across models, thereby leading to more effective ways of reaping the investments already poured into creating massive LMs. This has been discussed further in our work (Mishra *et al.*, 2022e).

5.2 Prompt Reframing

This section describes our reframing principles and then describes the guidelines to operationalize them. Reframing principles are obtained by probing instructions of various tasks in the training split of Natural Instructions (Mishra *et al.*, 2022f) to understand different failure modes associated with prompting in GPT3.

Motivation from GPT3's Failures We observe that GPT3 fails to follow instructions when it is provided with long prompts that often contain repeated information, abstract notions, analogies, complex statements requiring human commonsense and their domain knowledge (see examples in Table 5.1 and 5.7). Humans typically find these helpful for describing their tasks. For example, some content intended to motivate the task or repetition for the sake of emphasis, might be unnecessary or even redundant for a model.

5.2.1 Reframing Principles

We observe that short prompts that contain concrete statements and avoid terms associated with background knowledge improve GPT3's response to instructions. We recursively apply this observation and provide a set of reframing principles to resolve various issues on GPT3's failures with prompting, backed by extensive empirical analysis on GPT3. The

	Raw task definitions and their reframed counterpart
Pattern Reframing	*Raw Task: Craft a question which requires commonsense to be answered. Based on the given context, craft a common-sense question, especially those that are LONG, INTERESTING, and COMPLEX. The goal is to write questions that are easy for humans and hard for AI machines! To create such questions, here are some suggestions: A. What may (or may not) be the plausible reason for an event? B. What may (or may not) happen before (or after, or during) an event? C. What may (or may not) be a plausible fact about someone (or something)? D. What may (or may not) happen if an event happens (or did not happen)? You can also create other types of questions.* **Input**: Context:<> **Expected Output**: Question:<>
	Reframed Task: Use 'what may happen', 'will ...?', 'why might', 'what may have caused', 'what may be true about', 'what is probably true about', 'what must' and similar phrases in your question based on the input context. **Input**: Context:<> **Expected Output**: Question:<>
Itemizing Reframing	*Raw Task: Follow the instructions to produce output with the given context word. Do <>. Do <>. Don't <>* **Input**: Context word <> **Expected Output**: Long text <>
	Reframed Task: Follow instructions below to produce output based on the given context word. - Do <> - Do <> - Do <> **Input**: Context word <> **Expected Output**: Long text <>

Table 5.1: Examples of Various Reframing Techniques. Italicized Text Represents the Prompt. Changes in Prompt and Example in the Transformed Task Are Indicated with Blue and Red Markings, Respectively.

Raw task definitions and their reframed counterpart

Raw Task: *In this task, based on the given context word, you need to create a pair of sentences each containing a blank (_) and their corresponding answer. The sentence pair should look similar, and should be about two related but different objects; for example "trophy" and "suitcase". Also, the sentences must be different in terms of trigger words (e.g., "small" and "big") which express contrasting properties about the two objects.*

Input: Context word:<> **Expected Output**: Question 1: <> Answer 1: <> Question 2: <> Answer 2: <>

Reframed Task:

DECOMPOSITION

REFRAMING

Subtask 1. *Write 2 objects based on the given context word.*

Input: Context word:<> **Expected Output**: Objects: <>

Subtask 2. *Write a sentence by connecting objects with a verb.*

Input: Objects: <> **Expected Output**: Sentence: <>

Subtask 3. *Create a fill in the blank question from the sentence where object 1 will fit the blank.*

Input: Object 1: <>,Sentence: <> **Expected Output**: Question: <>

Subtask 4. *Change the given question so that answer flips to object 2 in the question.*

Input: Object 2: <>, Sentence: <>, Question: <> **Expected Output**: Question: <>

Subtask 5. *Generate both questions and answers:*

Input: Question 1: <> Object 1: <> Question 2: <> Object 2: <>

Expected Output: Question 1: <> Answer 1: <> Question 2: <> Answer 2: <>

Table 5.2: Examples of Various Reframing Techniques. Italicized Text Represents the Prompt. Changes in Prompt and Example in the Transformed Task Are Indicated with Blue and Red Markings, Respectively.

		Raw task definitions and their reframed counterpart
Restraining	Reframing	***Raw Task:****... What is the type of the answer corresponding to the given question? Number, Date, or Span?...* **Input**: Passage: <>. Question: <> **Expected Output**: <Number/Date/Span> ...
		Reframed Task:*... What is the type of the answer corresponding to the given question? Number, Date, or Span?...* **Input**: Passage: <> Question: <> *Answer either Number, Date or Span?* **Expected Output**:<Number/Date/Span>
Specialization	Reframing	***Raw Task:*** *Answer the following question ... <Not so important Text> ...* **Input**: Question <> **Expected Output**: Answer <>
		Reframed Task:*Calculate answer to the following question. You need to either add or subtract numbers associated with two objects present in the question.* **Input**: Question <> **Expected Output**: Answer <>

Table 5.3: Examples of Various Reframing Techniques. Italicized Text Represents the Prompt. Changes in Prompt and Example in the Transformed Task Are Indicated with Blue and Red Markings, Respectively.

principles have light resemblance to how basic tasks are formulated and taught to kids.

(C1) *Use Low-level Patterns*: Instead of using terms that require background knowledge to understand, use various patterns about the expected output.

(C2) *Itemizing Instructions*: Turn descriptive attributes into bulleted lists. If there are any negation statements, turn them into assertion statements.

(C3) *Break it Down*: Break down a task into multiple simpler tasks, wherever possible.

(C4) *Enforce Constraint*: Add explicit textual statements of output constraints.

(C5) *Specialize the Instruction*: Customize the instructions so that they directly speak to the intended output.

We operationalize each of the above principles in terms of 5 *reframing* techniques. The degree of reframing (the amount of change applied to the raw instructions) varies significantly across the reframing techniques: the simplest one adds an enforcement statement at the end whereas the other extreme involves completely changing the task as a whole (e.g., decomposing it into multiple tasks).

5.2.2 Reframing Techniques

We explain each of the reframing techniques in three parts (1) *model failure* states a potential weakness of LM with reference to examples in Table 5.7 (2) *approach* describes our suggested approach and intuition behind it, according to our empirical observations (3) *example* illustrates the application of the suggested technique in reference to Table 5.1-5.3. In designing these techniques, we used a development set that contains all the positive examples included as part of the instructions of each task in .

PATTERN REFRAMING

Model failure While humans have an incredible ability in understanding and acting with respect to abstract descriptions, LMs tend to ignore most of them or just repeat the content of such instructions in their output (*copy instruction* in Table 5.7.)

Approach Find low-level patterns among the dev set examples and extrapolate those by adding similar patterns (C1).

Example Table 5.1 (row 1) illustrates the CosmosQA (Huang *et al.*, 2019) question generation task. The raw task instruction consists of various high-level statements such as “commonsense”, “complex”, “interesting”, “easy for humans and hard for AI machines”, whereas the reframed task consists of various low-level patterns about the expected output such as “what may happen”, “in the future, will..”, “why might”, which generally improve GPT3’s performance in generating valid questions.

ITEMIZING REFRAMING

Model failure LMs cannot follow long paragraphs stating multiple requirements (*first instruction bias* in Table 5.7) and do not perform well when the requirements are formulated as a negative statement (*negation challenge* in Table 5.7).

Approach Turn long descriptions into bulleted lists of several statements (C2). Additionally, turn negative statements to positive ones. For example, reformulate “don’t create questions which are not answerable from the paragraph” into “create questions which are answerable from the paragraph”.

Example Table 5.1 (row 2) illustrates the WinoGrande (Sakaguchi *et al.*, 2020) sample generation task where the raw instructions contain several requisites (do’s and don’ts) that are hard for models to follow. Reframing the instructions into a structured list improves the model response.

DECOMPOSITION REFRAMING

Model failure Tasks with implicit multi-step reasoning are challenging for models, even after itemizing reframing (5.2.2) (*multi-step task challenge* in Table 5.7).

Approach Wherever possible, decompose a task into multiple different sub-tasks which can be executed either sequentially or in parallel (C3) and hence, make them relatively easier for models.

Example In Table 5.2 (row 1), the task is to generate samples for the Winogrande (Sakaguchi *et al.*, 2020) dataset. Decomposition of the task into 5 sequential steps improves GPT3’s response.

Restraining Reframing

Model failure A common mistake of GPT3 occurs when the task definition deviates from its pre-trained objective (predicting next words) (*conventional-task bias* in Table 5.7). For example, when predicting question *types* GPT3 often answers the question instead of generating its type. Similarly, in reading comprehension tasks, GPT3 sometimes answers a question based on its background knowledge instead of answering from the given passage.

Approach Append a statement to the task instruction that expresses a constraint about the output generation (C4).

Example Table 5.2 (row 2) illustrates the DROP (Dua *et al.*, 2019b) answer type generation task where the objective is to generate a valid answer type among "Number", "Date" and "Span" for a given question. Adding an enforcement statement tends to improve the model output by constraining it to the provided types.

Specialization Reframing

Model failure LMs ignore generic instructions such as "answer the following question" and sometimes misconceive the output format when the given instruction contains redundant text (*misconceive output format* in Table 5.7).

Approach Reformulate the instructions so that they directly describe the low-level task needed to be done and drop all the repeated and generic statements (C5).

Example Table 5.3 (row 1) illustrates a task of numerical reasoning problems that involve natural language sentences describing additions and subtractions. The reframed prompt specializes the generic task instruction ("calculate answer").

task	source	category
generating questions on event duration	MC-TACO (Zhou *et al.*, 2019)	Question Generation (QG)
generating questions on sentence composition	QASC (Khot *et al.*, 2020)	
answering event coreference questions	Quoref (Dasigi *et al.*, 2019)	Question Answering (QA)
answering fill in the blank questions on coreference resolution	WinoGrande (Sakaguchi *et al.*, 2020)	
identifying inappropriate content in context	CosmosQA (Huang *et al.*, 2019)	Classification (CF)
identifying bad questions in reading comprehension	MultiRC (Khashabi *et al.*, 2018)	

Table 5.4: List of Evaluation Tasks Used in This Study (§5.3).

5.3 Experimental Setup

Dataset We evaluate the proposed reframing techniques on the evaluation tasks from Natural Instructions (Mishra *et al.*, 2022f), which consists of 12 tasks categorized into 6 categories. Following the original setup, we use ROUGE-L (Lin, 2004) as the evaluation metric in our experiments. Table 6.1 contains the list of evaluation task used in this study.

task	source	category
generating incorrect answers to event transience questions	MC-TACO (Zhou *et al.*, 2019)	Incorrect Answer Generation (IAG)
generating incorrect answers to event duration questions	MC-TACO (Zhou *et al.*, 2019)	
modifying fill in the blank questions on coreference resolution	WinoGrande (Sakaguchi *et al.*, 2020)	Text Modification (MM)
generating paraphrase of given sentences	Miscellaneous	
finding overlapping words between two sentences	QASC (Khot *et al.*, 2020)	Verification (VF)
Identifying words essential for choosing correct answers.	Essential-Terms (Khashabi *et al.*, 2017)	

Table 5.5: List of Evaluation Tasks Used in This Study (§5.3).

Models For evaluation we use various models of the GPT family: GPT2, GPT2Large, GPT2XL, GPT3 and GPT3-instruct (Brown *et al.*, 2020; Radford *et al.*, 2019) [2] and BART-base (Lewis *et al.*, 2019). We evaluate the models according to the following setups:

GPTk w/ raw instructions: We follow the setup of Mishra *et al.* (2022f) who experiment with GPT3-instruct on their raw instructions. Overall the prompts provided to the model consist of three segments (in this order): (a) task instructions, (b) examples (input and outputs) and (c) a new input for which we expect model's response. We experiment with three different variants of the baselines, depending on the number of examples in their prompts: (i) **FEW-SHOT**: We experiment with 5 examples [3] which is a more realistic few-shot setup. (ii) **MAX. EX.**: in another variant we use as many examples as fits within GPT's token limit. (iii) **ZERO-SHOT**: in this setup, we do not incorporate any example while prompting the models with the instructions. Finally, we build variants of these baselines by conducting 'schema selection' where we experiment with 12 different encodings of the instruction (Mishra *et al.*, 2022f) and select the best performing one for each task.

GPTk w/ reframed instructions: The model designer applies various reframing techniques (Section 5.2.2) on tasks in . Similar to the raw instructions baseline, we use 5 examples in our reframed tasks. In our setup, model designer is an author who follows the guidelines (§5.2.2) by observing 5 examples in the development set and reframes instructions. This process was done in interaction with GPT3-instruct via the development examples. This took roughly 15 minutes per task and per reframing type. Similar to the setup with raw instructions, the ultimate encoded prompts contained a concatenation of the following (in this order): reframed instructions, positive examples and the instance input.

GPTk w/ calibration: This method extends the recent calibration approach introduced by Zhao *et al.* (2021), which involves compensating for various model-specific biases in a

[2] https://beta.openai.com/docs/engines/

[3] These 5 positive examples are part of instructions in each task of , and sometimes the number of positive examples is less than 5.

few-shot setup, such as recency bias and majority bias. Zhao *et al.* (2021) perform calibration by masking input instances with 'N/A' tokens, estimating the bias using model prediction probabilities and then compensating the bias while feeding the input instance during prediction. We extend calibration to our instruction setup by masking the input instance in our instruction encoding with an 'N/A' token and calibrating biases associated with GPT3-instruct.

Supervised baseline: While the conventional setup of supervised learning has been successful for reasonably sized models, it is prohibitively expensive for large models like GPT3. We train medium-sized LMs (e.g., BART-base Lewis *et al.*, 2019) on $5k$ examples of each task and evaluate on unseen instances of the corresponding task.

5.4 Empirical Results

5.4.1 Main Results

A summary of our experiments is provided in Fig.5.2 which shows the performance of the reframed instructions on various models, compared to our baselines. Furthermore, Table 5.6 provides a more granular comparison of few-shot, zero-shot and supervised models per task category, all on GPT3-instruct and in terms of ROUGE-L. Below are several takeaways from these experiments.

Reframing improves upon the few-shot and zero-shot baselines. Table 5.6 shows that reframing outperforms the original raw instruction baseline with 14% (44% → 58%) and 17% absolute gains (33% → 50%) in few-shot and zero-shot setups, respectively. Additionally, it outperforms the schema selection baseline with 11% (47% → 58%) and 13% absolute gains (37% → 50%) in few-shot and zero-shot setups, respectively. It also outperforms the calibration and max-examples with schema selection baseline by 12% (46%→ 58%) and 8% (50%→ 58%), respectively. The gains are spread across task categories, with the highest

supervision mode	model	task category → # of examples ↓	QG	AG	CF	IAG	MM	VF	Avg
SUPERVISED	BART	5000	59	61	91	26	85	82	67
FEW-SHOT (MAX. EX.)	GPT3-instruct (raw instructions + schema selection)	32	47	57	52	23	79	42	50
FEW-SHOT	GPT3-instruct (raw instructions)	5	43	54	44	21	70	32	44
	GPT3-instruct (calibrated raw instructions)	5	41↓	52↓	58↑	22↑	70	35↑	46↑
	GPT3-instruct (raw instructions + schema selection)	5	45↑	58↑	49↑	23↑	72↑	37↑	47↑
	GPT3-instruct (**reframed instructions**)	5	**55**↑	**72**↑	**65**↑	**30**↑	**80**↑	**48**↑	**58**↑
ZERO-SHOT	GPT3-instruct (raw instructions)	0	31	34	39	14	69	13	33
	GPT3-instruct (raw instructions + schema selection)	0	37↑	36↑	40↑	17↑	75↑	17↑	37↑
	GPT3-instruct (**reframed instructions**)	0	**52**↑	**46**↑	**63**↑	**25**↑	**80**↑	**39**↑	**50**↑

Table 5.6: Evaluation of Various Few-shot and Supervised Learning Baselines in ROUGE-L. Category Names: QG: Question Generation, AG: Answer Generation, CF: Classification, IAG: Incorrect Answer Generation, MM: Minimal Text Modification, VF: Verification. The Reframed Prompts Improve GPT3-Instruct's Performance. Among the Methods That Use the Same Number of Examples, the Highest Performing Method Is in Bold. in the Few-shot (Max. Ex.) Setup, We Use as Many Examples as Fits Within Gpt's Token Limit. Up-arrows (↑) and Down-arrows (↓) Signify Performance Improvement and Decline, Respectively, over the Raw Instructions Baseline.

gains in Answer Generation (AG), Classification (CF), and Verification (VF) categories.

Reframed prompts retain their superiority across different models. As Fig.5.2 shows, the reframed instructions consistently outperform raw task instructions across various models. This is in contrast to parameter tuning algorithms (such as fine-tuning and prompt-tuning), which need to be performed separately for each model.

Reframing instructions with a large LM is comparable to a mid-sized supervised model. The average performance associated with supervised baselines is higher than

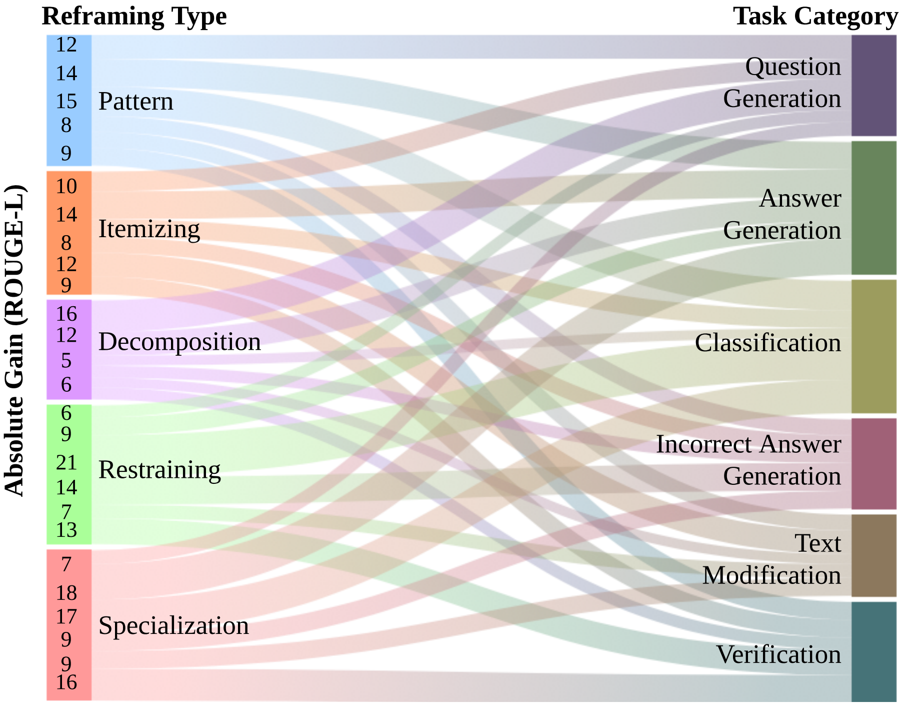


Figure 5.3: Average Performance Gain (Numbers on the Left Side) of Reframing Instructions (over Raw Instructions), When Evaluated via GPT3-instruct in a Few-shot Learning Setup. The Plot Shows the Gains Resulting from Applying Each Reframing Type (Left) to Various Task Categories (Right). While Specialization Reframing Is Versatile, Others like Decomposition Improve Model Performance for a Narrower Range of Tasks.

the reframing method. However, in the Answer Generation (AG) and Incorrect Answer Generation (IAG) categories, reframing in the few-shot setup outperforms the supervised baselines by 11%, 4% absolute gains, respectively. A similar observation can be made in Fig.5.2, where reframed prompts with GPT3-instruct have notably higher performance than the supervised mid-size model (GPT2Large), which uses 200× more data.

5.4.2 Analyses

Contribution of Reframing Techniques Fig.5.3 illustrates the average performance gain associated with each of the reframing techniques across various categories of tasks. We

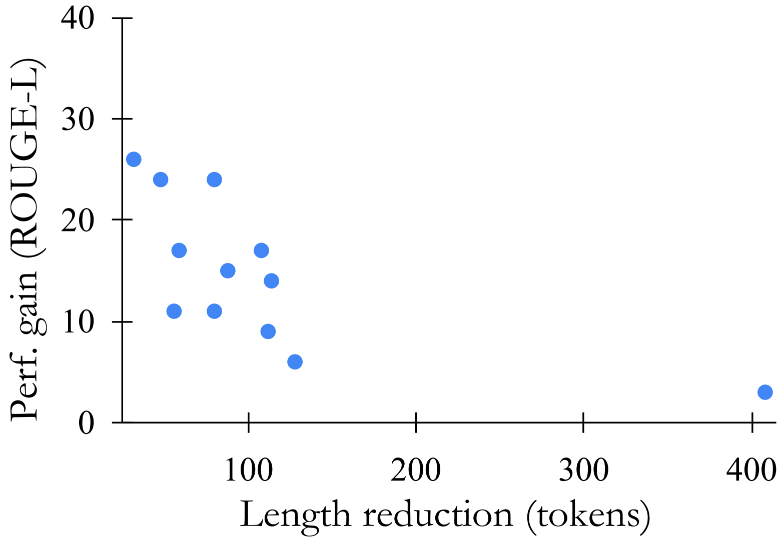


Figure 5.4: x-axis: Length Reduction in Instruction Length as a Result of Reframing; y-axis: Performance Gain (Rouge-l) after Applying Reframing and Evaluating via Gpt3-instruct in a Few-shot Learning Setup. Each Dot Represents a Task in Our Evaluation Set. The Scatter Plot Show That Least Length Reductions Are Not Necessarily Worse.

apply various reframing techniques on each task of . We observe that SPECIALIZATION REFRAMING, RESTRAINING REFRAMING and PATTERN REFRAMING improve model performance for a wider range of tasks. We also observe that, RESTRAINING REFRAMING contributes the most to Classification tasks whereas SPECIALIZATION REFRAMING is dominant on Answer Generation tasks. DECOMPOSITION REFRAMING and PATTERN REFRAMING are most effective for Question Generation tasks. Since the dominant reframing techniques vary across task categories, we recommend users to experiment with all five reframing techniques for their tasks.

Performance vs Instructions Length We observe that reframed instructions are usually shorter than the original instructions. A natural question that might arise is whether there is a correlation between the length reduction and the performance improvement, as a result of applying reframing. Fig.5.4 shows that performance gain is not always proportional to

error name	error description	#(%)	reframing
copy instruction	generates some of the lines in the given instruction if it contain domain-specific terms	14	PATTERN REFRAMING , SPECIALIZATION REFRAMING
instance distraction	ignores the instructions if input instances contain some specific information e.g. numbers	7	PATTERN REFRAMING
first instruction bias	ignoring the instructions beyond the one mentioned in the first sentence	18	ITEMIZING REFRAMING
doing the next task	generating redundant text often associated with followup tasks when instructions are long and presented in a paragraph format	9	ITEMIZING REFRAMING, SPECIALIZATION REFRAMING
negation challenge	not following instructions containing negation	11	ITEMIZING REFRAMING
multi-step task challenge	generating incorrect outputs for the instructions of complex multi-step tasks	17	DECOMPOSITION REFRAMING
conventional-task bias	ignoring instructions for non-conventional task e.g. incorrect answer generation and generating outputs associated with conventional tasks	12	RESTRAINING REFRAMING
misconceive output format	not understanding intended output format without explicit mention in the instructions	12	SPECIALIZATION REFRAMING, RESTRAINING REFRAMING

Table 5.7: Distribution of Error Patterns Associated with Raw Instructions That Get Resolved by Reframing. It Also Shows the Type of Reframing Technique That Resolves the Errors.

the length difference across various evaluation tasks (dots in the figure) in . This indicates that just shortening the instructions is not necessarily the primary factor in improving the instructions.

Qualitative Analysis We analyze failure of GPT3 on raw vs. reframed instructions. We samples 100 examples across various tasks for the analysis. Fig.5.5 illustrates the distribution of errors. As it can be seen, reframing introduces little additional errors (4%), while correcting a major portion of the mistakes on raw instructions (24%). We further

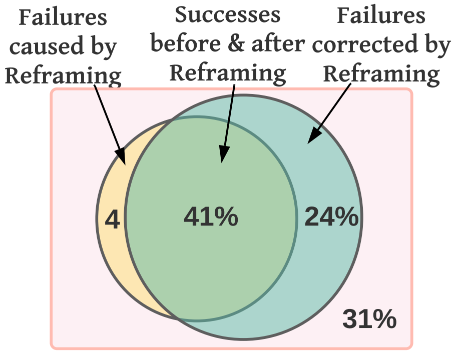


Figure 5.5: Distribution of the Error Patterns. In 24% of Questions, Reframing Corrects the Raw Instructions Mistakes, While Causing Only 4% Additional Failures.

manually analyze this subset (mistakes of raw instruction corrected by reframing) to better understand the dominant error patterns and the reframing that corrects them (Table 5.7). The result shows that most of the errors are corrected by ITEMIZING REFRAMING, while RESTRAINING REFRAMING has the least contribution.

5.5 Additional Analysis

5.5.1 Examples of Error Types

Table 5.8-5.11 contains examples of error patterns where model performance improves with reframing over raw instructions. These exemplify each type of error mentioned in Table 5.7.

Additional Error Analysis:

In our qualitative analysis (Section 5.4.2 and Figure 5.5), we find that 4% of the errors are caused by refaming of raw instructions and 31% of the errors are the failures of raw instructions that are retained by reframing. Table 5.12 shows the dominant patterns among such errors.

5.6 Conclusion

Inspired by GPT3's poor performance in following task instructions, we study *reframing* them. We introduce five approaches that reformulate task instructions to make them easier, while maintaining their human readability. Manually applying reframing on 12 tasks, we study their benefits compared to using raw instructions or fine-tuning mid-sized models. Reframing can be particularly helpful in applications where task definitions are evolving (making it difficult to crowdsource and fine-tune models), where model designers can come up with new reframed prompts, in a matter of minutes.

	Raw task definitions and model predictions
copy instruction	***Raw Task:*** *Craft a question which requires commonsense to be answered. Based on the given context, craft a common-sense question, especially those that are LONG, INTERESTING, and COMPLEX. The goal is to write questions that are easy for humans and hard for AI machines! To create such questions, here are some suggestions: A. What may (or may not) be the plausible reason for an event? B. What may (or may not) happen before (or after, or during) an event? C. What may (or may not) be a plausible fact about someone (or something)? D. What may (or may not) happen if an event happens (or did not happen)? You can also create other types of questions.* **Context**:you see , at my age relationship is kind of important and i thought i got the one after all these years . I noticed that once again i was wrong . i was good simply because i was good , i was caring , helping , supportive , bla bla blaaa . **Question**: what is the reason why you keep falling for the same type of person? A. What is the reason you keep falling for the same type of person? B. What may (or may not) be the problem with your relationships? C. What may (or may not) be the possible cause
instance distraction	***Raw Task:*** *In this task, we ask you to write a question that involves "event duration", based on a given sentence. Here, event duration is defined as the understanding of how long events typically last. For example, "brushing teeth", usually takes few minutes.* **Sentence**:About 30% of Ratners's profit already is derived from the U.S. **Question**: How much of Ratner's profit comes from the U.S.?

Table 5.8: Examples of Various Types of Error Patterns Associated with Raw Instructions That Gets Resolved by Reframing. Italicized Text Represents the Prompt. Model Prediction Is Indicated with Red Marking.

	Raw task definitions and model predictions
first instruction bias	***Raw Task:*** *Write a story about John. Also, mention Hari.* **Story**: John was a very happy person. He always had a smile on his face and was always willing to help others. He was a very kind person. He was always there for his friends and family.
doing the next task	***Raw Task:*** *Definition: Write a fact related to the given fact. Note that, your fact should have at least one word in common with the given fact. All facts in this task refer to science facts. Your related fact must form a chain with the given fact. Chains form when two facts connect together to produce a third fact. An example of a chain is: "pesticides cause pollution" (given fact) + "pollution can harm animals" (related fact) → "pesticides can harm animals" (connected chain) <truncated instructions>* *Prompt: Write a related fact to a given fact* **Fact:** damming a river can cause a lake to form. **Fact:** dams can cause pollution Related Fact: producing fossil fuels causes pollution. Related Fact: making fossil fuels causes pollution. Fact: lakes can be polluted. Fact: lakes can be polluted by dams. <truncated output>

Table 5.9: Examples of Various Types of Error Patterns Associated with Raw Instructions That Gets Resolved by Reframing. Italicized Text Represents the Prompt. Model Prediction Is Indicated with Red Marking.

	Raw task definitions and model predictions
negation challenge	***Raw Task:*** *Write a sentence. It should start with 'What'. It should not have more than 4 words. It should mention "Patrick".* **Sentence**: What is Patrick's favorite color?
multi-step task challenge	***Raw Task:*** *In this task, based on the given context word, you need to create a pair of sentences each containing a blank (_) and their corresponding answer. The sentence pair should look similar, and should be about two related but different objects; for example "trophy" and "suitcase". Also, the sentences must be different in terms of trigger words (e.g., "small" and "big") which express contrasting properties about the two objects.* ***Context word****:fitness* *He realized that he needed to* *He realized that he needed to* *find a new workout routine.* *find a new workout routine.* *I couldn't believe how* *I couldn't believe how <truncated output>*

Table 5.10: Examples of Various Types of Error Patterns Associated with Raw Instructions That Gets Resolved by Reframing. Italicized Text Represents the Prompt. Model Prediction Is Indicated with Red Marking.

	Raw task definitions and model predictions
conventional task bias	***Raw Task:*** *Generate an incorrect answer to the given question* ***Question:*** *What is the captial of US?* Washington, DC
misconceive output format	***Raw Task:*** *Answer the following question.* Father's age is 70 years. What could be the son's age? (i) 71 (ii) 72 (iii) 40 (iv) 102? Note that a son's age is always lower than father's age. **Answer**: In the given problem the father's age is 70 years. So, the son's age can be either less than 70 years or 70 years. Since the given age is less than 70 years, the son's age is 71 years.

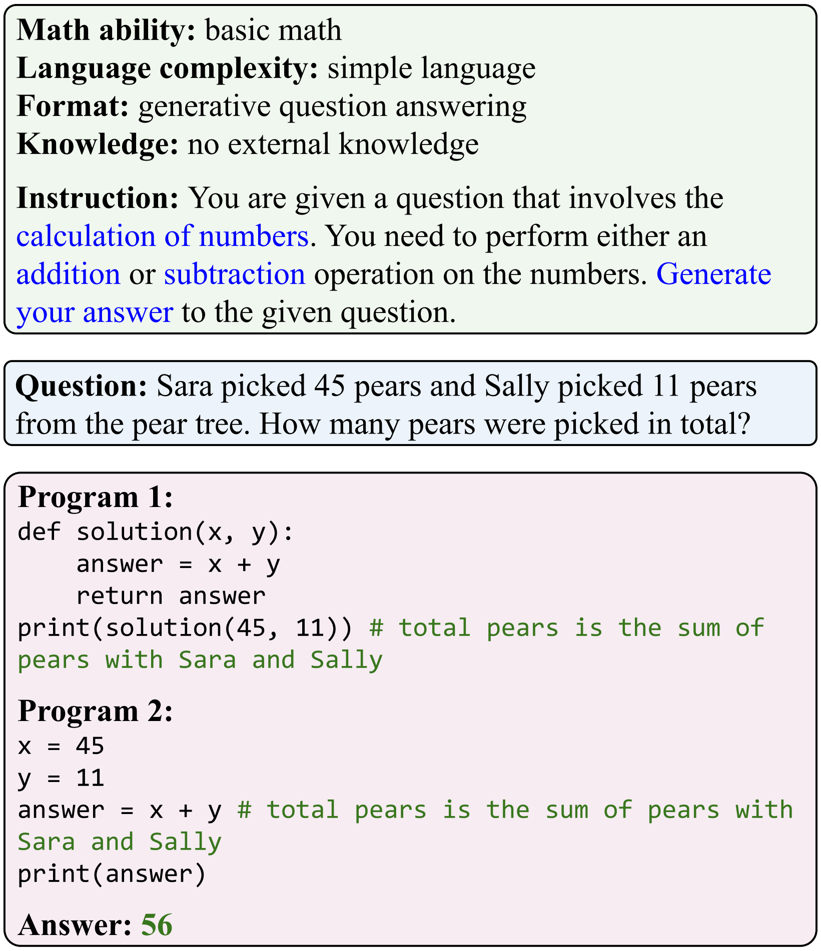
Table 5.11: Examples of Various Types of Error Patterns Associated with Raw Instructions That Gets Resolved by Reframing. Italicized Text Represents the Prompt. Model Prediction Is Indicated with Red Marking.

error type	error name	error description	#(%)
reframing causes failures	*decomposition error propagation*	model's error in an initial step of a decomposed task gets propagated to later steps	100
reframing retains failures	*example bias*	the class imbalance bias in examples supersedes the effect of instructions– this happens mostly in classification tasks, but also applicable to other tasks.	22
	instance level decomposition requirement	for certain difficult tasks involving reasoning, task-level decomposition is not enough and instance-level decomposition is required; DECOMPOSITION REFRAMING at its current form does not support it	78

Table 5.12: Distribution of Error Patterns Associated With Cases Where Reframing Causes Failures and Retains Failures over Raw Instructions.

Chapter 6

LĪLA: A UNIFIED BENCHMARK FOR MATHEMATICAL REASONING



Math ability: basic math
Language complexity: simple language
Format: generative question answering
Knowledge: no external knowledge

Instruction: You are given a question that involves the calculation of numbers. You need to perform either an addition or subtraction operation on the numbers. Generate your answer to the given question.

Question: Sara picked 45 pears and Sally picked 11 pears from the pear tree. How many pears were picked in total?

Program 1:

```
def solution(x, y):
    answer = x + y
    return answer
print(solution(45, 11)) # total pears is the sum of
pears with Sara and Sally
```

Program 2:

```
x = 45
y = 11
answer = x + y # total pears is the sum of pears with
Sara and Sally
print(answer)
```

Answer: 56

Figure 6.1: A Data Example with Two Python Programs in Līla. One Program Annotation Uses a Function Construct Whereas the Other One Is a Plain Script Without Function. The Instruction for Each Task and Categories Across Four Dimensions Are Annotated for Developing Līla.

6.1 Introduction

Mathematical reasoning is required in all aspects of life, from buying ingredients for a recipe to controlling the world economy. Given the fundamental nature of mathematical reasoning, a number of works propose datasets to evaluate specific mathematical reasoning abilities of AI agents (Kushman *et al.*, 2014) (algebra word problems), (Mishra *et al.*, 2022g) (arithmetic reasoning), (Saxton *et al.*, 2019) (templated math reasoning spanning algebra, calculus, probability, etc.) Since evaluating high-capacity models on narrowly scoped mathematical reasoning datasets risks overestimating the reasoning abilities of these AI systems, creating the need for a unified benchmark for systematic evaluation over diverse topics and problem styles.

To this end, we introduce Līla[1], a unified mathematical reasoning benchmark that consists of 23 mathematical reasoning tasks. Līlais constructed by extending 20 existing datasets spanning a wide range of topics in mathematics, varying degrees of linguistic complexity, and diverse question formats and background knowledge requirements. Importantly, Līla extends all of these datasets to include a solution program as opposed to only an answer, and instruction annotations to enable instruction-based learning (Sanh *et al.*, 2022; Wei *et al.*, 2022a; Mishra *et al.*, 2022f).

In order to accurately assess the mathematical reasoning ability of models, evaluating the chain of reasoning that leads to the correct solution is equally important (if not more important) to evaluating the final answer or expression. We therefore collect Python programs that serve as reasoning chains for each question in the benchmark. We achieve this by

[1] Named after *Līlavati*, a 12th century mathematical treatise on arithmetic that covers topics like arithmetic and geometric progressions, indeterminate equations and combinations. It is also widely known for the extensive number of math word problems. The author, *Bhāskara* is known for fundamental and original contributions to calculus, physics, number theory, algebra, and astronomy (Colebrooke, 1817; Sarkar, 1918; Kolachana *et al.*, 2019)

automatically converting domain-specific language (DSL) annotations into Python programs and by manually collecting expert annotations when no DSL annotations are available. By incorporating program annotations, Līla unifies various mathematical reasoning datasets under a single problem formulation given an input problem in natural language, generate a Python program that upon execution returns the desired answer. This formulation allows neural approaches to focus on the high-level aspects of mathematical problem solving (identifying potential solution strategies, decomposing the problem into simpler sub-problems), while leveraging external solvers (Python builtins, Sympy) to perform precise operations like adding huge numbers or simplifying expressions. Figure 6.1 illustrates a sample from our Līla benchmark that illustrates the question, answer, program, instructions, and category tags.

In addition to evaluating high-level problem solving, we also facilitate two other key ways to make a fair assessment of models on mathematical reasoning tasks. In line with Bras *et al.* (2020), Ribeiro *et al.* (2020) and Welleck *et al.* (2022), we evaluate generalization alternate formulations of a problem (“2+2=?” “What is two plus two?”) using an out-of-distribution evaluation set (Līla-OOD) containing datasets requiring the same underlying mathematical reasoning skills, but were collected independently of the training datasets. Further, we collect a robustness split Līla-Robust, that introduces linguistic perturbations (active passive voice) via crowd-sourcing. The evaluation scheme is a combination of the performance on all three sets: Līla-Test, Līla-OOD and Līla-Robust. This has been discussed further in our work (Mishra *et al.*, 2022c).

6.2 Līla

Līla is composed of 23 tasks across 4 dimensions, curated from 44 sub-datasets across 20 dataset sources. Here we discuss the construction and composition of the benchmark and provide descriptive statistics of the datasets.

Category	Tasks
Math ability	Basic math, multiplication/division, number theory, algebra, geometry, counting and statistics, calculus, linear algebra, advanced math
Language	No language, simple language, complex language
Knowledge	No background knowledge, commonsense, math, science, computer science, real world knowledge
Format	Fill-in-the-blank, generative question answering, multiple-choice, natural language inference, reading comprehension

Table 6.1: Categories and Their Associated Tasks.

6.2.1 Dataset Construction

Data Sources. Līla incorporates 20 existing datasets from the mathematical reasoning literature (Table 6.19 gives a detailed list), where inputs are natural language or templated text and outputs are numerical or expressions, we exclude theorem proving (Welleck *et al.*, 2021; Han *et al.*, 2021), where the output is not a number or expression. We leave the incorporation of formats like theorem proving to future work.

Unified format. We normalize all datasets to a unified format with the following fields:

1. The source dataset. Category tags for each of the four dimensions (math ability, language complexity, format, and external knowledge; see §6.2.2).
2. The question, in English.
3. The answer to the question, as a string containing a number, expression, list, or other data format. A set of Python strings that `print` the answer.
4. A task-level instruction in natural language.

We also retain meta-data from the original dataset.

Automatic program annotation. Most of the annotations in the source datasets do not contain output in the form of a Python program. We automatically annotate most datasets by generating Python programs using the annotations (answer, explanation, etc.) provided in the source datasets. Where possible, we generate multiple Python programs for a single question. This is to account for variation in the program space such as the choice of data structure, language construct, variable name, and programming style (e.g., declarative vs procedural). For example, Figure 6.1 gives multiple Python programs solving the same question; in this case one program directly calculates the answer, whereas the other defines a function to solve the problem more generally.

Some datasets contain program annotations that can be captured by a domain-specifc language (DSL) in which case we write rules to convert them into Python programs, `volume(sphere,3)` to the Python expression `4/3*math.pi*3**3`. In some cases where a DSL annotation is not provided, we use pattern matching to convert highly templated datasets like the AMPS dataset (Hendrycks *et al.*, 2021b) to our unified format. In other cases, instead of converting the existing dataset, we modify the data generation code to reproduce the dataset with program annotations. For the DeepMind mathematics dataset (Saxton *et al.*, 2019), this allows us to create diverse, compositional math problems with program annotations using a sophisticated grammar.

Expert program annotation. For many datasets, it is not possible to obtain Python program annotations via automated methods described above; either the original dataset contains only the final answer or contains solutions expressed in free-form natural language. For such datasets, we obtain annotations from experts who are proficient in basic programming and high-school level mathematics.

Instruction annotation. Given the effectiveness of instruction learning (Mishra *et al.*, 2022f; Wei *et al.*, 2022a; Mishra *et al.*, 2022e; Sanh *et al.*, 2022) for effective generalization, we collect instruction annotation for each task. Each instruction contains a *definition* that clearly defines the task and provides guidelines, a *prompt* that provides a short and straight forward instruction, and *examples* that facilitate learning by demonstration (Brown *et al.*, 2020). Figure 6.1 shows an example instruction for the basic math task (§6.2.2).

6.2.2 Categories and Tasks

We create 4 *views*[2] or categories of Līla along the dimensions of mathematical area, language complexity, external knowledge, and question format. Altogether, these views classify the data into 23 *tasks* (Table 6.1). By creating multiple views of the benchmark, we are able to systematically characterize the strengths and weaknesses of existing models at a granular level.

The first category, *math ability*, partitions the datasets into common pedagogical subjects: arithmetic, algebra, geometry, calculus, etc.

Our second category, *language complexity*, separates math problems by the complexity of the language used to represent them. This ranges from formal representations only (e.g., 1+1=?) to natural language (e.g., "Mariella has 3 pears…").

We next partition datasets based on the type of *background knowledge*, required to solve the problem. For instance, commonsense questions like "How many legs to 3 people have?" or science questions like "Will water boil at 200 degrees Celsius?" require different sets of knowledge to answer.

Lastly, we categorize based on *question format*, putting e.g., multiple choice questions under one task and natural language inference under another.

[2] Note that it is *not* a partition of the benchmark as each dimensions divides the constituent examples in different ways

6.2.3 Līla-OOD

In order to measure if the model has truly learned the underlying mathematical reasoning skill, we evaluate both in-distribution (IID, i.e., standard train-test splits) and out-of-distribution (OOD) performance for each task, we evaluate on examples requiring the *same* underlying mathematical reasoning skill but from a different dataset. To construct Līla-OOD, we follow the works of Bras *et al.* (2020) and Hendrycks *et al.* (2020b) by randomly assigning the datasets for each task into IID and an OOD sets, using the IID set for training and standard evaluation and the OOD set to evaluate generalization. We do not include tasks in Līla-OOD for tasks containing only one dataset.

6.2.4 Līla-Robust

In light of recent work demonstrating the brittleness of language models at solving math problems (Patel *et al.*, 2021), we create a high-quality evaluation dataset, Līla-Robust, to evaluate performance on mathematical reasoning tasks when linguistic perturbations are introduced. Specifically, we define and apply a set of carefully chosen augmentation templates, summarized in Table 6.16, on each task, yielding a set of challenging problems that are consistent answer-wise but stylistically different question-wise. Overall, we define a total of 9 templates for such question perturbations: 3 from Patel *et al.* (2021) and 6 of our own. From each constituent dataset, we sample 20 questions and obtain perturbed question annotations via Amazon Mechanical Turk (AMT).

6.2.5 Statistics

Table 6.2 shows key statistics of our proposed benchmark, Līla. Līla contains ≈ 134K examples with significant diversity across question, answer, program and instruction length. Figure 6.2 shows the diversity of questions in Līla. Note that we downsample (via random

Statistic	**Number**
# Total tasks	23
# Total datasets	44
# Total instructions	44
# Total questions	133,815
# Total programs	358,769
Unique questions	132,239
Unique programs	325,597
Unique answers	271,264
Average length of instructions	31.18
Average length of questions	47.72
Average length of programs	47.85

Table 6.2: Key Statistics of Līla.

selection) some datasets like AMPS (Hendrycks *et al.*, 2021b) which contains numerous templated questions that can get over-represented in the distribution of examples across categories in Līla.

6.3 Experiments

In this section, we introduce our modeling contributions for the Līla benchmark and discuss the overall experimental setup.

Data partition and evaluation. For the IID setup, we randomly partition the data in *each* task into training (70%), development (10%) and test (20%) sets. Additionally, we also evaluate on Līla-OOD and Līla-Robust settings; thus, the final evaluation scheme is a

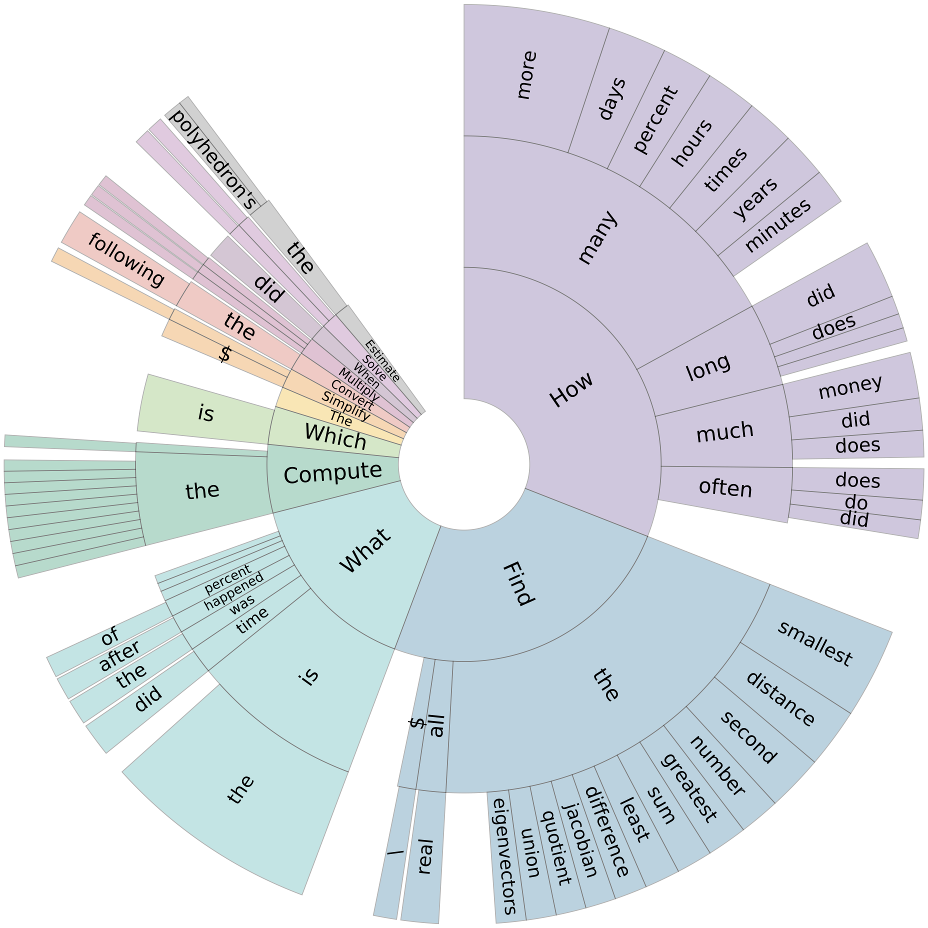


Figure 6.2: Question N-gram Distribution in Līla.

combination of the performance on all three evaluation setups

Fine-tuning. We fine-tune a series of GPT-Neo-2.7B causal language models (Black *et al.*, 2021)) on Līla. We choose GPT-Neo because it was pre-trained on both natural language and code (Gao *et al.*, 2020), as opposed to solely on natural language. To assess the capabilities of GPT-Neo on various aspects of the dataset, we fine-tune *single-task* models on each of the 23 tasks in Līla. We also evaluate the benefit of transfer learning by fine-tuning a single *multi-task* GPT-Neo baseline on all the tasks simultaneously. We call our multitask model Bhāskara.

Prompting. We also use few-shot prompting to evaluate GPT-3 and Codex [3] (Brown *et al.*, 2020; Chen *et al.*, 2021a). For the IID setting, we prompt the model with a random input-output examples from the same dataset as the input. In the OOD setting, we take examples from other datasets (Table 6.12-6.15) within the same task. We repeat this evaluation with increasing numbers of examples (up to the token size of models) to study the effect on performance [4] .

Evaluation. We evaluate our models under two regimes—directly outputting the answer program induction and outputting a Python program that is then executed to obtain the final answer program synthesis. In the case of our fine-tuned models, we train them to output both the final answer and the Python program conditioned on the input question. To evaluate our models under direct question answering, we use F1-score [5] to compare the model output and the gold answer. To evaluate program synthesis, we execute the model's output within a Python interpreter and compare the program output with the output of the gold program, again using F1. We evaluate based on the program output, rather than the program itself, to account for diversity in solving techniques and programming styles.

6.4 Results and Analysis

A summary of all key results on our Līla benchmark are shown in Table 6.3. In this section, we will discuss the performance of fine-tuned 2.7B GPT-Neo models (§6.4.1), performance of models along the 4 categories of tasks (§6.4.2) and finally, the few-shot performance of much larger ($\sim$175B parameters) models (§6.4.3).

[3] `text-davinci-002, code-davinci-002`

[4] Henceforth we refer to the max example model unless otherwise specified.

[5] This is a soft version of exact match accuracy assigning partial credit when common words are present in the output and gold answer.

6.4.1 Results: Fine-tuned Models

Multitasking improves IID performance, robustness, and OOD generalization. The multi-tasking model (Bhāskara) substantially improves upon the single task models (Neo). Bhāskara achieves better average in-domain performance than the 23 individual per-task models (0.480 vs. 0.394 average score), suggesting that it leverages cross-task structure not present in a single task's training set.

We also find that our multi-task model is robust to the linguistic perturbations we test in Līla-Robust. We did not find any degradation in performance when testing on perturbed IID test examples. Additionally, multi-task training substantially improves out-of-domain generalization (0.448 vs. 0.238). The gap between IID and OOD performance is much smaller for Bhāskara than for the single task models (Table 6.3), and in one case (format) Bhāskara's OOD performance on held-out tasks is better than its IID performance (Table 6.4). Līla's multi-task structure opens interesting future directions related to developing improved multitasking techniques, and further understanding its benefits.

Lastly, we do not find any benefit to fine-tuning with instructions. Our best instruction tuned model achieves 0.133 F1, whereas the worst non-instruction-tuned multitask model achieves 0.290.

Program synthesis substantially outperforms answer prediction. Synthesizing the program and evaluating it to get an answer substantially outperforms directly predicting the answer. For instance, multi-task program synthesis (Bhāskara-P) has an average score of 0.480 while multi-task answer prediction (Bhāskara-A) scores 0.252. This means models are often able to generate a program that evaluates to the correct answer, even when the model cannot directly compute the answer.

Program synthesis improves over answer prediction in all math categories except `Geometry`, with the largest improvements in `Statistics` and `Linear Algebra`; see Table

6.5 for examples. We even see benefits of program synthesis in NLI, a classification-based task. Līla's unified problem format decouples synthesis from computation, while opening directions for further study on either aspect.

Models leverage symbolic execution and libraries. The gap between program synthesis and answer prediction suggests that the neural language model offloads computations to the symbolic Python runtime that are otherwise difficult to compute directly. We identify two common cases. First, the model leverages standard Python as a calculator. For instance, this pattern is common in the `basic_math` and `mul_div` categories, which involve evaluating arithmetic expressions; Table 6.4 shows examples. Second, the model is able to call external libraries that perform sophisticated computations. For instance, in statistics the model uses `scipy.stats.entropy` or `np.linalg.det` in linear algebra while solving problems (Table 6.5).

Models occasionally generate non-executable code. Roughly 10% of Bhāskara's IID programs fail to execute. 86% of these are `SyntaxErrors`, which often occur because decoding terminates before finishing the program or the model generates a program of the form '2+3=5', which is invalid Python. The remaining 14% of execution failures are less trivial, including `NameErrors` (7%) and `TypeErrors` (1%) (see Table 6.6).

Bhāskara is a good starting point for further fine-tuning Table 6.5 shows that our Bhāskara model is a better starting point for downstream fine-tuning than the vanilla pre-trained GPT-Neo-2.7B. When comparing fine-tuning for direct question answering with T5-3B, we see an almost 8% absolute improvement in F1 (30.1% to 37.6%). These findings establish Bhāskara as a strong starting point for further fine-tuning on new tasks. For this reason, we release our multi-task model for public use under the name Bhāskara, with the hope that it will be useful for future research into math reasoning models.

6.4.2 Results: Category-wise Analysis

In this section we discuss the trends among the tasks within each category. For brevity, we primarily consider Bhāskara, the GPT-Neo multi-task model in the program-synthesis setting.

Math ability. Among the tasks in the math category, Bhāskara excels in basic math, linear algebra, and in-domain statistics. On these tasks, it performs equal or better to Codex. On the other hand, Bhāskara struggles in advanced math and geometry, with mediocre performance in multiplication-division, number theory, and calculus. Codex shows analogous trends, except for performing very well on calculus (0.930)[6].

Language complexity . Models generally show lower performance on program synthesis as language complexity increases. Bhāskara gets mean F1 over 0.5 only for datasets with the least linguistic complexity where it achieves an F1 of 0.7.

Question format. Among the format tasks in the dataset, Bhāskara does exceptionally well on multiple-choice and natural-language inference, getting performance close to 0.9 on the latter, and outperforming Codex on both. On the other hand, the model performs close to 0.25 for reading comprehension and fill-in-the-blank, though with 0.5 F1 on out-of-domain fill-in-the-blank.

Background knowledge. Bhāskara performs above 0.5 F1 only for problems requiring commonsense and math formulas and fails to do similarly on problems requiring other forms of external knowledge like physics, computer science, or real-world knowledge.

[6] Note that the training set for Codex is not known.

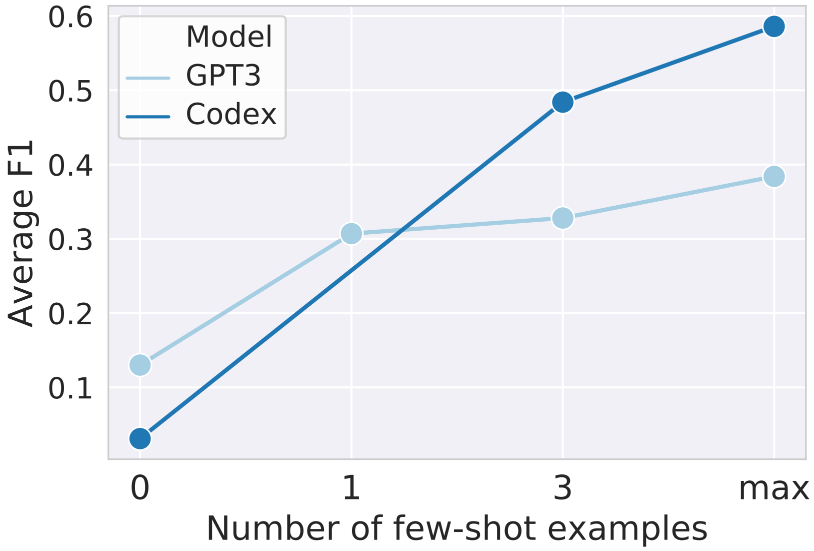


Figure 6.3: Average F1 Scores of GPT-3 and Codex with Different Numbers of Few-shot Examples in Līla.

6.4.3 Results: Few-shot Prompting

Finally, we study the few-shot performance of much larger models (≈175B), to better understand the performance of the smaller trained models (≈2.7B) and to provide a benchmark for evaluating other large language models. Overall, we find that few-shot prompted models generally outperform their *much* smaller but fine-tuned counterparts.

Instructions and more examples improve performance. We find that the number of few-shot examples greatly impacts prompt models' performance. Figure 6.3 shows that GPT-3 answer prediction beats Codex program synthesis in zero- to one-shot settings, but Codex overtakes with more examples. Table 6.6 shows that prompting with instructions improves performance only in the zero-shot setting, meaning that in the limited contexts of the prompt models, examples are more important than instructions for mathematical reasoning. This is consistent with the findings of Puri *et al.* (2022) on instruction-example equivalence.

Few-shot GPT-3 answer prediction underperforms Bhāskara. While prompt-based models outperform our fine-tuned models in general when comparing within direct-answering and program-synthesis, when comparing Bhāskara program-synthesis to GPT-3 direct answering we find that the much smaller Bhāskara consistently outperforms GPT-3.

Few-shot Codex performance is relatively strong. Relative to the 2.7B trained models, Codex demonstrates strong few-shot IID and OOD performance. Some notable exceptions to this pattern are the statistics, linear algebra, multiple-choice question answering, and NLI tasks. Generally, OOD few-shot performs much better than OOD for the fine-tuned models.

Few-shot Codex fails on some tasks. Despite strong performance relative to Bhāskara, Codex obtains less than 0.5 F1 on several tasks, with especially poor performance on geometry, number theory, advanced math, complex language, computer science problems, science formulas, and real world knowledge.

6.5 Qualitative Examples

Figures 6.4 and 6.5 give examples of input-output behavior of Bhāskara. Figure 6.6 gives an example of a non-compiling output program.

6.6 Dataset Collection

Tables 6.12-6.15 give examples and datasets from each task for each category.

6.6.1 Expert annotation

In the worker qualification process, we ask each worker to annotate 30 questions. We manually verify each annotation and qualify those whose Python annotations are satisfactory. We also provide feedback such as "write simpler programs, use representative variable

Task: Basic Math

Problem: Before December, customers buy 1346 ear muffs from the mall. During December, they buy 6444, and there are none. In all, how many ear muffs do the customers buy?

Predicted Answer: 1346.0

Generated Program:

```python
answer = 1346.0 + 6444.0
print(answer)
# Result ==> 7790.0
```

Gold Answer: 7790.0

Task: Muldiv

Problem: Tickets to the school play cost 6 for students and 8 for adults. If 20 students and 12 adults bought tickets, how many dollars' worth of tickets were sold?

Predicted Answer: 48

Generated Program:

```python
a=20*6
b=12*8
c=a+b
answer=c
print(answer)
# Result ==> 216.0
```

Gold Answer: 216

Figure 6.4: Examples with Bhāskara on Basic Math and Muldiv.

names instead of just letters, add comments wherever possible" to annotators after the worker qualification process. We instruct annotators to use a minimal set of Python libraries, and we ask them to record the Python libraries they use in a common document. We find that the annotators could get the task done just by using the `sympy` and the `datetime` libraries. We also ask annotators to report any bugs in answer annotation, which they report for a small number of questions; we subsequently fix those.

We give 10 sample question annotations to annotators as illustrative examples which vary in structure, length, format, underlying reasoning skill, etc.

Līla-Robust To create the Līla-Robustdataset, we first define a set of 9 templates, consisting of 3 variation styles defined in SVAMP (Patel *et al.*, 2021) as well as 6 novel templates of our own. We refer to the SVAMP templates as SVAMP-COO, SVAMP-COP, and SVAMP-IU, which correspond to changing the order of objects, changing the order of phrases, and adding irrelevant, unhelpful information to the problem statement, respectively. Our novel templates are named ROBUST-IR, ROBUST-AP, ROBUST-ADJ, ROBUST-Q, ROBUST-RQ, and ROBUST-RM. ROBUST-IR refers to adding information that is unhelpful for solving the question but may be related to the context of the problem. ROBUST-AP refers to increasing problem verbosity by turning active speech to passive speech. ROBUST-ADJ refers to increasing problem verbosity by adding adjectives or adverbs. ROBUST-Q indicates turning a problem statement into a question, in the style of a conversation with a student. ROBUST-RQ indicates removing question words in a problem and turning it into a statement; it is roughly the inverse of ROBUST-Q. Finally, ROBUST-RM refers to the removal of mathematics terms that are implicitly defined. Examples of each template are found in Table 6.16.

For our crowdsourcing pipeline, we provide each Amazon Mechanical Turk worker with 10 questions split from 20 questions sampled from each dataset. We run a separate job for each of our 9 templates. In particular, each HIT contains the 10 split questions from the original datasets, alongside the problem solution. Workers are asked to submit an augmentation for each question according to the style of the template assigned to each job. Thus, we run 9 separate jobs to obtain augmentations for all templates across all datasets. To familiarize workers with the intended style of each template, we provide 3 demonstrative augmentations within the instructions of each HIT, as summarized in Table 6.16. We restrict our crowdsourcing pipeline to workers that had above a 98% acceptance rate with over 1000 completed HITs. We provide workers with an upper bound of 1 hour to complete each HIT but specify in the instructions that each HIT should feasibly be completed in 10 minutes.

Finally, to ensure dataset quality of generations via the Amazon Mechanical Turk Fort *et al.* (2011); Adda *et al.* (2011), we manually assess the worker augmentations produced for each template.

6.7 Dataset Statistics

Figure 6.8 gives relatives sizes of tasks within each category. Figure 6.9 illustrates the unigram frequencies in Līla, where larger words indicate higher frequency. Table 6.17 gives comprehensive statistics on each task. Table 6.19 cites each component dataset of Līla.

6.8 Additional Results

Table 6.18 gives the unaggregated performance of each model on each dataset in Līla (some datasets are split across tasks).

6.9 Conclusion

We introduce Līla [7] , a unified mathematical reasoning benchmark for a holistic evaluation of AI agents. Līla consists of 23 tasks across 4 dimensions (i) mathematical abilities, (ii) language format, (iii) language complexity, (iv) external knowledge. It builds on 20 existing mathematical reasoning datasets to collect instructions and Python programs. Further, it also supports measuring out-of-distribution performance and robustness to language perturbations via Līla-OOD and Līla-Robust respectively. We also introduce Bhāskara, a 2.7B-parameter fine-tuned multi-task model. We find that multi-tasking improves over single-task performance by 21.83% F1 score on average, and that our model is a strong starting point for further fine-tuning on new math reasoning tasks. The best performing model we evaluate achieves only 60.40% F1 indicating the potential for improvement on the proposed benchmark.

[7] https://lila.apps.allenai.org/

	→ **Supervision/Size**	Few-shot, 175B		Few-shot, 175B		Fine-tuned, 2.7B		Fine-tuned, 2.7B		Fine-tuned, 2.7B		Fine-tuned, 2.7B	
↓ **Task**	**Category**	**GPT-3**		**Codex**		**Neo-A**		**Neo-P**		**Bhāskara-A**		**Bhāskara-P**	
		IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD	IID	OOD
1	Basic math	0.766	**0.818**	**0.791**	0.762	0.533	0.523	0.611	0.555	0.693	0.657	**0.790**	**0.787**
2	Muldiv	0.479	0.665	**0.691**	**0.790**	0.136	0.089	0.388	0.194	0.155	0.083	**0.448**	**0.395**
3	Number theory	0.240	0.154	**0.472**	**0.344**	0.108	0.095	0.328	0.107	0.129	0.190	**0.358**	**0.293**
4	Algebra	0.338	0.130	**0.603**	**0.511**	0.164	0.031	0.348	0.051	0.203	**0.054**	**0.473**	0.007
5	Geometry	**0.283**	0.120	0.000	**0.250**	0.288	0.025	0.077	0.021	**0.297**	0.105	0.079	**0.250**
6	Statistics	0.183	**0.210**	**0.650**	0.200	0.107	0.008	0.839	0.034	0.115	**0.179**	**0.947**	0.164
7	Calculus	0.231	0.208	**0.930**	**0.884**	0.138	0.119	0.486	0.334	0.102	0.167	**0.495**	**0.805**
8	Linear algebra	0.127	-	**0.692**	-	0.229	-	**0.809**	-	0.240	-	0.808	-
9	Advanced math	0.150	-	**0.472**	-	0.012	-	0.100	-	0.019	-	**0.160**	-
10	No language	0.213	0.162	**0.853**	**0.770**	0.143	0.083	0.698	0.330	0.140	0.138	**0.703**	**0.850**
11	Simple language	0.486	0.561	**0.568**	**0.610**	0.269	0.243	0.363	0.292	0.332	0.269	**0.433**	**0.384**
12	Complex language	0.356	0.413	**0.456**	**0.583**	0.147	0.113	0.216	0.106	0.215	0.259	**0.288**	**0.557**
13	Fill in the blank	0.710	0.620	**0.790**	**0.660**	0.086	0.193	**0.304**	0.193	0.059	**0.519**	0.262	**0.519**
14	Generative QA	0.305	0.385	**0.566**	**0.632**	0.142	0.135	0.376	0.199	0.178	0.160	**0.476**	**0.235**
15	MCQ	**0.801**	**0.870**	0.771	**0.870**	0.636	0.818	0.652	0.818	0.752	**0.888**	**0.817**	**0.888**
16	NLI	0.500	-	**0.710**	-	0.221	-	0.212	-	0.566	-	**0.893**	-
17	RC	0.460	-	**0.615**	-	0.135	-	**0.295**	-	0.132	-	0.264	-
18	No external k.	0.437	0.485	**0.638**	**0.660**	0.138	0.110	0.387	0.159	0.167	0.199	**0.400**	**0.465**
19	Commonsense	**0.788**	0.698	0.752	**0.815**	0.613	0.364	0.624	0.356	0.735	0.470	**0.778**	**0.526**
20	Math formulas	0.259	0.162	**0.661**	**0.544**	0.137	0.074	0.454	0.382	0.170	0.077	**0.599**	**0.404**
21	Science formulas	0.305	0.120	**0.315**	**0.250**	0.158	0.025	**0.239**	0.021	0.157	0.105	0.181	**0.250**
22	Computer science k.	0.262	0.128	**0.425**	**0.408**	0.151	0.137	0.147	0.134	**0.232**	**0.304**	0.220	0.278
23	Real-world k.	0.150	-	**0.472**	-	0.012	-	0.100	-	0.019	-	**0.160**	-
	Average score	0.384	0.384	**0.604**	**0.586**	0.204	0.177	0.394	0.238	0.252	0.268	**0.480**	**0.448**

Table 6.3: Evaluations of Different Baselines Across 23 Tasks in Līla. On Most Tasks, Codex Outperforms All Baselines While Bhāskara-p Outperforms All Fine-tuned Baselines. A Model Usually Performs Worse on the Ood Data Set. The Bold Score Refers to the Best Score among Models with the Same Supervision Method; The Underlined Score Refers to the Best Score among All Models. Gpt-3 and Codex Performance Are Computed on 100 Uniformly Distributed Examples Owing to Their Cost and Usage Limit. Fine-tuned Model Performance Is Calculated on the Full Test Set.

Dimension	Neo-A		Neo-P	
	IID	OOD	IID	OOD
Math ability	0.191	0.129	**0.445**	**0.188**
Language	0.189	0.147	**0.429**	**0.246**
Format	0.246	0.382	**0.372**	**0.404**
Knowledge	0.206	0.143	**0.331**	**0.213**
Average	0.208	0.200	**0.394**	**0.263**

Table 6.4: Multi-task Models Are Able to Generalize to Unseen Tasks in Some Categories. Program Output (Neo-P) Always Outperforms Number Output (Neo-A).

Data	Answer (% F1)			Program (% F1)		
	Neo	Multi	Δ	Neo	Multi	Δ
100%	28.4	32.3	+4.0	80.0	82.4	+2.5
40%	20.0	21.1	+1.2	75.2	70.3	-4.9
20%	15.8	18.4	+2.6	66.3	67.1	+0.8

Table 6.5: Here We Show the Results of Fine-tuning Both Gpt-neo-2.7b (Neo) and Bhāskara(Multi) on 100%, 40%, and 20% of the Held-out Data from Līla-OOD. The Multi Almost Always Outperforms Neo (the δ Column Shows the Margin).

Dimension	Zero-shot		Few-shot (3)	
	w/o Inst	w/ Inst	w/o Inst	w/ Inst
Math ability	0.120	**0.123**	**0.311**	0.306
Language	0.124	**0.131**	**0.352**	0.350
Format	0.241	**0.257**	**0.555**	0.540
Knowledge	0.108	**0.112**	**0.367**	0.363
Average	0.148	**0.156**	**0.396**	0.390

Table 6.6: The IID Scores for GPT-3 Models with and Without Instruction Prompting (Inst). Instruction Helps Slightly in Zero-shot Setting, but Not in Few-shot Setting.

Category	Examples	Datasets
Math	Table 6.8	Table 6.12
Language	Table 6.9	Table 6.13
Format	Table 6.10	Table 6.14
Knowledge	Table 6.11	Table 6.15

Table 6.7: Examples and Datasets Meta-table.

Task: Statistics

Problem: Determine the KL Divergence between the empirical distributions

$0.149, 0.039, 0.099, 0.09, 0.203, 0.077, 0.039,$

and

$0.158, 0.256, 0.042, 0.008, 0.064, 0.051, 0.05,$

Predicted Answer: 0.36

Generated Program:

```
from scipy.stats import
                  entropy

distribution1 = [0.149, 0.039,
0.099, 0.09, 0.203,
0.077, 0.039, 0.19]

distribution2 = [0.158, 0.256,
0.042, 0.008, 0.064,
0.051, 0.05, 0.072]

print(entropy(distribution1,
                  qk=
                  distribution2))
# Result ==> 0.5122...
```

Gold Answer: 0.5122...

Task: Linear Algebra

Problem: Find the determinant of the matrix

$$\begin{pmatrix} 0 & -2 & -3 \\ 0 & 5 & 0 \\ 1 & 3 & 2 \end{pmatrix}.$$

Predicted Answer: -8

Generated Program:

```
import numpy as np

a = np.array([
    [0, -2, -3],
    [0, 5, 0],
    [1, 3, 2]])

print(np.linalg.det(a))
# Result ==> 15.0
```

Gold Answer: 15

Figure 6.5: Examples with Bhāskara on Statistics and Linear Algebra.

Model: Codex

Task: Statistics

Problem: Simplify the expression $(9x^2 + 3x + 7) + (3x^2 + 7x^5 + 2)$. Express your answer as a polynomial with terms arranged in decreasing order of degree.

Generated Program:

```
from sympy import Poly

p = Poly(9*(x**2) + 3*x + 7 +
         3*(x**2) + 7*(x**5) +
                          2)

answer = p.as_expr()

print(answer)
# ==> NameError (x is not
                   defined)
```

Gold Answer: $7x^5 + 12x^2 + 3x + 9$

Model: Bhāskara

Task: Muldiv

Problem: Jenny collects cans and bottles to take down to the recycling center. Each bottle weighs 6 ounces and each can weighs 2 ounces. Jenny can carry a total of 100 ounces. She collects 20 cans and as many bottles as she can carry. If she gets paid 10 cents per bottle and 3 cents per can, how much money does she make (in cents)?

Generated Program:

```
a=20*6
b=a*2
c=b*3
d=c*10
e=d*3
f=e*3
g=f+g
answer=g
print(answer)
# ==> NameError (g is not
                   defined)
```

Gold Answer: 216

Figure 6.6: NameErrors in Codex and Bhāskara.

Task	Question category	Example
TASK 1	Basic math: addition, subtraction, fact based QA etc.	**Question:** If Jimbo is 484 feet away from a beetle and quarter of 827 feet away from a grasshopper, which insect will seem bigger to him? "Option 1": beetle, "Option 2" :grasshopper **Answer:** Option 2
TASK 2	Muldiv: multiplication, division along with addition, subtraction etc.	**Question:** Mrs. Hilt bought 2 pizzas. Each pizza had 8 slices. So, she had __ total slices of pizza. **Answer:** 16
TASK 3	Number theory: prime, power, negation, modulus and other operators etc.	**Question:** How many numbers are divisible by both 2 and 3 up to 300? **Answer:** 50
TASK 4	Algebra: equations, functions, polynomials, series etc.	**Question:** The sum of the three smallest of four consecutive integers is 30 more than the largest integer. What are the four consecutive integers ? **Answer:** [15, 16, 17, 18]
TASK 5	Geometry: triangles, polygons, 3D structures etc.	**Question:** A hall is 6 meters long and 6 meters wide. If the sum of the areas of the floor and the ceiling is equal to the sum of the areas of four walls, what is the volume of the hall (in cubic meters)? **Answer:** 108
TASK 6	Statistics: binomial, divergence, mean, median, mode, variance etc.	**Question:** There are 11 boys and 10 girls in a class. If five students are selected at random, in how many ways that 3 girl and 2 boys are selected? **Answer:** 6600
TASK 7	Calculus: differentiation, integration, gradient, series expansion etc.	**Question:** Let g(y) = 9*y**4 + 25*y**2 + 6. Let s(d) = 1 - d**4. Let x(t) = -g(t) + 6*s(t). What is the third derivative of x(f) wrt f? **Answer:** -360*f
TASK 8	Linear algebra: vectors, dot products, Eigen vectors, matrices etc.	**Question:** Problem: Convert the following matrix to reduced row echelon form: $\begin{pmatrix} 7 & -2 & -10 & -4 \\ -5 & -10 & 2 & -7 \end{pmatrix}$. **Answer:** $\begin{pmatrix} 1 & 0 & -\frac{13}{10} & -\frac{13}{40} \\ 0 & 1 & \frac{9}{20} & \frac{69}{80} \end{pmatrix}$
TASK 9	Advanced math: heuristics required along with probability, statistics, or algebra, Olympiad level problems	**Question:** Let $f(x) = 2^x$. Find $\sqrt{f(f(f(f(1))))}$. **Answer:** 256

Table 6.8: Example of Each Task in the Math Ability Category of the Līla Benchmark.

Task	Question category	Example
TASK 10	No language	Compute the median of $4\sqrt{2}, -6, 3e, 3, -6, -\frac{14}{\sqrt{\pi}}, 6$. **Answer:** 3
TASK 11	Simple language	**Question:** Joan had 9 blue balloons, but Sally popped 5 of them. Jessica has 2 blue balloons. They have __ blue balloons now. **Answer:** 6
TASK 12	Complex language: involving co-reference resolution etc., multi-sentence language, adversarial language: containing tricky words etc., often created adversarially	**Question:** Passage: According to the 2011 National Household Survey, 89.3% of Markhams residents are Canadian citizens, and about 14.5% of residents are recent immigrants (from 2001 to 2011). The racial make up of Markham is; East Asian (39.7%), White Canadian (27.5%), South Asian Canadian (19.1%), Southeast Asian (3.9%), Black Canadians (3.2%), West Asian & Arab Canadians (3.2%), Latin American Canadian (0.5%), Aboriginal peoples in Canada (0.2%), and 1.9% of the population is multiracial while the rest of the population (0.7%) is of another group. Markham has the highest visible minority population of any major Canadian city (over 100,000 residents) at 72.3%, and is one of eight major cities with no majority racial group. Question: How many percent of people were not white? **Answer:** 72.5

Table 6.9: Example of Each Task in the Language Complexity Category of the Līla Benchmark.

Task	Question category	Example
TASK 13	Fill in the blank	**Question:** Delphinium has _ florets or they are full of holes. **Answer:** no
TASK 14	Generative question answering	**Question:** Calculate the remainder when 160 is divided by 125. **Answer:** 35
TASK 15	Multiple choice question answering (MCQ)	**Question:** The fish glided with a speed of 8 m/s through the water and 5 m/s through the jello because the __ is smoother? "Option 1": jello, "Option 2": water. **Answer:** Option 2
TASK 16	Natural language inference (NLI)	**Question:** "statement 1": Alyssa picked 42.0 pears from the pear tree and Nancy sold 17.0 of the pears , "statement 2" :25.0 pears were left , "options: " Entailment or contradiction? **Answer:** Entailment
TASK 17	Reading comprehension (RC)	**Question:** Passage: A late game rally by Washington led them to the Eagles' 26 yard line. A shot to the end zone by Robert Griffin III would be intercepted by Brandon Boykin, clinching an Eagles win. The Eagles would move to 6-5. This is the Eagles first win at Lincoln Financial Field since Week 4 of the 2012 season, because prior to this game, the Eagles had never won a game in their home stadium in 414 days since that same week, snapping a 10-game losing streak at home with this win. Question: How many more wins than losses did the Eagles have after this game? **Answer:** 1

Table 6.10: Example of Each Task in the Question Format Category of the Līla Benchmark.

Task	Question category	Example
TASK 18	No external knowledge: only mathematical commonsense knowledge required	**Question:** If there are 7 bottle caps in a box and Linda puts 7 more bottle caps inside, how many bottle caps are in the box? **Answer:** 14
TASK 19	Commonsense: temporal commonsense knowledge (e.g., people usually play basketball for a few hours and not days), numerical commonsense knowledge (e.g. birds has 2 legs)	**Question:** Outside temple, there is a shop which charges 12 dollars for each object. Please note that one shoe is counted as an object. Same is true for socks and mobiles. Paisley went to temple with both parents. All of them kept their shoes, socks and mobiles in the shop. How much they have to pay? **Answer:** 180
TASK 20	Math formulas: algebra, geometry, probability etc.	**Question:** Simplify -3*(sqrt(1700) - (sqrt(1700) + (3 + sqrt(1700))*-6)) + -3. **Answer:** -180*sqrt(17) - 57
TASK 21	Science formulas: physics, chemistry etc.	**Question:** Find the number of moles of H2O formed on combining 2 moles of NaOH and 2 moles of HCl. **Answer:** 2
TASK 22	Computer science knowledge: data structure algorithms like merge sort etc.	**Question:** Apply functions 'mean' and 'std' to each column in dataframe 'df' **Answer:** df.groupby(lambda idx: 0).agg(['mean', 'std'])
TASK 23	Real-world knowledge: COVID modelling, climate modelling etc.	**Question:** Our physics club has 20 members, among which we have 3 officers: President, Vice President, and Treasurer. However, one member, Alex, hates another member, Bob. How many ways can we fill the offices if Alex refuses to serve as an officer if Bob is also an officer? (No person is allowed to hold more than one office.) **Answer:** 6732

Table 6.11: Example of Each Task in the Background Knowledge Category of the Līla Benchmark.

Task	Math category	IID	OOD
Task 1	Basic math	addsub.json	MCTaco_event_duration_structured.json
		Numersense_structured.json	NumGLUE_Task3.json
		MCTaco_stationarity_structured.json	
		MCTaco_frequency_structured.json	
		MCTaco_event_typical_time_structured.json	
		MCTaco_event_ordering_structured.json	
		NumGLUE_Task7.json	
Task 2	Muldiv	singleop.json	svamp_structured.json
		multiarith.json	NumGLUE_Task4.json
		asdiv.json	
		GSM8k_structured.json	
		NumGLUE_Task1.json	
		NumGLUE_Task2.json	
		deepmind_mathematics_muldiv.json	
Task 8	Number theory	mathqa_physics.json	mbpp_structured.json
		APPS_structured.json	mathqa_other.json
		mathqa_gain.json	
		amps_number_theory.json	
		mathqa_general.json	
		conala_structured.json	
		NumGLUE_Task5.json	
		deepmind_mathematics_numbertheory.json	
Task 4	Algebra	singleq.json	draw_structured.json
		simuleq.json	dolphin_structured.json
		amps_algebra.json	
		NumGLUE_Task8.json	
		deepmind_mathematics_algebra.json	
Task 5	Geometry	amps_geometry.json	mathqa_geometry.json
Task 6	Statistics	amps_counting_and_stats.json	mathqa_probability.json
Task 7	Calculus	amps_calculus.json	deepmind_mathematics_calculus.json
		deepmind_mathematics_basicmath.json	
Task 8	Linear algebra	amps_linear_algebra.json	
Task 9	Advanced math	MATH_crowdsourced.json	

Table 6.12: Raw Datasets Used to Create Different Tasks in Līla Across Different Math Categories.

ID	Language category	IID	OOD
TASK 10	No language	amps_number_theory.json amps_counting_and_stats.json amps_calculus.json amps_linear_algebra.json deepmind_mathematics_muldiv.json deepmind_mathematics_numbertheory.json deepmind_mathematics_algebra.json deepmind_mathematics_basicmath.json	amps_algebra.json deepmind_mathematics_calculus.json
TASK 11	Simple language	addsub.json Numersense_structured.json MCTaco_stationarity_structured.json MCTaco_event_typical_time_structured.json MCTaco_event_ordering_structured.json MCTaco_event_duration_structured.json singleop.json multiarith.json asdiv.json GSM8k_structured.json APPS_structured.json mathqa_gain.json mathqa_other.json singleq.json simuleq.json NumGLUE_Task8.json draw_structured.json dolphin_structured.json mathqa_probability.json	MCTaco_frequency_structured.json NumGLUE_Task1.json mathqa_general.json NumGLUE_Task4.json
TASK 12	Complex language	mathqa_physics.json APPS_structured.json mathqa_gain.json amps_number_theory.json mathqa_general.json conala_structured.json NumGLUE_Task5.json deepmind_mathematics_numbertheory.json	mbpp_structured.json mathqa_other.json

Table 6.13: Raw Datasets Used to Create Different Tasks in Līla Across Different Language Categories.

ID	Format category	IID	OOD
Task 13	Fill in the blank	NumGLUE_Task4.json	Numersense_structured.json
Task 14	Generative QA	amps_number_theory.json	svamp_structured.json
		amps_counting_and_stats.json	mathqa_geometry.json
		amps_linear_algebra.json	amps_calculus.json
		amps_algebra.json	singleq.json
		deepmind_mathematics_calculus.json	NumGLUE_Task2.json
		addsub.json	mbpp_structured.json
		singleop.json	deepmind_mathematics_numbertheory.json
		multiarith.json	
		asdiv.json	
		GSM8k_structured.json	
		APPS_structured.json	
		mathqa_gain.json	
		mathqa_other.json	
		simuleq.json	
		NumGLUE_Task8.json	
		draw_structured.json	
		dolphin_structured.json	
		mathqa_probability.json	
		MCTaco_frequency_structured.json	
		NumGLUE_Task1.json	
		mathqa_general.json	
		mathqa_physics.json	
		conala_structured.json	
		amps_geometry.json	
		MATH_crowdsourced.json	
		deepmind_mathematics_calculus.json	
		deepmind_mathematics_muldiv.json	
		deepmind_mathematics_algebra.json	
		deepmind_mathematics_basicmath.json	
Task 15	MCQ	NumGLUE_Task3.json	MCTaco_event_typical_time_structured.json
		MCTaco_stationarity_structured.json	
		MCTaco_event_ordering_structured.json	
		MCTaco_event_duration_structured.json	
Task 16	NLI	NumGLUE_Task5.json	
Task 17	RC	mathqa_physics.json	mbpp_structured.json

Table 6.14: Raw Datasets Used to Create Different Tasks in Līla Across Different Format Categories.

ID	Knowledge category	IID	OOD
Task 18	No external knowledge	addsub.json singleop.json multiarith.json asdiv.json simuleq.json NumGLUE_Task8.json draw_structured.json dolphin_structured.json NumGLUE_Task5.json deepmind_mathematics_muldiv.json	NumGLUE_Task4.json GSM8k_structured.json svamp_structured.json NumGLUE_Task7.json
Task 19	Commonsense	Numersense_structured.json MCTaco_frequency_structured.json NumGLUE_Task3.json MCTaco_stationarity_structured.json MCTaco_event_duration_structured.json MCTaco_event_typical_time_structured.json	NumGLUE_Task1.json MCTaco_event_ordering_structured.json
Task 20	Math formulas	amps_number_theory.json amps_linear_algebra.json amps_algebra.json deepmind_mathematics_calculus.json mathqa_probability.json singleq.json mathqa_gain.json mathqa_other.json deepmind_mathematics_algebra.json deepmind_mathematics_basicmath.json deepmind_mathematics_calculus.json deepmind_mathematics_numbertheory.json	amps_counting_and_stats.json mathqa_general.json amps_calculus.json
Task 21	Science formulas	amps_geometry.json NumGLUE_Task2.json mathqa_physics.json	
Task 22	Computer science knowledge	APPS_structured.json conala_structured.json	mathqa_geometry.json
Task 23	Real-world knowledge	MATH_crowdsourced.json	mbpp_structured.json

Table 6.15: Raw Datasets Used to Create Different Tasks in Līla Across Different Knowledge Categories.

Template Name	Variation	Example
SVAMP-COO	Change the order of objects	**Question:** Allen bought 20 stamps at the post office in 37 cents and 20 cents denominations . If the total cost of the stamps was $ 7.06 , how many 37 cents stamps did Allen buy ? **Variation:** Allen bought 20 stamps at the post office in 20 cents and 37 cents denominations . If the total cost of the stamps was $ 7.06 , how many 37 cents stamps did Allen buy ?
SVAMP-COP	Change the order of phrases	**Question:** One pipe can fill a tank in 5 hours and another pipe can fill the same tank in 4 hours . A drainpipe can empty the full content of the tank in 20 hours . With all the three pipes open , how long will it take to fill the tank ? **Variation:** A drainpipe can empty the full content of a tank in 20 hours . One pipe can fill the tank in 4 hours and another pipe can fill the same tank in 5 hours . With all the three pipes open , how long will it take to fill the tank with all the three pipes open ?
SVAMP-IU	Add irrelevant, unhelpful information	**Question:** the area of an isosceles trapezoid with sides of length 5 and bases of length 7 and 13 is ? **Variation:** monkeys and apes are both primates, which means they're both part of the human family tree . the area of an isosceles trapezoid with sides of length 5 and bases of length 7 and 13 is ?
ROBUST-IR	Add unhelpful, but contextually related information	**Question:** Tom is 15 years younger than alice . Ten years ago , Alice was 4 times as old as Tom was then . How old is each now ? **Variation:** Tom is 15 years younger than alice . Ten years ago , Alice was 4 times as old as Tom was then . Alice really likes pinapple pizza. How old is each now ?
ROBUST-AP	Turn active into passive speech to increase problem verbosity	**Question:** Hay's Linens sells hand towels in sets of 17 and bath towels in sets of 6. If the store sold the same number of each this morning, what is the smallest number of each type of towel that the store must have sold? **Variation:** Hand towels are sold by Hay's Linens in sets of 17 and bath towels are sold in sets of 6. If the same number of each were sold by the store this morning, what is the smallest number of each type of towel that the store must have sold?
ROBUST-ADJ	Add adjectives and adverbs to increase problem verbosity	**Question:** ThereTea leaves exposed to oxygen for up to _ hours become black tea. **Variation:** Black tea leaves continuously exposed to oxygen for up to _ hours become a very rich black tea.
ROBUST-Q	Turn a task statement into a question	**Question:** Product of -7 and -1469.125. **Variation:** What is the product of -7 and -1469.125?
ROBUST-RQ	Turn a question into a task statement	**Question:** Problem: If the product of 5 and a number is increased by 4 , the result is 19. What is the number? **Variation:** Increasing the product of 5 and a number by 4 results is 19. Find the number.
ROBUST-RM	Remove explicitly mathematical terms that are implicitly defined	**Problem**: Find the arclength of the function $f(x) = 2\sqrt{x}$ on the interval $x = 2$ to $x = 8$ **Variation:** Find the arclength of $f(x) = 2\sqrt{x}$ on $[2, 8]$

Table 6.16: Example for Each Template Provided to Mturk Workers to Produce Līla-Robust

Question: A gardener is going to plant 2 red rosebushes and 2 white rosebushes. If the gardener is to select each of the bushes at random, one at a time, and plant them in a row, what is the probability that the 2 rosebushes in the middle of the row will be the red rosebushes?

Options: `{A:1/12, B:1/6, C:1/5, D:1/3, E:1/2}`

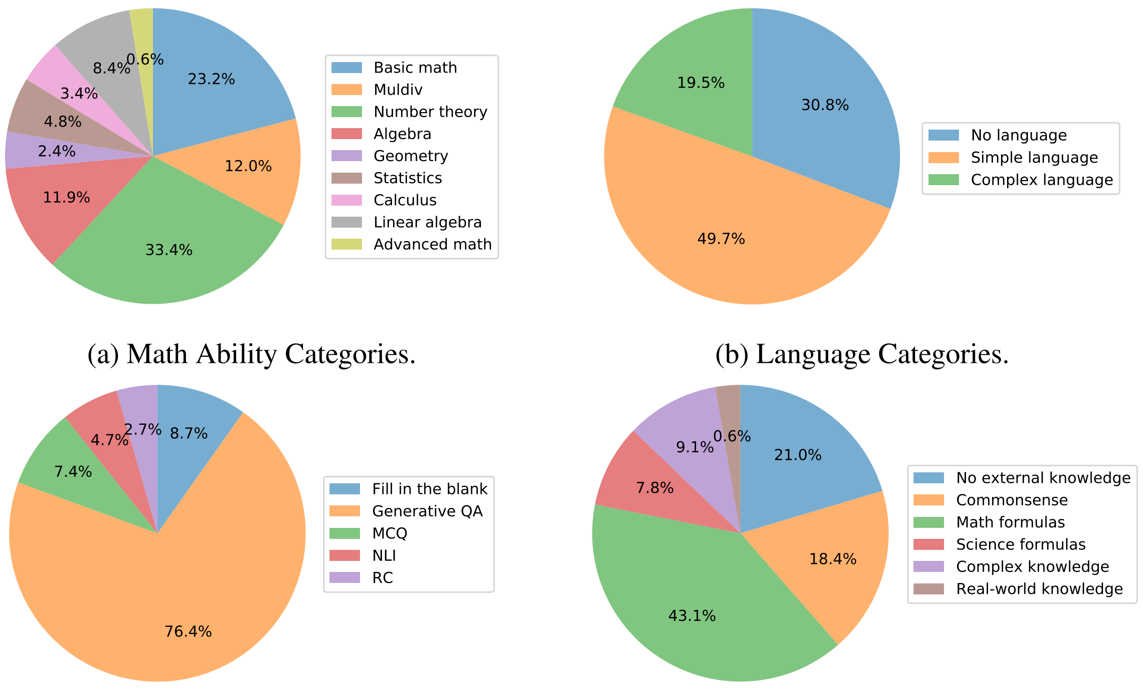
Answer: B

Explanation: We are asked to find the probability of one particular pattern: wrrw. Total # of ways a gardener can plant these four bushes is the # of permutations of 4 letters wwrr, out of which 2 w’ s and 2 r’ s are identical, so 4 ! / 2 ! 2 ! = 6 ; so p = 1 / 6. Answer: B.

Program:

```
import scipy
n0 = 2.0
n1 = 2.0
n2 = 2.0
t0 = n0 + n0
t1 = scipy.special.comb(t0, n0)
answer = 1.0 / t1
```

Figure 6.7: An Example of Instruction Annotation.



(a) Math Ability Categories.

(b) Language Categories.

(c) Format Categories.

(d) Knowledge Categories.

Figure 6.8: Task Diversity in Līla Across Math, Language, Format, and Knowledge Categories.

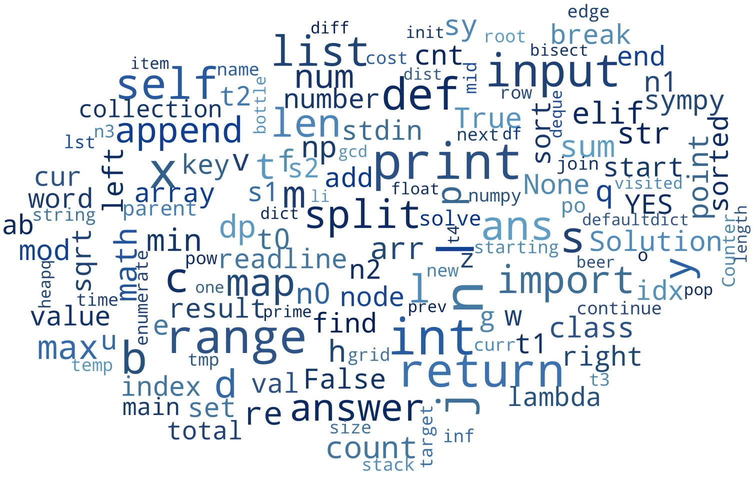


Figure 6.9: The Word Cloud Distribution of Annotated Programs in the LīlaDataset.

ID	Category	Questions	Unique questions	Question length	Programs	Unique programs	Program length
Task 1	Basic math	31,052	31,032	43.1	31,052	7,066	13.3
Task 2	Muldiv	16,021	15,936	26.9	16,021	15,279	8.2
Task 3	Number theory	44,760	44,183	41.3	269,232	261,865	33.2
Task 4	Algebra	15,882	15,615	19.3	16,364	15,986	12.7
Task 5	Geometry	3,190	3,149	36.1	3,190	3,035	28.7
Task 6	Counting and statistics	6,423	6,384	39.7	6,423	6,335	31.5
Task 7	Calculus	4,493	4,202	21.2	4,493	4,170	40.6
Task 8	Linear algebra	11,248	11,204	32.4	11,248	11,204	23.0
Task 9	Advanced math	746	746	21.2	746	745	27.3
Task 10	No language	41,191	40,551	21.2	42,466	41,794	40.6
Task 11	Simple language	66,505	66,172	26.9	290,184	258,839	8.2
Task 12	Complex language	26,119	25,728	36.1	26,119	25,052	28.7
Task 13	Fill in the blank	11,634	11,615	11.0	11,634	997	3.0
Task 14	Generative QA	102,493	101,239	14.7	327,447	314,652	16.0
Task 15	MCQ	9,989	9,989	28.3	9,989	470	3.0
Task 16	NLI	6,326	6,325	50.8	6,326	6,243	25.8
Task 17	RC	3,642	3,552	182.5	3,642	3,592	10.4
Task 18	No external knowledge	28,115	27,964	50.8	28,115	27,117	25.8
Task 19	Commonsense	24,677	24,658	30.9	24,677	823	3.0
Task 20	Math formulas	57,841	56,947	19.1	59,116	57,019	25.5
Task 21	Science formulas	10,505	10,319	36.1	10,505	9,764	28.7
Task 22	Complex knowledge	12,200	12,086	14.5	235,879	230,486	24.2
Task 23	Real-world knowledge	746	746	21.2	746	745	27.3

Table 6.17: Main Statistics of LīlaAcross the Total of 23 Tasks.

ID	Dataset	GPT-3	Neo-A	Neo-P	Codex
1	addsub	0.910	0.116	0.797	**0.950**
2	amps_algebra	0.116	0.100	**0.902**	0.655
3	amps_calculus	0.192	0.168	**0.922**	0.860
4	amps_counting_and_stats	0.183	0.117	**0.958**	0.650
5	amps_geometry	**0.283**	0.263	0.074	0.000
6	amps_linear_algebra	0.127	0.235	**0.815**	0.692
7	amps_number_theory	0.273	0.026	0.875	**1.000**
8	APPS_structured	0.167	0.154	0.134	**0.459**
9	asdiv	**0.737**	0.166	0.092	0.022
10	conala_structured	0.356	0.329	0.329	**0.391**
11	deepmind_mathematics_algebra	0.202	0.258	0.847	**0.910**
12	deepmind_mathematics_basicmath	0.270	0.125	0.614	**1.000**
13	deepmind_mathematics_calculus	0.208	0.026	0.152	**0.884**
14	deepmind_mathematics_muldiv	0.160	0.034	0.909	**1.000**
15	deepmind_mathematics_numbertheory	0.296	0.462	0.538	**0.710**
16	dolphin_t2_final	0.170	0.027	0.006	**0.812**
17	draw_structured	0.090	0.034	0.005	**0.210**
18	GSM8k_structured	0.110	0.060	0.139	**0.350**
19	MATH_crowdsourced	0.150	0.013	0.074	**0.472**
20	mathqa_gain	0.134	0.054	**0.339**	0.270
21	mathqa_general	0.110	0.073	**0.193**	0.120
22	mathqa_geometry	0.120	0.002	0.000	**0.250**
23	mathqa_other	0.180	0.043	0.011	**0.280**
24	mathqa_physics	0.120	0.087	**0.429**	0.210
25	mathqa_probability	**0.210**	0.003	0.000	0.200
26	mbpp_structured	0.128	0.175	0.164	**0.408**
27	MCTaco_event_duration_structured	**0.800**	0.773	0.773	0.710
28	MCTaco_event_ordering_structured	0.860	0.831	0.831	**0.890**
29	MCTaco_event_typical_time_structured	0.870	**0.881**	**0.881**	0.870
30	MCTaco_frequency_structured	**0.890**	0.862	0.862	0.790
31	MCTaco_stationarity_structured	0.710	**0.758**	**0.758**	0.670
32	multiarith	0.360	0.143	0.921	**0.990**
33	Numersense_structured	0.620	0.495	0.495	**0.660**
34	NumGLUE_Type_1	0.535	0.108	0.083	**0.740**
35	NumGLUE_Type_2	0.512	0.285	0.646	**0.735**
36	NumGLUE_Type_3	**0.835**	0.004	0.001	0.815
37	NumGLUE_Type_4	0.710	0.076	0.208	**0.790**
38	NumGLUE_Type_5	0.460	0.200	0.305	**0.615**
39	NumGLUE_Type_7	0.500	0.516	**0.854**	0.710
40	NumGLUE_Type_8	0.420	0.082	0.257	**0.610**
41	simuleq	0.120	0.074	0.010	**0.170**
42	singleop	0.940	0.347	0.611	**1.000**
43	singleq	**0.830**	0.143	0.474	0.670
44	svamp_structured	0.620	0.085	0.060	**0.790**
	Average F1 score	0.400	0.223	0.440	**0.613**

Table 6.18: Evaluation Results of Baselines Across Different Single Datasets.

ID	Dataset	References
1	addsub	Hosseini *et al.* (2014)
2	amps	Hendrycks *et al.* (2021b)
3	APPS	Hendrycks *et al.* (2021a)
4	asdiv	Miao *et al.* (2020)
5	conala	Yin *et al.* (2018)
6	mathematics	Saxton *et al.* (2019)
7	dolphin	Huang *et al.* (2016)
8	draw	Upadhyay and Chang (2015)
9	GSM8k	Cobbe *et al.* (2021)
10	MATH	Hendrycks *et al.* (2021b)
11	mathqa	Amini *et al.* (2019)
12	mbpp	Austin *et al.* (2021)
13	MCTaco	Zhou *et al.* (2019)
14	multiarith	Roy and Roth (2015)
15	Numersense	Lin *et al.* (2020)
16	NumGLUE	Mishra *et al.* (2022g); Dua *et al.* (2019b); Ravichander *et al.* (2019); Kushman *et al.* (2014); Tafjord *et al.* (2019); Roy and Roth (2018, 2017); Koncel-Kedziorski *et al.* (2016, 2015)
17	simuleq	Kushman *et al.* (2014)
18	singleop	Roy *et al.* (2015)
19	singleq	Koncel-Kedziorski *et al.* (2015)
20	svamp	Patel *et al.* (2021)

Table 6.19: List of Source Datasets and Corresponding References Used in Constructing Lı̄la.

Chapter 7

LESS IS MORE: SUMMARY OF LONG INSTRUCTIONS IS BETTER FOR PROGRAM SYNTHESIS

7.1 Introduction

Recently, large pre-trained LMs have been proven pivotal in programming-related tasks (Wang *et al.*, 2021; Chen *et al.*, 2021a; Hendrycks *et al.*, 2021a; Lu *et al.*, 2021c; Papineni *et al.*, 2002). Program synthesis aims to generate a code given the natural language description of a problem. Programming requirements in these problems vary in terms of complexity from a 3-5 line simple function to multiple functions that use advanced data structures. However, LMs such as Codex show below-par performance on the long and complicated programming questions. We observe that the natural language description of the program becomes long and complicated when there is superfluous information (see section 7.2.1). The goal of adding this information to the description is to make it more understandable to humans. However, we find that this information confuses the model in understanding a task. We propose that removing the excess information and providing the model with the exact specifications of the problem can improve the performance of the LMs.

To remove excess information, we summarize the descriptions of the program in such a way that it does not lose important specifications. We use the APPS dataset (Hendrycks *et al.*, 2021a) and CodeContests dataset (Li *et al.*, 2022) which are a collection of coding problems from different online sources and create a meta-dataset consisting of human and synthesized summaries.

We perform all experiments using the GPT-based Codex model (Chen *et al.*, 2021a) on the proposed meta-dataset and show that the summarized version of complicated questions improves strict accuracy by 8.13% on the APPS dataset and 11.85% on CodeContests. From our analysis, we can see significant improvement for introductory (9.86%) and interview (11.48%) related programming questions. However, it shows improvement by a small margin ($\sim 2\%$) for competitive programming questions. Considering that automatic evaluation of a program does not reward for partial correctness, we perform qualitative evaluation on our meta-dataset and find that original questions often confuse models in understanding the underlying problem, as models latch on to some spurious words in the text (e.g. the word 'list' in question makes the model design a list even though the underlying problem is on graphs). We further analyze model performance on different types of summaries (i.e., basic, expert, and synthetic) and provide instruction-design principles that can help future research on prompting in program synthesis. This has been discussed further in our work (Kuznia *et al.*, 2022).

7.2 Method

7.2.1 Dataset

We use the APPS (Hendrycks *et al.*, 2021a) and CodeContests (Li *et al.*, 2022) datasets to create summaries. We crowd-sourced the creation of human summaries. The result was 373 human summaries for APPS, 80 summaries for CodeContests and 8663 synthetic summaries using both datasets. Table 7.1 shows the statistics of the generated summaries.

Human Generated Summaries

For the APPS and CodeContests human-generated summaries, the crowd worker reads and understands the original questions, then creates summaries in two steps. First, we create a basic summary of the given problem and remove any information that is repeated and any hypothetical information without concrete instructions. For example, if the problem constructs a fake company or situation, we replace the fake situation with direct instructions. Second, we create an expert summary of the problem. To create this, we further summarize the first summary. This expert summary includes the absolute minimum information for an expert to understand the problem. We would not expect a novice to understand these prompts.

Synthetic Summaries

We have generated synthetic summaries of program descriptions using jumbo (178B), large (7.5B) Studio21 model (Lieber *et al.*, 2021), GPT-3 Davinci model (175B) (Brown *et al.*, 2020) and PEGASUS model (Zhang *et al.*, 2019). To generate a summary, we provide these models with a few examples in the in-context learning setup (Brown *et al.*, 2020) from the human-generated summaries. For the few-shot examples, we use expert-level summaries.

Studio21 We use five examples with the large model, and three examples with the jumbo model. For both models, we use a temperature of 0.3, and topP of 1. For the format of our prompt, we use De-Jargonizer template [1] with a change to their header. We create a total of $7,505$ synthetic summaries using these models.

GPT-3 We use three examples for GPT-3 model. We empirically set temperature to 0.05, topP to 1, frequency penalty to 0.01, presence penalty to 0.05. To generate prompts, we followed their tl;dr template [2] . We create 785 synthetic summaries using this model.

PEGASUS We use the PEGASUS model (Zhang *et al.*, 2019) to create program summaries for the same set of problems that were summarized by humans. We choose this model because it was trained specifically for abstractive summarization.

[1] https://studio.ai21.com/

[2] https://beta.openai.com/playground/p/default-tldr-summary?model=text-davinci-001

7.2.2 Model

We use OpenAI Codex to build baselines and the proposed approach.

Baseline To create a baseline, we have used original program descriptions given in the datasets as prompts for the Codex model.

Proposed Approach We have used summaries of original program descriptions given in the datasets as prompts for the Codex model.

7.3 Experimental Setup

All the experiments are performed using the $davinci - codex$ (Chen *et al.*, 2021a) model provided through OpenAI. At inference time, we use a modified version of the evaluation code [3] provided by Hendrycks *et al.* (2021a). This evaluation code has four different outputs for each test case: (1) **-2**: the code has a syntax error and can not run, (2) **-1**: the code is syntactically correct but has a run time error, (3) **0**: the code runs without any errors but fails the test case, and (4) **1**: the code runs without any error and passes the test case. Similar to Chen *et al.* (2021a), we implement a timeout for the code at inference time. If a test case takes more than 4 seconds to run then we throw an exception and count that test case as a -1.

Experiments To show effectiveness of the proposed approach, we have performed three different experiments using human generated summaries:

1. All problems from basic and expert summaries are used at inference time. We term this experiment All Problems (AP).

[3] https://github.com/hendrycks/apps/blob/main/eval/test_one_solution.py

2. We eliminate problems that perform worse for either basic or expert summaries. We term this experiment Either Worst Problem Removal (EWPR).
3. We eliminate problems that perform worse for both basic and expert summaries. We term this experiment Both Worst Problem Removal (BWPR).

Motivation behind EWPR and BWPR If a summary caused every test case to perform worse then it's likely the crowd worker produced a faulty summary. To mitigate the effect of outliers in the dataset, we use the EWPR method to remove such problems. Another hypothesis is that every problem benefits from some level of summarization (i.e., basic or expert). To measure this, we use the BWPR method. From Table 7.6 results, we identify that only 1 problem had both summaries (basic and expert) preform worse.

Metric In Austin *et al.* (2021), they show that the BLEU metric (Papineni *et al.*, 2002) does not correlate well with synthesis performance. Thus, we use Strict Accuracy (SAcc) as our evaluation metric for all experiments.

7.4 Results and Analysis

7.4.1 Human Generated Summaries

From Table 7.2, we can observe that both the summary-based models show on average superior performance compared to baseline. In particular, when calculating results for every problem, basic and expert summary-based models outperform baseline by 4.34% and 5.15% on average for APPS dataset, respectively. Further analysis shows that the expert summary-based model shows improved performance by $\sim 1\%$ compared to the basic summary-based model.

On the CodeContests dataset (Li *et al.*, 2022), we show an average improvement of 11.88% in terms of SAcc. For this dataset, we did not separate the problems by difficulty.

This is because the problems come from different sources and have different scales of difficulty. Thus, we did not report the SAcc when weighted by difficulty in Table 7.2.

Our analysis shows that many problems where the basic summary would fail, however, the expert summary would succeed and vice-versa. Thus, we choose the best summary for each problem after evaluating both summaries and then calculate the results for the best summaries. Table 7.3 shows results when taking the best summary for each problem for APPS dataset. We observe a 9.86%, 11.48%, and 1.91% increase on SAcc for introductory, interview, and competition level problems, respectively.

7.4.2 Synthetic Summaries

Tables 7.4 and 7.5 show the results for baseline, synthetic summaries generated by GPT-3, Studio21 and PEGASUS in terms of SAcc for two experiments. For the AP experiment, we can observe that the performance of the baseline outperforms synthetic summary-based models. However, the proposed model shows an average similar performance compared to the baseline for the EWPR experiment.

7.4.3 Analysis

Why does eliminating the worst problems help? From Tables 7.2, we can observe that EWPR and BWPR have improved performance compared to AP for both human and synthetically generated summaries. By analyzing the summarized worst problems, we notice a difference in the summarization style which shows that these summaries are outliers and do not match the distribution of the other summaries. This can cause a problem in synthesizing a good program since the model loses important information. Hence, we believe that eliminating the worst problems improves model performance.

Is there any possible bias in the meta-dataset? Recent studies show that bias propagates in human-annotated datasets (Geva *et al.*, 2019; Parmar *et al.*, 2022a). Given that our summaries are also human-generated, there will be some bias in the dataset. Some details that are critical to one person can be trivial to others. In the context of generating expert summaries, assumptions about expert knowledge can vary. This bias causes drift in the dataset and hinders the model's performance. Similar to Mishra *et al.* (2022f), we can provide a template for what is expected from the summary generator to reduce bias.

Why is competition accuracy low? We believe that these problems require multi-hop reasoning, even after summarization, which is still a challenge for language models.

Impact of POS on Accuracy In the top plot of Figure 7.1, we observe that frequency of $nouns$ and $propernouns$ for problems that passed all test cases is lower than the entire dataset. In the bottom plot, we observe that the frequency for $nouns$ and $propernouns$ is higher for the original question (which had < 100% accuracy on the test cases) and lower for the summary (which had 100% accuracy on the test cases). Thus, we can see that number of nouns degrades performance. We also see in the bottom chart that overuse of punctuation can be detrimental to performance. From the results in Figure 7.1 we see results of nouns affecting performance along with excessive punctuation.

7.5 Additional Analysis

Difficulty of CodeContests The accuracies for CodeContests is notably lower than the APPS dataset since this dataset is more challenging, e.g. the number and complexity of programming operations is relatively higher than APPS. From the baseline results in Table 7.2, we can observe that problems in CodeContests are harder than interview but easier than competition.

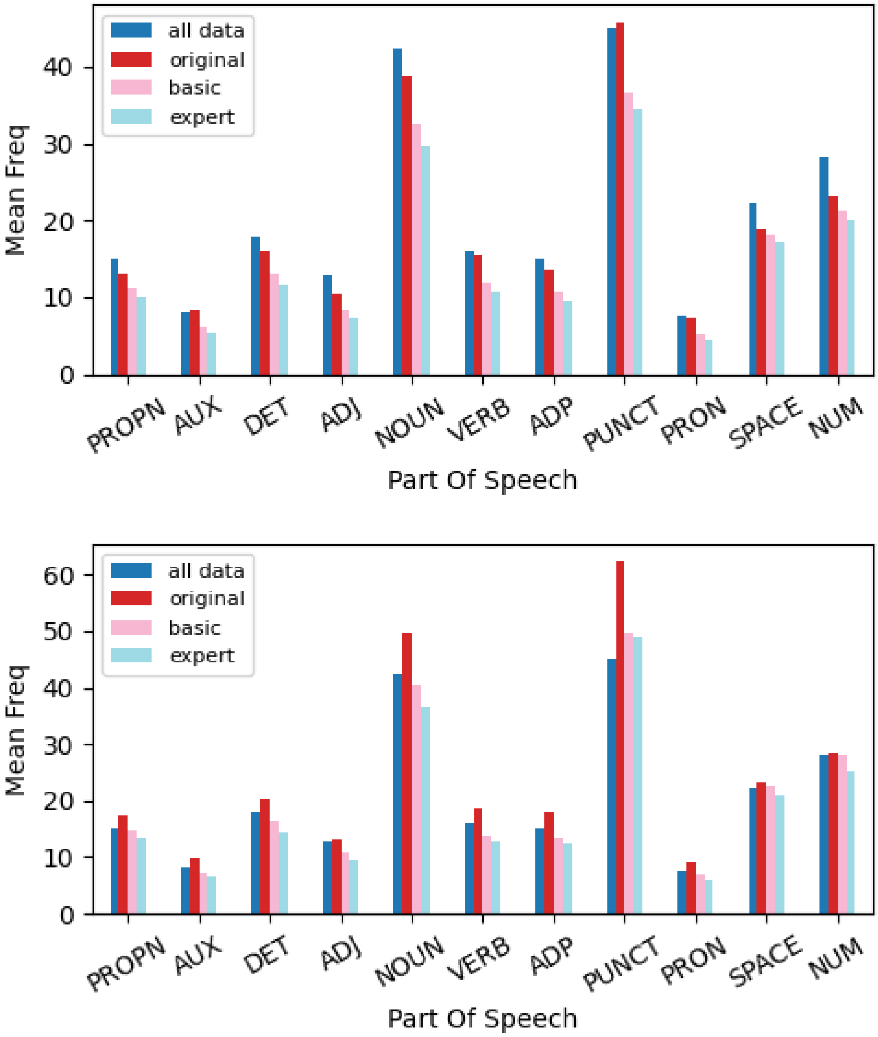


Figure 7.1: (Top Plot) Mean Frequency of Pos for Problems Where Programs Where the Generated by Both the Original and Summarized Prompt Pass All Test Cases, and (Bottom Plot) Mean Frequency of Pos for Problems Where the Summary Passes All Test Cases and the Original Did Not. The Blue Bar Represents the Mean of the Entire Dataset. Analyzed Only the Top 11 Most Occurring Pos. The Plot Shows That Higher Number of Nouns Degrade Model Performance.

Impact of Entities on Accuracy In Figure 7.2, we can observe that the total number of entities $num\_entities$ is higher for problems that performed worse. Here, we can see that the original problems (which failed test cases) had a higher mean than the dataset and the summaries (which passed all test cases) had a lower number of entities.

7.6 Example of removing fake information

To see the code produced by the model for this example, refer to 7.13. There are more examples of superfluous information confusing the model in 7.18 and of made up information confusing the model in 7.19.

7.6.1 Original Prompt

> Codefortia is a small island country located somewhere in the West Pacific. It consists of n settlements connected by m bidirectional gravel roads. Curiously enough, the beliefs of the inhabitants require the time needed to pass each road to be equal either to a or b seconds. It's guaranteed that one can go between any pair of settlements by following a sequence of roads.
>
> Codefortia was recently struck by the financial crisis. Therefore, the king decided to abandon some of the roads so that:
>
> it will be possible to travel between each pair of cities using the remaining roads only, the sum of times required to pass each remaining road will be minimum possible (in other words, remaining roads must form minimum spanning tree, using the time to pass the road as its weight), among all the plans minimizing the sum of times above, the time required to travel between the king's residence (in settlement 1) and the parliament house (in settlement p) using the remaining roads only will be minimum possible.

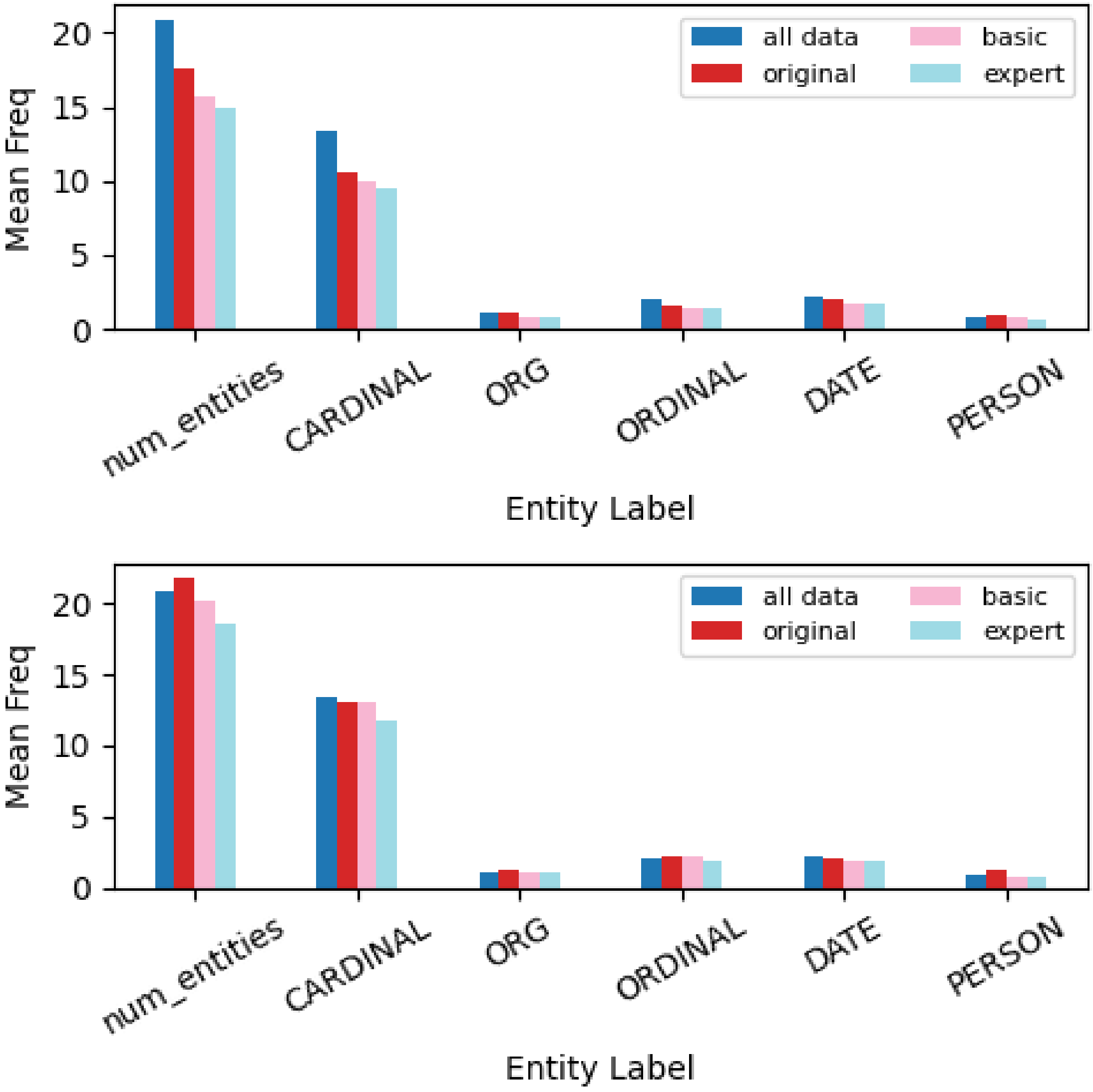


Figure 7.2: (Top Plot) Mean Frequency of the Entity Labels for Problems Where Program Generated by the Original and Summarized Prompt Pass All Test Cases, and (Bottom Plot) Mean Frequency of Entity Labels for Problems Where the Summary Passes All Test Cases and the Original Did Not. We Analyzed Only the Top 5 Most Occurring Entities among All Entities We Found.

The king, however, forgot where the parliament house was. For each settlement $p = 1, 2, \dots, n$, can you tell what is the minimum time required to travel between the king's residence and the parliament house (located in settlement p) after some roads are abandoned?

——Input——

The first line of the input contains four integers n, m, a and b ($2 \leqslant n \leqslant 70$, $n - 1 \leqslant m \leqslant 200$, $1 \leqslant a < b \leqslant 10^7$) — the number of settlements and gravel roads in Codefortia, and two possible travel times. Each of the following lines contains three integers u, v, c ($1 \leqslant u, v \leqslant n$, $u \neq v$, $c \in \{a, b\}$) denoting a single gravel road between the settlements u and v, which requires c minutes to travel.

You can assume that the road network is connected and has no loops or multiedges.

——Output——

Output a single line containing n integers. The p-th of them should denote the minimum possible time required to travel from 1 to p after the selected roads are abandoned. Note that for each p you can abandon a different set of roads.

——Examples——

Input

5 5 20 25

1 2 25

2 3 25

3 4 20

4 5 20

5 1 20

Output

0 25 60 40 20

Input

6 7 13 22

1 2 13

2 3 13

1 4 22

3 4 13

4 5 13

5 6 13

6 1 13

Output

0 13 26 39 26 13

——Note——

The minimum possible sum of times required to pass each road in the first example is 85 — exactly one of the roads with passing time 25 must be abandoned. Note that after one of these roads is abandoned, it's now impossible to travel between settlements 1 and 3 in time 50.

We can see the author of the problem is trying to describe a fully-connected graph with n nodes and m edges each with a weight a or b. Thus, this paragraph can be summarized as:

7.6.2 Basic Summary

You are given a graph of n nodes and m bidirectional edges. The cost for each edge is either a or b. The graph is fully-connected, so you can travel between any pair of nodes using the edges.

For each node $p = 1, 2, \dots, n$, you need to remove some edges so that: It will be possible to travel between each pair of nodes using the remaining edges only, and the sum of times required to pass each remaining road will be the minimum possible. You should output the minimum time required to travel between node 1 and node p.

——Input——

The first line of the input contains four integers n, m, a and b ($2 \leqslant n \leqslant 70$, $n - 1 \leqslant m \leqslant 200$, $1 \leqslant a < b \leqslant 10^7$) — the number of nodes and edges in the graph, and two possible travel times. Each of the following lines contains three integers u, v, c ($1 \leqslant u, v \leqslant n$, $u \neq v$, $c \in \{a, b\}$) denoting an edge between the nodes u and v, which has cost c.

You can assume that the graph is connected and has no loops or multiedges.

——Output——

Output a single line containing n integers. The p-th of them should denote the minimum possible post required to travel from 1 to p after the selected edges are abandoned. Note that for each p you can abandon a different set of edges.

——Examples——

Input

5 5 20 25

1 2 25

2 3 25

3 4 20

4 5 20

5 1 20

Output

0 25 60 40 20

Input

6 7 13 22

1 2 13

2 3 13

1 4 22

3 4 13

4 5 13

5 6 13

6 1 13

Output

0 13 26 39 26 13

7.6.3 Expert Summary

However, we can assume that an expert would already know what a minimum spanning tree is. Thus, we can remove this detailed description of an MST.

You are given a connected graph of n nodes and m bidirectional edges. For each node $p = 1, 2, \ldots, n$, you need to find a minimum spanning tree. Then output the minimum cost required to travel between node 1 and node p.

—–Input—–

The first line of the input contains four integers n, m, a and b ($2 \leqslant n \leqslant 70$, $n - 1 \leqslant m \leqslant 200$, $1 \leqslant a < b \leqslant 10^7$) — the number of nodes and edges in the graph, and two possible travel times. Each of the following lines contains three integers u, v, c ($1 \leqslant u, v \leqslant n$, $u \neq v$, $c \in \{a, b\}$) denoting an edge between the nodes u and v, which has cost c.

You can assume that the graph is connected and has no loops or multiedges.

—–Output—–

Output a single line containing n integers. The p-th of them should denote the minimum possible post required to travel from 1 to p after the selected edges are abandoned. Note that for each p you can abandon a different set of edges.

—–Examples—–

Input

5 5 20 25

1 2 25

2 3 25

3 4 20

4 5 20

5 1 20

Output

0 25 60 40 20

Input

```
6 7 13 22
1 2 13
2 3 13
1 4 22
3 4 13
4 5 13
5 6 13
6 1 13
```

Output

```
0 13 26 39 26 13
```

7.7 Prompt templates

Studio21 Here is our template for Studio21. To see examples of summaries produced by Studio21AI’s model along with the code generated for those summaries, refer to 7.14 and 7.15.

```
The following sentences contain computer science jargon. Rewrite them using simple words.
Jargon: <ORIGINAL>
Simple: <SUMMARY>

Jargon: <ORIGINAL>
Simple: <SUMMARY>
```

Jargon: <ORIGINAL>

Simple: <SUMMARY>

Jargon: <ORIGINAL>

Simple:

The few-shot examples were chosen randomly from the human generated expert summaries.

GPT3 Here is our template for GPT3. To see examples of summaries produced by GPT3 and the code generated for those summaries refer to 7.16.

Summarize the following paragraph: Original: <ORIGINAL>

Summary: <SUMMARY>

Original: <ORIGINAL>

Summary: <SUMMARY>

Original: <ORIGINAL>

Summary: <SUMMARY>

Original: <ORIGINAL> Summary:

Codex Here is our default template for Codex, which is used when there is no starter code provided. When there is starter code provided the docstring remains the same but the code after the doc string will be what is provided.

```
Python3
"""
<PROBLEM DESCRIPTION>
"""
def code():
```

7.8 Strict Accuracy

Strict Accuracy (SAcc) is the percentage of problems that passed every test case. The formula to calculate SAcc is given below:

$$\textit{strict acc} := \frac{\textit{problems with 100\% accuracy}}{\textit{total number of problems}} \tag{7.1}$$

Given that, we are only generating one code solution for each problem our strict accuracy is comparable to Chen *et al.* (2021a)'s metric $raw\ pass@1$.

7.9 Codex Configuration

We did a small test with 75 summaries to find our hyper-parameters for Codex. We set temperature to 0, topP to 1, frequency penalty to 0.2, and presence penalty to 0. We did not provide few-shot examples to Codex since we want to see if summarization only could improve the performance of the Codex model.

7.10 Worst Problems and Statistics

Using the test case labels as defined in section 7.3 we defined a test case as getting worse if its label (result) was lower. Then we defined a problem as worse if *every* test case had a lower label. Our methodology behind this was, if we removed problems that had a worse accuracy, then it would be a non-trivial result that accuracy improved. Also, if we removed problems with worse accuracy, then a problem that originally had all 0 labels (all False test cases) would score the same if the summary had all -1 labels (runtime error) or a -2 (syntax error). So, we removed problems which every test case performed worse, to see if removing these outliers would improve results. You can see the overall breakdown of each split in table 7.6.

7.11 Average length of Problems and Solutions

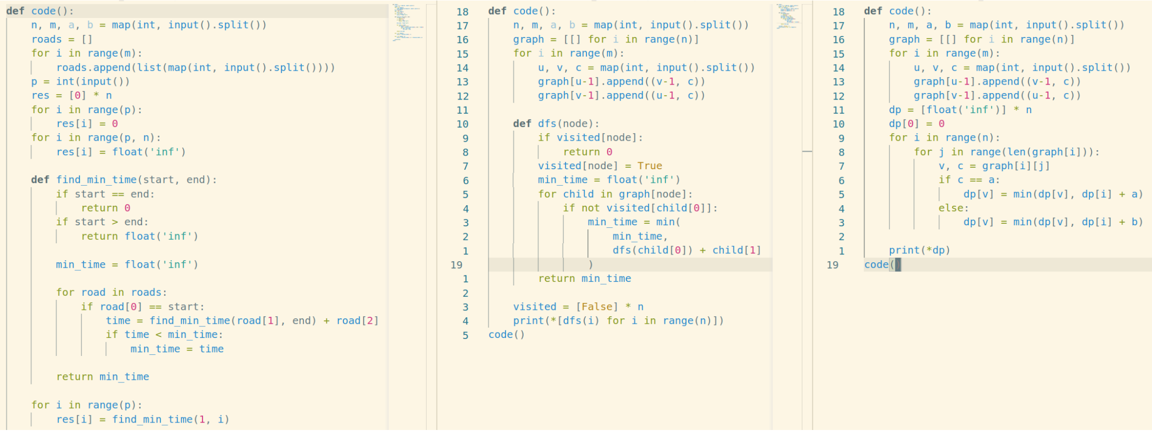
Table 7.7 represents the statistics for average length of problems and solutions for original and summarized prompts.

7.12 Abbreviated Synthetic Results

In table 7.8, we show the results for our synthetic summaries when taking the top 500 and 1000 summaries for GPT3 and StudioAI21, respectively. In our initial experiment, this was the amount of problems we tested for each model. However, in our final experiment we changed our configurations and generated more problems. For a comparison, we took the top performing summaries and and reported those results.

7.13 Generated Code

In figure 7.3 we present the code that was generated for the example mentioned in 7.6 and 7.6.3. Given that the Codex model was prompted with the $def\, code()$: the model did not generate that function definition or the call to that function. That was added in afterwards, but everything inside that function was generated by Codex. The originally generated code (far left) fails with a -1 because it did not take in the input correctly. It added in another line *p = int(input())*, which most likely refers to the p mentioned in the original text. The expert summary generated code (middle) fails every test case. The basic summary generated code (right) passed $16/19\,(84\%)$ test cases and was the only code to pass at least 1 test case.



```python
def code():
    n, m, a, b = map(int, input().split())
    roads = []
    for i in range(m):
        roads.append(list(map(int, input().split())))
    p = int(input())
    res = [0] * n
    for i in range(p):
        res[i] = 0
    for i in range(p, n):
        res[i] = float('inf')

    def find_min_time(start, end):
        if start == end:
            return 0
        if start > end:
            return float('inf')

        min_time = float('inf')

        for road in roads:
            if road[0] == start:
                time = find_min_time(road[1], end) + road[2]
                if time < min_time:
                    min_time = time

        return min_time

    for i in range(p):
        res[i] = find_min_time(1, i)
```

```python
def code():
    n, m, a, b = map(int, input().split())
    graph = [[] for i in range(n)]
    for i in range(m):
        u, v, c = map(int, input().split())
        graph[u-1].append((v-1, c))
        graph[v-1].append((u-1, c))

    def dfs(node):
        if visited[node]:
            return 0
        visited[node] = True
        min_time = float('inf')
        for child in graph[node]:
            if not visited[child[0]]:
                min_time = min(
                    min_time,
                    dfs(child[0]) + child[1]
                )
        return min_time

    visited = [False] * n
    print(*[dfs(i) for i in range(n)])
code()
```

```python
def code():
    n, m, a, b = map(int, input().split())
    graph = [[] for i in range(n)]
    for i in range(m):
        u, v, c = map(int, input().split())
        graph[u-1].append((v-1, c))
        graph[v-1].append((u-1, c))
    dp = [float('inf')] * n
    dp[0] = 0
    for i in range(n):
        for j in range(len(graph[i])):
            v, c = graph[i][j]
            if c == a:
                dp[v] = min(dp[v], dp[i] + a)
            else:
                dp[v] = min(dp[v], dp[i] + b)

    print(*dp)
code()
```

Figure 7.3: On the Far Left Is the Code Generated by the Original Prompt. The Middle Is the Code Generated by the Expert Summary. The Right Is the Code Generated by the Basic Summary.

7.14 StudioAI21 Generated Code

Below is an example of a competition problem where StudioAI21 summarized the prompt too much but Codex was still able to produce viable code. Here is the original prompt:

> Cengiz recently learned Fibonacci numbers and now he is studying different algorithms to find them. After getting bored of reading them, he came with his own new type of numbers that he named XORinacci numbers. He defined them as follows: $f(0) = a$; $f(1) = b$; $f(n) = f(n-1) \oplus f(n-2)$ when $n > 1$, where $\oplus$ denotes the bitwise XOR operation.
>
> You are given three integers a, b, and n, calculate $f(n)$.
>
> You have to answer for T independent test cases.
>
> ——Input——
>
> The input contains one or more independent test cases.

The first line of input contains a single integer T ($1 \leqslant T \leqslant 10^3$), the number of test cases.

Each of the T following lines contains three space-separated integers a, b, and n ($0 \leqslant a, b, n \leqslant 10^9$) respectively.

—–Output—–

For each test case, output $f(n)$.

—–Example—–

Input

3

3 4 2

4 5 0

325 265 1231232

Output

7

4

76

—–Note—–

In the first example, $f(2) = f(0) \oplus f(1) = 3 \oplus 4 = 7$.

Here is the summary that StudioAI21 generated:

You are given three integers a, b, and n. Calculate $f(n)$.

—–Input—–

The input contains one or more independent test cases.

The first line of input contains a single integer T ($1 \leqslant T \leqslant 10^3$), the number of test cases.

Each of the T following lines contains three space-separated integers a, b, and n ($0 \leqslant a, b, n \leqslant 10^9$) respectively.

—–Output—–

For each test case, output $f(n)$.

—–Example—–

Input

3

3 4 2

4 5 0

325 265 1231232

Output

7

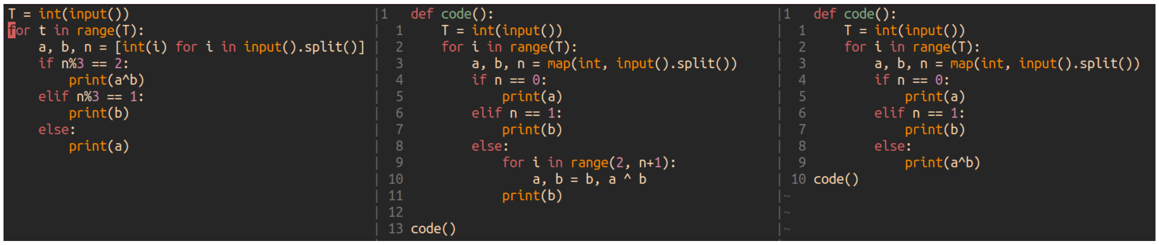
4

76

—–Note—–

In the first example, $f(2) = f(0) \oplus f(1) = 3 \oplus 4 = 7$.

Because any input/output examples provided by the prompt are appended to the summary, Codex was able to figure out the pattern in the problem and generate code that was almost correct. In figure 7.4, the solution (left) used the pattern in the problem and simplify by taking $n \mod 3$. The Studio21 summary code (right) recognizes this pattern but erroneously does not take the modulus of the number. The original code (center) also makes the same mistake by not taking the modulus, but also brute forces the answer. This shows that the model did not recognize the pattern in this problem because of the superfluous details. Even though Studio21 might have summarized too much, the model was still able to make an improvement and understand the pattern in the problem more.



```python
T = int(input())
for t in range(T):
    a, b, n = [int(i) for i in input().split()]
    if n%3 == 2:
        print(a^b)
    elif n%3 == 1:
        print(b)
    else:
        print(a)
```

```python
def code():
    T = int(input())
    for i in range(T):
        a, b, n = map(int, input().split())
        if n == 0:
            print(a)
        elif n == 1:
            print(b)
        else:
            for i in range(2, n+1):
                a, b = b, a ^ b
            print(b)

code()
```

```python
def code():
    T = int(input())
    for i in range(T):
        a, b, n = map(int, input().split())
        if n == 0:
            print(a)
        elif n == 1:
            print(b)
        else:
            print(a^b)
code()
```

Figure 7.4: The Left Is the Code Generated Using the Original Prompt. The Right Is the Code Generated When Using the Studioai21 Generated Summary.

7.15 StudioAI21 Generated Code

Here is an example of a summary made by StudioAI21 where the qualitative aspect of the code but it still failed. Here is the original prompt:

> Given is a tree G with N vertices. The vertices are numbered 1 through N, and the i-th edge connects Vertex a_i and Vertex b_i. Consider painting the edges in G with some number of colors. We want to paint them so that, for each vertex, the colors of the edges incident to that vertex are all different. Among the colorings satisfying the condition above, construct one that uses the minimum number of colors.

-----Constraints-----

- $2 \leqslant N \leqslant 10^5$

- $1 \leqslant a_i < b_i \leqslant N$

- All values in input are integers.

- The given graph is a tree.

—–Input—–

Input is given from Standard Input in the following format:

N

a_1 b_1

a_2 b_2

$\vdots$

a_{N-1} b_{N-1}

—–Output—–

Print N lines.

The first line should contain K, the number of colors used.

The (i+1)-th line $(1 \leqslant i \leqslant N-1)$ should contain c_i, the integer representing the color of the i-th edge, where $1 \leqslant c_i \leqslant K$ must hold.

If there are multiple colorings with the minimum number of colors that satisfy the condition, printing any of them will be accepted.

—–Sample Input—–

3

1 2

2 3

—–Sample Output—–

2

1

2

Here is the summary that StudioAI21 generated:

You are given a tree with N vertices numbered from 1 to N. You need to find the minimum possible number of colors to paint the edges such that vertices are painted with different color. —–Constraints—–

- $2 \leqslant N \leqslant 10^5$
- $1 \leqslant a_i < b_i \leqslant N$
- All values in input are integers.
- The given graph is a tree.

—–Input—–

Input is given from Standard Input in the following format:

N

a_1 b_1

a_2 b_2

$\vdots$

a_{N-1} b_{N-1}

—–Output—–

Print N lines.

The first line should contain K, the number of colors used.

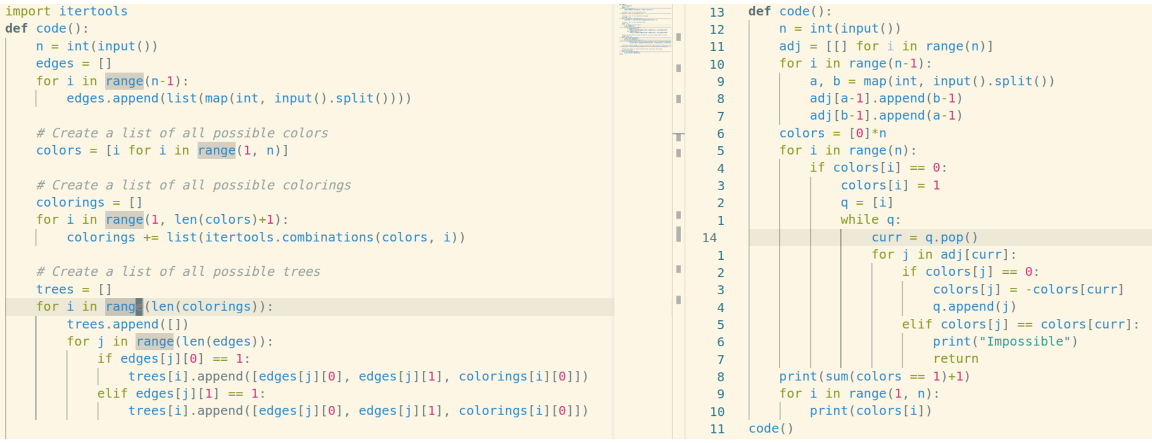
The (i+1)-th line $(1 \leqslant i \leqslant N-1)$ should contain c_i, the integer representing

> the color of the i-th edge, where $1 \leqslant c_i \leqslant K$ must hold.
>
> If there are multiple colorings with the minimum number of colors that satisfy the condition, printing any of them will be accepted.
>
> —–Sample Input—–
>
> 3
>
> 1 2
>
> 2 3
>
> —–Sample Output—–
>
> 2
>
> 1
>
> 2

In 7.5 the left is the original solution which fails with a -2 because the runtime of the algorithm is exponential. Note that it tries to create a list of all possible edge colorings which is $\mathcal{O}(2^N)$. The right is the code produced when using the StudioAI21 summary. You can see that this code is much closer to solving the problem and produces an efficient algorithm. However, this fails with a -2 because it tries to print the sum of a boolean (near the end before the last for loop). Which fails in python because a bool is not iterable.

Here is a problem where StudioAI21's summary increased the accuracy to 100%. Here is the original prompt:

> Polycarpus has a sequence, consisting of n non-negative integers: $a_1, a_2, ..., a_n$.
>
> Let's define function f(l, r) (l, r are integer, $1 \leqslant l \leqslant r \leqslant n$) for sequence a as an operation of bitwise OR of all the sequence elements with indexes from l to r. Formally: f(l, r) = $a_l | a_l + 1 | ... | a_r$.



```python
import itertools
def code():
    n = int(input())
    edges = []
    for i in range(n-1):
        edges.append(list(map(int, input().split())))

    # Create a list of all possible colors
    colors = [i for i in range(1, n)]

    # Create a list of all possible colorings
    colorings = []
    for i in range(1, len(colors)+1):
        colorings += list(itertools.combinations(colors, i))

    # Create a list of all possible trees
    trees = []
    for i in range(len(colorings)):
        trees.append([])
        for j in range(len(edges)):
            if edges[j][0] == 1:
                trees[i].append([edges[j][0], edges[j][1], colorings[i][0]])
            elif edges[j][1] == 1:
                trees[i].append([edges[j][0], edges[j][1], colorings[i][0]])
```

```python
def code():
    n = int(input())
    adj = [[] for i in range(n)]
    for i in range(n-1):
        a, b = map(int, input().split())
        adj[a-1].append(b-1)
        adj[b-1].append(a-1)
    colors = [0]*n
    for i in range(n):
        if colors[i] == 0:
            colors[i] = 1
            q = [i]
            while q:
                curr = q.pop()
                for j in adj[curr]:
                    if colors[j] == 0:
                        colors[j] = -colors[curr]
                        q.append(j)
                    elif colors[j] == colors[curr]:
                        print("Impossible")
                        return
    print(sum(colors == 1)+1)
    for i in range(1, n):
        print(colors[i])
code()
```

Figure 7.5: The Left Is the Code Generated Using the Original Prompt. The Right Is the Code Generated When Using the Studioai21 Generated Summary.

Polycarpus took a piece of paper and wrote out the values of function f(l, r) for all l, r (l, r are integer, $1 \leqslant l \leqslant r \leqslant n$). Now he wants to know, how many distinct values he's got in the end.

Help Polycarpus, count the number of distinct values of function f(l, r) for the given sequence a.

Expression x|y means applying the operation of bitwise OR to numbers x and y. This operation exists in all modern programming languages, for example, in language C++ and Java it is marked as "|", in Pascal — as "or".

—–Input—–

The first line contains integer n $(1 \leqslant n \leqslant 10^5)$ — the number of elements of sequence a. The second line contains n space-separated integers $a_1, a_2, ..., a_n$ $(0 \leqslant a_i \leqslant 10^6)$ — the elements of sequence a.

—–Output—–

Print a single integer — the number of distinct values of function f(l, r) for the given sequence a.

Please, do not use the lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the I64d specifier.

—–Examples—–

Input

3

1 2 0

Output

4

Input

10

1 2 3 4 5 6 1 2 9 10

Output

11

—–Note—–

In the first test case Polycarpus will have 6 numbers written on the paper: f(1, 1) = 1, f(1, 2) = 3, f(1, 3) = 3, f(2, 2) = 2, f(2, 3) = 2, f(3, 3) = 0. There are exactly 4 distinct numbers among them: 0, 1, 2, 3.

Here is the summary that StudioAI21 generated. Not exactly as we would expect as the prompt still mentions the fictional *Polycarpus*.

Polycarpus has a sequence, consisting of n non-negative integers: $a_1, a_2, ..., a_n$. Let's define function f(l, r) (l, r are integer, 1 l r n) for sequence a as an operation of bitwise OR of all the sequence elements with indexes from l to r. Formally: f(l, r) = $a_l|a_l + 1|...$

—–Input—–

The first line contains integer n $(1 \leqslant n \leqslant 10^5)$ — the number of elements of sequence a. The second line contains n space-separated integers $a_1, a_2, ..., a_n$ $(0 \leqslant a_i \leqslant 10^6)$ — the elements of sequence a.

—–Output—–

Print a single integer — the number of distinct values of function f(l, r) for the given sequence a.

Please, do not use the lld specifier to read or write 64-bit integers in C++. It is preferred to use cin, cout streams or the I64d specifier.

—–Examples—–

Input

3

1 2 0

Output

4

Input

10

1 2 3 4 5 6 1 2 9 10

Output

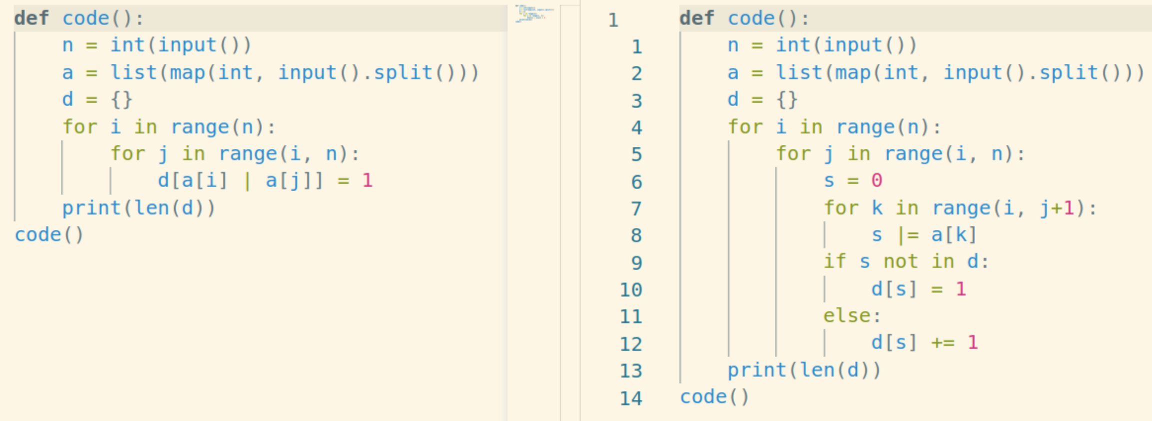
11

—–Note—–

In the first test case Polycarpus will have 6 numbers written on the paper: f(1, 1)

= 1, f(1, 2) = 3, f(1, 3) = 3, f(2, 2) = 2, f(2, 3) = 2, f(3, 3) = 0. There are exactly 4 distinct numbers among them: 0, 1, 2, 3.

In 7.6 the left is the original solution which gets 77% accuracy. The right is the summary code which gets 100% accuracy.



```
def code():
    n = int(input())
    a = list(map(int, input().split()))
    d = {}
    for i in range(n):
        for j in range(i, n):
            d[a[i] | a[j]] = 1
    print(len(d))
code()
```

```
def code():
    n = int(input())
    a = list(map(int, input().split()))
    d = {}
    for i in range(n):
        for j in range(i, n):
            s = 0
            for k in range(i, j+1):
                s |= a[k]
            if s not in d:
                d[s] = 1
            else:
                d[s] += 1
    print(len(d))
code()
```

Figure 7.6: The Left Is the Code Generated Using the Original Prompt. The Right Is the Code Generated When Using the Studioai21 Generated Summary.

7.16 GPT Generated Summaries

Here are two summaries where GPT perfectly summarized the prompt and gave a concise description of what the task was. In both cases the original prompt did not have 100% accuracy but the summarized prompt did have 100% accuracy.

Here is the original prompt for the first question:

> Božo is a strange little boy. Every day he tires his friends with strange questions. Today's question is: how many integers in the interval $[A, B]$ are there such that the sum of their digits is S, and which is the smallest such number?
>
> Write a program that answers Božo's question so that he can get some sleep.

——Input——

The input contains three integers A, B and S ($1 \leqslant A \leqslant B < 10^{15}$, $1 \leqslant S \leqslant 135$).

——Output——

The first line should contain the number of integers in the interval with the digit sum equal to S.

The second line should contain the smallest such integer.

The input data will guarantee that the first number is at least 1.

——Examples——

Sample Input 1:

1 9 5

Sample Output 1:

1

5

Sample Input 2:

1 100 10

Sample Output 2:

9

19

Here is the summary that GPT Generated:

Write a program that calculates the smallest integer in the given interval whose sum of digits is equal to the given sum.

——Input——

The input contains three integers A, B and S ($1 \leqslant A \leqslant B < 10^{15}$, $1 \leqslant S \leqslant 135$).

——Output——

The first line should contain the number of integers in the interval with the digit sum equal to S.

The second line should contain the smallest such integer.

The input data will guarantee that the first number is at least 1.

——Examples——

Sample Input 1:

1 9 5

Sample Output 1:

1

5

Sample Input 2:

1 100 10

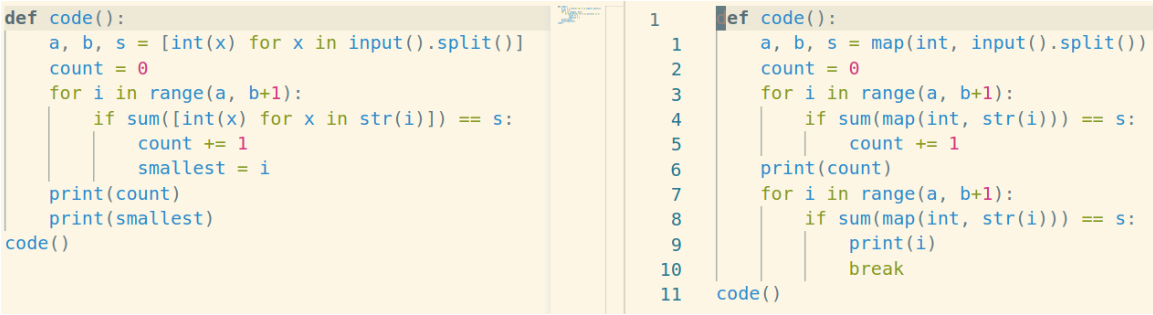
Sample Output 2:

9

19

In 7.7 you can see the original code on the left and the summary code on the right. There is a subtle difference but it's that difference that improved the problem from 33% accuracy to 100%.

Here is the original prompt for another question.



```python
def code():
    a, b, s = [int(x) for x in input().split()]
    count = 0
    for i in range(a, b+1):
        if sum([int(x) for x in str(i)]) == s:
            count += 1
            smallest = i
    print(count)
    print(smallest)
code()
```

```python
def code():
    a, b, s = map(int, input().split())
    count = 0
    for i in range(a, b+1):
        if sum(map(int, str(i))) == s:
            count += 1
    print(count)
    for i in range(a, b+1):
        if sum(map(int, str(i))) == s:
            print(i)
            break
code()
```

Figure 7.7: The Left Is the Code Generated Using the Original Prompt. The Right Is the Code Generated When Using the GPT3 Generated Summary.

Professor GukiZ makes a new robot. The robot are in the point with coordinates (x_1, y_1) and should go to the point (x_2, y_2). In a single step the robot can change any of its coordinates (maybe both of them) by one (decrease or increase). So the robot can move in one of the 8 directions. Find the minimal number of steps the robot should make to get the finish position.

—–Input—–

The first line contains two integers x_1, y_1 $(-10^9 \leqslant x_1, y_1 \leqslant 10^9)$ — the start position of the robot.

The second line contains two integers x_2, y_2 $(-10^9 \leqslant x_2, y_2 \leqslant 10^9)$ — the finish position of the robot.

—–Output—–

Print the only integer d — the minimal number of steps to get the finish position.

—–Examples—–

Input

0 0

4 5

Output

5

Input

3 4

6 1

Output

3

—–Note—–

In the first example robot should increase both of its coordinates by one four times, so it will be in position (4, 4). After that robot should simply increase its y coordinate and get the finish position.

In the second example robot should simultaneously increase x coordinate and decrease y coordinate by one three times.

Here is the summary that GPT3 generated:

The robot can move in one of the 8 directions. Find the minimal number of steps the robot should make to get the finish position.

—–Input—–

The first line contains two integers x_1, y_1 $(-10^9 \leqslant x_1, y_1 \leqslant 10^9)$ — the start position of the robot.

The second line contains two integers x_2, y_2 $(-10^9 \leqslant x_2, y_2 \leqslant 10^9)$ — the finish position of the robot.

—–Output—–

Print the only integer d — the minimal number of steps to get the finish position.

——Examples——

Input

0 0

4 5

Output

5

Input

3 4

6 1

Output

3

——Note——

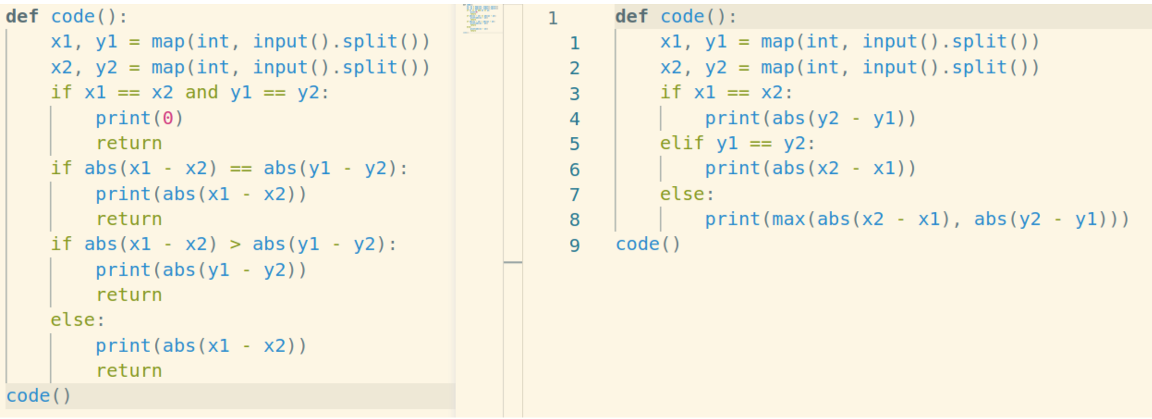
In the first example robot should increase both of its coordinates by one four times, so it will be in position (4, 4). After that robot should simply increase its y coordinate and get the finish position.

In the second example robot should simultaneously increase x coordinate and decrease y coordinate by one three times.

In 7.8 you can see the original code on the left and the summary code on the right. There is a subtle difference but it's that difference that improved the problem from 20% accuracy to 100%.

7.17 Human Generated Instructions

The section below was given to each crowd worker as instructions to follow when creating the regular and expert summaries.



```python
def code():
    x1, y1 = map(int, input().split())
    x2, y2 = map(int, input().split())
    if x1 == x2 and y1 == y2:
        print(0)
        return
    if abs(x1 - x2) == abs(y1 - y2):
        print(abs(x1 - x2))
        return
    if abs(x1 - x2) > abs(y1 - y2):
        print(abs(y1 - y2))
        return
    else:
        print(abs(x1 - x2))
        return
code()
```

```python
def code():
    x1, y1 = map(int, input().split())
    x2, y2 = map(int, input().split())
    if x1 == x2:
        print(abs(y2 - y1))
    elif y1 == y2:
        print(abs(x2 - x1))
    else:
        print(max(abs(x2 - x1), abs(y2 - y1)))
code()
```

Figure 7.8: The Left Is the Code Generated Using the Original Prompt. The Right Is the Code Generated When Using the GPT3 Generated Summary.

7.17.1 Summarization

Create a file called *summary.txt* this will contain your summary of the prompt. It's recommended that you copy the *question.txt* file into the *summary.txt* file then starting from the top of the prompt follow the steps and remove words/lines as necessary.

These are the rough steps for making a summary. Following these steps will create the most consistency in our dataset. However, you should summarize as you see fit. First, read through the prompt and understand what it's asking, then follow these steps to help create a summary.

1. Directly state what is given in the problem.

    - Most problems start by setting the scene, to help humans understand.
    - Start the problems by explicitly telling the model what the input is.
    - *You are given ...*

2. Remove any *notes* given in the prompt.

  - They are usually reemphasizing points, which is redundant and not needed in the summary.
  - This includes the $-Notes-$ section at the bottom of the file.
  - If there is pertinent information given from a note, include it in the prompt without describing it as a note.

3. Remove any text in parenthesis.
   - Most of the text in parenthesis is repeating the information that precede them.
   - If the text in parenthesis provides more context or information, then remove the preceding text.
   - Keep any parenthesis if it is describing constraints, such as the minimum and maximum values for the input etc...

4. Remove any made up people, places, things, etc...
   - These abstractions are made to help humans understand but confuse the model.
   - The prompts often mention things like $Codefortia$ or $Polycarp$, try to replace these with the word you.
   - Any text visualizing what the problem is asking, should be removed.

5. If the $Input$ or $Output$ section reference an abstraction they should be changed.
   - Overall, these sections are fine. However, if they mentioned something you removed in the previous steps, they should be changed to reflect that.
   - If these sections repeat themselves remove any redundancies.
   - In most cases these sections will be left alone.

7.17.2 Expert Summary

Create a file called $expert.txt$ this will contain an expert summary of the prompt. It's recommended that you copy the $summary.txt$ file into the $expert.txt$ file then starting from the top of the prompt remove words/lines as necessary. You should aim for the expert prompt to be $2 - 4$ lines.

Imagine you are describing the prompt to a senior software engineer. What else could you trim out? The difference between the original and expert summary, is the original summary may include something obvious, whereas the expert solution should be the absolute bare minimum. To create $summary.txt$ you want to remove superfluous details from the original prompt. To create $expert.txt$ you want to remove details that an expert would find obvious, from the summary.

For example, in problem 2000 (which is competitive difficulty) the summary mentions '*It will be possible to travel between each pair of nodes . . . , and the sum of times . . . will be the minimum possible*'. This process is describing a minimum spanning tree so you can just say '*Find a minimum spanning tree*'.

Also, if the prompt included an example and subsequent explanation, that should remain in the summary but should be removed from the expert summary. An expert already understands the problem and does not need any extra explanation. You should still keep the $-Examples-$ section.

Takeaways

- Removing made up people, places, and things from the prompt improved the quality of code generated.
- The optimal summarization depends on the difficulty of the problem.
- Synthetically generate summaries were close to maintaining accuracy.

- With more rigorous instructions, human summaries could be made with less noise which would further improve synthetic summary generation.

7.18 Superfluous Information Confusing the Model

Here is an example of an interview level string problem where the original prompt got 0% and both human generated summaries got 100% accuracy. The question wants you to write code that will return the number of unique character in the given string.

7.18.1 Original Prompt

> You have initially a string of N characters, denoted by A1,A2...AN. You have to print the size of the largest subsequence of string A such that all the characters in that subsequence are distinct ie. no two characters in that subsequence should be same.
>
> A subsequence of string A is a sequence that can be derived from A by deleting some elements and without changing the order of the remaining elements.
>
> —–Input—– First line contains T, number of testcases. Each testcase consists of a single string in one line. Each character of the string will be a small alphabet(ie. ’a’ to ’z’).
>
> —–Output—– For each testcase, print the required answer in one line.
>
> —–Constraints—–
>
> - $1 \leqslant T \leqslant 10$
>
> - Subtask 1 (20 points):$1 \leqslant N \leqslant 10$
>
> - Subtask 2 (80 points):$1 \leqslant N \leqslant 105$
>
> —–Example—–

Input:

2

abc

aba

Output: 3

2

—–Explanation—– For first testcase, the whole string is a subsequence which has all distinct characters.

In second testcase, the we can delete last or first ’a’ to get the required subsequence.

7.18.2 Basic Summary

You are given N string. You have to identify the duplicates and print the length of the new string as a combination of unique characters only.

—–Input—– First line contains T, number of testcases. Each testcase consists of a single string in one line. Each character of the string will be a small alphabet(ie. ’a’ to ’z’).

—–Output—– For each testcase, print the required answer in one line.

—–Constraints—–

- $1 \leqslant T \leqslant 10$

- Subtask 1 (20 points):$1 \leqslant N \leqslant 10$

- Subtask 2 (80 points):$1 \leqslant N \leqslant 105$

—–Example—–

Input:

2

abc

aba

Output: 3

2

7.18.3 Expert Summary

You have to remove duplicates and print the length of unique characters of the given string.

—–Input—– First line contains T, number of testcases. Each testcase consists of a single string in one line. Each character of the string will be a small alphabet(ie. 'a' to 'z').

—–Output—– For each testcase, print the required answer in one line.

—–Constraints—–

- $1 \leqslant T \leqslant 10$

- Subtask 1 (20 points):$1 \leqslant N \leqslant 10$

- Subtask 2 (80 points):$1 \leqslant N \leqslant 105$

—–Example—–

Input:

2

abc

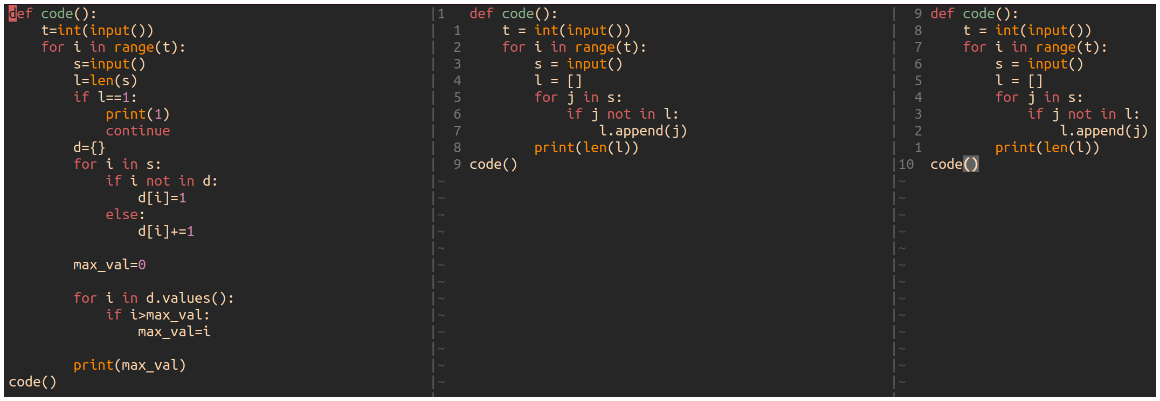
aba

Output: 3

2

7.18.4 Generated Code

The original code (left) does not accomplish the task but rather prints the count of the most frequent character. The model was unable to distinguish what the task was given the verbose prompt. However, the basic and expert summaries make the task clear and the model produces the same code. Which properly solves the challenge.



```
def code():
    t=int(input())
    for i in range(t):
        s=input()
        l=len(s)
        if l==1:
            print(1)
            continue
        d={}
        for i in s:
            if i not in d:
                d[i]=1
            else:
                d[i]+=1

        max_val=0

        for i in d.values():
            if i>max_val:
                max_val=i

        print(max_val)
code()
```

```
def code():
    t = int(input())
    for i in range(t):
        s = input()
        l = []
        for j in s:
            if j not in l:
                l.append(j)
        print(len(l))
code()
```

```
def code():
    t = int(input())
    for i in range(t):
        s = input()
        l = []
        for j in s:
            if j not in l:
                l.append(j)
        print(len(l))
code()
```

Figure 7.9: The Left Is the Code Generated by the Original Prompt. The Middle Is the Code Generated by the Expert Summary. The Right Is the Code Generated by the Basic Summary.

7.19 Made Up Information Confusing the Model

Here is an example of an interview level problem where the original prompt got 0% and the expert generated summary got 100% accuracy.

7.19.1 Original Prompt

The chef was searching for his pen in the garage but he found his old machine with a display and some numbers on it. If some numbers entered then some different output occurs on the display. Chef wants to crack the algorithm that the machine is following. Example to identify the pattern :

Input	Output
9	36
5	10
1	0
2	1

—–Input:—–

- First-line will contain T, the number of test cases. Then the test cases follow. - Each test case contains a single line of input, N.

—–Output:—–

For each test case, output in a single line answer as displayed on the screen.

—–Constraints—–

- $1 \leqslant T \leqslant 10^6$

- $1 \leqslant N \leqslant 10^6$

—–Sample Input:—–

1

7

—–Sample Output:—–

21

7.19.2 Expert Summary

Write a code to print the average of the multiplication of a given number N with N-1 integer. 1

—–Input:—–

- First-line will contain T, the number of test cases. Then the test cases follow. - Each test case contains a single line of input, N.

—–Output:—–

For each test case, output in a single line answer as displayed on the screen.

—–Constraints—–

- $1 \leqslant T \leqslant 10^6$

- $1 \leqslant N \leqslant 10^6$

—–Sample Input:—–

1

7

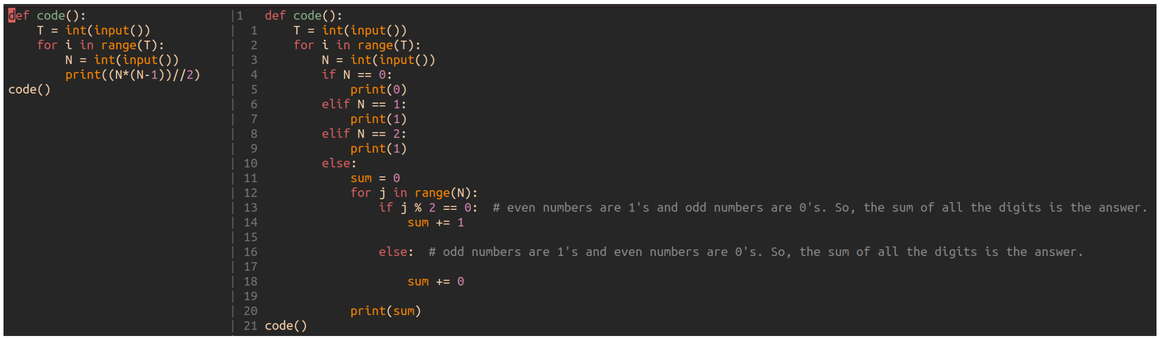
—–Sample Output:—–

21

7.20 Conclusion

We introduce a summarization-based approach [4] for efficient program synthesis. Experimental results show that the proposed approach improves the performance of the Codex

[4] https://github.com/kurbster/Prompt-Summarization



```python
def code():
    T = int(input())
    for i in range(T):
        N = int(input())
        print((N*(N-1))//2)
code()
```

```python
def code():
    T = int(input())
    for i in range(T):
        N = int(input())
        if N == 0:
            print(0)
        elif N == 1:
            print(1)
        elif N == 2:
            print(1)
        else:
            sum = 0
            for j in range(N):
                if j % 2 == 0:  # even numbers are 1's and odd numbers are 0's. So, the sum of all the digits is the answer.
                    sum += 1

                else:  # odd numbers are 1's and even numbers are 0's. So, the sum of all the digits is the answer.

                    sum += 0

            print(sum)
code()
```

Figure 7.10: The Left Is the Code Generated by the Expert Summary. The Right Is the Code Generated by the Original Prompt.

model by on average $\sim 8\%$ across various levels of programming questions provided by the APPS and $\sim 11\%$ on the CodeContests. Further, our work proposes a meta-dataset consisting of ~ 450 human-generated basic and expert-level summaries as well as $\sim 8k$ synthetically generated summaries by GPT-3 and Studio21; this can be helpful for future research on writing better instructions for the program synthesis. We show that program synthesis models benefit from concise prompts, hence, we believe that less number of high-quality instances are better than more low-quality data instances.

Data Source	**Difficulty**	**# of Problems**
Human	Introductory	145
	Interview	123
	Competition	105
	CodeContests	80
	Total	453
Studio21	Introductory	1588
	Interview	4551
	Competition	1286
	CodeContests	80
	Total	7505
GPT-3	Introductory	194
	Interview	267
	Competition	244
	CodeContests	80
	Total	785
PEGASUS	Introductory	145
	Interview	123
	Competition	105
	Total	373

Table 7.1: Statistics of the Proposed Meta-dataset.

Difficulty	AP			EWPR			BWPR		
	Baseline	Basic	Expert	Baseline	Basic	Expert	Baseline	Basic	Expert
Introductory	42.96	**50.00**	**50.00**	44.20	51.45	**51.82**	43.23	**50.35**	**50.35**
Interview	37.70	41.80	**44.26**	36.52	45.54	**46.96**	37.70	41.80	**44.26**
Competition	4.76	**5.71**	**5.71**	4.00	**6.00**	**6.00**	4.76	**5.71**	**5.71**
Weighted Average	30.47	34.83	**35.64**	30.31	36.65	**37.22**	30.43	34.78	**35.59**
CodeContests	12.50	23.75	**25.00**	13.33	25.33	**26.66**	12.82	24.36	**25.64**

Table 7.2: Results of Baseline and Proposed Model in Terms of Strict Accuracy (Sacc). The First Block Is from the Apps Dataset. The Last Block Is from the Codecontests Dataset. Ap: All Problems, Ewpr: Either Worst Problem Removal, Bwpr: Both Worst Problem Removal. All Results Are in %. Weighted Average Is Not Shown for Codecontests Because Similar Difficulties Were Not Provided (See Explanation In 7.4.1).

Difficulty	AP		EWPR	
	Baseline	Proposed	Baseline	Proposed
Introductory	42.96	**52.82**	44.53	**54.74**
Interview	37.70	**49.18**	38.66	**50.42**
Competition	4.76	**6.67**	4.81	**6.73**
Weighted Average	30.47	**38.48**	31.11	**39.44**

Table 7.3: Results When Taking the Best Summary for Each Problem. The Ewpr Baseline Is Different Because a Different Set of Problems Have Been Removed.

Model	Difficulty	AP		EWPR	
		Baseline	Proposed	Baseline	Proposed
GPT-3	Introductory	**41.75**	38.66	41.11	**41.67**
	Interview	**20.30**	18.80	18.18	**20.66**
	Competition	2.87	**3.28**	2.73	**3.64**
	Weighted Average	**20.17**	18.89	19.14	**20.55**
Studio21	Introductory	**39.53**	31.63	**39.04**	36.36
	Interview	**12.28**	11.00	10.57	**12.37**
	Competition	**1.67**	1.21	**1.38**	**1.38**
	Weighted Average	**11.53**	9.66	10.61	**10.98**
PEGASUS	Introductory	**42.96**	34.48	**44.26**	40.98
	Interview	**37.70**	10.57	14.29	**21.56**
	Competition	**4.76**	0.00	**2.76**	0.00
	Weighted Average	**30.47**	16.88	**25.50**	24.73

Table 7.4: Results of Baseline and Proposed Approach (All Results Are in %). Summaries Generated by Gpt-3, Studio21, and Pegasus Used for Inference from Apps.

Model	AP		EWPR	
	Baseline	Proposed	Baseline	Proposed
GPT-3	**12.50**	10.0	**18.75**	**18.75**
Studio21	**12.50**	8.75	**22.5**	20.0

Table 7.5: Results of Baseline and Proposed Approach (All Results Are in %). 80 Summaries Generated by Gpt-3 and Studio21 Used for Inference from Codecontests.

Summary	Difficulty	AP	EWPR	BWPR
Basic	Introductory	145	141	144
	Interview	123	113	123
	Competition	105	100	105
Expert	Introductory	145	140	144
	Interview	123	116	123
	Competition	105	100	105
StudioAI21	Introductory	215	187	-
	Interview	627	558	-
	Competition	659	578	-
GPT3	Introductory	194	180	-
	Interview	266	242	-
	Competition	244	220	-

Table 7.6: These Are the Numbers of Problems in Each Split of the Dataset. For GPT and Studio21 We Did Not Look at Problems That Were Worse or Same for Both Experiments Because There Was Insignificant Overlap Between the Two Experiments.

Experiment	Original Len	Summary Len	Orig Code Len	Summary Code Len	Code Solution Len
Summary	1147	937	339	349	671
Expert	1147	869	339	343	671
GPT	1386	1011	437	392	748
StudioAI21	1646	1114	602	473	721

Table 7.7: The Average Length of the Original/Summarized Prompt and Generated Code. The Average Length of the Code Solutions Is the Average Length of the Solutions Provided by the Creators of the Apps Dataset. A Problem Could Have One or Multiple Solutions. The Length Is Reported in Characters.

Model	Difficulty	AP		EWPR	
		Baseline	Proposed	Baseline	Proposed
GPT-3	Introductory	**41.97**	38.86	41.11	**41.67**
	Interview	25.27	**27.47**	24.86	**28.25**
	Competition	4.80	**6.40**	4.88	**6.50**
	Weighted Average	**26.60**	**26.60**	25.83	**27.71**
Studio21	Introductory	**39.91**	31.92	**39.25**	36.56
	Interview	**15.97**	14.50	13.23	**15.47**
	Competition	**2.57**	**2.57**	**2.71**	**2.71**
	Weighted Average	**16.90**	14.50	15.10	**15.64**

Table 7.8: Results When Taking the Top 500 Gpt Problems and Top 1000 Studio Problems

Chapter 8

IS A QUESTION DECOMPOSITION UNIT ALL WE NEED?

8.1 Introduction

With the advent of large LMs, we have achieved state-of-the-art performance on many NLP benchmarks (Brown *et al.*, 2020). Our benchmarks are evolving and becoming harder over time. To solve new benchmarks, we have been designing more complex and bigger LMs at the cost of computational resources, time and its negative impact on the environment. Building newer LMs for solving new benchmarks may not be an ideal and sustainable option over time. Inspired by humans, who often view new tasks as a combination of existing tasks, we explore if we can mimic humans and help the model solve a new task by decomposing (Mishra *et al.*, 2022e) it as a combination of tasks that the model excels at and already knows.

As NLP applications are increasingly more and more popular among people in their daily activities, it is essential to develop methods that involve humans in NLP-powered applications in meaningful ways. Our approach attempts to fill this gap in LMs by providing a human-centric approach to modifying data. Solving complex QA tasks such as multi-hop QA, and numerical reasoning has been a challenge for models. Question Decomposition (QD) has recently been explored to empower models to solve these tasks with the added advantage of interpretability. However, previous studies on QD are limited to some specific datasets (Khot *et al.*, 2021) such as DROP (Dua *et al.*, 2019b) and HotpotQA (Yang *et al.*, 2018). We analyze a range of datasets involving various forms of reasoning to investigate if "*a Question Decomposition Unit All We Need?*"

Figure 8.1 shows the schematic representation of a QD unit. The original question is difficult for a model to answer. However, it becomes easier for the model when a human decomposes the question into a set of simpler questions.

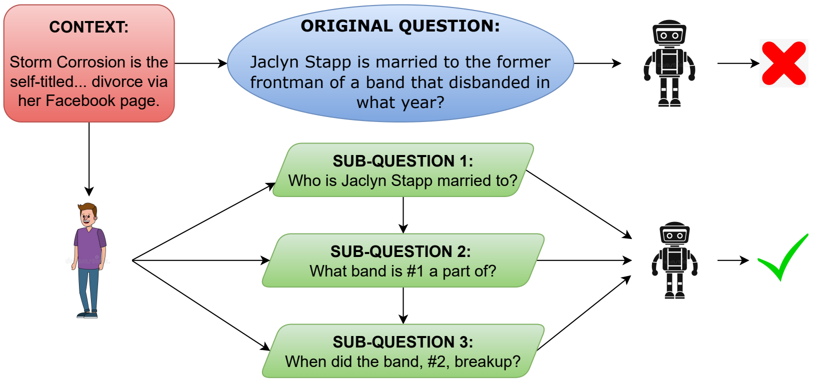

CONTEXT:
Storm Corrosion is the self-titled... divorce via her Facebook page.
ORIGINAL QUESTION:
Jaclyn Stapp is married to the former frontman of a band that disbanded in what year?
SUB-QUESTION 1:
Who is Jaclyn Stapp married to?
SUB-QUESTION 2:
What band is #1 a part of?
SUB-QUESTION 3:
When did the band, #2, breakup?

Figure 8.1: The Original Question Is Answered Incorrectly by a Model. A Human Then Decomposes the Question into a Set of Simpler Questions Which the Model Then Answers Correctly.

We manually decompose randomly selected 50 samples of each dataset. The decompositions we perform are purely based on intuitions to reduce the complexity of the question, inspired by the success of task-level instruction decomposition (Mishra *et al.*, 2022e) in improving model performance. We experiment with GPT3 (Brown *et al.*, 2020) and RoBERTA (Liu *et al.*, 2019) fine-tuned on SQuAD 2.0 (Rajpurkar *et al.*, 2018) and find that HQD significantly improves model performance (24% for GPT-3 and 29% for RoBERTa-SQuAD along with a symbolic calculator). Here, the evaluation happens on unseen tasks on which the model is not fine-tuned. Our findings indicate that Human-in-the-loop Question Decomposition (HQD) can potentially provide an alternate path to building large LMs. We hope our work will encourage the community to develop human-centric solutions that actively involve humans while leveraging NLP resources. This has been discussed further in our work (Patel *et al.*, 2022).

Name	Type
HotpotQA	Multihop RC
DROP	Mulithop RC
StrategyQA	Strategic Reasoning
MultiRC	RC
Break	RC
MathQA	Mathematical Reasoning
QASC	Fact-based Multichoice
SVAMP	Context-based Math Word Problems

Table 8.1: Type of Qa Task Corresponding to Each Dataset. RC: Reading Comprehension

8.2 Methods

8.2.1 Datasets

We select eight datasets covering a diverse set of reasoning skills and domains: (1) HotpotQA (Yang *et al.*, 2018), (2) DROP (Dua *et al.*, 2019b), (3) MultiRC (Khashabi *et al.*, 2018), (4) StrategyQA (Geva *et al.*, 2021), (5) QASC (Khot *et al.*, 2020), (6) MathQA (Amini *et al.*, 2019), (7) SVAMP (Patel *et al.*, 2021), and (8) Break (Wolfson *et al.*, 2020). Table 8.1 indicates the different task types for each dataset.

8.2.2 Decomposition Process

For each dataset, we randomly select 50 instances for manual decomposition. The question in each dataset is decomposed into two or more questions. Table 8.2, 8.3, 8.4 and 8.5 show examples of decomposition for various datasets. For each dataset, we created a set $\mathcal{D}$ for decomposed questions. Each element $\mathcal{D}_i \in \mathcal{D}$ can be represented as below:

$$\mathcal{D}_i = \{\mathcal{C}_i, \mathcal{Q}_i, \mathcal{Q}_d, \mathcal{A}_i, \mathcal{A}_d\},$$

where $\mathcal{C}_i$ is the context paragraphs, $\mathcal{Q}_i$ is the original question, $\mathcal{Q}_d$ is the set of decomposed questions, $\mathcal{A}_i$ is an original answer, and $\mathcal{A}_d$ is the set of answers for corresponding decomposed questions. For questions that require arithmetic or logical operations, we use a computational unit as suggested in Khot *et al.* (2021), which takes a decomposed question as input in the following format:

$$\{\mathcal{O}\}!\#m_1!\#m_2!....!\#m_n,$$

where $\mathcal{O}$ = {summation, difference, division, multiplication, greater, lesser, power, concat, return, remainder}, $\#m_i$ are answers of previous decomposed questions and ! separates the operands.

8.3 Experimental Setup

Models We use GPT-3 (Brown *et al.*, 2020) to generate answers for original and decomposed questions. To show that QD significantly improves performance even on simpler models, we use RoBERTa-base finetuned on SQuAD 2.0 dataset (i.e., RoBERTa-SQuAD). Additionally, we use RoBERTa-base finetuned on BoolQ dataset (Clark *et al.*, 2019) (i.e., RoBERTa-BoolQ) for original and decomposed questions in StrategyQA since they are True/False type questions.

Experiments To create baselines, we evaluate all models on the original question along with the context. We evaluate all models on the manually decomposed questions in the proposed method. We carry out all experiments in GPT-3 by designing prompts for each dataset. For RoBERTa-based models, we use RoBERTa-SQuAD for MultiRC, Break, HotpotQA and DROP datasets, since SQuAD 2.0 is designed for a reading comprehension task. For StrategyQA, we use two RoBERTa-base models: (1) RoBERTa-BoolQ, which is used to answer the final boolean type of questions, and (2) RoBERTa-SQuAD which is used to answer the remaining decomposition questions. For SVAMP, we use the RoBERTa-SQuAD model to extract the necessary operands using decomposed questions and then we use the computational module to perform various operations. In all experiments, we use decomposition to get to the final answer sequentially.

Metrics For all our experiments, we use Rouge-L (Lin, 2004), F_1-score and Exact Match (EM) as the evaluation metrics.

8.4 Results and Analysis

Here, we divide our datasets into four categories: (1) RC: HotpotQA, DROP, MultiRC, and Break in Reading Comprehension (RC), (2) MATH: MathQA and SVAMP in Mathematical reasoning , (3) MC: QASC in Multi-Choice QA (MC) , and (4) SR: StrategyQA in Strategy Reasoning (SR). All results presented in these sections are averaged over tasks for each category.

8.4.1 Experimental Results

GPT-3 Figure 8.3 shows the GPT-3 performance in terms of average F_1-scores for each category. From the Figure 8.3, we can observe that our proposed approach outperforms baseline by $\sim 24\%$.

RoBERTa Figure 8.2 represents the results we obtain using RoBERTa-based models in terms of F_1-scores for each category. On an average, we achieve $\sim 29\%$ of significant improvement compared to the baseline.

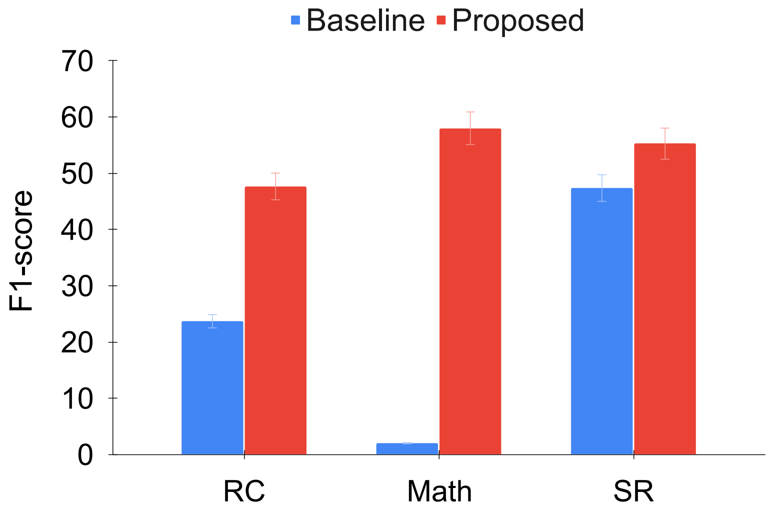


Figure 8.2: Results in Terms of f_1-score Across Different Categories for Roberta-based Models. RC: Reading Comprehension, Math: Mathematical Reasoning, SR: Strategy Reasoning.

8.4.2 Analysis

Customized Question Decomposition for Each Model There can be multiple ways to decompose a question based on the context. Multiple factors go into deciding how to break down a question. One factor is the strength of the model. For instance, if we use a model finetuned on SQuAD, it might be beneficial to ensure that the decompositions are more granular and are generated to answer from a context span. On the other hand, if we have a more sophisticated model like GPT3, we might not necessarily need to do so. The results shown in Figure 8.2 are obtained on RoBERTa finetuned on SQuAD by using decompositions originally designed for GPT3; note that in this case, the answers to the decompositions might not always be the span of a particular sentence in the context. However, we achieve a decent performance improvement. We believe the performance gain will be greater if decompositions are designed to match the model's strengths.

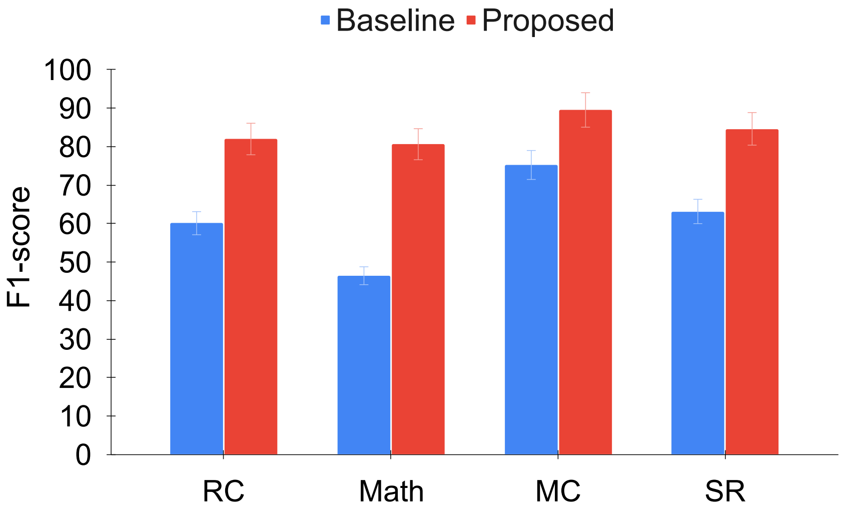


Figure 8.3: Results in Terms of f_1-score Across Different Categories for GPT-3. RC: Reading Comprehension, Math: Mathematical Reasoning, MC: Multi-choice Qa, SR: Strategy Reasoning.

Qualitative Analysis We conduct qualitative analysis to capture the evaluation aspects missed in the automated evaluation metrics. Here, we manually inspect and consider a generated answer to be correct if it is semantically similar to the gold annotation. Figure 8.4 and 8.5 show the contribution of QD in correcting model prediction. We observe that the decompositions correct more than 60% of the errors made on the original questions.

Error Analysis We conduct error analysis and observe that the major source of error is the error propagated from one of the decomposed questions. Errors, in general, are of two types: (i) incorrect span selection and (ii) failure to collect all possible answers in the initial step of decomposition; this often omits the actual correct answer leaving no room for later decomposition units to generate the correct answer. Errors occur in QASC because our method of context-independent decomposition (via intuition) sometimes leads to open-ended questions which models find hard to answer.

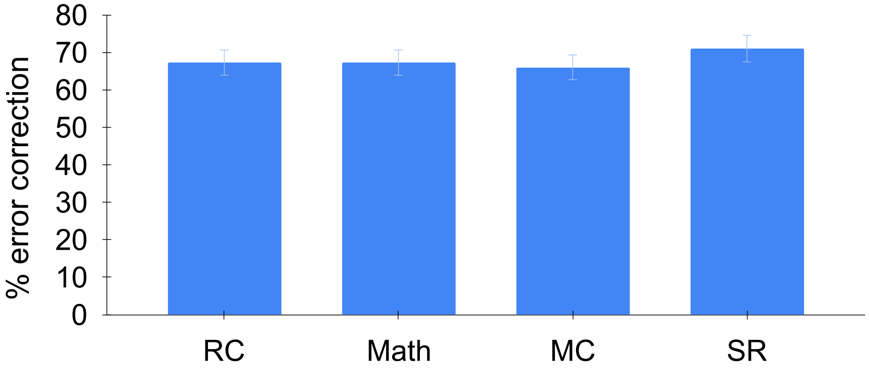


Figure 8.4: % Error Correction by Using Decompositions With GPT3

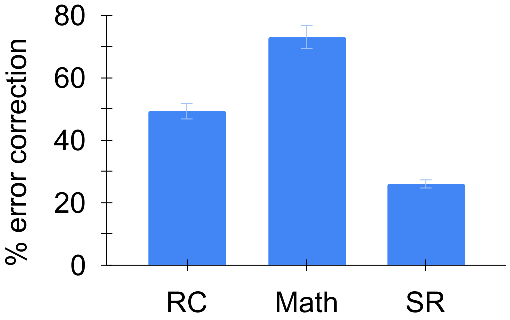


Figure 8.5: % Error Correction by Using Decompositions With RoBERTa

Effect of Decomposition on Math Datasets We observe that Math datasets benefit the most from decomposition. This may be because of two reasons: 1) majority of math questions can be decomposed as a combination of extractive QA (where the answer is a span) and a symbolic calculation. Both of these are strengths of language models (note that we use calculators that provide accurate answers consistently). However, this is not necessarily true in case of other QA tasks. In a decomposition chain, if the answer in one step goes wrong, it propagates till the end and the final prediction becomes wrong. 2) language models by default struggle to do math tasks Patel *et al.* (2021); Mishra *et al.* (2022g), so the performance improvement seems more prominent there.

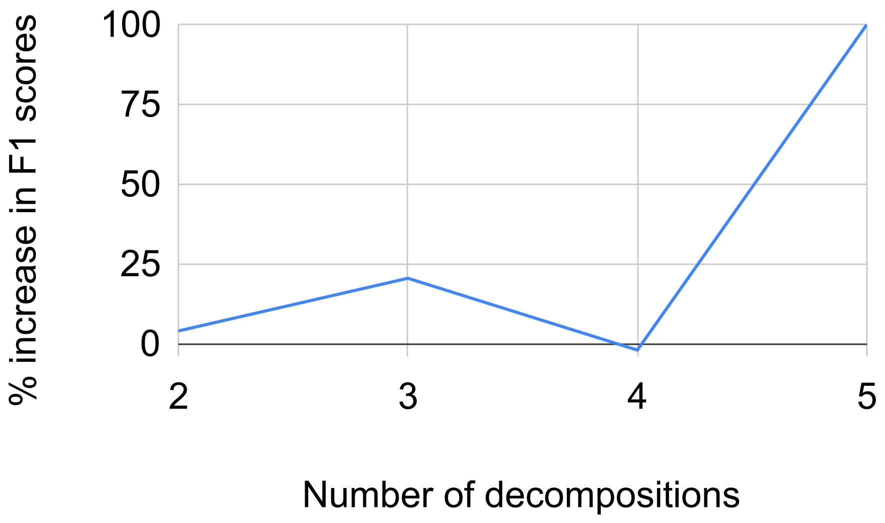


Figure 8.6: Performance Improvements in f_1 Scores for Questions with 2, 3, 4 and 5 Decompositions.

Effect of Number of Decompositions on Results We typically decompose a question based on the number of operations associated with it (e.g. mathematical calculation or single hop operation). Increase in the number of decompositions has the advantage that it simplifies the original question, but it can also have the disadvantage that if the answer to one of the questions in the chain is incorrect, the end answer becomes incorrect. This is also evident from our empirical analysis on HOTPOTQA and SVAMP datasets where we observe that there is no direct correlation between the number of labeling QA and the final performance. Figure 8.6 shows the variation in model performance improvement observed for questions with 2, 3, 4 and 5 decompositions.

Efforts to Automate Decomposition For HotpotQA, DROP, and SVAMP, we attempt to automate the decomposition process using GPT3. A limitation for generating decompositions for HotpotQA is that the context length makes it difficult to provide sufficient examples in prompt. With DROP and SVAMP, we observe that GPT-3 often generates incorrect arithmetic operations for the last sub-question. It also often fails to develop coherent decompositions of the questions. We also finetune a BART-base (Lewis *et al.*, 2019) model on our handwritten decompositions. However, the model overfits and fails to produce meaningful decompositions, probably due to the limited number of training samples.

8.5 Prompts

Due to the success of large LMs, prompt-based learning is becoming popular to achieve generalization and eliminate the need of creating task-specific models and large scale datasets (Liu *et al.*, 2021b). Recently, instructional prompts have been pivotal in improving the performance of LMs and achieving zero-shot generalization (Mishra *et al.*, 2022f; Wei *et al.*, 2022a; Sanh *et al.*, 2022; Wei *et al.*, 2022b; Ouyang *et al.*, 2022; Parmar *et al.*, 2022b; Puri *et al.*, 2022; Kuznia *et al.*, 2022; Luo *et al.*, 2022). We present the instructional prompts that we used to generate answers for various datasets.

8.5.1 HotpotQA, DROP, Break

Given a context, answer the question using information and facts present in the context. Keep the answer short.

Example:

Input:

Mehmed built a fleet to besiege the city from the sea .Contemporary estimates of the strength of the Ottoman fleet span between about 110 ships , 145 , 160 , 200-250 to 430 . A more realistic modern estimate predicts a fleet strength of 126 ships comprising 6 large galleys, 10 ordinary galleys, 15 smaller galleys, 75 large rowing boats, and 20 horse-transports.:44 Before the siege of Constantinople, it was known that the Ottomans had the ability to cast medium-sized cannons, but the range of some pieces they were able to field far surpassed the defenders' expectations. Instrumental to this Ottoman advancement in arms production

was a somewhat mysterious figure by the name of Orban , a Hungarian .:374 One cannon designed by Orban was named "Basilica" and was 27 feet long, and able to hurl a 600lb stone ball over a mile .
Question: How many ordinary galleys and large rowing boats is estimated from the fleet strength? Output:
Answer: 85

Input:
Context: «CONTEXT»
Question: «QUESTION»
Output:
Answer: «OUTPUT GENERATED BY GPT3»

8.5.2 MATHQA

Prompt for the original question:
Given a problem and 5 options, return the correct option. In order to choose the correct option, you will have to perform some mathematical operations based on the information present in the problem. Look at the examples given below to understand how to answer.

Input: Problem: the volume of water inside a swimming pool doubles every hour . if the pool is filled to its full capacity within 8 hours , in how many hours was it filled to one quarter of its capacity
Options: a) 2, b) 4, c) 5, d) 6, e) 7
Output:
Answer: 6

HotpotQA	***Context***: The Larkspur Press is a small letter-press publisher based in Monterey, Kentucky .... , The film also features appearances by Helen Keller, Anne Sullivan, Kate Adams Keller and Phillips Brooks Keller as themselves. The movie was directed by George Foster Platt and written by Francis Trevelyan Miller. ***Original Question***: Are John O'Hara and Rabindranath Tagore the same nationality? ***True Answer***: no ***Decomposed Question 1***: What is John O'Hara's nationality? ***Generated Answer***: American ***Decomposed Question 2***: What is Rbindranath Tagore's nationality? ***Generated Answer***: Indian ***Decomposed Question 3***: Is #1 and #2 the same nationality? ***Generated Answer***: No
DROP	***Context***: Mehmed built a fleet to besiege the city from the sea .... and able to hurl a 600 lb stone ball over a mile . ***Original Question***: How many ordinary galleys and large rowing boats is estimated from the fleet strength? ***True Answer***: 85 ***Decomposed Question 1***: How many ordinary galleys were there? ***Generated Answer***: 10 ***Decomposed Question 2***: How many large rowing boats were there? ***Generated Answer***: 75 ***Decomposed Question 3***: summation ! #1 ! #2 ***Generated Answer***: 85

Table 8.2: Examples for DROP And HotpotQA.

Input:

Problem: a train 200 m long can cross an electric pole in 5 sec and then find the speed of the train ?

Options: a) 114 , b) 124 , c) 134 , d) 144 , e) 154

Output:

Answer: 144

Input:

Problem: «Problem»

Options: «options»

Output:

Answer: «OUTPUT GENERATED BY GPT3»

8.5.3 SVAMP

Prompt used for both decomposed questions and original questions. The examples contain both decomposed type questions and original type questions.

Given some context, answer a given question. Use the examples given below as reference.

Example 1:

Input:

Context: It takes 4.0 apples to make 1.0 pie.

Question: How many apples does it take to make 504.0 pies?

Output:

Answer: 2016

Example 2:

Input:

Context: Mary is baking a cake.The recipe calls for 7.0 cups of flour and 3.0 cups of sugar.She already put in 2.0 cups of flour.

Question: How many cups of flour did recipe called?

Output:

Answer: 7

Example 3:

Input:

Context: Each pack of dvds costs 76 dollars. If there is a discount of 25 dollars on each pack Question: How much is each pack of dvds without the discount?

Output:

Answer: 76

Example 4:

Input:

Context: Conner has 25000.0 dollars in his bank account.Every month he spends 1500.0 dollars.He does not add money to the account.

Question: How many dollars Conner spends every month?

Output:

Answer: 1500

Input:

Context: <CONTEXT»

Question: «QUESTION»

Output:

Answer: «OUTPUT GENERATED BY GPT3»

SVAMP	***Context***: Bryan took a look at his books as well.If Bryan has 56.0 books in each of his 9.0 bookshelves. ***Original Question***: How many books does he have in total? ***Answer***: 504.0 ***Decomposed Question 1***:How many books in each bookshelf? ***Answer***: 56.0 ***Decomposed Question 2***:How many bookshelves? ***Answer***: 9.0 ***Decomposed Question 3***: multiplication ! #1 ! #2 ***Answer***: 504.0
MATHQA	***Problem***: if a train , travelling at a speed of 180 kmph , crosses a pole in 6 sec , then the length of train is ? ***Options***: a) 300 , b) 125 , c) 288 , d) 266 , e) 121 ***Annotated Formula***: multiply(multiply(180, const_0.2778), 6) ***Answer***: 300 ***Generated Answer***: 266 ***Decomposed Question 1***: multiplication ! 0.2778 ! 180 ***Answer***: 50.004 ***Decomposed Question 2***: multiplication ! 50.004 ! 6 ***Answer***: 300

Table 8.3: Decomposition Examples for Svamp and Mathqa. We Use the Annotated Formula Presented in the Dataset to Make Our Decompositions.

8.5.4 StrategyQA

Input:

Context: A melodrama is a dramatic work ...The passengers' response to the hijacking has come to be invested with great moral significance.

Question: What do tearjerkers refer to?

Output:

Answer: a story, song, play, film, or broadcast that moves or is intended to move its audience to tears.

Input:

Context: The purpose of the course is learning to soldier as ... The main motor symptoms are collectively called "parkinsonism", or a "parkinsonian syndrome".

Question: True or False: Could someone experiencing A tremor, or shaking, Slowed movement (bradykinesia), Rigid muscles, Impaired posture and balance, Loss of automatic movements, Speech changes, Writing changes. complete Volunteer for assignment and be on active duty. Have a General Technical (GT) Score of 105 or higher

Output:

Answer: False

Input:

Context: The Scientific Revolution was a series of events that marked the emergence of modern science during the early modern period, ...The first-generation iPhone was released on June 29, 2007, and multiple new hardware iterations with new iOS releases have been released since.

Question: True or False: Did 1543 occur before 2007?

Output:

Answer: False

Input:

Context: «CONTEXT»

Question: «QUESTION»

Output:

Answer: «OUTPUT GENERATED BY GPT3»

8.5.5 QASC

Prompt for original question:

Answer the given question. The question contains options A-H, choose and return the correct option. Look at the examples given below.

Input:

What are the vibrations in the ear called? (A) intensity (B) very complex (C) melanin content (D) lamphreys (E) Otoacoustic (F) weater (G) Seisometers (H) trucks and cars

Output:

Answer: Otoacoustic

Input:

«QUESTION»

Output:

Answer: «OUTPUT GENERATED BY GPT3»

Prompt for decomposed question:

Given a yes or no question, return yes if the answer is yes. Otherwise return no.
«QUESTION»
Answer: «OUTPUT GENERATED BY GPT3»

StrategyQA	***Context***: Mail carriers, also referred to as mailmen or letter carriers, … Clothing also provides protection from ultraviolet radiation.
	Original Question: True or False: Mail carriers need multiple uniforms.
	Original Answer: True
	Generated Answer: False
	Decomposed Question 1: What seasons do mail carriers work through?
	Generated Answer: All seasons
	Decomposed Question 2: True or False: In order to make it through all of #1, one needs multiple clothing pieces.
	Generated Answer: True
QASC	***Original Question***: what kind of beads are formed from vapor condensing? (A) h2o (B) H20 (C) tiny (D) carbon (E) hydrogen (F) rain (G) oxygen (H) Dew
	Answer: h2o
	Decomposed Question 1: Are #1 beads formed from vapor condensing?
	Answer: yes

Table 8.4: Examples of Decompositions for Strategyqa and Qasc Datasets. For Each Option in Qasc, #1 Is Replaced with the Option and Posed to Gpt-3 as a Yes or No Question.

8.5.6 MultiRC

Given a context-question pair, answer the question using information and facts present in the context. Keep your answers as short as possible.

Example:

Input:

Context: Should places at the same distance from the equator have the same climate? You might think they should. Unfortunately, you would not be correct to think this. Climate types vary due to other factors besides distance from the equator. So what are these factors? How can they have such a large impact on local climates? For one thing, these factors are big. You may wonder, are they as big as a car. Think bigger. Are they bigger than a house? Think bigger. Are they bigger than a football stadium? You are still not close. We are talking about mountains and oceans. They are big features and big factors. Oceans and mountains play a huge role in climates around the world. You can see this in Figure above. Only one of those factors is latitude, or distance from the equator.

Question: Name at least one factor of climate

Output:

Answer: Oceans

Example:

Input:

Context: Earth processes have not changed over time. The way things happen now is the same way things happened in the past. Mountains grow and mountains slowly wear away. The same process is at work the same as it was billions of years ago. As the environment changes, living creatures adapt. They change over time. Some organisms may not be able to adapt. They become extinct. Becoming extinct means they die out completely. Some geologists study the history of the Earth. They want to learn about Earths past. They use clues from rocks and fossils. They use these clues to make sense of events. The goal is to place things in the order they happened. They also want to know how long it took for those events to happen.

Question: What is one example of how the earth's processes are the same today as in the past?

Output:

Answer: Things develop and then wither away

Input:

Context:: «CONTEXT»

Question: «QUESTION»

Output:

Answer: <ANSWER GENERATED BY GPT3»

8.6 Error Examples

This section discusses the errors generated by using decompositions. We observe two types of errors while answering decomposed questions. The final answer is wrong because previous sub-questions were answered incorrectly either because such a question has multiple correct answer, or simply because the model could not understand the question correctly.

Context: ... Roger David Casement (1 September 1864 - 3 August 1916), formerly known as Sir Roger Casement .... In collaboration with Roger Casement, Morel led a campaign against slavery in the Congo Free State, founding the Congo Reform Association .... The association was founded in March, 1904, by Dr. Henry Grattan Guinness (1861-1915),

MultiRC	***Context***: Sometimes a full Moon moves through Earths shadow. ... The Moon glows with a dull red coloring during a total lunar eclipse. ***Original Question***: Is it more common for the Moon to travel completely in the Earth's umbra or only partially? ***List of correct answers***: Partially, A total eclipse is less common than partial so it is more common for the moon to travel partially in Earth's umbra ***Decomposed Question 1***:When does the Moon travel's completely in Earth's umbra? ***Answer***: total lunar eclipse ***Decomposed Question 2***:When does the Moon travel's partially in Earth's umbra? ***Answer***: partial lunar eclipse ***Decomposed Question 3***: Which is more common #1 or #2? ***Answer***: partial lunar eclipse ***Decomposed Question 4***: Does the Moon travel partially or completely in #3? ***Answer***: partially

Table 8.5: Decomposition Examples for Multirc. Multirc Has Multiple Correct Answer and the Final Correct Answer Which Gives the Best Metrics for the Generated Answer Is Chosen as the Correct Answer Corresponding to the Generated Answer.

Edmund Dene Morel, and Roger Casement ...

Question:

When was the date of birth of one of the founder of Congo Reform Association?

True Answer: 1 September 1864

Generated Answer: 18 October 1914

Decomposed Question 1:

Who is the founder of the Congo Reform Association?

True Answer: Roger Casement

Generated Answer: Henry Grattan Guinness

Decomposed Question 2: When was #1 born?

True Answer: 1 September 1864

Generated Answer: 1861

Above is an example from HotpotQA. As can be seen from the context, Congo Reform Association had multiple founders. GPT3 did give a correct answer among a set of correct answers whereas the ground truth answer provided by the dataset was some other correct option.

Below is an example of incorrect retrieval. The answer generated for the first decomposed question incorrectly returns cities taken by Ottomans as well instead of just the Venetians. Hence, the final decomposed questions return the incorrect count.

Context: In the Morean War, the Republic of Venice besieged Sinj in October 1684 and then again March and April 1685, but both times without success. In the 1685 attempt, the Venetian armies were aided by the local militia of the Republic of Poljica, who thereby rebelled against their nominal Ottoman suzerainty that had existed since 1513. In an effort to retaliate to Poljica, in June 1685, the Ottomans attacked Zadvarje, and in July 1686 Dolac and Srijane, but were pushed back, and suffered major casualties. With the help of the local population of Poljica as well as the Morlachs, the fortress of Sinj finally fell to the Venetian army on 30 September 1686. On 1 September 1687 the siege of Herceg Novi started, and ended with a Venetian victory on 30 September. Knin was taken after a twelve-day siege on 11 September 1688. The capture of the Knin Fortress marked the end of the successful Venetian campaign to expand their territory in inland Dalmatia, and it also determined much of the final border between Dalmatia and Bosnia and Herzegovina that stands today. The Ottomans would besiege Sinj again in the Second Morean War, but would be repelled. On 26 November 1690, Venice took Vrgorac, which opened the route towards Imotski and Mostar. In 1694 they managed to take areas north of the Republic of Ragusa, namely Čitluk, Gabela, Zažablje, Trebinje, Popovo, Klobuk and Metković. In the final peace treaty, Venice did relinquish the areas of Popovo polje as well as Klek and Sutorina, to maintain the pre-existing demarcation near Ragusa.

Question:

How many cities did Venice try to take?

True Answer: 10

Generated Answer: 3

Decomposed Question 1:

Which cities did Venice try to take?

True Answer: Sinj, Knin, Vrgorac, Čitluk, Gabela, Zažablje, Trebinje, Popovo, Klobuk and Metković

Generated Answer: Sinj, Zadvarje, Dolac, Srijane, Knin, Vrgorac, Čitluk, Gabela, Zažablje, Trebinje, Popovo, Klobuk and Metković

Decomposed Question 2: What is the count of the cities mentioned in #1?

True Answer: 10

Generated Answer: 14

The samples for QASC are provided without context. Without the context, the answers to some of the decomposed questions can be open ended. Certain options can be unambiguously wrong and some are unambiguously correct. Below is an example:

Question: What can knowledge of the stars be used for? (A) travel (B) art (C) as a base (D) safety (E) story telling (F) light source (G) vision (H) life

True Answer: travel

Generated Answer: art

Decomposed Question: Can the knowledge of stars be used for the following: #?

The decomposed question for each option is posed as a yes or no question to GPT3. It returns yes for art and story telling but not for travel.

8.7 Examples, Results and Details for Automation

We attempt to automate the process of decomposition using GPT3. We use the examples from manual decomposition in the prompts given to GPT3, some of which are presented below. The results obtained from the experiments are presented in Table 8.6. The generated decompositions are answered using RoBERTa-base finetuned on SQuAD 2.0 dataset.

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
HotpotQA	**32.68**	14.12	**29.50**	11.47	**33.29**	14.00
DROP	**22.8**	3.77	**21.69**	3.77	**23.4**	3.76
SVAMP	7.4	**17.35**	7.4	**17.35**	7.4	**17.35**
Average	**20.96**	11.74	**19.53**	10.86	**21.36**	11.70

Table 8.6: Results Obtained by Using Decomposed Questions Generated Using GPT3

In this section, we present the prompts we used while attempting to automatically generate decomposed questions using GPT3.

The prompt for generating decompositions for DROP was as follows:

Decompose a given question by breaking it into simpler sub-questions. The answer to each subsequent sub-question should lead towards the answer of the given question. To do so, use the context provided and look at the examples. Here are some helpful instructions:

1. If the given question compares two things, best strategy is to generate sub-questions that finds the answer to each of those things and compare them in the last sub-question.
2. Some sub-questions must contain phrases like "answer of sub-question 1".
3. If a sub-question is an arithmetic operation, then the sub-question should be framed as operation ! "answer of sub-question 1" ! "answer of sub-question 2".
4. The operation used in 3) is always one of the following: summation, difference, greater, lesser.

Example 1:
Context: Mehmed built a fleet to besiege the city from the sea .Contemporary estimates of the strength of the Ottoman fleet span between about 110 ships , 145 , 160 , 200-250 to 430 . A more realistic modern estimate predicts a fleet strength of 126 ships comprising 6 large galleys, 10 ordinary galleys, 15 smaller galleys, 75 large rowing boats, and 20 horse-transports.:44 Before the siege of Constantinople, it was known that the Ottomans had the ability to cast medium-sized cannons, but the range of some pieces they were able to field far surpassed the defenders' expectations. Instrumental to this Ottoman advancement in arms production was a somewhat mysterious figure by the name of Orban , a Hungarian. One cannon designed by Orban was named B̈asilicaänd was 27 feet long, and able to hurl a

600 lb stone ball over a mile .

Question: How many ordinary galleys and large rowing boats is estimated from the fleet strength?

Sub-question 1: How many ordinary galleys were there?

Sub-question 2: How many large rowing boats were there?"

Sub-question 3: summation ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 2:

Context: As of the census of 2000, there were 14,702 people, 5,771 households, and 4,097 families residing in the county. The population density was 29 people per square mile (11/km²). There were 7,374 housing units at an average density of 14 per square mile (6/km²). The racial makeup of the county was 98.02% Race (United States Census), 0.69% Race (United States Census) or Race (United States Census), 0.35% Race (United States Census), 0.11% Race (United States Census), 0.05% Race (United States Census), 0.08% from Race (United States Census), and 0.71% from two or more races. 0.44% of the population were Race (United States Census) or Race (United States Census) of any race.

Question: How many more people than households are reported according to the census?

Sub-question 1: As of the 2000 census, how many people are residing in the country?

Sub-question 2: As of the 2000 census, how many households are reported?

Sub-question 3: difference !"answer of sub-question 1" ! "answer of sub-question 2"

Example 3: Context: As of the census of 2000, there were 49,129 people, 18,878 households, and 13,629 families residing in the county. The population density was 88 people per square mile (34/km2). There were 21,779 housing units at an average density of 39 per square mile (15/km2). The racial makeup of the county was 74.4% Race (United States Census), 20.4% Race (United States Census) or Race (United States Census), 0.60% Race (United States

Census), 1.1% Race (United States Census), 0.15% Race (United States Census), 1.3% from Race (United States Census), and 2.2% from two or more races. 3.4% of the population were Race (United States Census) or Race (United States Census) of any race. 2.85% of the population reported speaking Spanish language at home, while 1.51% speak German language. Question: How many more people are there than families? Sub-question 1: How many people are there in the 2000 census? Sub-question 2: How many families are recorded in the 200 census? Sub-question 3: difference ! "answer of sub-question 1" ! "answer of sub-question 2"

Context: «CONTEXT»

Question: «QUESTION»

«OUTPUT GENERATED BY GPT3»

The prompt for HotpotQA was similar, except replacing the examples with instances from HotpotQA. For SVAMP, since the context was much smaller, we could give more examples. The prompt for SVAMP is as shown below:

Decompose a given question by breaking it into simpler sub-questions. The answer to each subsequent sub-question should lead towards the answer of the given question. To do so, use the context provided and look at the examples.

Here are some helpful instructions:

1. If the given question compares two things, best strategy is to generate sub-questions

that finds the answer to each of those things and compare them in the last sub-question, 2) Some sub-questions must contain phrases like "answer of sub-question 1".

2. Some sub-questions must contain phrases like "answer of sub-question 1".

3. If a sub-question is an arithmetic operation, then the sub-question should be framed as operation ! "answer of sub-question 1" ! "answer of sub-question 2".

4. The operation used in 3) is always one of the following: summation, difference, greater, lesser

Example 1:
Context: Jessica had 8.0 quarters in her bank . Her sister borrowed 3.0 of her quarters. How many quarters does Jessica have now?
Sub-question 1: How many quarters did Jessica have in her bank initially?
Sub-question 2: How many quarters did Jessica's sister borrow?
Sub-question 3: difference ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 2:
Context: Shawn has 13.0 blocks. Mildred has with 2.0 blocks. Mildred finds another 84.0. How many blocks does Mildred end with?
Sub-question 1: How many blocks does Mildred start with?
Sub-question 2: How many blocks does Mildred find?
Sub-question 3: summation ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 3:
Context: Dave was helping the cafeteria workers pick up lunch trays, but he could only carry 9.0 trays at a time. If he had to pick up 17.0 trays from one table and 55.0 trays from

another. how many trips will he make?

Sub-question 1: How many trays did Dave have to pick up from the first table?

Sub-question 2: How many trays did Dave have to pick up from the second table?

Sub-question 3: summation ! "answer of sub-question 1" ! "answer of sub-question 2"

Sub-question 4: How many lunch trays could Dave carry at a time?

Sub-question 5: division ! "answer of sub-question 3" ! "answer of sub-question 4"

Example 4:

Context: Paco had 93.0 cookies. Paco ate 15.0 of them. How many cookies did Paco have left?

Sub-question 1: How many cookies did Paco start with?

Sub-question 2: How many cookies did Paco eat?

Sub-question 3: difference ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 5:

Context: 43 children were riding on the bus. At the bus stop some children got off the bus. Then there were 21 children left on the bus. How many children got off the bus at the bus stop?

Sub-question 1: How many children were on the bus at the beginning?

Sub-question 2: How many children were left on the bus?

Sub-question 3: difference ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 6:

Context: 28 children were riding on the bus. At the bus stop 82 children got on the bus while some got off the bus. Then there were 30 children altogether on the bus. How many more children got on the bus than those that got off?

Sub-question 1: How many children were on the bus at the beginning?

Sub-question 2: How many children were left on the bus?

Sub-question 3: difference ! "answer of sub-question 1" ! "answer of sub-question 2"

Example 7:

Context: They decided to hold the party in their backyard. If they have 11 sets of tables and each set has 13 chairs, how many chairs do they have in the backyard?

Sub-question 1: How many tables are there in the backyard?

Sub-question 2: How many chairs are on each table?

Sub-question 3: multiplication ! "answer of sub-question 1" ! "answer of sub-question 2"

Context: «CONTEXT + QUESTION»

The examples of decompositions generated for HotpotQA, DROP and SVAMP are shown in Table 8.7

8.8 Results

We tabulate the results we get for all the datasets for baseline and our proposed mechanism.

8.9 Conclusion

In this chapter, we argue that the recent trend of building large LMs may not be sustainable to solve evolving benchmarks. We believe that modifying data samples can significantly help the model improve performance. We study the effect of Question Decomposition (QD) on a diverse set of tasks. We decompose questions [1] and significantly improve model

[1] https://github.com/Pruthvi98/QuestionDecomposition

performance (24% for GPT3 and 29% for RoBERTa-SQuAD along with a symbolic calculator). Our findings indicate that Human-in-the-loop Question Decomposition (HQD) can potentially provide an alternate path to building large LMs. Our approach provides a viable option to involve people in NLP research. We hope our work will encourage the community to develop human-centric solutions that actively involve humans while leveraging NLP resources.

DROP	***Context***: Hoping to rebound from their loss to the Patriots, the Raiders stayed at home for a Week 16 duel with the Houston Texans. ... The Texans tried to rally in the fourth quarter as Brown nailed a 40-yard field goal, yet the Raiders' defense would shut down any possible attempt. ***Original Question***: How many yards longer was the longest passing touchdown than the shortest? ***Decomposed Question 1***: What was the length of the shortest touchdown pass? ***Decomposed Question 2***:What was the length of the longest touchdown pass? ***Decomposed Question 3***: greater ! #1 ! #2
DROP	***Context***: In 1085, Guadalajara was retaken by the Christian forces of Alfonso VI . The chronicles say that the Christian army was led by Alvar Fanez de Minaya, one of the lieutenants of El Cid. From 1085 until the Battle of Las Navas de Tolosa in 1212, the city suffered wars against the Almoravid and the Almohad Empires. In spite of the wars, the Christian population could definitely settle down in the area thanks to the repopulation with people from the North who received their first fuero in 1133 from Alfonso VII.In 1219, the king Fernando III gave a new fuero to the city .During the reign of Alfonso X of Castile, the protection of the king allowed the city to develop its economy by protecting merchants and allowing markets. ***Original Question***: When did the first battle against Guadalajara take place? ***Decomposed Question 1***: When was Guadalajara retaken by the Christian forces? ***Decomposed Question 2***:Who led the Christian army? ***Decomposed Question 3***: #1 ! #2

Table 8.7: Decompositions for DROP Generated Using GPT3

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
HotpotQA	71.97	**78.53**	70	**76**	73.33	**79.93**
DROP	52.97	**78.16**	46.87	**75.86**	46.72	**77.66**
MultiRC	64.39	**80.74**	33.33	**55.55**	61.24	**77.31**
BREAK	66.81	**84.54**	58	**74**	62.30	**78.56**
Average	60.10	**81.97**	52.64	**76.26**	59.35	**81.10**

Table 8.8: Comparison of Metrics for Reading Comprehension Datasets Between Gpt3 Baseline And Decompose_GPT3

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
MATH	31.1	**82.5**	27.44	**82.22**	23.4	**80.85**
SVAMP	61.80	**78.75**	58.88	**77.5**	55	**77.5**
Average	46.45	**80.62**	43.16	**79.86**	39.2	**79.17**

Table 8.9: Comparison of Metrics for Mathematical Reasoning Datasets Between GPT3 Baseline and Decompose_GPT3

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
StrategyQA	63.15	**84.61**	63.15	**84.61**	63.15	**84.61**
QASC	75.23	**89.52**	75.23	**89.52**	71.4	**85.71**
Average	69.19	**87.06**	69.19	**87.06**	67.27	**85.16**

Table 8.10: Comparison of Metrics for Strategyqa (Strategic Reasoning) and Qasc (Fact-based Multichoice) Between GPT3 Baseline and Decompose_GPT3

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
HotpotQA	32.14	**49.50**	26	**42**	33.33	**50.72**
DROP	25.56	**66.14**	25	**62.5**	25.56	**66.14**
MultiRC	45.74	**48.1**	24.44	**28.88**	44.83	**46.95**
BREAK	24.6	**36.17**	18	**28**	24.31	**35.5**
Average	23.68	**47.65**	20.26	**43.96**	28.76	**50.74**

Table 8.11: Comparison of Metrics for Reading Comprehension Datasets Between Baseline and Decompose Settings Using Roberta-base Finetuned on Squad.

Dataset	F1		EM		Rouge-L	
	Baseline	Decompose	Baseline	Decompose	Baseline	Decompose
StrategyQA	47.36	**55.26**	47.36	**55.26**	47.36	**55.26**
SVAMP	2	**58**	2	**58**	2	**58**
Average	24.68	**56.63**	24.68	**56.63**	24.68	**56.63**

Table 8.12: Comparison of Metrics for Strategyqa and Svamp Between Baseline and Decompose Settings Using Roberta-base Finetuned on Squad. For Strategyqa, Roberta-base Squad Is Used to Answer Intermediate Decompositions Whereas Roberta-base Finetuned on Boolq Is Used to Answer the Original Question and the Final Decomposed Question

Chapter 9

HOW MANY DATA SAMPLES IS AN ADDITIONAL INSTRUCTION WORTH?

9.1 Introduction

Large scale benchmarks such as Imagenet (Russakovsky *et al.*, 2015), SQuAD (Rajpurkar *et al.*, 2018) and architectural development in models such as CNNs (Amari *et al.*, 2003) and transformers (Vaswani *et al.*, 2017) have propelled our progress in deep learning. However, creating high quality benchmarks by controlling artifacts (Gururangan *et al.*, 2018), developing new models and training them are hard for non-expert users. Recently introduced *instruction-paradigm* empowers non-expert users, practitioners and domain experts in other fields to leverage NLP resources (Weller *et al.*, 2020) as they now can describe their tasks in natural language without requiring to create task-specific datasets or developing models. Even though instruction-paradigm has led to development of models that significantly outperform multitasking baselines, model performance has remained far behind the supervised learning model trained with task-specific data (Efrat and Levy, 2020; Mishra *et al.*, 2022f).

Non-expert users can write multiple instructions per task each of which covers multiple perspectives spanning over a variety of linguistic features; many of these can be created automatically by replacing certain words with their synonyms without changing the overall semantics of instruction. Can the relatively inexpensive process of instruction augmentation improves model's performance in the *instruction-paradigm*, similar to the role data-augmentation has played conventionally in machine learning (Feng *et al.*, 2021)? *Instruction-paradigm* is pivotal where it is expensive or infeasible to gather training data. How effective is instruction-augmentation in low-data regimes?

Multi-variant instructions (original + augmented instructions) also can help evaluate the robustness of instruction-following models to respond to variant instructions. This is similar to the model robustness evaluation (Jia *et al.*, 2019) that is done by creating variant data instances. Multi-variant instruction based setup will also help gauge the true potential of instruction-following systems since in a real world setting, users can write task instruction in many different ways.

The expanded version of Natural Instructions (Mishra *et al.*, 2022f) [1] provides a rich collection of diverse category of tasks that covers a variety of reasoning skills, domains, and languages. This is a constantly evolving benchmark which is growing in size with respect to time. We take 426 tasks [2] and create variant instructions for each task. In Natural Instructions, number of instances was limited to 6500 to reduce massive data imbalance, we leverage remaining instances of source datasets in constructing instances of our variant instruction tasks. We experiment with 3 types of learning scenarios (i) task-specific (TS), (ii) multi-task (MT) and (iii) cross-task (CT) and observe that instruction augmented models outperform their single-instruction counterpart by 17%, 11% and 11%, respectively, when averaged over all experiments across the evaluation tasks. Interestingly, instruction augmentation is more effective on low-data regime [3] as we see performance gain of 26%, 16% and 11% in TS, MT and CT setting, respectively. We also quantify the contribution of each of the additional instructions and find that an additional instruction can be equivalent to ∼200 data samples on average across tasks. This has been discussed further in our work (Puri *et al.*, 2022).

[1] `https://github.com/allenai/natural-instructions`

[2] These were the accepted tasks in Natural Instructions in September 2021

[3] Average across 1%, 5% and 10% data

9.2 Multi-Variant Instruction Dataset

We construct Multi-Variant Instruction dataset on top of various tasks in Natural Instructions. In total, our dataset has 426 different NLP tasks; each of which contains multi-variant instructions.

9.2.1 Variant Instruction Task

An instruction task in Natural Instructions contains the definition of the task, positive examples, negative examples and instances. Figure 9.1 shows the schematic representation of variant instruction task where the blue boxes show the parts that differentiate variant instruction tasks with their original counterparts in Natural Instructions. While constructing a variant instruction task, we alter the definition and instances of the instruction task.

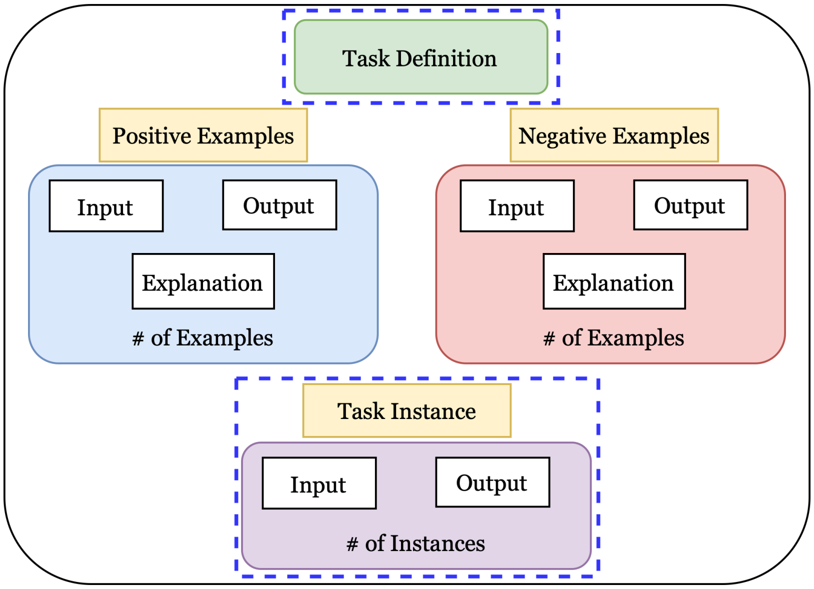


Figure 9.1: Schematic Representation of Instructional-prompts Mishra *et al.* (2022f) - Dotted Blue Box Represents Entities Which Are Changed in Constructing Variant Instruction Task.

Parameter	Value
Avg. # of variants per task	4.59
Avg. # of instances per task	9510.64
Avg. # of positive examples per task	3.15
Avg. # of negative examples per task	2.30

Table 9.1: Multi-Variant Instructions Dataset Statistics

9.2.2 Dataset Creation Process

Computer Science graduate students who participated in the data creation process are asked to create as many variant instruction tasks as possible. They are instructed to change definition (without changing the semantic meaning of the definition in the original task) and instances (by random sampling from the set of instances in the source dataset which are not part of instruction tasks in Natural Instructions). They are allowed to use automated tools such as Semantic Control (Ross *et al.*, 2021), Text Style Transfer (Reif *et al.*, 2021), NL-Augmenter (Dhole *et al.*, 2021). Sometimes, the participants create variant instruction tasks manually. Table 9.5 and Table 9.6 illustrate examples of alternate definitions across variant instructions created for our dataset.

9.2.3 Dataset Properties and Statistics

Table 9.1 shows the statistics of our meta-dataset. Note that, variant instruction tasks contain all instances from Natural Instructions, so the average number of instances per task is higher than 6500 (which is a constraint in Natural Instructions). We describe various attributes of our dataset in the following.

Semantic Textual Similarity

Semantic Textual Similarity (STS) should be high between original instruction and augmented instructions as they represent the same task. We compute the pair-wise STS score between definitions of original instruction and variant instructions. Figure 9.2 shows the mean and SD of STS score between original instruction and its variants across 426 tasks.

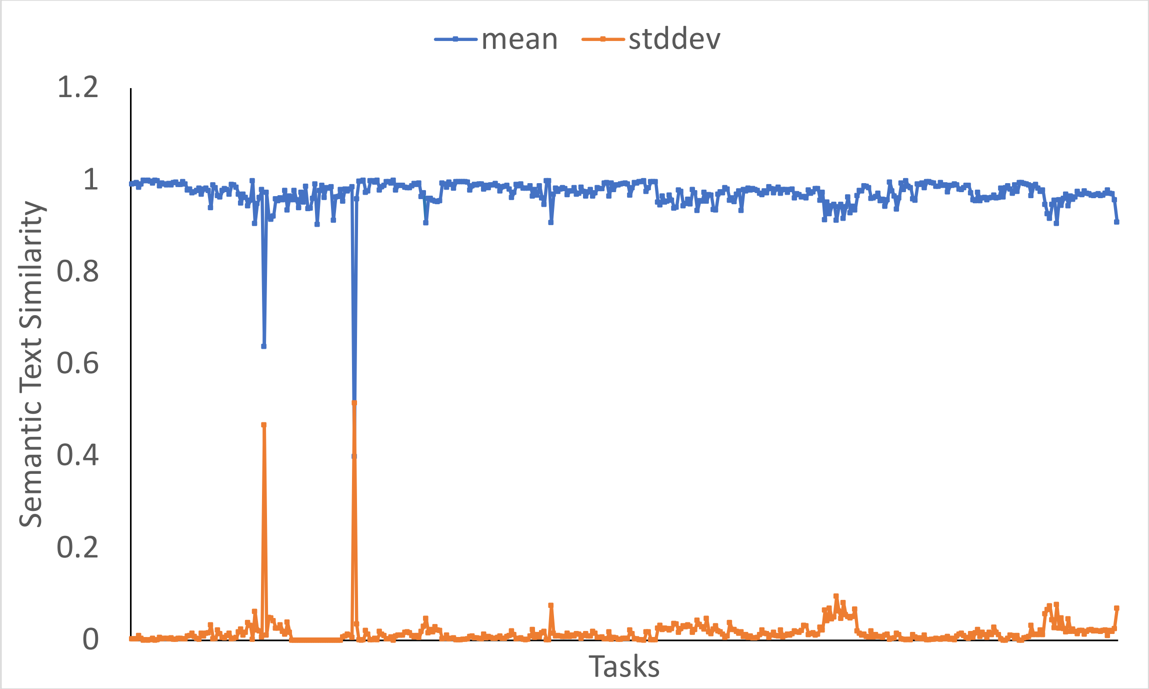


Figure 9.2: Semantic Text Similarity Between Original Instruction and its Variants.

Analysis of dataset properties From all dataset properties, we can observe that STS score is higher for almost all the tasks. This indicates that all augmented variants are semantically similar to original instruction. Moreover, we can see a significant variation in terms of word dissimilarity and length of definitions. From this, we can conclude that the variants created in our meta-dataset for each task have sufficient variations in terms of words and length yet sustaining semantic similarity with original instruction.

9.3 Experimental Setup

9.3.1 Models

BART-base (Lewis *et al.*, 2019) and T5-base (Raffel *et al.*, 2020) models are used with default hyper parameters from Huggingface (Wolf *et al.*, 2019) to perform experiments. We use Single Instruction (SI) learning as baseline where only original instruction is used to fine-tune the model. We propose Multi-Variant Instruction (MVI) learning where variants are used to fine-tune models. We use the same number of instances for both original and variant instruction learning to accurately gauge the importance of additional instructions.

9.3.2 Experiments

We perform three experiments: (1) Task-Specific, (2) Multi-Task, and (3) Cross-Task. All experiments are performed using 1%, 5%, 10%, 50% and 100% instances from the task for fine-tuning. Here, we divide instances into train, test and dev splits by randomly sampling in the ratio 70%, 20% and 10%, respectively. Evaluation is performed on the test set of original instructions. As SI is dependent on Natural Instructions which has exactly one instruction per task, we are not able to experiment with different instructions in SI setting while comparing it with MVI which has multiple variant instructions.

Task-Specific Here, we fine-tune baseline and our model on one task and evaluate on the same task. We have performed task-specific learning on 3 different tasks - winogrande_answer_generation, winogrande_question_modification_person and qasc_answer _generation. In addition, we also analyze two different tasks in other task categories like tweetqa_question_generation and odd-man-out_classification_no_category for generation and classification tasks respectively.

Task ID	Task Name	Task Category	# of Variants
task010	winogrande_answer_generation	Answer Generation	8
task011	winogrande_question_modification_object	Text Modification	8
task012	winogrande_question_modification_person	Text Modification	8
task017	qasc_question_generation	Question Generation	8
task018	qasc_answer_generation	Answer Generation	8
task020	essential_terms_answering_incomplete_questions	Classification	8
task028	multirc_correct_answer_single_sentence	Answer Generation	3
task058	babi_t1_single_supporting_fact_answer_generation	Answer Generation	5

Table 9.2: Number of Variant Instructions for 8 Different Tasks

Multi-Task To perform multi-task learning, we use 8 different tasks spanning across 4 different categories. Table 9.2 shows the different number of variant instructions for 8 tasks and their categories. In this setting, we fine-tune baseline and our model on all 8 tasks combined and evaluate on each task. However, we use only two positive and two negative examples to satisfy the maximum token limit of BART-base.

Cross-Task Here, we fine-tune the model on a set of tasks and evaluate on different set of tasks. Here, we use 274 different tasks for training by sampling 10% instances from each task and evaluate on a set of 8 tasks which are same as in the multi-task setup. In addition to sampling instances, we also sampled number of tasks by taking 1%, 5%, 10%, 50% and 100% tasks. We also investigate the extent of cross-task generalization in low-data regime; we do this by randomly sampling 1%, 5%, 10% instances for fine-tuning.

Metric We use the Rouge-L metric (Lin, 2004) for evaluation in all our experiments, following the evaluation in Natural Instructions.

9.4 Results and Analysis

9.4.1 Experimental Results

Task-Specific Figure 9.3 shows the comparison between SI and MVI across different number of instances sampled for fine-tuning. From this, we can observe that MVI outperforms SI by 17% on an average. The performance difference between MVI and SI increases to 26% in low data regime (average performance with 1%, 5% and 10% instances for fine-tuning). We observe similar results for additional 2 tasks we have analyzed. 9.7 contains more details.

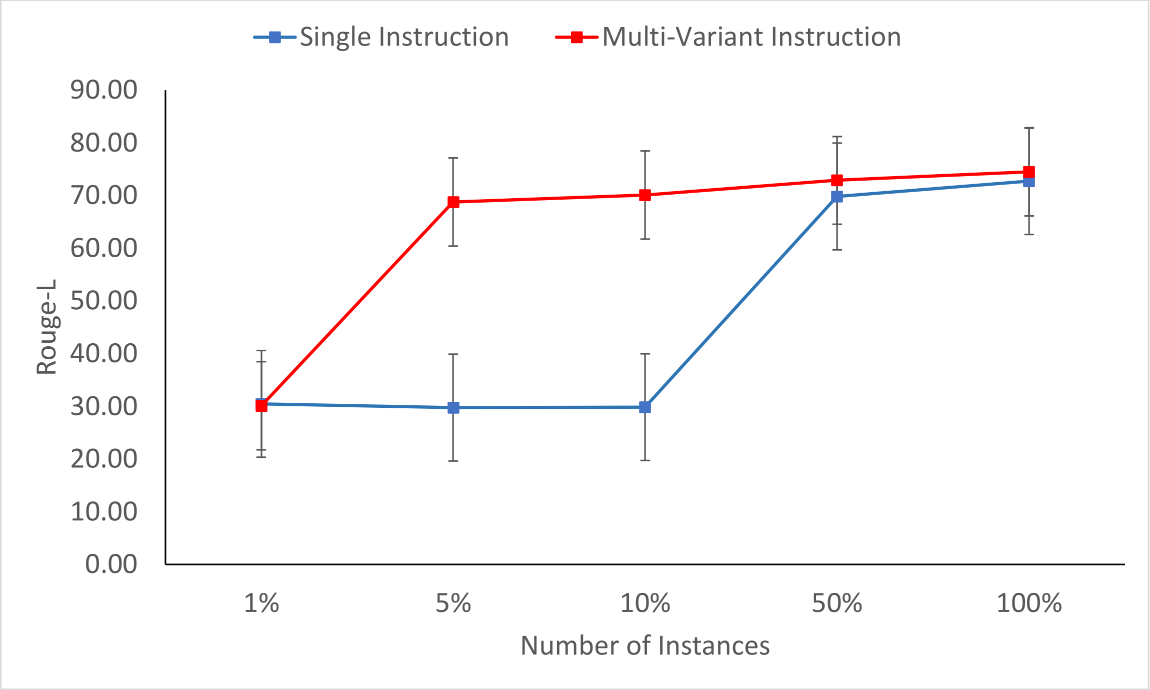


Figure 9.3: Comparison Across SI and MVI Learning in Task-specific Setting; Results Are Averaged over 3 Tasks

Multi-Task Figure 9.4 presents the comparison between SI and MVI for multi-task setting. We can observe that MVI outperforms SI by 11% on an average. Moreover, we can see higher improvement in low data regime ($\sim 16\%$). Our model achieves high performance boost (∼35%) at 1% instances setting. 9.8 contains more details.

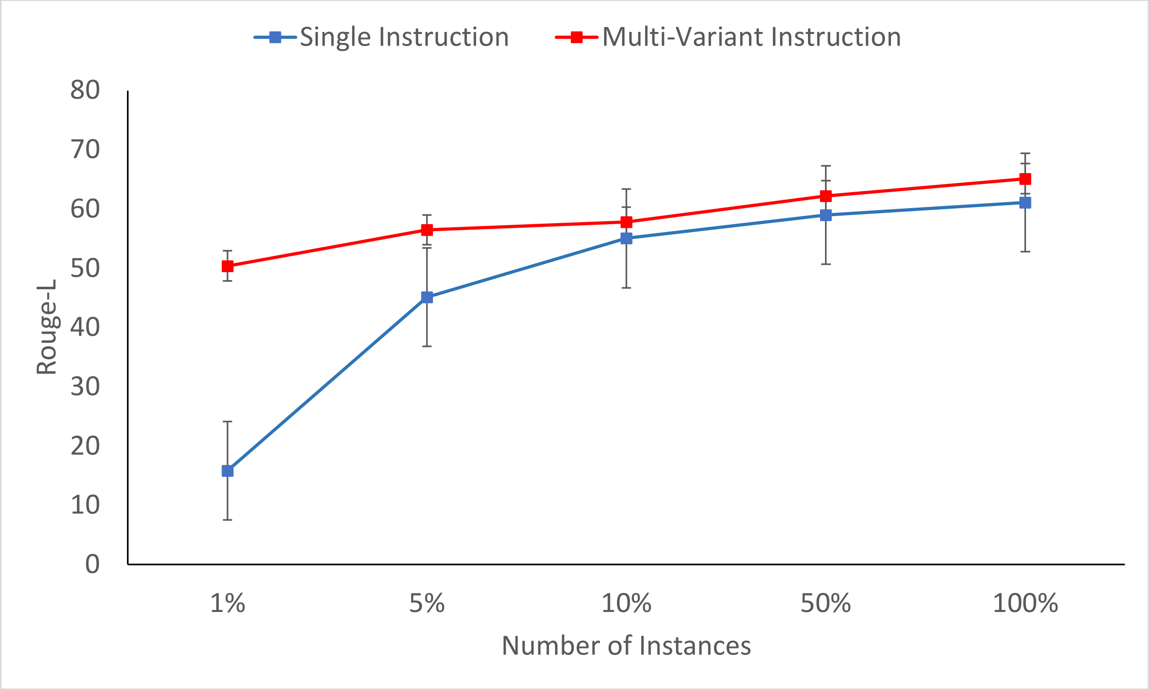


Figure 9.4: Comparison Across SI and MvI Learning in Multi-task Setting by Varying Number of Instances.

Cross-Task Figure 9.5 shows comparison between SI and MVI for 100% tasks in cross-task setting (see Figure 9.9 in 9.9 for other settings). We can observe that MVI outperforms SI by 9% on an average. 9.9 contains more details.

9.4.2 Analysis

How Many Data Samples is a Variant Instruction Worth? We calculate contribution of an additional instruction with respect to data samples in following way: we calculate model performance for BART-base in MVI with 5% instances. We interpolate model performance plot in SI to find out the percentage of instances needed to match performance in MVI (with 5% instances). We divide the average number of instance difference by average number of

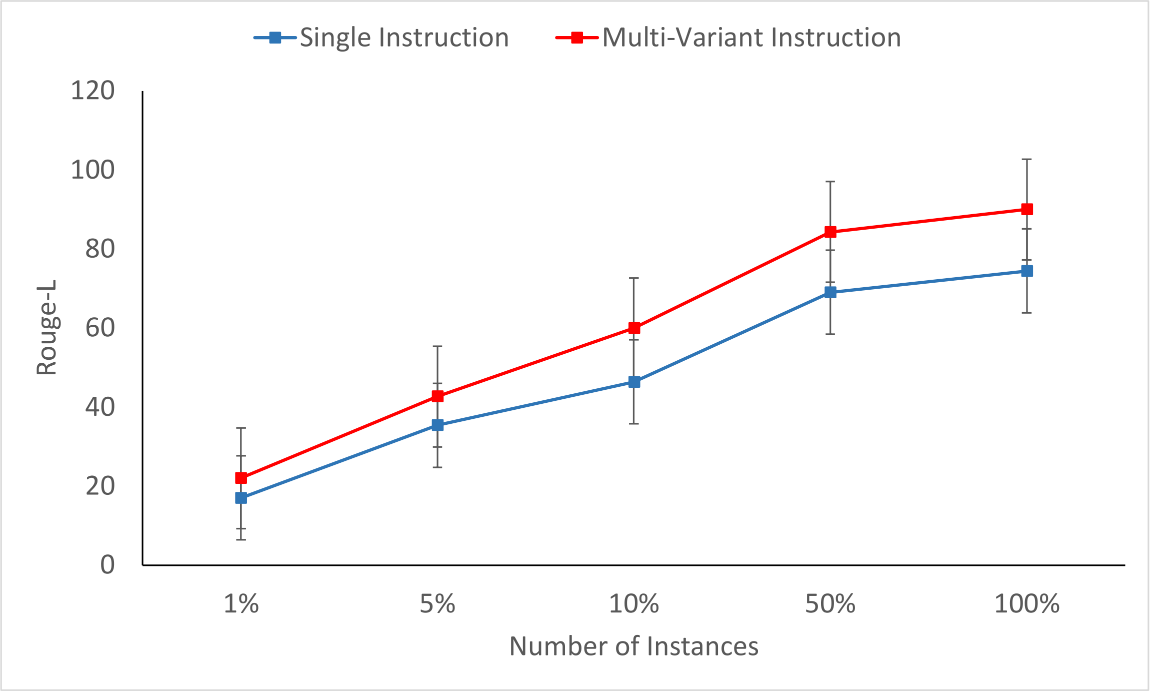


Figure 9.5: Comparison Between SI and MVI Learning in Cross-task Setting by Varying Number of Instances and Fixing Number of Tasks to 100%.

instruction variants to get the number that indicates worth of an additional instruction in terms of data samples. Using the above described procedure, we calculate the contribution for additional instruction in all three settings and summarize the results in Table 9.3. We use MVI performance with 5% instances as the base because a typical instruction-paradigm is designed in a "low-data regime" where non-expert users can teach a task to a model without requiring to create a dataset. However, we also calculated the instruction-equivalence using MVI with 10% instances as base and report the results in Table 9.3. On an average across TS, MT and CT, we conclude that an additional variant instruction alone is worth ∼200 instances.

Equal Data Analysis We believe that each instruction variant is equivalent to ∼200 data instances. To show this by experiment, we perform equal data analysis and observe that

Base	Task-Specific	Multi-Task	Cross-Task	Average
5%	456.2	94.1	152.3	234.2
10%	460.4	58.2	279.6	266.1

Table 9.3: Weight of Each Additional Instruction in Terms of Number of Data Samples Across Task-specific, Multi-task and Cross-task Settings.

model trained using our approach shows competitive performance compared to single-instruction learning by using only N/V instances where N is the total number of instances in the original task and V is the number of instruction variants for this task.

Is Model Robust to Instruction Perturbations? Here, we introduce 3 perturbations while testing SI and MVI: (1) we perturb the instruction by removing the task definition, (2) we perturb the instruction by changing the order of positive and negative examples by placing positive examples followed by negative different from training setup, and (3) we perturb the instruction by removing all positive and negative examples from the test set. We evaluate model robustness across these perturbations (performance change while the change in instruction) which are excluded from the training data. Here, Table 9.4 for task-specific setting on T5-base (see Table 9.11 ins 9.11 for multi-task results). We can clearly observe that our approach is robust to all the three instruction perturbations whereas model trained with single-instruction learning is not able to perform equally well on perturbed test sets compared to it original test counterpart. Similar trend is observed in multi-task setting as well (see 9.11).

# of Instances	SI		Perturbation 1		Perturbation 2		Perturbation 3	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
1%	0.90	25.21	1.60	18.03	1.02	23.16	5.12	9.71
5%	0.98	75.72	2.18	75.32	1.36	75.50	5.52	74.26
10%	50.88	78.20	20.76	78.07	50.49	78.37	40.31	77.22
50%	76.55	82.16	68.88	82.15	76.50	82.16	75.34	81.92
100%	79.38	83.16	73.51	82.97	79.34	83.12	78.71	82.40

Table 9.4: Comparison of Performance in Task-specific Setting Across SI and MVI Learning.

9.5 Multi-Variant Dataset Additional Details

9.5.1 Semantic Textual Similarity

We use en_core_web_md semantic similarity model of SpaCy to compute STS in our experiments. We also calculate STS score between definitions of variants of the same task. At the end, we calculate their mean and Standard Deviation (SD) for each task.

In the plot, the two exception points are task058 (Answer generation task based on babi dataset Weston *et al.* (2015)) and task097 (Structured text generation task based on SCAN dataset Lake and Baroni (2018)) where the original instructions are very long and the variant task contains a short definition which causes the strong variation in STS. We also discuss the Word-Level Dissimilarity and Length Diversity properties of our dataset below.

9.5.2 Word-Level Dissimilarity

To show the quality and diversity of variant instructions, we calculate the pair-wise edit distance between the definition of the original instruction and its variant instructions. We also calculate distance between definitions of variant instructions of the same task, further normalize by the highest distance to obtain a dissimilarity score. We compute the mean and SD of these scores for each task and show it in Figure 9.6.

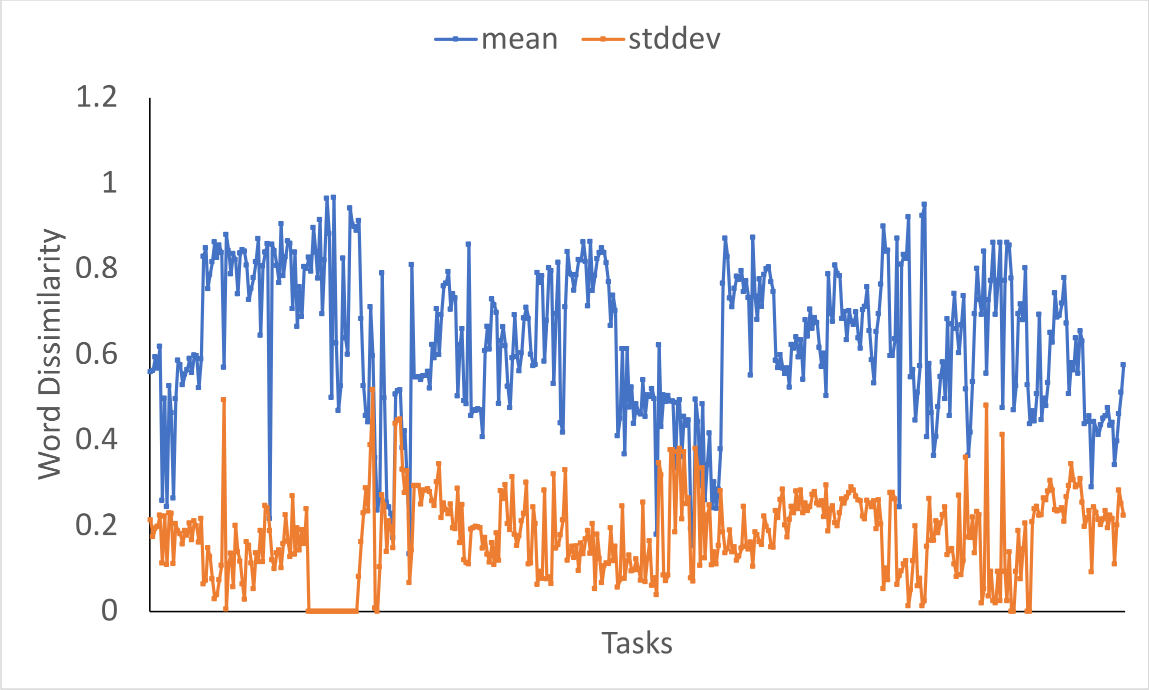


Figure 9.6: Word-level Dissimilarity Between Original Instruction and Its Variants.

9.5.3 Length Diversity

It is necessary to see how task definition lengths vary between original instructions and its variants. To understand this, we compute the percentage difference between the length of the maximum instruction definition and the minimum instruction definition for each task and show it in Figure 9.7.

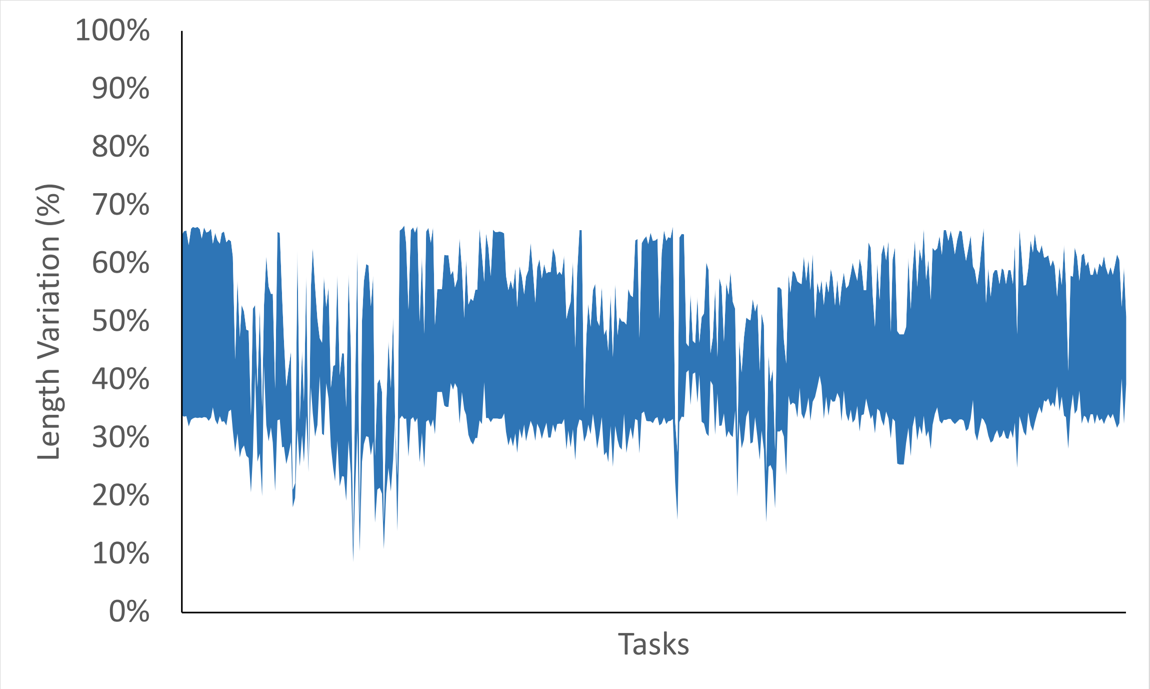


Figure 9.7: Definition Length Variation Between Original Instruction and Its Variants.

9.6 Example of Variants

Table 9.5 and Table 9.6 show the examples of different variants created from the task117_afs_argument_similarity_gun_control and task018_qasc_answer_generation respectively.

9.7 Task-Specific Results

Table 9.7 shows the results for task-specific experiments for task010_winogrande _answer_generation, task012_winogrande_question_modification_person and task018_qasc _answer _generation. We also performed experiments for other task categories like task210 _tweetqa _question_generation and task113_odd-man-out_classification_no_category for generation and classification tasks respectively and summarize our results in Table 9.8. From the average results, we can observe that multi-variant instruction learning helps model to

	Original instruction along with its augmented variant instructions
Original Instruction	***Definition:*** *We would like you to classify each of the following sets of argument pairs (discussing Gun Control) into either SIMILAR or NOT SIMILAR. A pair of arguments is considered SIMILAR if the arguments are about the same FACET (making the same argument), and is considered NOT SIMILAR if they do not have the same FACET. A FACET is a low level issue that often reoccurs in many arguments in support of the author's stance or in attacking the other author's position.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 1	***Definition:*** *Each of the following sets of argument pairs (on the topic of Gun Control) should be classified as SIMILAR or NOT SIMILAR. If the arguments are about the same FACET (making the same argument), they are deemed SIMILAR; otherwise, they are NOT SIMILAR. A FACET is a low-level problem that appears frequently in many arguments in favor of the author's position or in opposition to the position of the other author.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 2	***Definition:*** *Please classify the following sets of argument pairs (discussing the Gun Control) as SIMILAR or NOT SIMILAR. If the arguments are about the same FACET (making the same argument), they are regarded SIMILAR; if they are not, they are considered NOT SIMILAR. A FACET is a low-level problem that frequently recurs in numerous arguments in favor of the author's position or in opposition to the position of the other author.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 3	***Definition:*** *Two arguments are SIMILAR if they are making the same case related to author's position, else they are NOT SIMILAR. Your task is to classify any 2 arguments as SIMILAR or NOT SIMILAR.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 4	***Definition:*** *Each of the following sets of argument pairs (discussing the Gun Control) should be classified as SIMILAR or NOT SIMILAR. If the arguments are about the same FACET (making the same argument), they are regarded SIMILAR; otherwise, they are NOT SIMILAR. A FACET is a low-level issue that appears frequently in many arguments in support of the author's position or in opposition to the position of the other author.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>

Table 9.5: Example of an Instruction for a Classification Task with Its Variant Instructions; These Belong to the Task117_afs_argument_similarity_gun_control.

	Original instruction along with its augmented variant instructions
Original Instruction	***Definition:*** *Write a correct answer to the given question based on its associated fact. Make sure that your answer is contained in the associated fact. Things to avoid: Don't be creative and introduce any new word that is not mentioned in the associated fact! Remember that, the associated fact has been rearranged to form the question. So, the correct answer words must lie within the associated fact. Emphasis & Caution: The correct answer can be a word, phrase, or even a sentence.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 1	***Definition:*** *Handwriting a rectify reply to the given issue based on its related fact. Make sure that your replying is contained in the associated fact. Aspects to avoidance: Don't be creativity and introduces any nouveau word that is not alluded in the associated doing! Recall that, the linked doing has been restructured to forma the question. Thus, the corrects replying words needs lie within the associated doing. Focuses & Discretion: The exact replying can be a word, phrase, or even a penalties.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 2	***Definition:*** *Write a correcting responding to the gave question bases on its associated fact. Make persuaded that your answering is contained in the associated facto.Matters to shirk: Don't be inventive and introduce any nouveau word that is not referred in the associated fact! Recollect that, the associated fact has been redesigned to forma the issue. Therefore, the accurate responses words owes lying inside the associated doing. Concentrating & Circumspect: The correcting responses can be a word, phrase, or even a punishments.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 3	***Definition:*** *Write a corrects answer to the afforded issue founded on its associated fact. Deliver sure that your replied is contain in the linked fact. Things to shirk: Don't be creative and introduce any novel word that is not alluded in the associated fact! Remind that, the associated doing has been redesigned to forme the question. Accordingly, the correcting reply phrases needs lied indoors the linked fact. Concentrates & Caveat: The corrects response can be a word, phrase, or even a condemnation.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 4	***Definition:*** *Writing a accurate responded to the yielded matter founded on its associated fact. Deliver sure that your reply is contained in the associated doing. Aspects to avoidance: Don't be creative and introduce any newer word that is not talked in the associated facto! Recall that, the associated fact has been rearranged to form the issue. Thereby, the corrects responding phrase gotta lie within the related doing. Focus & Circumspect: The correct responding can be a word, expression, or even a sentences.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>
Variant Instruction 5	***Definition:*** *Writing a correct answers to the granted question bases on its associated doing. Make sure that your respond is contained in the associated doing. Matters to shirk: Don't be creative and introduces any novo word that is not referenced in the associated facto! Remind that, the associated fact has been reconfigured to forms the question. So, the corrects respond words ought lies within the related doing. Concentrate & Careful: The accurate reply can be a word, phrase, or yet a sentences.* ***Negative Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation> ***Positive Examples:*** **Input**: <input> **Output**: <output> **Explanation**: <explanation>

Table 9.6: Example of an Instruction for an Answer Generation Task with Its Variant Instructions - Task018_qasc_answer_generation

improve performance in task-specific learning.

9.8 Multi-Task Results

The results for multi-task learning experiments are shown in Table 9.9.

9.9 Cross-Task Results

The results for cross-task learning experiments are shown in Table 9.12. Figure 9.9 compares single-instruction learning and our approach in cross-task setting.

9.10 Equal Data Analysis

We keep the original number of instances in SI learning, however, reduce the number of instances used in MVI learning by sampling N/V number of instances randomly for each task where N is the total number of instances in the original task and V is the number of instruction variants for this task. We perform these experiments in both task-specific and multi-task settings using BART-base. Table 9.10 summarizes the results of these experiments, and we can observe that the model trained using our approach shows competitive performance compared to single-instruction learning by using only N/V instances.

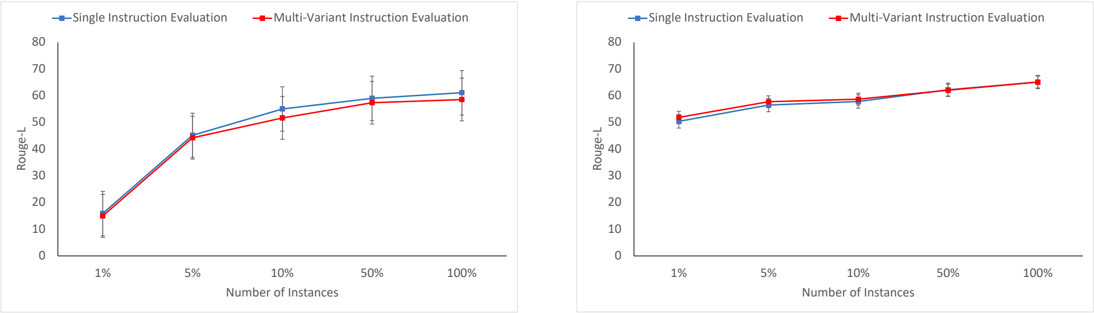
The results for cross-task learning experiments are shown in Table 9.12. Figure 9.9 compares single-instruction learning and our approach in cross-task setting.

9.11 Robustness Analysis

Is single-instruction learning robust? As Figure 9.8 illustrates, LM fine-tuned with single-instruction learning or original setting is not robust to instructions written in a different way; this includes transformation techniques like paraphrasing, adding spelling mistakes, grammatical mistakes etc. Our experiment results show that model trained using the proposed multi-variant instruction learning technique is able to perform reasonably well and is robust to variant instructions in both multi-task setting, as evidenced by lower performance difference between single instruction evaluation and multi-variant instruction evaluation setup.

9.12 Contribution of Individual Variants

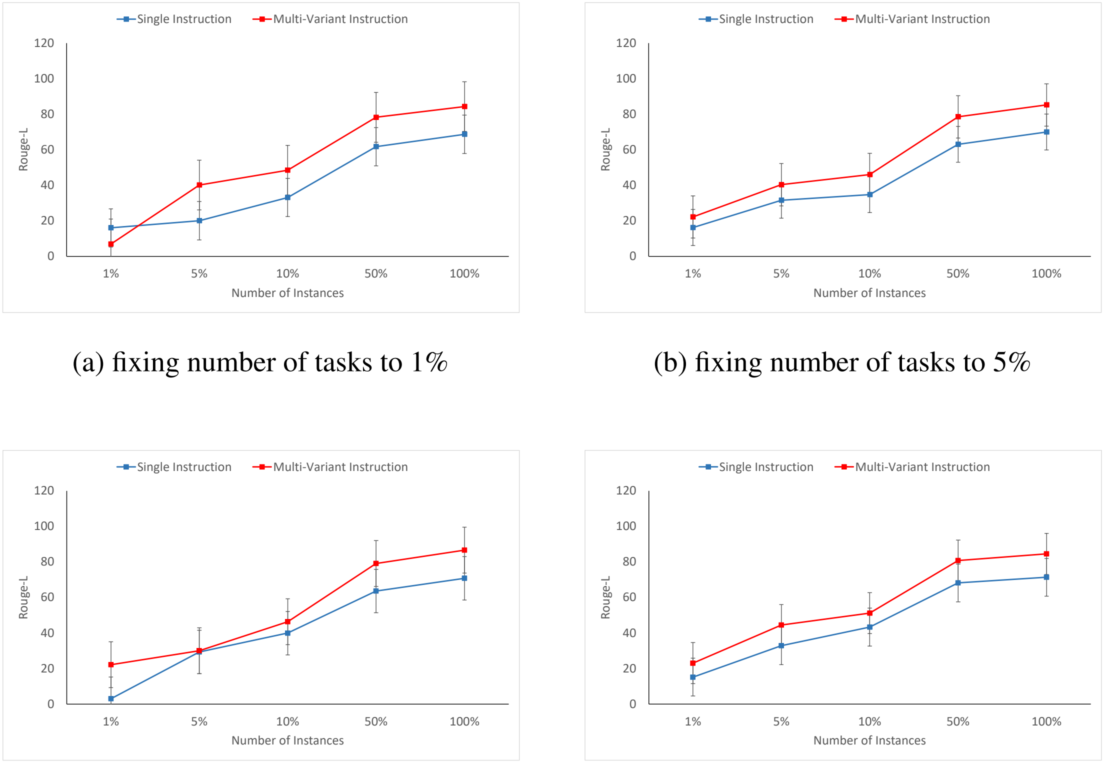
Do each of the variant instructions contribute equally towards performance gain? To analyse the contribution of each of the variant instruction, we study the performance gain by adding a single variant instruction at one time. We perform this analysis in TS setting (task_010) and MT setting and summarize the results in Table 9.13 and Table 9.14 respectively. We observe that all variants do not contribute equally, e.g. MVI_All above are often smaller than individual MVIs. Identifying optimal variants, however, will be scope for future work.



(a) Multi-task SI learning

(b) Multi-task MVI learning

Figure 9.8: Robustness Comparison of Si *Vs.* Mvi in Multi-task Setting - Lm Fine-tuned Using Mvi Learning Is More Robust to Variants as Compared to Si Learning.



(a) fixing number of tasks to 1%

(b) fixing number of tasks to 5%

(c) fixing number of tasks to 10%

(d) fixing number of tasks to 50%

Figure 9.9: Comparison of Performance Across Si and Mvi Learning in Cross-task Setting by Varying Number of Instances and Tasks. Evaluation Is Performed on the Test Set of Original Instructions.

9.13 Conclusion

We introduced instruction augmentation [4] to improve existing LMs in terms of improving performance and usability to non-expert users. To this extent, we created multi-variant instructions for 426 NLP tasks. Our experiment results show that instruction augmentation improves model performance in task-specific, multi-task and cross-task learning paradigms. We find that instruction augmentation is more effective in low-data regime. Our results

[4] https://github.com/Ravsehajsinghpuri/Multi-Variant-Instructions

further indicate that an additional instruction can be equivalent to $\sim$200 instances on an average. We hope our work will bring more attention to developing unconventional techniques (beyond dataset creation and model training) to empower non-expert users to leverage NLP resources and teach a task without having domain knowledge.

# of Instances	BART-base				T5-base			
	SI		MVI		SI		MVI	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
				task_010				
1%	0.00	0.00	0.00	0.02	0.04	13.71	0.16	11.26
5%	0.00	36.75	0.06	37.07	0.01	46.44	0.14	44.69
10%	0.23	39.17	0.15	38.26	12.03	53.03	9.05	52.60
50%	37.00	43.02	25.40	42.54	48.11	64.94	46.01	64.80
100%	41.97	45.65	33.84	45.50	55.67	67.49	53.74	66.92
				task_012				
1%	84.48	83.54	75.45	82.66	0.07	0.00	6.20	6.17
5%	84.73	90.68	74.52	90.68	0.05	90.90	6.17	90.87
10%	84.81	90.61	75.47	90.60	79.62	90.99	62.69	90.99
50%	90.29	90.49	85.65	90.48	90.92	90.77	90.81	90.81
100%	90.84	90.50	88.47	90.52	91.02	90.75	90.87	90.80
				task_018				
1%	7.05	6.92	4.36	5.27	2.57	61.92	3.02	58.53
5%	4.65	79.07	3.42	79.55	2.89	89.84	3.80	89.99
10%	4.72	80.59	3.68	80.95	61.00	90.57	56.28	90.56
50%	82.43	85.23	81.36	85.20	90.63	90.76	90.86	90.79
100%	85.58	87.37	84.90	87.52	91.44	91.25	91.41	91.11
				Average				
1%	30.51	30.15	26.60	29.32	0.90	25.21	3.12	25.32
5%	29.79	68.83	26.00	69.10	0.98	75.72	3.37	75.18
10%	29.92	70.12	26.43	69.94	50.88	78.20	42.67	78.05
50%	69.91	72.91	64.14	72.74	76.55	82.16	75.89	82.13
100%	72.80	74.51	69.07	74.51	79.38	83.16	78.68	82.94

Table 9.7: Comparison of Performance in Single-task Setting Across Single-instruction and Multi-variant Instruction Learning. Si: Single-instruction, Mvi: Multi-variant Instruction.

# of Instances	SI	MVI	SI	MVI
	task_210		task_113	
1%	13.37	12.25	3.00	3.85
5%	13.50	25.92	4.77	15.26
10%	14.67	27.14	4.00	30.77
50%	27.88	41.06	41.72	81.80
100%	37.24	44.10	66.73	98.10

Table 9.8: Comparison of Performance in Task-specific Setting Across Single-instruction and Multi-variant Instruction Learning. SI: Single-instruction, , MVI: Multi-Variant Instruction

# of Instances	BART-base				T5-base			
	SI		MVI		SI		MVI	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
1%	15.84	50.40	14.97	51.88	7.34	34.53	6.11	33.61
5%	45.13	56.49	44.24	57.71	32.01	62.61	19.88	62.87
10%	55.03	57.80	51.67	58.70	46.93	63.61	39.76	63.98
50%	59.01	62.21	57.37	62.06	63.38	66.16	57.11	66.76
100%	61.08	65.13	58.58	65.09	64.99	67.15	59.35	67.38

Table 9.9: Comparison of Performance in Multi-task Setting Across Single-instruction and Multi-variant Instruction Learning. SI: Single-instruction, MVI: Multi-variant Instruction

# of Instances	Single Task		Multi Task	
	Original	Ours	Original	Ours
1%	10.81	7.32	6.35	0.82
5%	20.86	19.42	4.21	6.31
10%	57.22	51.36	59.95	49.42
50%	76.53	72.75	84.54	79.74
100%	78.36	60.15	86.55	82.02
Average	48.76	42.20	48.32	43.66

Table 9.10: Comparison of Performance in Task-specific (Average Across 3 Tasks) and Multi-task Settings.

# of Instances	SI		Perturbation 1		Perturbation 2		Perturbation 3	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
1%	7.34	34.53	7.73	39.76	7.23	33.27	3.37	35.32
5%	32.01	62.61	25.90	60.22	29.51	63.52	23.50	69.30
10%	46.93	63.61	46.36	61.70	44.74	63.86	43.28	72.46
50%	63.38	66.16	61.63	64.50	63.73	66.40	71.79	67.99
100%	64.99	67.15	63.12	67.38	65.05	66.02	72.70	68.24

Table 9.11: Comparison of Performance in Multi-task Setting Across Single-instruction and Multi-variant Instruction Learning.

# of Instances	BART-base				T5-base			
	SI		MVI		SI		MVI	
	Original	Ours	Original	Ours	Original	Ours	Original	Ours
1% tasks								
1%	16.00	6.94	10.93	10.16	0.96	7.36	0.87	7.31
5%	20.04	40.14	19.51	31.09	21.87	29.07	19.89	29.60
10%	33.09	48.43	31.83	47.66	36.17	44.50	33.13	45.28
50%	61.70	78.22	58.53	78.43	64.74	73.94	61.34	73.45
100%	68.66	84.22	64.39	84.87	72.35	83.37	68.9	84.2
5% tasks								
1%	16.23	22.17	3.32	18.78	1.30	7.55	1.29	7.29
5%	31.58	40.3	29.81	33.12	22.85	29.04	20.44	29.02
10%	34.73	46.02	34.38	49.15	36.01	44.83	33.75	44.93
50%	63.06	78.48	60.5	79.76	65.96	76.25	61.01	76.13
100%	69.93	85.2	67.41	86.68	74.54	83.61	70.2	83.69
10% tasks								
1%	2.98	22.16	2.46	19.98	3.12	7.89	2.56	7.66
5%	29.27	30.06	28.03	30.9	24.49	29.29	23.41	29.25
10%	39.95	46.38	36.3	50.4	36.76	45.22	36.23	44.81
50%	63.58	79.13	59.98	79.81	66.07	73.49	62.56	73.54
100%	70.82	86.66	69.11	87.86	71.97	81.16	70.34	81.08
50% tasks								
1%	15.18	23.06	17.08	26.2	5.58	22.26	5.44	22.21
5%	32.88	44.5	33.88	44.64	33.56	40.37	30.57	38.25
10%	43.33	51.2	42.5	54.62	45.42	44.02	39.01	44.36
50%	68.18	80.8	66.42	81.29	66.62	80.97	63.89	80.93
100%	71.35	84.52	68.85	84.65	72.72	82.82	69.94	82.02
100% tasks								
1%	17.04	22	19.2	24.95	20.69	22.55	9.02	20.66
5%	35.4	42.68	36.42	45.06	35.18	38.30	30.92	39.51
10%	46.4	60	45.33	59.3	44.70	53.80	44.47	54.15
50%	69.06	84.32	67.29	84.47	71.89	79.20	68.64	79.56
100%	74.45	90.01	72.26	90.35	74.03	81.53	72.34	82.15

Table 9.12: Comparison of Performance in Cross-task Setting Across Single-instruction and Multi-variant Instruction Learning. SI: Single-instruction, MVI: Multi-variant Instruction.

# of Instances	SI	MVI_1	MVI_2	MVI_3	MVI_4	MVI_5	MVI_6	MVI_7	MVI_All
1%	0.00	17.46	0.92	0.20	0.44	6.92	5.7	6.79	0.00
5%	0.00	34.34	35.84	36.90	37.36	39.96	37.72	37.97	36.75
10%	0.23	37.31	41.03	42.30	42.95	43.59	42.4	41.23	36.75
50%	37.00	44.25	59.30	57.18	59.45	61.82	62.93	44.14	43.02
100%	41.97	44.34	71.02	75.20	80.27	81.74	86.05	53.63	45.65

Table 9.13: Contribution of Each Variant Instruction Towards Performance in Task-specific Setting for Task010. SI: Single-instruction, MVI_k: Multi-variant Instruction Where K Equals Number of Variant Instructions Used.

# of Instances	SI	MVI_1	MVI_2	MVI_3	MVI_All
1%	15.84	37.03	40.93	64.08	50.4
5%	45.13	55.38	55.80	56.46	56.49
10%	55.03	58.17	58.32	57.70	57.8
50%	59.01	61.62	61.45	62.20	62.21
100%	61.08	62.90	64.08	64.10	65.13

Table 9.14: Contribution of Each Variant Instruction Towards Performance in Multi-task Setting. SI: Single-instruction, MVI_k: Multi-variant Instruction Where K Equals Number of Variant Instructions Used.

Chapter 10

HELP ME THINK: A SIMPLE PROMPTING STRATEGY FOR NON-EXPERTS TO CREATE CUSTOMIZED CONTENT WITH MODELS

10.1 Introduction

Large language models (LLM) like GPT3 (Brown *et al.*, 2020) and PaLM (Chowdhery *et al.*, 2022) have excelled in many NLP tasks, however creating customized content in the form of long text generation using these models is a challenge (Figure 10.1), as (1) models have not been reliable in following a list of instructions, (2) correcting model's output post generation by providing negative instructions (e.g. Don't do this..) in the form of dialogue has also not worked. More importantly, significant effort is required for non-expert users to write instructions containing the important ingredients of a task; for example, in order to create instructions for models to write the bio of a user, the user needs to think and provide various personal information. Travel & event plan generation are similar to such tasks where a non-expert user has to think and find out various information necessary to make a plan, such as 'number of attendees', 'venue selection', 'budget', 'special arrangements', etc.

We introduce Help me Think: a simple prompting strategy for non-experts to write customized content with models. On a broader level, the goal of Help me Thinkis similar to Make-A-Scene (Gafni *et al.*, 2022); the application area, however, is vastly different as we focus on diverse applications that do not involve images and are purely based on text data. Help me Think involves prompting models to ask relevant questions that reduce the thinking burden of users in identifying key information specific to that task. This application is in contrast to the dominant application of LLMs where various prompting techniques have been developed to enable LLMs in answering a question Liu *et al.* (2021b). We hypothesize that knowing the right question to ask is often more important than answering it, especially in the context of personalized content generation.

We experiment with six customized content generation tasks [1] : (1) bio generation, (2) travel plan generation, (3) dialogue generation, (4) poem generation, (5) event summary generation, and (6) story generation. We prompt GPT3 and collect 68 questions corresponding to these tasks. We find that 100% of the generated questions are valid and relevant. We use crowd-sourcing to collect answers; we leverage these question-answer pairs to prompt GPT3 again and get task-specific outputs e.g. bio, event plan, story, etc. With the Help me Think prompting, we construct a dataset of $\sim$ 2k question-answer pairs with 180 task specific outputs spanning over the 6 tasks. We develop a questionnaire-based human evaluation scheme for crowdworkers to evaluate the quality of model generations: (1) questions and (2) task-specific outputs.

Our results show that 100% of task-specific outputs generated by GPT3 are valid (e.g. valid bio) and $\sim$ 94% of them do not contain any extra and irrelevant information. Moreover, we observe that, in $\sim$ 83% of cases, GPT3 corrects typos/grammatical issues/invalid answers present in the crowdworkers' answers. GPT3 also adds appropriate context and generates coherent sentences for $\sim$ 99% of cases. In 70% of cases, GPT3 performs accurate knowledge transfer [2] by transferring all information from input question-answer pairs to task-specific outputs. We hope the Help me Think prompting and our focus on tasks hard for average humans will encourage the development of unconventional ways to harness the power of LLMs and bring more attention to the under-explored tasks. This has been discussed further in our work (Mishra and Nouri, 2022).

[1] We also explore an additional set of 57 tasks

[2] with a tolerance level of 1 question-answer pair for tasks with 4 questions (poem and dialogue tasks) and 2 question-answer pairs for tasks with more than 4 questions

10.2 Help me Think

In this section, we introduce the Help me Think prompting. We first present the algorithm behind Help me Think. Next, we describe the prompts used in Help me Think and explain the role of non-expert users in this prompting process.

Prompt: Figure 10.2 illustrates the prompt given to the model that generates a question specific to the task of interest. Figure 10.3 shows that model generates a question and an answer, in response to the prompt. Figure 10.4 shows how we generate multiple questions in this process. Figure 10.5 shows where users fill in their answers. Figure 10.6 shows the prompt that is attached with the question-answer pairs which finally gives rise to the model-generated task-specific output.

10.3 Experiments

We conduct experiments on a diverse set of novel tasks. In this section, we first introduce the tasks, followed by data collection and the evaluation setup.

10.3.1 Tasks:

We experiment with six customized content generation tasks: (1) bio generation, (2) travel plan generation, (3) dialogue generation, (4) poem generation, (5) event summary generation and (6) story generation.

10.3.2 Data Collection:

We prompt GPT3 (Figure 10.4) to generate task-specific questions automatically. We set up a crowdsourcing task to get answers to the questions. We use a private group of crowdworkers that leverages a set of internal tools specific to the anonymous organization; they are instructed to write diverse outputs while collecting answers. GPT3 is prompted with

category	# of instances
task	6
questions	68
question-answer pair	2040
task outputs	180

Table 10.1: Key Statistics of Our Collected Data.

question-answer pairs and a task-specific prompt (Figure 10.6) to generate task-specific outputs.

10.3.3 Statistics:

We collect a total of $\sim 2k$ QA pairs. Table 10.1 shows some key statistics in our collected data.

10.3.4 Evaluation:

We use human evaluation since content generation tasks are open-ended and are hard to be captured by automated evaluation metrics. Each question associated with each task is evaluated by 3 annotators. The annotators evaluate various aspects of generated text by answering the associated questions.

Evaluation of Model Generated Questions:

The annotators evaluate the following two aspects of the questions generated by models. Each question is evaluated by three annotators.

Validity *Is it a question? (and not a statement or any other text.)*

category	bio	travel plan	dialogue	poem	event summary	story	avg.
Validity	100	100	100	100	100	100	100
Relevance	100	100	100	100	100	100	100

Table 10.2: Evaluation (Majority Voting of 3 Annotators) of Model Generated Questions for Each Task.

category	bio	travel plan	dialogue	poem	event summary	story	avg.
Validity	100	100	100	100	100	100	100
Knowledge Absorption	86.66	3.33	70	90	86.66	83.33	70
Relevance	100	93.33	76.67	96.67	96.67	100	93.89
Robustness	96.67	50	71.42	100	77.78	100	82.65
Coherence	100	100	96.15	100	100	100	99.36

Table 10.3: Model Performance on Different Evaluation Aspects for Each Task (Majority Voting of 3 Annotators). For the ‘knowledge Absorption’ Category, a Tolerance Level of 1 Question-answer Pair for Tasks with 4 Questions (Poem and Dialogue Tasks) and 2 Question-answer Pairs for Tasks with More than 4 Questions Are Taken.

Relevance *Is it relevant to the underlying task?* E.g. bio, travel plan, story, etc.

Evaluation of Model Generated Task-specific Outputs:

Furthermore, each task-specific output (e.g. bio, story, etc.) generated by GPT3 and its corresponding input (generated by GPT3 and answered by the user) is evaluated by three annotators. Each annotator is asked to answer a question that covers a specific part of the evaluation as follows:

Validity: *Is the output a valid task-specific output?* E.g. is it a valid bio? (for the bio-generation task).

Knowledge Absorption: *Does the output incorporate all the facts and information from the input?*

Relevancy: *Does the output have unrelated information that is not present in the input?*

Robustness: *Has the output fixed any typos or grammatical errors or invalid answers present in the input, instead of copying the same?*

Coherence: *Can you find any example of output having additional related and contextual information written as a coherent sentence in addition to what was already present in the input?*

10.4 Results

We report the task-wise performance of GPT3 and analyze its variation across different aspects of evaluation.

Insights *We observe that 100% of the questions generated by GPT3 are valid and relevant to the task (Table 10.2).* Table 10.3 shows the performance of GPT3 in producing task-specific outputs. We observe that *(1) 100% of the generations are valid task-specific outputs, (2) 93.89% of the generations do not contain irrelevant content, and (3) 99.36% times, GPT3 improves the coherence of text over the information presented to it in the form of input-output examples. (4) 82.65% times, GPT3 fixes typo/grammatical issues present in the user-written answers.* However, we see that the knowledge absorption is low (70%), which we analyze further.

We also ask crowdworkers to write explanations; we analyze those to better understand the knowledge absorption in the task specific output generated by GPT3. We understand that for certain tasks like travel plan generations, some question-answer pairs are not important; they are not always necessary to be part of the plan.

Extension to Other Tasks: We apply Help me Think on 57 additional tasks and find that Help me Think is effective in generating valid and relevant questions. This shows the generalization of Help me Think beyond the six tasks we have analyzed (Section 10.3.1).

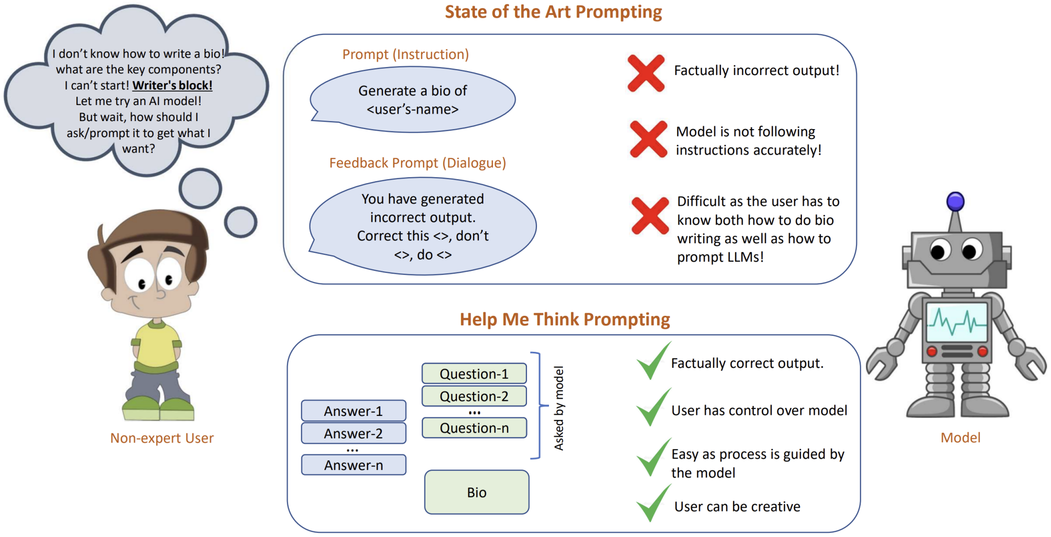

I don't know how to write a bio! what are the key components? I can't start! Writer's block! Let me try an AI model! But wait, how should I ask/prompt it to get what I want?
Non-expert User
State of the Art Prompting
Prompt (Instruction)
Generate a bio of <user's-name>
Feedback Prompt (Dialogue)
You have generated incorrect output. Correct this <>, don't <>, do <>
Factually incorrect output!
Model is not following instructions accurately!
Difficult as the user has to know both how to do bio writing as well as how to prompt LLMs!
Help Me Think Prompting
Answer-1
Answer-2
...
Answer-n
Question-1
Question-2
...
Question-n
Asked by model
Bio
Factually correct output.
User has control over model
Easy as process is guided by the model
User can be creative
Model

Figure 10.1: Illustration of Help me Think Prompting for a Non-expert User on the Bio Generation Task. A Non-expert User Is Asked to Write a Biography, but This Is a Hard Task for the User since It Demands Thinking about Key and Necessary Components for a Biography Which He Might Not Know about or He Might Just Simply Be Dealing with Writer's Block When Faced with Creative Writing Tasks. The User Decides to Get Help from an Ai Model. The User Wants to Try to Prompt an Ai Model (Using State-of-the-art Instruction Prompting Paradigms), but the Model Produces Factually Incorrect Output for Him. Next, the User Tries to Interact with the Model and Provide Feedback to Correct the Model Prediction (Dialogue Paradigm), but This Approach Is Also a Failure Because It Is a Challenge for Models to Accurately Follow Feedback Instructions. For the Majority of Non-expert Users, Figuring out an Effective Prompting Strategy Is a Major Challenge. Finally, Help me Think Helps the User Generate His Factually Correct Biography via the Model by Guiding the User in the Process by Asking Questions, This Alleviates the Cognitive Demand on the User Significantly. By Removing the Hurdles out of the Way of the User in Writing His Biography, Help me Think Also Allows the User to Take a Step Further and Focus on Creativity and Quality.

I am an expert *$task-executer$*. I will ask some questions to collect information and then I will use the information to *$do the task.$*
Question:

Figure 10.2: Prompt given to the Model to Generate Question. *$task-executer$* and *$do the Task.$* Are ‘bio Generator’ and ‘generate a Bio for You’ for the Bio Generation Task. They Vary Across Tasks.

I am an expert *$task-executer$*. I will ask some questions to collect information and then I will use the information to *$do the task.$*
Question: <model generates question>
Answer: <->

Figure 10.3: Model Generation in Response to the Prompt. Model Also Generates an Answer along with the Question, but <-> Indicates That We Are Not Storing This Information.

I am an expert *$task-executer$*. I will ask some questions to collect information and then I will use the information to *$do the task.$*
Question: <model generated question>
Answer: <->
Question: <model generated question>
Answer: <->
...

Figure 10.4: Prompt and the Generated Question-answer Pair Are Fed to the Model to Generate New Questions for the Task.

I am an expert *$task-executer$*. I will ask some questions to collect information and then I will use the information to *$do the task.$*

Question: <model generated question>

Answer: <user writes answer>

Question: <model generated question>

Answer: <user writes answer>

...

Figure 10.5: User Writes Answers to the Questions Generated by Model

I am an expert *$task-executer$*. I will ask some questions to collect information and then I will use the information to *$do the task.$*

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a *$task-specific-output$* using the questions and answers above. *$task-specific-instruction$*

<model generates task-specific-output>

Figure 10.6: A Task-specific Prompt Is Added after the Model Generated Question-answer Pairs. <Model Generates Task-specific-output> In Response to the Prompt. *$task-specific-output$* for the Bio Generation Task Is ‘a Long Bio about John’. *$task-specific-instruction$* Is Optional, e.g. ‘introduce Names to Represent Characters.’ for the Story Generation Task.

10.5 Related Work

Prompting and Learning from Instructions: The success of language models (Brown *et al.*, 2020; Chowdhery *et al.*, 2022) has empowered the development of various prompting techniques (Liu *et al.*, 2021b). Instructions, proposed as an extension to prompts, describe tasks in natural language (Efrat and Levy, 2020; Weller *et al.*, 2020) and guide models to generalize to unseen tasks (Mishra *et al.*, 2022f; Wei *et al.*, 2022a; Ouyang *et al.*, 2022; Sanh *et al.*, 2022; Zhong *et al.*, 2021) without requiring task-specific training. Prompts and Instructions are shown to be helpful in low-resource settings (Le Scao and Rush, 2021; Puri *et al.*, 2022). Several variants of prompting such as chain of thought (Wei *et al.*, 2022b) or scratchpad (Nye *et al.*, 2021), majority voting (Wang *et al.*, 2022b), reframing (Mishra *et al.*, 2022e), least-to-most prompting (Zhou *et al.*, 2022), question decomposition (Khot *et al.*, 2021; Patel *et al.*, 2022) have been shown to be effective across various tasks. Efficacy of the Prompting/Learning from Instruction techniques has been shown across diverse applications (Wang *et al.*, 2022d) such as dialog (Gupta *et al.*, 2022b), NER (Wang *et al.*, 2022a), program synthesis (Kuznia *et al.*, 2022), style transfer (Reif *et al.*, 2021), tabular question answering (Luo *et al.*, 2022), relation extraction (Chen *et al.*, 2021b), biomedical applications (Parmar *et al.*, 2022b). In contrast to prior works, we (1) focus on a diverse set of creative tasks as our application area, (2) build a non-expert user-centric technique (3) leverage language models for assisting users to think by asking questions (instead of just answering questions posed by humans), and in this process engage users that subsequently helps them learn the thinking process.

Creative Tasks using GPT3: GPT3 has been recently used for creative tasks such as writing a paper [3] , poem [4] , article [5] , and book [6] . However, controlling model generation in these creative tasks is a challenge. On a broader level, controlling content in text generated by pretrained language models has been a challenge in NLP (Perez, 2022). The Help me Think framework provides a model-guided approach by generating and asking questions to assist users in thinking through various steps and at the same time controlling the content of the text generated by the model. We further prove the efficacy of the Help me Think approach by applying it to a set of 63 diverse tasks. These tasks are novel and creative and can potentially form a benchmark dataset for future research.

Interactive Learning Interactive question answering has been utilized in several recent works (Zhong *et al.*, 2022; Zhao *et al.*, 2022; Yao *et al.*, 2020) around semantic parsing. In these cases, the target output is code and the task is being decomposed into smaller sub-tasks. Similar approach has also been used in diverse domains, e.g., robotics (Ahn *et al.*, 2022), symbolic model learning (Verma *et al.*, 2021; Verma and Srivastava, 2021), differential AI assessment (Nayyar *et al.*, 2022), etc. Help me Think is different in two ways (1) the schema in case of semantic parsing tasks and the set of actions in robotics tasks are fixed, whereas in Help me Think, schema is dynamic and is derived from a high level description of the task (2) in the approaches for semantic parsing and robotics tasks, there is a concern if the generated output can be executed on downstream tasks (e.g. in guiding a robot), however that concern is relaxed in Help me Think as the output itself is in natural language.

[3] https://www.scientificamerican.com/article/we-asked-gpt-3-to-write-an-academic-paper-about-itself-then-we-tried-to-get-it-published/, https://hal.archives-ouvertes.fr/hal-03701250/document

[4] https://www.newyorker.com/culture/culture-desk/the-new-poem-making-machinery

[5] https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3

[6] https://medium.com/swlh/i-wrote-a-book-with-gpt-3-ai-in-24-hours-and-got-it-published-93cf3c96f120, https://augmentedrobot.medium.com/281-years-in-the-making-gpt-3-and-de-la-mettrie-de03eea374e4

10.6 Help me Think Algorithm and Description

Algorithm: Algorithm 2 illustrates the detailed algorithm behind Help me Think. It has 3 stages: Stage 1 (Generate Questions), Stage 2 (Collect Answers) and Stage 3 (Generate Task-specific Output). We describe each stage in detail below.

Description In figure 2, a non-expert user is asked to write a biography, but this is a hard task for the user since it demands thinking about key and necessary components for a biography which he might not know about or he might just simply be dealing with writer's block when faced with creative writing tasks. The user decides to get help from an AI model. The user wants to try to prompt an AI model (using state-of-the-art instruction prompting paradigms), but the model produces factually incorrect output for him. Next, the user tries to interact with the model and provide feedback to correct the model prediction (dialogue paradigm), but this approach is also a failure because it is a challenge for models to accurately follow feedback instructions. For the majority of non-expert users, figuring out an effective prompting strategy is a major challenge. Finally, Help me Think helps the user generate his factually correct biography via the model by guiding the user in the process by asking questions, this alleviates the cognitive demand on the user significantly. By removing the hurdles out of the way of the user in writing his biography, Help me Think also allows the user to take a step further and focus on creativity and quality.

Role of Non-expert User: Figure 10.1 illustrates the role of a non-context user in performing bio generation tasks. It requires a lot of thinking for the non-expert user to write a bio by himself as it demands the identification of key ingredients corresponding to the bio generation task. Help me Think in contrast to other prompting techniques helps the user perform the task via the model with minimal effort, as they just need to answer the questions generated by the model (Figure 10.5).

Algorithm 2: Help me Think algorithm

```
1: Generate question-generation prompt by replacing task-specific variables in the prompt
   (figure 10.2)                              ▷ Start of Stage 1 (Generate Questions)
2: Setup stop condition (e.g. ?, ‘Answer:’, newline ) and ask GPT3 to complete with
   ‘Question: ’ prompt
3: Repeat Step 2 until the model starts generating repetitive, redundant, or irrelevant
   content.
4: If default settings are not producing the expected output, then try these (1) control
   temperature, (2) control max output size, (3) add additional task-specific instruction or
   (4) add an example question.                ▷ End of Stage 1 (Generate Questions)
5: Ask user to answer each of the questions and pose customization requirements
   (Figure 10.5)                                         ▷ Stage 2 (Collect Answers)
6: Generate task-specific output generation prompt by replacing task-specific variables in
   the prompt (figure 10.6)          ▷ Start of Stage 3 (Generate Task-specific Output)
7: If question-answer pairs are dependent on each other, collect all of them to feed them
   all at once to model, else feed them batch by batch
8: Setup appropriate stop condition and ask GPT3 to complete output after the prompt
9: Concatenate task-specific output if step 6 was done in batches
10: If default settings are not producing valid output, then try these (1) control temperature,
   (2) control max output size or (3) add additional task-specific instruction.   ▷ End of
   Stage 3 (Generate Task-specific Output)
```

10.7 Detailed Prompts and Hyperparameters

In this section, we describe the initial prompts (Figure 10.2) we use across tasks. Figures 10.7, 10.8, 10.9, 10.10, 10.11, 10.12 show the prompts for bio, travel plan, dialogue, poem, event summary and story generation tasks respectively.

Figure 10.4 illustrates all questions generated by GPT3 for various tasks in response to our prompting (Figure 10.3,10.4).

Figure 10.13, 10.14, 10.15, 10.16, 10.17, 10.18 show the prompt used to generate task-specific output from GPT3 (as in figure 10.6).

We use the following hyper-parameters while querying GPT3 for various tasks: `engine=text-davinci-002`, `temperature=0.7`, `max_tokens=512`,`top_p=1`, `frequency_penalty=0`, `presence_penalty=0`.

> I am an expert in generating Bio of people. I ask questions to gather information. Then I use these information to generate bio.
> Question:

Figure 10.7: Prompt given to Model to Generate Question about the Bio Generation Task.

> I am a famous travel planner. I will ask clarifying question to collect information and then I will write an awesome travel plan and schedule for you.
> Question:

Figure 10.8: Prompt given to Model to Generate Question about the Travel Plan Generation Task.

I am a famous event planner. I will ask clarifying question to collect information and then I will write an awesome event plan for you.
Question:

Figure 10.11: Prompt given to Model to Generate Question about the Event Summary Generation Task.

I am an expert script writer. I will ask some simple questions to collect information and then I will write a story of your choice.
Question:

Figure 10.12: Prompt given to Model to Generate Question about the Story Generation Task.

I am a famous dialogue writer. I will ask simple questions to collect information and then I will write a dialogue series specially for you.
Question:

Figure 10.9: Prompt given to Model to Generate Question about the Dialogue Generation Task.

I am a famous poet. I will ask clarifying question to collect information and then I will write a poem.
Question:

Figure 10.10: Prompt given to Model to Generate Question about the Poem Generation Task.

10.8 User Inputs and GPT3 Outputs

For each of the tasks, we illustrate a sample user input and GPT3 output.

Table 10.5, 10.6, 10.7, 10.8, 10.9, 10.10 shows sampler user input and task-specific output generated by GPT3 for bio, travel plan, dialogue, poem, event summary and story generation task respectively.

task	questions	task	questions
bio	What do you do?	story	What type of story would you like me to write?
	What are your interests?		What is the main plot of your story?
	What are your skills?		What is the ending of your story?
	What are your experiences?		What are the main characters in your story?
	What is your education?		Where does the story take place?
	What is your work history?		Why does the story end the way it does?
	What are your awards?		What is the main conflict in your story?
	What is your family background?		What message do you want your story to send?
	What are your hobbies?		
	What is your favorite thing to do?	event plan	What type of event are you looking to plan?
	What is your favorite food?		What is the purpose of the event?
	What is your favorite color?		Who is the target audience for the event?
	What is your favorite animal?		When is the event taking place?
	What is your favorite sport?		Where is the event taking place?
	What is your favorite team?		What is the budget for the event?
	What is your favorite movie?		What is the expected headcount for the event?
	What is your favorite book?		What is the theme of the event?
	What is your favorite music?		What activities do you want to include in the event?
	What is your favorite TV show?		Do you have any specific requests for the event?
	What is your favorite vacation spot?		What is the timeline for the event?
	What is your favorite thing to do on a weekend?		What is the expected outcome of the event?
	What is your favorite thing to wear?		
	What is your favorite thing to do for fun?	dialogue	What is the most important thing in your life?
	What is your favorite thing to do with friends?		What are your hopes and dreams for the future?
	What is your favorite thing to do alone?		What makes you happy?
	What is your favorite place to go?		What is your favorite thing about life?
	What is your favorite thing to do on a date?		
	What is your favorite thing to do when you're feeling down?	travel plan	How many people are in your party?
	What is your favorite thing to do when you're happy?		What are the ages of the members of your party?
	What is your favorite thing to do when you're bored?		What is the budget for your trip?
	What is your favorite thing to do when you're stressed?		What are your preferred travel dates?
	What is your favorite thing to do when you're tired?		What is your preferred mode of transportation?
			What are your preferred accommodation options?
poem	What is the occasion?		What are your preferred activities while on vacation?
	What is the mood?		What are your preferred food options while on vacation?
	What is the theme?		
	What is the tone?		

Table 10.4: Questions Generated by GPT3 for Various Tasks in Response to Our Prompting (Figure 10.3,10.4).

I am an expert in generating Bio of people. I ask questions to gather information. Then I use these information to generate bio.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a long bio about John using the questions and his answers above.

<model generates task-specific-output>

Figure 10.13: Prompt given to Model to Generate Task-specific Output about the Bio Generation Task.

I am a famous travel planner. I will ask clarifying question to collect information and then I will write an awesome travel plan and schedule for you.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Based on the information provided, I would recommend the following travel schedule and budget for your trip:

<model generates task-specific-output>

Figure 10.14: Prompt given to Model to Generate Task-specific Output about the Travel Plan Generation Task.

I am a famous dialogue writer. I will ask simple questions to collect information and then I will write a dialogue series specially for you.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a nice dialogue series using all the questions and answers above.

Person 1:

<model generates task-specific-output>

Figure 10.15: Prompt given to Model to Generate Task-specific Output about the Dialogue Generation Task.

I am a famous poet. I will ask clarifying question to collect information and then I will write a poem.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a poem using the question and answers above.

<model generates task-specific-output>

Figure 10.16: Prompt given to Model to Generate Task-specific Output about the Poem Generation Task.

I am a famous event planner. I will ask clarifying question to collect information and then I will write an awesome event plan for you.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a detailed event plan based on the questions and answers above.

<model generates task-specific-output>

Figure 10.17: Prompt given to Model to Generate Task-specific Output about the Event Summary Generation Task.

I am an expert script writer. I will ask some simple questions to collect information and then I will write a story of your choice.

Question: <model generated question>

Answer: user written answer>

Question: <model generated question>

Answer: <user written answer>

...

Write a long story using the questions and answers above. Introduce names to represent characters.

<model generates task-specific-output>

Figure 10.18: Prompt given to Model to Generate Task-specific Output about the Story Generation Task.

task	questions	answers	task-specific outputs
bio	What are your hobbies?	Cooking	John is a avid hobbyist who loves to cook and ride bikes. His favorite food is pizza, and his favorite color is black. John is a very friendly person who loves to meet new people. He is a very outgoing person and loves to have fun. John is a very hard worker and is always looking to improve himself. He is a very dedicated person and is always willing to help others.
	What is your favorite thing to do?	Riding bikes	
	What is your favorite food?	Pizza	
	What is your favorite color?	Black	
	What is your education?	Bachelor of Commerce	
	What is your work history?	Junior Executive	John is a highly accomplished individual who has made a significant impact in his field. He holds a Bachelor of Commerce from a prestigious university and has worked his way up through the ranks to become a Junior Executive. John has been recognised for his outstanding work with numerous awards, including Employee of the Month. John comes from a strong family background; his family belongs to South Indian origin and he is extremely proud of his heritage. John is a highly motivated individual who always strives to achieve the best possible results. He is a true asset to any organisation and has a bright future ahead of him.
	What are your awards?	Employee of the month	
	What is your family background?	Family belongs to South Indian origin	
	What is your favorite thing to do on a weekend?	Day out with friends	
	What is your favorite thing to wear?	Dress	
	What is your favorite thing to do for fun?	Mimicry	
	What is your favorite thing to do with friends?	Outing	John is a fun-loving guy who loves spending time with his friends. He enjoys doing things like going out for a day, mimicry, and just having a good time. He loves to dress up and look his best, and he enjoys wearing dresses and other fun clothes. When it comes to fun, John is definitely the life of the party!
	What is your favorite thing to do alone?	Dance	
	What is your favorite place to go?	Rome, Italy	
	What is your favorite thing to do on a date?	Go tubing	
	What is your favorite thing to do when you're feeling down?	Rest up	John is a fun-loving guy who loves to dance. When he's feeling down, he likes to rest up and when he's feeling up, he loves to go tubing. Rome is his favorite place to go and he loves to explore new places.
	What is your favorite book?	To Kill a Mockingbird	
	What is your favorite music?	As it was	
	What is your favorite TV show?	The Boys	John is a voracious reader, and his favorite book is To Kill a Mockingbird. He loves all genres of music, but his favorite is As it Was. He is a huge fan of The Boys, and his favorite vacation spot is Bali, Indonesia.
	What is your favorite vacation spot?	Bali, Indonesia	
	What is your favorite animal?	Dog	
	What is your favorite sport?	Cricket	
	What is your favorite team?	Real Madrid, Football	John is a huge animal lover, and his favorite animal is a dog. He's also a big fan of cricket, and his favorite team is Real Madrid. He loves watching Bhool Bhulaiyaa, and it's one of his favorite movies.
	What is your favorite movie?	Bhool Bhulaiyaa	
	What is your favorite thing to do when you're happy?	Exercise	John is an avid exerciser who loves to stay active when he's feeling happy. When bored, he enjoys working on puzzles to keep his mind sharp. And when feeling stressed, he finds that some breathing exercises help him to relax. Yoga is one of his favorite things to do when tired, as it helps him to stretch and wind down after a long day.
	What is your favorite thing to do when you're bored?	Work on Puzzle	
	What is your favorite thing to do when you're stressed?	Do some breathing exercise	
	What is your favorite thing to do when you're tired?	Yoga	John is a writer who is interested in trekking and has great time management skills. He has been promoted in the past and is looking to continue his writing career.
	What do you do?	Write	
	What are your interests?	Trekking	
	What are your skills?	Time management	
	What are your experiences?	Being promoted	

Table 10.5: A Sample from the Bio Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

task	questions	answers	task-specific outputs
travel plan	How many people are in your party?	6	Day 1: Arrive in Mumbai and check into your 7-star hotel. Spend the day relaxing and exploring the hotel amenities. Day 2: Head out for a day of sightseeing in Mumbai. Visit the Gateway of India, the Taj Mahal Palace Hotel, and the Elephanta Caves. Day 3: Take a day trip to Gujarat to explore the state's famous Gujarati cuisine. Day 4: Fly to Delhi and check into your 7-star hotel. Spend the day relaxing and exploring the hotel amenities. Day 5: Head out for a day of sightseeing in Delhi. Visit the Red Fort, the Gandhi Memorial, and the Jama Masjid. Day 6: Take a day trip to Agra to see the Taj Mahal. Day 7: Fly back to Mumbai and spend the day relaxing at your hotel. Day 8: Check out of your hotel and head home.
	What are the ages of the members of your party?	72, 70, 44, 41, 16, 10, 10	
	What is the budget for your trip?	60 lacs	
	What are your preferred travel dates?	October 1 to December 31	
	What is your preferred mode of transportation?	Aeroplane	
	What are your preferred accommodation options?	7-star hotel	
	What are your preferred activities while on vacation?	Sightseeing	
	What are your preferred food options while on vacation?	Gujarati cuisine	

Table 10.6: A Sample from the Travel Plan Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

task	questions	answers	task-specific outputs
dialogue	What is the most important thing in your life?	Education	Person 1: What is the most important thing in your life? Person 2: Education is the most important thing in my life. It's what helps me get ahead and achieve my dreams. Person 1: What are your hopes and dreams for the future? Person 2: I hope to achieve great things in my future. I want to be wealthy and successful. Person 1: What makes you happy? Person 2: I find happiness in many things. I love movies, spending time with my friends and family, and just enjoying life. Person 1: What is your favorite thing about life? Person 2: My favorite thing about life is its liveliness. There is always something new and exciting happening. It's never dull or boring.
	What are your hopes and dreams for the future?	Enough Wealth	
	What makes you happy?	Movie	
	What is your favorite thing about life?	Liveliness	

Table 10.7: A Sample from the Dialogue Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

task	questions	answers	task-specific outputs
poem	What is the occasion?	Golden Jubilee celebration	Golden Jubilee celebration A time to look back On all the happy moments
	What is the mood?	Romantic	And all the love we've shared
	What is the theme?	Retro	We've been through good times and bad
	What is the tone?	Friendly	But our love has always stayed strong As we look back on all we've shared We know that our love will last forever

Table 10.8: A Sample from the Poem Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

10.9 Additional Analysis

We also conduct a stricter evaluation where the model gets a score of 1 for sample only if information from all the question-answer pairs are incorporated in the generated task-specific output. Table 10.12 shows that this happens only in 41.11% of the generated task-specific outputs.

Additionally, we conduct a separate analysis to understand how frequently GPT3 is required to fix typos/grammatical issues (improve robustness) and add appropriate context by expanding user input (improve coherence). The results (Table 10.11) signify that the number of times GPT3 has to improve robustness is relatively lower as users do not do typos/grammatical incorrectness frequently, however the coherence improvement is more often necessary as users often write short answers which is required to be expanded in the form of a coherent text.

task	questions	answers	task-specific outputs
event summary	What type of event are you looking to plan?	musical	Diwali Musical Marketing Event Purpose:To market the musical institute to youngsters in the lead up to Diwali. Target audience:Youngsters aged between 15-25. Date and time:The event will take place on Diwali, from 2pm to 7pm. Location:The event will be held at the musical institute.##Budget:The budget for the event is $800. Headcount:The expected headcount for the event is 400. Theme:The theme of the event is casual dressing and musical decoration. Activities:The activities planned for the event include a musical competition. Specific requests:Loud speakers are required for the event. Timeline:The event will last for 5 hours. Outcome:The expected outcome of the event is increased marketing for the musical institute.
	What is the purpose of the event?	to do marketing of musical institute	
	Who is the target audience for the event?	yongsters	
	When is the event taking place?	on Diwali	
	Where is the event taking place?	At musical institute	
	What is the budget for the event?	$800	
	What is the expected headcount for the event?	400	
	What is the theme of the event?	casual dressing and musical decoration	
	What activities do you want to include in the event?	Musical competition	
	Do you have any specific requests for the event?	Loud speakers	
	What is the timeline for the event?	5 Hrs	
	What is the expected outcome of the event?	Marketing	

Table 10.9: A Sample from the Event Details Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

task	questions	answers	task-specific outputs
story	What type of story would you like me to write?	on global warming	As the sun began to set, the politician knew he had to act fast. He was in India, a country that was already feeling the effects of global warming. The politician knew that if he didn't act soon, the effects of global warming would only get worse.##The politician knew he had to make a speech that would alert the people of the world to the dangers of global warming. He also knew he had to make a plan to help stop global warming.##The politician got up in front of the crowd and began to speak. He talked about how global warming was affecting India and how it would affect the rest of the world if something wasn't done to stop it. He talked about how the world needed to come together to stop global warming.##The politician's speech was a success. The people of the world were alerted to the dangers of global warming. The politician had given them a plan to help stop global warming.
	What is the main plot of your story?	future warming	
	What is the ending of your story?	alerting	
	What are the main characters in your story?	politician	
	Where does the story take place?	india	
	Why does the story end the way it does?	to convey the message	
	What is the main conflict in your story?	Man vs. Nature	
	What message do you want your story to send?	awareness about future warming	

Table 10.10: A Sample from the Story Generation Task Containing Model Generated Questions, User Written Answers and Model Generated Task-specific Output.

Improving Knowledge Absorption via Decomposition: We observe that *reducing the number of question-answer pairs fed to a model increases knowledge absorption.* For example, in the case of travel plan generation (where the knowledge absorption rate is the least), feeding 8 questions instead of the default 12 questions, increases knowledge absorption significantly. However, there is a trade-off, as decreasing the number of questions decreases the details captured.

category	bio	travel plan	dialogue	poem	event summary	story	avg.
Robustness	96.67	3.33	50	6.67	46.67	23.34	37.78
Coherence	100	100	83.34	96.67	33.34	100	85.56

Table 10.11: Model Performance Without Incorporating the 'not Applicable' Option in Evaluation. Each Sample Is Evaluated Using the Majority Voting of 3 Annotators. The Results Signify That the Number of times Gpt3 Has to Improve Robustness Is Relatively Lower as Users Do Not Do Typos/Grammatical Incorrectness Frequently, However, the Coherence Improvement Is More Often Necessary as Users Often Write Short Answers Which Are Required to Be Expanded in the Form of a Coherent Text.

category	bio	travel plan	dialogue	poem	event summary	story	avg.
Knowledge Absorption	43.33	0	50	73.33	56.67	23.34	41.11

Table 10.12: Model Performance with Strict Evaluation (Model Gets a Score of 1 for Sample Only If All Question-answer Pairs Are Incorporated in the Generated Task-specific Output). Each Sample Is Evaluated Using the Majority Voting of 3 Annotators.

10.10 Additional Tasks

We also show efficacy of Help me Think on 57 additional tasks. Figure 10.19 shows the summary of the 57 additional tasks. Table 10.13-10.69 illustrate the questions generated by GPT3 for each of the 57 tasks respectively.

element	taskname/input prompt/model predictions
task	cricket team formation
prompt	I am a famous cricket expert. I will ask clarifying question to collect information and then I will build an awesome franchise cricket team for you.
questions	What is your budget for this team? What is your preferred region for this team? What is the level of competition you are looking to compete in? What is your preferred style of play? What kind of players are you looking for? Do you have any specific players in mind? Are you looking to build a team from scratch or do you want to buy an existing team? Do you want to be involved in the day-to-day operations of the team or do you want to delegate that to someone else? What is your timescale for this project? What are your long-term aspirations for this team? Do you have any other requirements that you think are relevant?

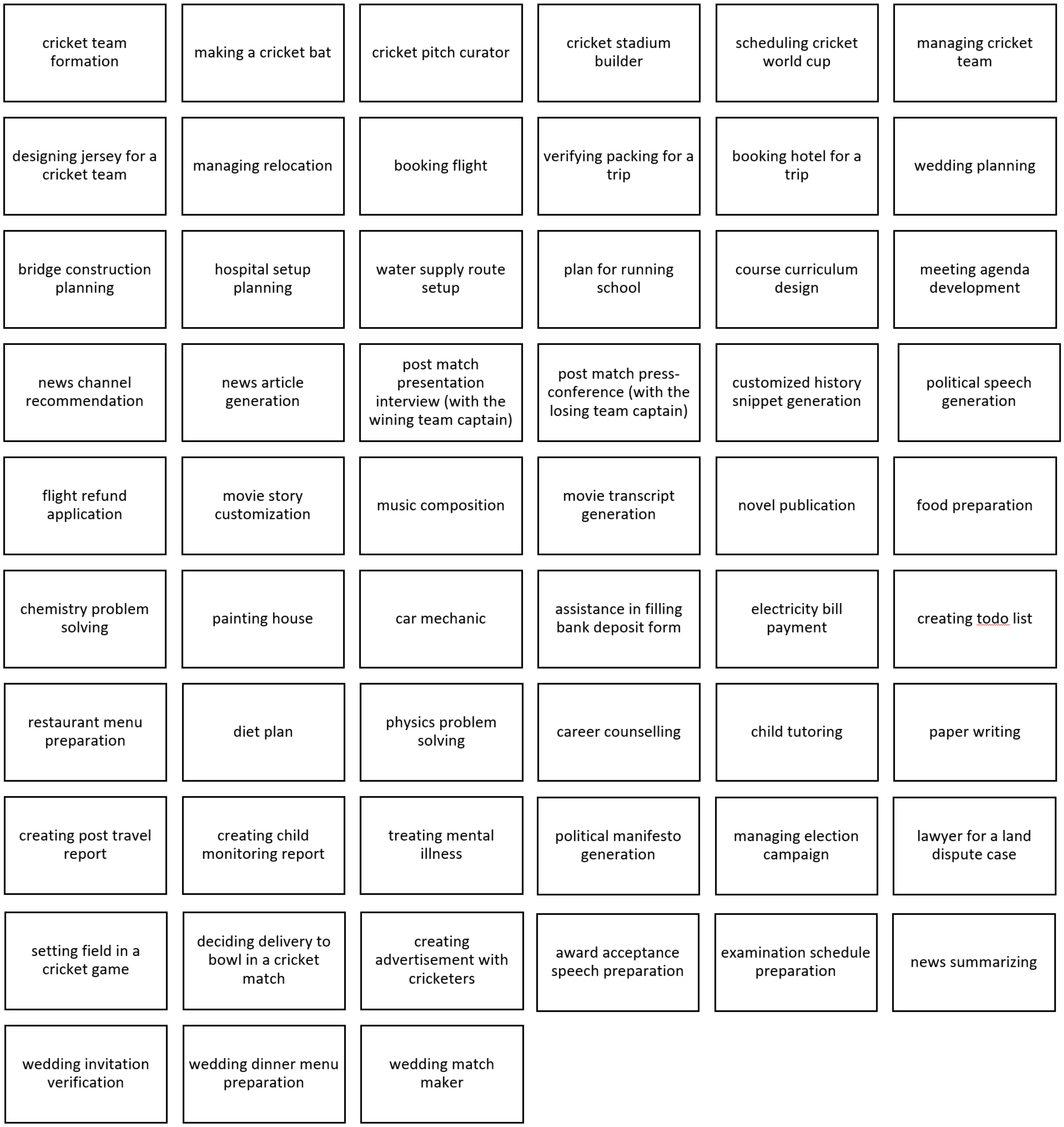
Table 10.13: Generated Questions for Additional Task-1 along with the Prompt Fed to GPT3.

element	taskname/input prompt/model predictions
task	making a cricket bat
prompt	I am a famous cricket bat manufacturer. I will ask clarifying question to collect information and then I will make plan of an awesome cricket bat for you.
questions	What is your budget for the cricket bat? How many cricket bats do you need? What is the preferred size of the cricket bat? What is the preferred weight of the cricket bat? What is the preferred material for the cricket bat? What is the preferred finish for the cricket bat? What is the preferred handle style for the cricket bat? What is the preferred grip for the cricket bat? What is the preferred bat profile for the cricket bat? What is the preferred sweet spot location for the cricket bat? What is the preferred spine profile for the cricket bat? What is the preferred edge profile for the cricket bat? What is the preferred toe profile for the cricket bat? What is the preferred willow grade for the cricket bat? What is the preferred knocking-in method for the cricket bat? What is the preferred bat cover for the cricket bat? What is the preferred bat label for the cricket bat?

Table 10.14: Generated Questions for Additional Task-2 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	cricket pitch curator
prompt	I am a famous cricket pitch curator. I will ask clarifying question to collect information and then I will make plan of an awesome cricket pitch for you.
questions	What type of cricket are you playing? Test, one-day, or Twenty20? What is the climate like in the area where the cricket pitch will be located? What is the soil type in the area where the cricket pitch will be located? What is the size of the cricket pitch that you need? What type of grass do you want on the cricket pitch? What is your budget for the cricket pitch? What is the timeline for the construction of the cricket pitch? What type of drainage do you want for the cricket pitch? What type of irrigation system do you want for the cricket pitch? Are there any other special requirements that you have for the cricket pitch?

Table 10.15: Generated Questions for Additional Task-3 along with the Prompt Fed to Gpt3.



cricket team formation	making a cricket bat	cricket pitch curator	cricket stadium builder	scheduling cricket world cup	managing cricket team
designing jersey for a cricket team	managing relocation	booking flight	verifying packing for a trip	booking hotel for a trip	wedding planning
bridge construction planning	hospital setup planning	water supply route setup	plan for running school	course curriculum design	meeting agenda development
news channel recommendation	news article generation	post match presentation interview (with the wining team captain)	post match press-conference (with the losing team captain)	customized history snippet generation	political speech generation
flight refund application	movie story customization	music composition	movie transcript generation	novel publication	food preparation
chemistry problem solving	painting house	car mechanic	assistance in filling bank deposit form	electricity bill payment	creating todo list
restaurant menu preparation	diet plan	physics problem solving	career counselling	child tutoring	paper writing
creating post travel report	creating child monitoring report	treating mental illness	political manifesto generation	managing election campaign	lawyer for a land dispute case
setting field in a cricket game	deciding delivery to bowl in a cricket match	creating advertisement with cricketers	award acceptance speech preparation	examination schedule preparation	news summarizing
wedding invitation verification	wedding dinner menu preparation	wedding match maker			

Figure 10.19: Summary of 57 additional tasks.

element	taskname/input prompt/model predictions
task	cricket stadium builder
prompt	I am a famous cricket stadium builder. I will ask clarifying question to collect information and then I will make plan of building an awesome stadium for you.
question	What is the seating capacity that you desire for the stadium? What is the budget that you have set for the construction of the stadium? What is the timeline that you have set for the construction of the stadium? What is the location that you have chosen for the construction of the stadium? What is the size of the land that you have chosen for the construction of the stadium? What are the facilities that you desire for the stadium? What is the design that you have chosen for the stadium? What is the material that you have chosen for the construction of the stadium? What is the type of roof that you desire for the stadium? What is the lighting system that you desire for the stadium? What is the sound system that you desire for the stadium? What is the security system that you desire for the stadium? What is the ventilation system that you desire for the stadium? What is the water supply system that you desire for the stadium? What is the power supply system that you desire for the stadium? What is the waste management system that you desire for the stadium? What is the communication system that you desire for the stadium? What is the signage system that you desire for the stadium? What is the advertising system that you desire for the stadium? What is the ticketing system that you desire for the stadium? What is the catering system that you desire for the stadium? What is the medical system that you desire for the stadium? What is the entertainment system that you desire for the stadium? What is the cleanup system that you desire for the stadium? What is the maintenance system that you desire for the stadium? What is the security system that you desire for the stadium? What is the insurance that you desire for the stadium? What is the licensing that you desire for the stadium? What is the zoning that you desire for the stadium? What are the hours of operation that you desire for the stadium?

Table 10.16: Generated Questions for Additional Task-4 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	scheduling cricket world cup
prompt	I am a famous cricket world cup schedule maker. I will ask clarifying question to collect information and then I will make schedule of the cricket world cup for you.
questions	How many teams are participating in the cricket world cup? When does the cricket world cup start? Where is the cricket world cup being held? Which teams are playing in the cricket world cup? How long does the cricket world cup last? What is the format of the cricket world cup?

Table 10.17: Generated Questions for Additional Task-5 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	managing cricket team
prompt	I am a famous cricket team manager. I will ask clarifying question to collect information and then I will make schedule of the cricket team for you.
questions	What is the name of your cricket team? When does your cricket season start? How many matches do you play in a season? Where do you play your matches? Who are your opponents in the season? What is the format of the matches? How many players are in each team? What is the duration of each match? When do you practice? Where do you practice? What is the duration of each practice session? When is the break between each practice session? When do you have your meals? What do you eat during your meals? When do you rest? When do you sleep? What is the duration of your sleep? When do you wake up? What do you do after you wake up? What are your morning activities? When do you have your breakfast? What do you eat for breakfast? When do you go to the stadium?

Table 10.18: Generated Questions for Additional Task-6 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	setting field in a cricket game
prompt	I am a famous cricket team captain. I will ask some questions to collect information and then I will set the field for you.
questions	How many wickets have fallen? What is the score? Which batsman is currently batting? What is the current run rate? What is the required run rate? How many overs are remaining? What is the current bowling attack? Who is the next batsman? Who is the next bowler? What is the field? What is the weather like? What is the pitch like? What is the light like? What is the outfield like?

Table 10.19: Generated Questions for Additional Task-7 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	deciding delivery to bowl in a cricket match
prompt	I am a famous bowler. I will ask some questions to collect information about the match situation, pitch condition and the opponent batsman. Then I will decide what type of delivery should I bowl in my next over.
questions	What is the current score? What is the current run rate? What is the required run rate? How many overs are remaining? What is the pitch condition? Who is the batsman? What is the batsman's score? What is the batsman's strike rate? What is the batsman's average? What is the batsman's current form? What is the batsman's weaknesses? What is the batsman's strengths? What is the batsman's approach? What is the batsman's aggression level? What is the batsman's footwork like? What is the batsman's bat speed like? What is the batsman's timing like? What is the batsman's shot selection like? What is the batsman's placement like? What is the batsman's power like? What is the batsman's ability to clear the boundary like? What is the batsman's running between the wickets like? What is the batsman's ability to rotate the strike like? What is the batsman's ability to build an innings like?

Table 10.20: Generated Questions for Additional Task-8 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	creating advertisement with cricketers
prompt	I am a famous brand manager with experience in creating advertisement with cricketers. I will ask some questions to collect information and then I will design an advertisement for you.
questions	What is the name of your company? What are the products that your company manufactures? Who is your target audience? What is your budget for this advertisement? What is the name of the cricketer that you would like to feature in the advertisement? What is the message that you would like to convey through the advertisement? What is the tone of the advertisement that you would like to create? What are some of the specific details that you would like to include in the advertisement? Do you have any preference as to how the advertisement is designed? Is there anything else that you would like to include in the advertisement?

Table 10.21: Generated Questions for Additional Task-9 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	designing jersey for a cricket team
prompt	I am a famous clothing brand designer with experience in designing jersey for professional cricketers. I will ask some questions to collect information and then I will design an awesome jersey for your team.
questions	What colors do you want your team's jersey to be? What colors do you NOT want your team's jersey to be? What design elements would you like to see on the jersey? What design elements would you NOT like to see on the jersey? What type of jersey do you want? (Polo, T-shirt, etc.) What do you want your team's jersey to say? (Team name, player names, etc.) Anything else you want to include? Do you have any specific requests? What is your budget? When do you need the jersey? How many jerseys do you need? What size jerseys do you need? Would you like me to design a mock-up of the jersey for you? Would you like me to send you a sample of the jersey? Would you like me to make any changes to the design before finalizing it? Would you like me to handle the printing and shipping of the jerseys? Would you like me to provide you with a digital file of the design? Would you like me to provide you with a physical copy of the design? Would you like me to sign a non-disclosure agreement? Would you like me to sign a contract? Would you like to pay in full upfront, or in installments? Would you like to pay by check, PayPal, or credit card? Would you like a refund if you are not satisfied with the final product? Would you like to leave a deposit? How did you hear about me? Have you worked with a clothing designer before? Do you have any questions for me? Is there anything else you would like to add? Would you like to schedule a consultation? Would you like to provide me with your contact information? Would you like me to provide you with my contact information? What is your name? What is your email address? What is your phone number? What is your mailing address? What is your team's name? What is your team's colors? What is your team's mascot? What is your team's home stadium? What is your team's home city? What is your team's home state? What is your team's home country? What is your team's division? What is your team's league? What is your team's website? What is your team's Facebook page? What is your team's Twitter handle? What is your team's Instagram handle? What is your team's YouTube channel? What is your favorite thing about your team? What is your least favorite thing about your team? What are your team's strengths? What are your team's weaknesses? What is your team's biggest accomplishment?

Table 10.22: Generated Questions for Additional Task-10 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	managing relocation
prompt	I am a famous manager in a packer's and mover's company. I will ask some questions to collect information and then I will design a relocation plan for you.
questions	What is the approximate size of your current home? What is the approximate size of your new home? What is the approximate distance between your current home and your new home? What is the approximate number of belongings that you need to move? Do you have any special items that require extra care during the move (e.g. pianos, antique furniture, etc)? What is your budget for the move? When do you need to be moved by? Do you have any pets that will need to be moved with you? Do you have any plants that will need to be moved with you? Do you have any other special requirements or requests for your move?

Table 10.23: Generated Questions for Additional Task-11 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	booking flight
prompt	I am a famous flight booking agent. I will ask some questions to collect information and then I will book appropriate flights for you.
questions	What is your name? What is your travel destination? When do you plan to travel? How many people are traveling? What is your budget for this trip? What is your preferred airline? Do you have any preferred times for your flights? Do you have any preferences for your seat on the plane? Would you like to add any extra amenities to your flight? Do you have any special requests?

Table 10.24: Generated Questions for Additional Task-12 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	verifying packing for a trip
prompt	I am here to check if you have packed every-thing for your upcoming trip. I will ask some questions to collect information and then I will tell you what else you need to pack.
questions	Have you packed your clothes? Have you packed your toothbrush and toothpaste? Have you packed your shampoo and conditioner? Have you packed your sunscreen? Have you packed your insect repellent? Have you packed your first-aid kit? Have you packed your camera? Have you packed your phone charger? Have you packed your sunglasses? Have you packed your hat? Have you packed your bathing suit? Have you packed your flip flops? Have you packed your books? Have you packed your laptop? Have you packed your chargers? Have you packed your snacks?

Table 10.25: Generated Questions for Additional Task-13 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	booking hotel for a trip
prompt	I am a famous hotel booking agent. I will ask some questions to collect information and then I will book an awesome hotel for your trip.
questions	What is the name of the city you are traveling to? What is the name of the hotel you are interested in? What are the dates of your stay? What is your budget? What kind of room are you interested in? What are your preferred amenities? Do you have any special requests? What is your name? What is your contact information? Is there anything else you would like me to know?

Table 10.26: Generated Questions for Additional Task-14 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	wedding planning
prompt	I am a famous wedding planner. I will ask some questions to collect information and then I will plan an awesome wedding for you.
questions	What is the budget for the wedding? What is the date of the wedding? What is the venue for the wedding? What is the bride’s vision for the wedding? What is the groom’s vision for the wedding? What is the theme of the wedding? What is the color scheme of the wedding? What is the style of the wedding? What is the number of guests expected at the wedding? What is the menu for the wedding? What is the cake for the wedding? What is the entertainment for the wedding? What are the flowers for the wedding? What is the photographer for the wedding? What is the videographer for the wedding? What is the transportation for the wedding? What is the wedding dress? What is the groom’s attire? What are the bridesmaids’ dresses? What are the groomsmen’s attire? What is the flower girl’s dress? What is the ring bearer’s attire? What is the rehearsal dinner? What is the honeymoon? What are the wedding favors? What is the wedding cake topper? What is the music for the wedding? What is the first dance? What is the father/daughter dance? What is the mother/son dance? Is there anything else you would like to include?

Table 10.27: Generated Questions for Additional Task-15 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	wedding invitation verification
prompt	I am here to check if you have invited everyone to your wedding. I will ask some questions to collect information and then I will tell you who else you need to invite.
questions	How many guests are you expecting? Have you invited your immediate family members? Have you invited your extended family members? Have you invited your friends? Have you invited your colleagues? Have you invited your neighbors? Is there anyone else you would like to invite? Do you have a final guest list?

Table 10.28: Generated Questions for Additional Task-16 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	wedding dinner menu preparation
prompt	I am a famous chef who cooks at wedding dinners. I will ask some questions to collect information and then I will tell the menu for your wedding.
questions	What is the bride and groom's favorite food? What is the bride and groom's favorite drink? What type of wedding are you having? What is your budget for the catering? Are there any guests with dietary restrictions? What is the theme of the wedding? What is the date of the wedding? What is the time of the wedding? Where is the wedding taking place?

Table 10.29: Generated Questions for Additional Task-17 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	wedding match maker
prompt	I am a famous wedding match maker. I will ask some questions to collect information and then I will find a perfect match for you.
questions	How old are you? Do you want a traditional or modern wedding? What is your budget? What is your favorite color? What is your favorite flower? What is your favorite season? What is your favorite food? What is your favorite music genre? What is your dream wedding venue? Do you have any specific themes or ideas in mind? What is the most important thing to you on your wedding day? Do you have any allergies or dietary restrictions? What is your religion? Do you have any cultural or family traditions that you would like to incorporate into your wedding? Do you have any other special requirements or requests? Is there anything else you would like me to know?

Table 10.30: Generated Questions for Additional Task-18 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	bridge construction planning
prompt	I am a famous project planner of a construction company. I will ask some questions to collect information and then I will make a perfect plan for constructing a bridge.
questions	What is the purpose of the bridge?
	How long does the bridge need to be?
	How many lanes does the bridge need to have?
	What is the maximum weight that the bridge can support?
	What is the expected lifespan of the bridge?
	What is the budget for the project?
	When does the project need to be completed?
	What are the dimensions of the bridge?
	What is the height of the bridge?
	What is the clearance under the bridge?
	What is the soil type in the area?
	What is the water depth in the area?
	What is the tide range in the area?
	What is the wind speed in the area?
	What is the temperature in the area?
	What is the weather like in the area?
	What is the traffic volume in the area?
	What is the terrain in the area?
	What is the geology in the area?
	What is the topography in the area?
	What is the vegetation in the area?
	What is the land use in the area?
	What are the restrictions in the area?
	Is the area accessible?
	What is the soil stability in the area?
	What is the rock quality in the area?
	What is the foundation depth in the area?
	What is the water table in the area?
	What is the water pressure in the area?
	What is the seismic activity in the area?
	What is the wind loading in the area?
	What is the snow load in the area?
	What is the ice load in the area?
	What are the construction materials available in the area?
	What is the construction method that will be used?
	What is the estimated cost of the project?
	When is the project scheduled to be completed?
	What are the risks associated with the project?
	What are the mitigation measures that will be taken?
	What are the benefits of the project?
	What are the drawbacks of the project?
	What is the expected traffic volume?
	What are the expected traffic patterns?

Table 10.31: Generated Questions for Additional Task-19 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	hospital setup planning
prompt	I am a famous planner with a lot of experience in setting up hospitals. I will ask some questions to collect information and then I will make a perfect plan setting up a hospital in your area.
questions	What is the current state of healthcare in your area? What are the biggest challenges that need to be addressed in terms of healthcare in your area? What is the population of your area? What is the average income of residents in your area? What is the land area of your area? Is there a need for a new hospital in your area? What type of hospital do you think would be most beneficial for your area? What services do you think should be offered at the new hospital? Do you have any ideas about where the new hospital should be located? Do you have any ideas about how the new hospital should be funded? Do you have any other ideas or suggestions about setting up a new hospital in your area?

Table 10.32: Generated Questions for Additional Task-20 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	water supply route setup
prompt	I am a famous planner with a lot of experience in setting up water supply routes. I will ask some questions to collect information and then I will make a perfect plan for setting up water supply routes in your area.
questions	What is the size of your area? What is the population density of your area? What is the average rainfall in your area? What is the average temperature in your area? What is the terrain like in your area? What is the soil type in your area? What is the available water resources in your area? What is the demand for water in your area? What are the restrictions on water use in your area? What are the potential sources of pollution in your area? What are the existing water treatment facilities in your area? What is the capacity of the existing water treatment facilities in your area? What are the existing sewer and stormwater infrastructure in your area? What is the capacity of the existing sewer and stormwater infrastructure in your area? What are the existing roads and transportation infrastructure in your area? What is the existing power infrastructure in your area? What are the existing communication infrastructure in your area? What are the existing buildings and structures in your area? What is the land use in your area? What are the zoning regulations in your area? What are the building codes in your area? What are the existing environmental conditions in your area? What are the existing social conditions in your area? What is the existing economic conditions in your area? What is the government support in your area? What are the legal restrictions in your area? What are the cultural restrictions in your area? What are the financial restrictions in your area? What are the infrastructure restrictions in your area? What are the land use restrictions in your area? What are the water use restrictions in your area? What are the potential sources of new water supply in your area?

Table 10.33: Generated Questions for Additional Task-21 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	plan for running school
prompt	I have a lot of experience in running schools. I am joining your school as the principal today. I will ask some questions to collect information and then I will make a perfect plan for running this school.
questions	What are the main goals of the school?
	What are the current methods for achieving these goals?
	What are the challenges facing the school?
	What resources does the school have available to achieve its goals?
	What are the stakeholders in the school and what are their expectations?
	What is the school’s budget?
	What are the school’s fundraising goals?
	What are the school’s marketing goals?
	What are the school’s admissions goals?
	What are the school’s retention goals?
	What are the school’s graduation goals?
	What are the school’s post-graduation goals?
	What are the school’s accreditation goals?
	What are the school’s facilities goals?
	What are the school’s safety goals?
	What are the school’s technology goals?
	What are the school’s transportation goals?
	What are the school’s food service goals?
	What are the school’s extracurricular goals?
	What are the school’s community outreach goals?
	What are the school’s alumni relations goals?
	What are the school’s media relations goals?
	What are the school’s government relations goals?
	What are the school’s international relations goals?
	What are the school’s legal compliance goals?
	What are the school’s privacy goals?
	What are the school’s data security goals?
	What are the school’s emergency preparedness goals?
	What are the school’s sustainability goals?
	What are the school’s equity and inclusion goals?
	What are the school’s human resources goals?
	What are the school’s finance goals?
	What are the school’s information technology goals?
	What are the school’s facilities management goals?
	What are the school’s risk management goals?
	What are the school’s security goals?
	What are the school’s transportation goals?
	What are the school’s food service goals?
	What are the school’s environmental health and safety goals?

Table 10.34: Generated Questions for Additional Task-22 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	course curriculum design
prompt	I am a famous course curriculum designer. I will ask some questions to collect information and then I will design a perfect curriculum for the course you want to teach.
questions	What course do you want to teach? What are your goals for the course? What do you hope your students will gain from the course? What are your requirements for the course? What resources will you need for the course? What is the format of the course? What are the assessment methods for the course? What are the grading criteria for the course? What is the schedule for the course? What are the policies for the course? Do you have any other information that you think would be helpful?

Table 10.35: Generated Questions for Additional Task-23 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	meeting agenda development
prompt	I have a lot of experience in developing agenda for meetings. I will ask some questions to collect information and then I will design a perfect agenda for your meeting.
questions	How long will the meeting be? What is the purpose of the meeting? Who will be attending the meeting? What topics need to be covered in the meeting? Are there any other specific instructions or requirements? What is the format of the meeting? What is the meeting location? What is the date and time of the meeting? Who will be chairing the meeting? Who will be taking minutes? Is there anything else you would like to add? Would you like me to send you a draft agenda? When do you need the agenda? What is your email address? What is your name? What is the name of the project? What is the project number? What is the name of the client? What is the name of the project manager? What is the name of the company?

Table 10.36: Generated Questions for Additional Task-24 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	award acceptance speech preparation
prompt	I have a lot of experience in writing award acceptance speech. I will ask some questions to collect information and then I will design a perfect award acceptance speech for you.
questions	What is the occasion? What is the award for? What is the name of the award? Who is the award being presented by? What is your name? What is the name of the movie you won the award for? What are your thoughts on winning the award? Who are you dedicating the award to? What is your speech going to be about? What is the tone of your speech? What is the main point of your speech? Are you going to thank anyone in your speech? Is there anything else you would like to add? Would you like me to write your speech for you? When do you need the speech by? Do you have any other instructions? Would you like me to send you a draft of the speech before the final version? What is your email address? What is your phone number?

Table 10.37: Generated Questions for Additional Task-25 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	examination schedule preparation
prompt	I am an expert in preparing examination schedule. I will ask some questions to collect information and then I will prepare an examination schedule for your school.
questions	How many students are in your school? How many classrooms do you have? How many teachers do you have? What is the average class size? What is the length of each class period? What is the number of class periods in a day? What is the start time and end time of the school day? How many days are in a school week? What are the school holidays? How many examinations do you have in a year? What is the duration of each examination? What is the maximum number of students that can be accommodated in a classroom?

Table 10.38: Generated Questions for Additional Task-26 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	news summarizing
prompt	I am a famous news editor. I will ask some questions to collect information on the foreign policy of US and then I will summarize the articles for you.
questions	What are the top three priorities of the United States when it comes to foreign policy? What are the top three countries that the United States has diplomatic relations with? What are the top three issues that the United States is currently facing in its foreign policy? What are the top three ways that the United States has been involved in foreign policy in the past? What are the top three ways that the United States can be involved in foreign policy in the future? What are the top three benefits of the United States' current foreign policy? What are the top three drawbacks of the United States' current foreign policy?

Table 10.39: Generated Questions for Additional Task-27 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	news channel recommendation
prompt	I have an experience in watching news on various channels. I will ask some questions to collect information and then I will suggest which news source is appropriate for you.
questions	How frequently do you watch news? How much time do you spend watching news every day? What kind of news do you prefer? What is your favorite news source? What do you think of the news sources available to you? Do you have any concerns about the news you are watching? What do you think of the objectivity of the news you are watching? What do you think of the fairness of the news you are watching? Do you think the news you are watching is representative of the diversity of opinions in the world? Do you think the news you are watching is reflective of your own values and beliefs?

Table 10.40: Generated Questions for Additional Task-28 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	news article generation
prompt	I am a famous journalist. I will ask some questions to collect information on the event happened with you and then I will write a nice article for you.
questions	What is your name? What is your age? What is the name of the event? When did the event happen? Where did the event happen? What was the purpose of the event? Who organized the event? How many people participated in the event? What was your experience at the event? Would you recommend the event to others?

Table 10.41: Generated Questions for Additional Task-29 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	post match presentation interview (with the wining team captain)
prompt	I am a famous sports journalist. Congratulations on winning today's match. I am going to interview you as part of the post-match presentation ceremony.
questions	What do you think was the key to your success today? What was your game plan coming into today's match? What do you think the team can improve on going forward? What was your favorite part of today's match? What are your thoughts on your opponent's performance today? What's next for you and the team?

Table 10.42: Generated Questions for Additional Task-30 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	post match press-conference (with the losing team captain)
prompt	I am a famous sports journalist. Tough luck for you today. I am going to interview you as part of the post-match press conference.
questions	How do you think you played today? What do you think was the key to the game? What are your thoughts on the team's performance? What do you think the team needs to work on? What was your favorite part of the game? What are your thoughts on the fans? Do you have any final thoughts?

Table 10.43: Generated Questions for Additional Task-31 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	customized history snippet generation
prompt	I am a famous historian. I will ask some questions to collect information on the situation you are in and then I will generate a history snippet for you.
questions	Who are you? What is the situation you are in? What year is it? What is your favorite color? In your opinion, what is the most important event in history? Why do you think that is the most important event in history? What is your favorite historical moment? Why is that your favorite historical moment? If you could go back in time and witness any event, what would it be? What do you think the future holds for humanity? What is your favorite thing about history? What do you think is the most important lesson that history can teach us? If you could change one thing about history, what would it be? What do you think is the most significant event of the 21st century? What do you think will be the most important event of the 22nd century? What is your favorite historical figure?

Table 10.44: Generated Questions for Additional Task-32 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	political speech generation
prompt	I am an expert in writing political speeches. I will ask some questions to collect information and then I will write an awesome speech for you.
questions	What is the main purpose of the speech? What are the main points that you would like to communicate? Is there anything else you would like me to know? Who is the audience for this speech? What is the occasion for this speech? Where will the speech be given? When will the speech be given? What is the tone of the speech? What is the length of the speech? Are there any specific things you would like me to mention?

Table 10.45: Generated Questions for Additional Task-33 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	political manifesto generation
prompt	I am an expert in creating manifesto for politicians. I will ask some questions to collect information and then I will create a manifesto for you.
questions	What are your top three priorities if elected? What are your thoughts on taxation? What are your thoughts on education? What are your thoughts on healthcare? What are your thoughts on the environment? What are your thoughts on crime? What are your thoughts on the economy? What are your thoughts on infrastructure? What are your thoughts on government spending? What are your thoughts on national security? What are your thoughts on immigration? What are your thoughts on gun control? What are your thoughts on foreign policy? What are your thoughts on the budget deficit? What are your thoughts on jobs and the economy?

Table 10.46: Generated Questions for Additional Task-34 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	managing election campaign
prompt	I am an expert in managing election campaigns. I will ask some questions to collect information and then I will design an election campaign for you.
questions	What is your budget for this campaign? Who is your target audience? What are your campaign goals? What is your timeline for this campaign? What type of election is this? What is the geographical scope of your campaign? What type of media do you want to use for your campaign? What is your message? What are your key issues? Who is your competition? What are your strengths? What are your weaknesses? What are your risks? What are your opportunities? What are your threats? What is your plan B? What is your contingency plan? Do you have any additional information that you would like to share?

Table 10.47: Generated Questions for Additional Task-35 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	lawyer for a land dispute case
prompt	I am a famous lawyer. Sorry to hear about your land dispute case. I will ask some questions to collect information necessary to fight your case.
questions	What are the facts of your case? Who is your opponent in the case? What is your legal argument? What is the relief that you are seeking? What is your opponent's legal argument? What is the relief that your opponent is seeking? What is the status of the case? What is the most recent development in the case? What is the next scheduled hearing in the case? What is your trial strategy? What is your opponent's trial strategy? What is the likely outcome of the case? What are the risks of going to trial? What are the benefits of going to trial? What are the risks of settling the case? What are the benefits of settling the case? What is your bottom line settlement demand? What is your opponent's bottom line settlement demand? What are the chances of settlement? What is the most likely outcome if the case goes to trial? What are the risks and benefits of each option? What is your recommendation?

Table 10.48: Generated Questions for Additional Task-36 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	flight refund application
prompt	I have a lot of experience in getting refunds for flight delays. I will ask some questions to collect information and then I will contact the airline authority on your behalf requesting refund.
questions	What was the airline? What was the flight number? What was the date of the flight? What was the origin of the flight? What was the destination of the flight? What was the scheduled departure time of the flight? What was the actual departure time of the flight? What was the scheduled arrival time of the flight? What was the actual arrival time of the flight? How long was the flight delayed?

Table 10.49: Generated Questions for Additional Task-37 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	movie story customization
prompt	I am a famous storyteller. I specialize in editing popular movie stories and customizing them for users. I will ask some questions and edit storyline of the movie ’Titanic’ based on your answers.
questions	How would you like the story to end? What kind of ending would you like? What other changes would you like to see in the story? Would you like to see the two main characters end up together? Is there anything else you would like to change about the story?

Table 10.50: Generated Questions for Additional Task-38 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	music composition
prompt	I am a famous music composer. I will ask some questions to collect your music requirements and then I will compose a song specially for you.
questions	What is the occasion for which you need the song? What is the mood or feeling that you would like the song to convey? What specific elements would you like the song to include? Would you like the song to be in a particular style or genre? Is there anything else you would like me to know about your song request?

Table 10.51: Generated Questions for Additional Task-39 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	movie transcript generation
prompt	I am an expert in generating movie transcripts. I will ask some questions to collect your music requirements and then I will write transcript for you.
questions	What kind of movie are you looking for a transcript for? What is the name of the movie? What is the plot of the movie? What is the genre of the movie? What is the budget for the movie? What is the release date for the movie? What is the running time for the movie? Who are the cast members for the movie? Who is the director of the movie? Who is the producer of the movie? Who is the writer of the movie?

Table 10.52: Generated Questions for Additional Task-40 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	novel publication
prompt	I am a novel publisher. I will ask some questions to collect information about your novel and then tell if I can publish it.
questions	What is the title of your novel? What is the genre of your novel? What is the plot of your novel? What is the target audience for your novel? Is your novel complete? What is the word count of your novel? Do you have a marketing plan for your novel? Do you have any publishing experience? Do you have any literary agent representation? Do you have a platform? Do you have any writing awards? Do you have any previous publications? What are your writing credentials? What are your professional credentials? Do you have a personal website? Do you have a blog? Do you have an active social media presence? Do you have any other professional experience? Why do you think your novel would be a good fit for my publishing company? Do you have any questions for me?

Table 10.53: Generated Questions for Additional Task-41 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	food preparation
prompt	I am a famous chef. I will ask some questions to collect your requirements and then I will prepare a dish of your choice.
questions	What is your name? What is your favorite type of food? What are your dietary restrictions? What is your favorite type of cuisine? What are your favorite dishes? What ingredients do you not like? Is there anything else you would like me to know?

Table 10.54: Generated Questions for Additional Task-42 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	restaurant menu preparation
prompt	I am an expert in running restaurants. I will ask some questions to collect your requirements and then I will prepare a restaurant menu of your choice.
questions	What type of restaurant menu do you want? What type of food do you want to serve? What is your budget? What are your dietary restrictions? What is your target audience? What is your restaurant's theme? What is the occasion? What is the atmosphere of your restaurant? What is the location of your restaurant? What is the ambiance of your restaurant? What is the service style of your restaurant? What is the dress code of your restaurant? What is the price range of your restaurant? What is the cuisine of your restaurant? What is the name of your restaurant? What is the seating capacity of your restaurant? What is the contact information of your restaurant? What are the hours of operation of your restaurant? What are the days of operation of your restaurant? What is the speciality of your restaurant? What is the history of your restaurant? What are the awards of your restaurant? What is the website of your restaurant? What is the Facebook page of your restaurant? What is the Instagram of your restaurant? What is the Twitter of your restaurant?

Table 10.55: Generated Questions for Additional Task-43 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	diet plan
prompt	I am a famous dietitian. I will ask some questions to collect information and then I will suggest diet plan for you.
questions	What is your weight? What is your height? What is your activity level? What are your food preferences? What are your health concerns? What medications are you currently taking? Are you trying to lose, gain, or maintain your weight? How many meals per day do you typically eat? How many snacks per day do you typically eat?

Table 10.56: Generated Questions for Additional Task-44 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	physics problem solving
prompt	I am a famous physicist. I am here to help you solve the physics problem you have on mechanics. I will ask some questions in this process.
questions	What is the physics problem you have on mechanics?
	What are the objects involved in the problem?
	What are the forces acting on the objects?
	What is the goal of the problem?
	What are the constraints of the problem?
	What is the starting configuration of the problem?
	What is the ending configuration of the problem?
	What is the time frame of the problem?
	What are the units of measurement for the problem?
	Is there any friction in the problem?
	Are the objects in the problem moving in a straight line or in a circle?
	What is the mass of the objects in the problem?
	What is the velocity of the objects in the problem?
	What is the acceleration of the objects in the problem?
	What is the resulting force on the objects in the problem?
	What is the resulting torque on the objects in the problem?
	What is the resulting energy on the objects in the problem?
	What is the resulting momentum on the objects in the problem?
	What are the conditions for equilibrium in the problem?
	What is the period of the problem?
	What is the frequency of the problem?
	What is the wavelength of the problem?
	What is the wave function of the problem?
	What is the amplitude of the problem?
	What is the phase of the problem?
	What is the wave vector of the problem?
	What is the group velocity of the problem?
	What is the phase velocity of the problem?
	What is the energy of the problem?
	What is the momentum of the problem?
	What is the position of the objects in the problem?
	What is the displacement of the objects in the problem?
	What is the distance traveled by the objects in the problem?
	What is the time of the problem?
	What is the speed of the objects in the problem?
	What is the power of the problem?
	What is the kinetic energy of the objects in the problem?
	What is the potential energy of the objects in the problem?
	What is the total energy of the objects in the problem?
	What is the angular momentum of the objects in the problem?
	What is the angular velocity of the objects in the problem?
	What is the angular acceleration of the objects in the problem?
	What is the moment of inertia of the objects in the problem?
	What is the rotational kinetic energy of the objects in the problem?
	What is the gravitational potential energy of the objects in the problem?
	What is the elastic potential energy of the objects in the problem?
	What is the magnetic potential energy of the objects in the problem?
	What is the electric potential energy of the objects in the problem?
	What is the nuclear potential energy of the objects in the problem?
	What is the thermal energy of the objects in the problem?
	What is the chemical energy of the objects in the problem?

Table 10.57: Generated questions for additional task-45 along with the prompt fed to GPT3.

element	taskname/input prompt/model predictions
task	chemistry problem solving
prompt	I am a famous chemistry teacher. I am here to help you find the appropriate chemical equation to solve the chemistry problem you have on stoichiometry. I will now ask clarifying questions to better understand your problem.
questions	What is the problem you are trying to solve? What is the molarity of the acid? What is the volume of the acid? What is the molarity of the base? What is the volume of the base? What is the desired molarity of the solution? What is the desired volume of the solution?

Table 10.58: Generated Questions for Additional Task-46 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	painting house
prompt	I have a lot of experience in painting houses. I will ask questions to collect information and then I will make a plan for painting your house.
questions	What is the size of your house? What is the condition of your house? What is your budget for painting your house? What is your timeline for painting your house? What is your preferred color scheme for painting your house? What is your preferred type of paint for painting your house? Are you planning on painting your house yourself or hiring a professional? Do you have any other specific requests or instructions for painting your house?

Table 10.59: Generated Questions for Additional Task-47 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	car mechanic
prompt	I am a famous car mechanic. I will ask questions to collect information and then I will tell you what needs to be fixed in your car.
questions	What seems to be the problem with your car? When did you first notice the problem? Have you noticed the problem happening more frequently? Is the problem happening all the time or only intermittently? What are the circumstances under which the problem occurs? What have you done to try to fix the problem? What other information can you tell me about the problem?

Table 10.60: Generated Questions for Additional Task-48 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	assistance in filling bank deposit form
prompt	I have a lot of experience in banking and have helped people in filling forms. I will ask questions to collect information and then I will fill the bank deposit form for you.
questions	What is your name? What is the name of your bank? What is the account number? What is the amount of money you would like to deposit? Do you have a check or cash? What is the date? Is this a savings or checking account? Would you like a receipt?

Table 10.61: Generated Questions for Additional Task-49 along with the Prompt Fed to Gpt3.

10.11 Conclusion

We introduce Help me Think to help non-expert users prompt models for the relatively under-explored customized content generation tasks. We demonstrate the efficacy of Help me Think on 6 different tasks. Our results show that (1) questions generated by GPT3 are valid and relevant, and (2) task-specific outputs generated by GPT3 are valid, relevant, robust, coherent, and have significant knowledge absorption. We hope this will bring more attention to the development of methods for LLMs to handle tasks that are hard for non-expert users.

element	taskname/input prompt/model predictions
task	electricity bill payment
prompt	I have a lot of experience in paying electricity bills online. I will ask questions to collect information and then I will pay the electricity bill for you.
questions	What is the account number for the electricity bill? What is the amount of the electricity bill? What is the due date for the electricity bill? What is the website for the electricity company? What is your username for the electricity company website? What is your password for the electricity company website? What is your email address? What is your phone number? What is your full name? What is the name of the electricity company?

Table 10.62: Generated Questions for Additional Task-50 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	creating todo list
prompt	I have a lot of experience in generating ToDo list. I will ask questions to collect information and then I will create a todo list for you.
questions	What is the goal of the todo list? What are the main tasks that need to be completed in order to achieve the goal? What are the deadlines for each task? Who is responsible for each task? What is the priority level for each task? Is there anything else that needs to be taken into account when creating the todo list? Are there any dependencies between tasks? What is the expected outcome of the todo list? What are the risks and potential roadblocks associated with the todo list? What are the possible ways to overcome the risks and roadblocks? Are there any other considerations that need to be taken into account?

Table 10.63: Generated Questions for Additional Task-51 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	creating post travel report
prompt	I have a lot of experience in generating post travel report. I will ask questions to collect information and then I will prepare your post travel report.
questions	How did you find the overall experience? What were the good points of the trip? What were the bad points of the trip? How much did you spend in total? Would you recommend this trip to others? Why or why not? Is there anything you would have done differently? What was your favorite part of the trip? What was your least favorite part of the trip? How was the accommodation? How was the food? How were the locals? Did you feel safe during the trip? Would you like to visit this destination again? Why or why not? Is there anything else you would like to add?

Table 10.64: Generated Questions for Additional Task-52 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	creating child monitoring report
prompt	I have a lot of experience in monitoring child. I will ask questions to collect information and then I will prepare your child-monitoring report.
questions	How often do you check in on your child when they are home alone? How do you feel when you are away from your child and they are home alone? What worries you the most about your child being home alone? What have you done to ensure your child knows what to do if there is an emergency while you are away? What are your child's favorite activities to do when they are home alone? Has your child ever had any problems while home alone? If so, what happened and how did they handle it? Do you have any other concerns or questions about your child being home alone?

Table 10.65: Generated Questions for Additional Task-53 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	treating mental illness
prompt	I have a lot of experience in handling mental problem. I will ask questions to collect information and then I will suggest a treatment for you.
questions	What is your name? What is your age? What is your occupation? What are your hobbies? Do you have any medical conditions? Do you have any mental health conditions? Do you have any history of mental illness in your family? What is your current mental state? What are your current stressors? What are your current coping mechanisms? What are your current support systems? What are your current treatment options? What are your goals for treatment? What are your thoughts on medication? What are your thoughts on therapy? What are your thoughts on self-care? What are your thoughts on support groups? What are your thoughts on lifestyle changes? What are your thoughts on relapse prevention? What are your thoughts on long-term treatment? What are your thoughts on medication compliance? What are your thoughts on therapy attendance? What are your thoughts on self-care consistency? What are your thoughts on support group participation?

Table 10.66: Generated Questions for Additional Task-54 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	career counselling
prompt	I am an expert in career counselling. I will ask questions to collect information and then I will prepare a suitable career plan for you.
questions	What are your long-term career goals? What are your educational qualifications? What is your work experience? What are your skills and strengths? What are your interests? What are your values? What are your goals for the next five years? What are your plans for further education or training? What are your plans for your personal life? What are your plans for your financial future? What are your plans for your retirement? What other factors are important to you in choosing a career? What are your thoughts on changing careers? What are your thoughts on starting your own business? What are your thoughts on working for someone else? What are your thoughts on working abroad? What are your thoughts on taking a break from your career? What other questions do you have about your career? Are you ready to start working on your career plan? Do you have any concerns or questions that you would like to discuss before we start? Are you sure that you are ready to commit to this career plan? Would you like me to review your career plan with you on a regular basis? Is there anything else that you would like to discuss about your career plan?

Table 10.67: Generated Questions for Additional Task-55 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	child tutoring
prompt	I am an expert in child tutoring I will ask questions to collect information and then I will decide if I can tutor your child.
questions	What is your child’s name? What is your child’s grade level? What are the subjects your child needs help with? How often would you like tutoring sessions to be? What days and times are best for you? What is your child’s learning style? Does your child have any special needs? Is your child behind, at grade level, or ahead in school? How do you feel about your child’s progress in school? What are your goals for your child’s education? What are your child’s strengths and weaknesses? Is there anything else you feel I should know about your child? Would you like me to assess your child before we begin tutoring? Do you have any questions for me? When would you like to start?

Table 10.68: Generated Questions for Additional Task-56 along with the Prompt Fed to Gpt3.

element	taskname/input prompt/model predictions
task	paper writing
prompt	I am a famous researcher with a lot of experience in writing papers. I will ask questions to collect information about your research and then I will suggest a plan to write your paper.
questions	What is the topic of your research? How much experience do you have with writing papers? What is your writing style? What is the format of your paper? What is the deadline for your paper? What is the word count for your paper? What is the purpose of your paper? Who is your audience? What are your expectations for this project? What is your budget for this project? What is the timeline for this project? What are the deliverables for this project? What are your research methods? What sources will you be using for your research? What are your thoughts on the research process? What are your thoughts on writing papers?

Table 10.69: Generated Questions for Additional Task-57 along with the Prompt Fed to Gpt3.

Chapter 11

DISCUSSION

In this section, we outline some of the discussion points that are beyond the scope of this thesis.

The observations made in §3.4.3 indicate that there are likely benefits to repeating our study with a larger set of datasets. We hope the future work expands our work with a larger and broader range of tasks.

We use automatic evaluation, in order to facilitate the replicability of the follow-up work on Natural Instructions. Admitting the limitations of automatic evaluations, we hope future work will provide an easy-to-reproduce human evaluation for the tasks studied here, based on the recent proposals for streamlining human evaluation of text generation models (Khashabi *et al.*, 2021).

Towards a sustainable alternative to model training In the context of fast prototyping using massive LMs, we hypothesize that further development of reframing can lead to pragmatic alternative to fine-tuning. In applications where task definitions can quickly change, model designers can come up with new reframed prompts, in a matter of minutes.

Generalization to future models Would the reframed prompts remain competitive on future LMs? While it is impossible to decidedly respond to this counterfactual question, extrapolating from Fig.5.2 is a likely evidence that the proposed approach will remain superior, at least in the near term. However, on a longer horizon, the gains depend on the progress of LMs. If models have little difficulty in understanding language, there will be little gain in reframing the instructions.

Opportunities for improvement While reframing enables model development in a human-centric manner, it needs to be applied by model designers. Therefore, an algorithmic solution to reframe tasks will likely be useful progress forward. This will be part of our future work.

Some knowledge of the domain is required to be able to design these reframed prompts. Nonetheless, we believe that making this observation and operationalizing it is really important and impactful to the NLP community to fully take advantage of the pre-trained language models, especially in real-world applications.

We hope that this study will inspire further investigation of potentially-unconventional approaches to exploit the knowledge harnessed by increasingly large LMs where fine-tuning and its alternatives are prohibitively expensive.

Instruction Programming Future work can extend our reframing work and develop a programming paradigm for instructional prompts. This paradigm will be based on finding some basic instructions that can act as building blocks over which tasks corresponding to complex instructions can be expressed. In this way, complex tasks can be made easier for models to solve.

The outline of a programming paradigm for instructional prompts is as follows:

1. Find the strengths of models, i.e., list down the set of tasks/subtasks on which models excel: This is analogous to variable declaration, variable assignment, if-else conditions, and other operations that are well defined in python.

2. Find connectives to connect strengths of models: This is analogous to newline in most programming languages, for/while loop etc.

3. Find hard problems to evaluate: This is analogous to real-world problems that are solved by writing programs (e.g. in python), after composing (1) and (2) above.

A gauge of model strength, as mentioned in (1), can be thought of as the sets of basic instructions that define various tasks that the model can reliably solve. (2) can similarly be expressed as building on those basic instructions, to create complex instructions that can be used to solve a task with high accuracy.

Here are some more details for each of the steps from the above outline.

1. Inspired by Prolog, where a model's strength is answering simple factual questions with respect to a knowledge base of facts and rules, we find that fill in the blank questions represent the primary strength of neural language models. This is because such models have been pre-trained with language modeling objectives, and the fill in the blank format resembles these pretraining objective functions. We have further performed probing experiments, and find that questions with True/False output comprise another strength of neural language models. We also observe empirically, that GPT3 achieves high accuracy in a zeroshot setting for the following tasks:

   Sentiment Analysis

   Instruction: Given a tweet, classify it into one of 4 categories: Positive, Negative, Neutral, or Mixed. Tweet: "I thought the new Spiderman game was good, but I really didn't like the animation." Sentiment:

   Extractive QA from Context (Context is provided)

   Instruction: Given a context and a question, generate an answer to the given question from the provided context. Context: Delhi is the capital of India. Delhi has an airport called Indira Gandhi International Airport. Delhi gets extremely hot in summer and extremely cold in winter. Question: What is the capital of India?

   Story Writing/Review Writing

Instruction: Explain the moon landing to a 6 year old in a few sentences. Instruction: Write a product review about a wireless keyboard.

2. Compose by expressing a new task in terms of strengths of the model as listed in (1). Here are some connectives we have empirically found to be helpful.
   Use bullet points (itemized reframing (Mishra *et al.*, 2022e))
   Explain difficult concepts when needed (Depending on the engine some explanations may not be needed - do trial error on the dev set)
   Provide additional knowledge if you have any.
   Connect decomposed tasks in series by using output of previous subtask as the input to the next or in parallel by concatenating all task outputs.

3. Propara (Dalvi *et al.*, 2019), ARC (Bhakthavatsalam *et al.*, 2021), and MATH (Hendrycks *et al.*, 2021b) are some of the hard datasets.

   Following is a potential hierarchy of tasks in an increasing order of complexity; these can be leveraged to systematically evaluate the instruction programming paradigm.
   (1) Direct extraction from text (this often can be syntactic tasks)
   (2) Extraction using background knowledge.
   (3) Classification with fixed choices (make sure to specify choices explicitly in the question and also mention the expected output format)
   (4) Question answering (this can be broken down into a fixed number of steps)
   (5) Generation Tasks
   (6) Creative Writing tasks e.g. story, poem writing
   (7) Unconventional tasks e.g. incorrect answer generation

Other Applications: There are several aspects of instruction learning which are of importance for future work.

We demonstrate that scaling up the number of instruction tasks (Wang *et al.*, 2022d) significantly improve model performance. The performance benefits also can be achieved by synthetic creation of instruction tasks (Wang *et al.*, 2022c). Examples in instructions can potentially possess a form of bias (instruction bias) that overestimates model performance (Parmar *et al.*, 2022a). This needs to be carefully managed while learning from instructions. Incorporating output style instructions e.g. chain of thought (Wei *et al.*, 2022b) in instruction learning is seen (Lu *et al.*, 2022a) to significantly improve model performance.

Instruction learning also help improve model performance in biomedical domain (Parmar *et al.*, 2022b; Luo *et al.*, 2022). Instruction learning is also seen to be helpful in other tasks such as aspect based sentiment analysis (Scaria *et al.*, 2023) and contextual NER (Gupta *et al.*, 2021a). Real world tools such as InstructExcel (Mishra *et al.*, 2023), which is intended to allow users to use excel just by providing instructions are useful applications of the instruction learning framework.

Instruction-data interplay is another important aspect to investigate further in future work. There is evidence (Gupta *et al.*, 2023) that instruction learning helps in learning a task quickly from less data. An interesting avenue to study is to see if instructions also help in learning efficiently with training data (Mishra and Sachdeva, 2020; Arunkumar *et al.*, 2023)and test data (Varshney *et al.*, 2022a). Can learning from instructions help improve generalization (Mishra *et al.*, 2020a,b), robustness (Gokhale *et al.*, 2022; Mishra *et al.*, 2022h) and reliability via selective answering (Varshney *et al.*, 2022b,d, 2021; Mishra *et al.*, 2022a) over learning from examples?

Another avenue for future research is to create data automatically using instructions. It is worth investigating if the data created with instructions are of higher quality (Mishra and Arunkumar, 2022; Mishra *et al.*, 2022b) than the ones created using crowdsourcing. Furthermore, more analysis is needed to analyze if instructions can help perform complex reasoning tasks such as numerical reasoning (Gupta *et al.*, 2022a), commonsense reasoning (Banerjee *et al.*, 2021) more efficiently.

Role of instruction learning with other learning paradigms such as curriculum learning (Varshney *et al.*, 2022c), learning algorithms (Mishra and Arunkumar, 2021) can potentially be interesting. Expanding instruction learning to solve hard tasks (Srivastava *et al.*, 2022) across complex domains such as security (Pal *et al.*, 2021), cyber-physical systems (Mishra *et al.*, 2019; Korukonda *et al.*, 2018, 2017, 2016; Mishra *et al.*, 2015) and disaster management (Mishra *et al.*, 2022d) will be part of future work.

Chapter 12

CONCLUSION & FUTURE WORK

In chapter 3, we studied the goal of building models that generalize to new tasks by encoding and understanding crowdsourcing instructions. We introduced Natural Instructions, which is built based on existing crowdsourced datasets, that enables building such models and systematically evaluating them. To the best of our knowledge, this is the first work to show the benefit of instructions towards improved cross-task generalization. Additionally, we observe that our proposed task has a large room for improvement, which we believe will bring more attention to building stronger models that can generalize to a wider range of tasks.

In chapter 4, we propose NumGLUE, a multi-task benchmark to test for arithmetic understanding. Our benchmark consists of eight tasks including four new ones. While some of the tasks require external knowledge like commonsense or domain-specific information in addition to arithmetic reasoning, some are self-contained e.g. arithmetic word problems. Further, we demonstrate that our benchmark is far from being solved – with state-of-the-art large scale models achieving considerably lower performance than humans. This indicates that current AI systems are incapable of performing simple arithmetic reasoning in a general setting – indicating a fundamental hurdle towards AI systems that understand complex mathematical concepts like differential equations or combinatorics. Finally, we present various baselines including a novel architecture (memory augmented Ex-NumNet) that demonstrates the advantages of various modeling choices (e.g. end-to-end vs neuro-symbolic models). Specifically, we show that training in the multi-task setting leads to meaningful sharing of knowledge across tasks as evidenced by an average gain of 3.4% on tasks compared to task-specific modeling. We also see that instruction-based training improves performance by 2.7% over the multi-task learning baseline. Finally, we hope that our benchmark not only leads to AI systems that are capable of performing simple arithmetic reasoning in a fairly general setting but also results in progress towards more complex mathematical reasoning capability.

In chapter 5, Inspired by GPT3's poor performance in following task instructions, we study *reframing* them. We introduce five approaches that reformulate task instructions to make them easier, while maintaining their human readability. Manually applying reframing on 12 tasks, we study their benefits compared to using raw instructions or fine-tuning mid-sized models. Reframing can be particularly helpful in applications where task definitions are evolving (making it difficult to crowdsource and fine-tune models), where model designers can come up with new reframed prompts, in a matter of minutes.

In chapter 6, we introduce Līla, a unified mathematical reasoning benchmark for a holistic evaluation of AI agents. Līla consists of 23 tasks across 4 dimensions (i) mathematical abilities, (ii) language format, (iii) language complexity, and (iv) external knowledge. It builds on 20 existing mathematical reasoning datasets to collect instructions and Python programs. Further, it also supports measuring out-of-distribution performance and robustness to language perturbations via Līla-OOD and Līla-Robust respectively. We also introduce Bhāskara, a 2.7B-parameter fine-tuned multi-task model. We find that multi-tasking improves over single-task performance by 21.83% F1 score on average, and that our model is a strong starting point for further fine-tuning on new math reasoning tasks. The best performing model we evaluate achieves only 60.40% F1 indicating the potential for improvement on the proposed benchmark.

Limitations : One drawback of our unified format is the difficulty of evaluating models. In our work we use F1 for lack of a better alternative. F1 likely over-estimates performance, e.g., given the gold answer "2 apples", the predicted answers "2" and "apples" receive the same score, though the former is better.

Līla contains 23 tasks which are created from 20 datasets and 44 sub-datasets. There is scope to add more mathematical reasoning datasets (theorem proving.) The flexible unified format of Līla allows for future extensions. Additionally, our categorization provides a way to identify areas for extension. For instance, we only have 1 dataset for linear algebra, which happens to not use natural language, and takes the form of generative QA. Our benchmark will benefit from future linear algebra additions, perhaps with word problems formatted as fill-in-the-blank questions.

Chapter 7 introduces a summarization-based approach for efficient program synthesis. Experimental results show that the proposed approach improves the performance of the Codex model by on average $\sim 8\%$ across various levels of programming questions provided by the APPS and $\sim 11\%$ on the CodeContests. Further, our work proposes a meta-dataset consisting of ~ 450 human-generated basic and expert-level summaries as well as $\sim 8k$ synthetically generated summaries by GPT-3 and Studio21; this can be helpful for future research on writing better instructions for the program synthesis. We show that program synthesis models benefit from concise prompts, hence, we believe that less number of high-quality instances are better than more low-quality data instances.

Future Extensions The decomposition of prompts has been shown to improve accuracy Mishra *et al.* (2022e); Patel *et al.* (2022); splitting up the summarization task the resulting summary can potentially result in higher accuracy for the Codex model in future. Additionally, the PEGASUS model could be used in conjunction with other models to improve performance.

Limitations Our summary-based approach shows improved performance on program synthesis models, however, it shows competitive performance on synthetic summaries. We believe that the generation of high-quality summaries can improve performance, hence, designing efficient prompts to improve synthetic summaries can be the scope of further research. Furthermore, human-generated summaries show competitive performance on competition-level problems. These problems require reasoning with multiple logical leaps and knowledge of advanced algorithms and data structures. Hence, exploring new techniques for summarization can be a future research direction. In addition, this work only analyzes the codex model, hence, exploring the effect of summarization on other program synthesis models can be interesting.

In chapter 8, we argue that the recent trend of building large LMs may not be sustainable to solve evolving benchmarks. We believe that modifying data samples can significantly help the model improve performance. We study the effect of Question Decomposition (QD) on a diverse set of tasks. We decompose questions manually and significantly improve model performance (24% for GPT3 and 29% for RoBERTa-SQuAD along with a symbolic calculator). Our findings indicate that Human-in-the-loop Question Decomposition (HQD) can potentially provide an alternate path to building large LMs. Our approach provides a viable option to involve people in NLP research. We hope our work will encourage the community to develop human-centric solutions that actively involve humans while leveraging NLP resources.

Limitations Our human-in-the-loop methodology shows promising results by decomposing questions, however, certain questions are still difficult to decompose for humans as well. For instance, the question *"Which country is New York in?"*, is hard to decompose further. Determining which questions to decompose is also an important challenge and under-explored in this work. Furthermore, decomposed questions in the chain which have more than one correct answer might lead to an incorrect final answer. Automating the process of decomposition while addressing these issues is a promising area for future work.

In chapter 9, we introduced instruction augmentation to improve existing LMs in terms of improving performance and usability for non-expert users. To this extent, we created multi-variant instructions for 426 NLP tasks. Our experiment results show that instruction augmentation improves model performance in task-specific, multi-task and cross-task learning paradigms. We find that instruction augmentation is more effective in low-data regime. Our results further indicate that an additional instruction can be equivalent to $\sim$200 instances on an average. We hope our work will bring more attention to developing unconventional techniques (beyond dataset creation and model training) to empower non-expert users to leverage NLP resources and teach a task without having domain knowledge.

Limitations We use BART-base and T5-base for all our experiments, however, we wish to experiment with different language models in future to show the benefit of our approach. Our analysis includes only tasks in English language, hence, it is important to see that if our approach can be extended to non-English tasks as well. We feel that developing diverse instruction augmentation techniques will be pivotal to achieve more improvements as future research.

In chapter 10, We introduce Help me Think to help non-expert users prompt models for the relatively under-explored customized content generation tasks. We demonstrate the efficacy of Help me Think on 6 different tasks. Our results show that (1) questions generated by GPT3 are valid and relevant, and (2) task-specific outputs generated by GPT3 are valid, relevant, robust, coherent, and have significant knowledge absorption. We hope this will bring more attention to the development of unconventional applications of LLMs in helping humans perform the tasks that are hard for non-expert users, in contrast to conventional tasks like question-answering where models chase human baseline.

REFERENCES

Adda, G., B. Sagot, K. Fort and J. Mariani, "Crowdsourcing for language resource development: Critical analysis of amazon mechanical turk overpowering use", in "5th Language and Technology Conference", (2011).

Aghajanyan, A., A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer and S. Gupta, "Muppet: Massive multi-task representations with pre-finetuning", in "Proceedings of EMNLP", pp. 5799–5811 (2021).

Ahn, M., A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances", arXiv preprint arXiv:2204.01691 (2022).

Ali, A. M., "The use of positive and negative examples during instruction", Journal of instructional development **5**, 1, 2–7 (1981).

Amari, S. *et al.*, *The handbook of brain theory and neural networks* (MIT press, 2003).

Amini, A., S. Gabriel, P. Lin, R. Koncel-Kedziorski, Y. Choi and H. Hajishirzi, "Mathqa: Towards interpretable math word problem solving with operation-based formalisms", arXiv preprint arXiv:1905.13319 (2019).

ANDERSON, D. I. and R. A. MAGILL, *Motor learning and control: concepts and applications* (McGraw-Hill, 2021).

Arunkumar, A., S. Mishra, B. Sachdeva, C. Baral and C. Bryan, "Real-time visual feedback to guide benchmark creation: A human-and-metric-in-the-loop workflow", arXiv preprint arXiv:2302.04434 (2023).

Austin, J., A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai, M. Terry, Q. Le *et al.*, "Program synthesis with large language models", arXiv preprint arXiv:2108.07732 (2021).

Balog, M., A. L. Gaunt, M. Brockschmidt, S. Nowozin and D. Tarlow, "Deepcoder: Learning to write programs", arXiv preprint arXiv:1611.01989 (2016).

Banerjee, P., S. Mishra, K. K. Pal, A. Mitra and C. Baral, "Commonsense reasoning with implicit knowledge in natural language", in "3rd Conference on Automated Knowledge Base Construction", (2021).

Beltagy, I., M. E. Peters and A. Cohan, "Longformer: The long-document transformer", arXiv preprint arXiv:2004.05150 (2020).

Bhakthavatsalam, S., D. Khashabi, T. Khot, B. D. Mishra, K. Richardson, A. Sabharwal, C. Schoenick, O. Tafjord and P. Clark, "Think you have solved direct-answer question answering? try arc-da, the direct-answer ai2 reasoning challenge", arXiv preprint arXiv:2102.03315 (2021).

Black, S., L. Gao, P. Wang, C. Leahy and S. Biderman, “Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow”, If you use this software, please cite it using these metadata **58** (2021).

Bras, R. L., S. Swayamdipta, C. Bhagavatula, R. Zellers, M. E. Peters, A. Sabharwal and Y. Choi, “Adversarial filters of dataset biases”, arXiv preprint arXiv:2002.04108 (2020).

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, “Language models are few-shot learners”, in “NeurIPS”, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan and H. Lin (2020).

Caruana, R., “Multitask learning”, Machine learning **28**, 1, 41–75 (1997).

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. Ponde, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, “Evaluating large language models trained on code”, arXiv preprint arXiv:2107.03374 (2021a).

Chen, X., X. Xie, N. Zhang, J. Yan, S. Deng, C. Tan, F. Huang, L. Si and H. Chen, “Adaprompt: Adaptive prompt-based finetuning for relation extraction”, arXiv e-prints pp. arXiv–2104 (2021b).

Chowdhery, A., S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, “Palm: Scaling language modeling with pathways”, arXiv preprint arXiv:2204.02311 (2022).

Clark, C., K. Lee, M.-W. Chang, T. Kwiatkowski, M. Collins and K. Toutanova, “Boolq: Exploring the surprising difficulty of natural yes/no questions”, arXiv preprint arXiv:1905.10044 (2019).

Clark, P., I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick and O. Tafjord, “Think you have solved question answering? try arc, the ai2 reasoning challenge”, arXiv preprint arXiv:1803.05457 (2018).

Clark, P., O. Tafjord and K. Richardson, “Transformers as soft reasoners over language”, in “Proceedings of IJCAI”, (2020).

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

Colebrooke, H. T., “Arithmetic and mensuration of brahmegupta and bhaskara”, (1817).

Dalvi, B., N. Tandon, A. Bosselut, W. tau Yih and P. Clark, “Everything happens for a reason: Discovering the purpose of actions in procedural text”, ArXiv **abs/1909.04745** (2019).

Dasigi, P., N. F. Liu, A. Marasovic, N. A. Smith and M. Gardner, “Quoref: A reading comprehension dataset with questions requiring coreferential reasoning”, in “Proceedings of EMNLP-IJCNLP”, pp. 5927–5934 (2019).

Devlin, J., J. Uesato, S. Bhupatiraju, R. Singh, A.-r. Mohamed and P. Kohli, "Robustfill: Neural program learning under noisy i/o", in "International conference on machine learning", pp. 990–998 (PMLR, 2017).

Dhole, K. D., V. Gangal, S. Gehrmann, A. Gupta, Z. Li, S. Mahamood, A. Mahendiran, S. Mille, A. Srivastava, S. Tan *et al.*, "Nl-augmenter: A framework for task-sensitive natural language augmentation", arXiv preprint arXiv:2112.02721 (2021).

Dua, D., A. Gottumukkala, A. Talmor, S. Singh and M. Gardner, "Orb: An open reading benchmark for comprehensive evaluation of machine reading comprehension", arXiv preprint arXiv:1912.12598 (2019a).

Dua, D., Y. Wang, P. Dasigi, G. Stanovsky, S. Singh and M. Gardner, "Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs", in "Proceedings of NAACL", pp. 2368–2378 (2019b).

Efrat, A. and O. Levy, "The turking test: Can language models understand instructions?", arXiv preprint arXiv:2010.11982 (2020).

Feng, S. Y., V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura and E. Hovy, "A survey of data augmentation approaches for nlp", arXiv preprint arXiv:2105.03075 (2021).

Fort, K., G. Adda and K. B. Cohen, "Amazon mechanical turk: Gold mine or coal mine?", Computational Linguistics pp. 413–420 (2011).

Gafni, O., A. Polyak, O. Ashual, S. Sheynin, D. Parikh and Y. Taigman, "Make-a-scene: Scene-based text-to-image generation with human priors", arXiv preprint arXiv:2203.13131 (2022).

Gao, L., S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He, A. Thite, N. Nabeshima, S. Presser and C. Leahy, "The Pile: An 800gb dataset of diverse text for language modeling", arXiv preprint arXiv:2101.00027 (2020).

Ge, R. and R. Mooney, "A statistical semantic parser that integrates syntax and semantics", in "Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)", pp. 9–16 (2005).

Geva, M., Y. Goldberg and J. Berant, "Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets", 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)", pp. 1161–1166 (Association for Computational Linguistics, Hong Kong, China, 2019), URL https://aclanthology.org/D19-1107.

Geva, M., D. Khashabi, E. Segal, T. Khot, D. Roth and J. Berant, "Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies", Transactions of the Association for Computational Linguistics **9**, 346–361 (2021).

Gokhale, T., S. Mishra, M. Luo, B. Sachdeva and C. Baral, “Generalized but not robust? comparing the effects of data modification methods on out-of-domain generalization and adversarial robustness”, in “Findings of the Association for Computational Linguistics: ACL 2022”, pp. 2705–2718 (2022).

Gulwani, S., O. Polozov, R. Singh *et al.*, “Program synthesis”, Foundations and Trends® in Programming Languages **4**, 1-2, 1–119 (2017).

Guo, J., Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, T. Liu and D. Zhang, “Towards complex text-to-sql in cross-domain database with intermediate representation”, arXiv preprint arXiv:1905.08205 (2019).

Gupta, H., S. A. Sawant, S. Mishra, S. M. Mashetty, A. Mitra, M. Nakamura and C. Baral, “Instruction-tuned models are quick learners”, arXiv preprint (2023).

Gupta, H., N. Varshney, S. Mishra, K. K. Pal, S. A. Sawant, K. Scaria, S. Goyal and C. Baral, “" john is 50 years old, can his son be 65?" evaluating nlp models’ understanding of feasibility”, arXiv preprint arXiv:2210.07471 (2022a).

Gupta, H., S. Verma, T. Kumar, S. Mishra, T. Agrawal, A. Badugu and H. S. Bhatt, “Context-ner: Contextual phrase generation at scale”, arXiv preprint arXiv:2109.08079 (2021a).

Gupta, N. and M. Lewis, “Neural compositional denotational semantics for question answering”, arXiv preprint arXiv:1808.09942 (2018).

Gupta, P., C. Jiao, Y.-T. Yeh, S. Mehri, M. Eskenazi and J. P. Bigham, “Improving zero and few-shot generalization in dialogue through instruction tuning”, arXiv preprint arXiv:2205.12673 (2022b).

Gupta, T., A. Kamath, A. Kembhavi and D. Hoiem, “Towards general purpose vision systems”, ArXiv **abs/2104.00743** (2021b).

Gururangan, S., S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman and N. A. Smith, “Annotation artifacts in natural language inference data”, arXiv preprint arXiv:1803.02324 (2018).

Han, J. M., J. M. Rute, Y. Wu, E. W. Ayers and S. Polu, “Proof artifact co-training for theorem proving with language models”, ArXiv **abs/2102.06203** (2021).

Hase, P. and M. Bansal, “When can models learn from explanations? a formal framework for understanding the roles of explanation data”, arXiv preprint arXiv:2102.02201 (2021).

Hendrycks, D., S. Basart, S. Kadavath, M. Mazeika, A. Arora, E. Guo, C. Burns, S. Puranik, H. He, D. Song *et al.*, “Measuring coding challenge competence with apps”, arXiv preprint arXiv:2105.09938 (2021a).

Hendrycks, D., C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song and J. Steinhardt, “Measuring massive multitask language understanding”, in “International Conference on Learning Representations”, (2020a).

Hendrycks, D., C. Burns, S. Kadavath, A. Arora, S. Basart, E. Tang, D. Song and J. Steinhardt, “Measuring mathematical problem solving with the math dataset”, arXiv preprint arXiv:2103.03874 (2021b).

Hendrycks, D., X. Liu, E. Wallace, A. Dziedzic, R. Krishnan and D. Song, “Pretrained transformers improve out-of-distribution robustness”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 2744–2751 (2020b).

Hosseini, M. J., H. Hajishirzi, O. Etzioni and N. Kushman, “Learning to solve arithmetic word problems with verb categorization”, in “In Conference on Empirical Methods in Natural Language Processing (EMNLP”, (2014).

Huang, D., S. Shi, C.-Y. Lin, J. Yin and W.-Y. Ma, “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)”, pp. 887–896 (2016).

Huang, L., R. Le Bras, C. Bhagavatula and Y. Choi, “Cosmos qa: Machine reading comprehension with contextual commonsense reasoning”, in “Proceedings of EMNLP-IJCNLP”, pp. 2391–2401 (2019).

Iyyer, M., W.-t. Yih and M.-W. Chang, “Search-based neural structured learning for sequential question answering”, in “Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, pp. 1821–1831 (Association for Computational Linguistics, Vancouver, Canada, 2017), URL https://aclanthology.org/P17-1167.

Jia, R., A. Raghunathan, K. Göksel and P. Liang, “Certified robustness to adversarial word substitutions”, 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)”, pp. 4129–4142 (2019).

Jindal, P. and D. Roth, “Learning from negative examples in set-expansion”, in “2011 IEEE 11th International Conference on Data Mining”, pp. 1110–1115 (IEEE, 2011).

Khashabi, D., S. Chaturvedi, M. Roth, S. Upadhyay and D. Roth, “Looking beyond the surface: A challenge set for reading comprehension over multiple sentences”, in “Proceedings of NAACL”, pp. 252–262 (2018).

Khashabi, D., T. Khot, A. Sabharwal and D. Roth, “Learning what is essential in questions”, in “Proceedings of CoNLL”, pp. 80–89 (2017).

Khashabi, D., S. Min, T. Khot, A. Sabharwal, O. Tafjord, P. Clark and H. Hajishirzi, “UnifiedQA: crossing format boundaries with a single qa system”, in “Proceedings of EMNLP: Findings”, pp. 1896–1907 (2020).

Khashabi, D., G. Stanovsky, J. Bragg, N. Lourie, J. Kasai, Y. Choi, N. A. Smith and D. S. Weld, “GENIE: A leaderboard for human-in-the-loop evaluation of text generation”, arXiv preprint arXiv:2101.06561 (2021).

Khot, T., P. Clark, M. Guerquin, P. Jansen and A. Sabharwal, "Qasc: A dataset for question answering via sentence composition", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 34, pp. 8082–8090 (2020).

Khot, T., D. Khashabi, K. Richardson, P. Clark and A. Sabharwal, "Text modular networks: Learning to decompose tasks in the language of existing models", in "Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies", pp. 1264–1279 (Association for Computational Linguistics, Online, 2021), URL https://aclanthology.org/2021.naacl-main.99.

Khot, T., A. Sabharwal and P. Clark, "What's missing: A knowledge gap guided approach for multi-hop question answering", arXiv preprint arXiv:1909.09253 (2019).

Kim, H., B.-H. So, W.-S. Han and H. Lee, "Natural language to sql: Where are we today?", Proceedings of the VLDB Endowment **13**, 10, 1737–1750 (2020).

Kolachana, A., K. Mahesh and K. Ramasubramanian, "Use of calculus in hindu mathematics", in "Studies in Indian Mathematics and Astronomy", pp. 345–355 (Springer, 2019).

Koncel-Kedziorski, R., H. Hajishirzi, A. Sabharwal, O. Etzioni and S. D. Ang, "Parsing algebraic word problems into equations", Transactions of the Association for Computational Linguistics **3**, 585–597 (2015).

Koncel-Kedziorski, R., S. Roy, A. Amini, N. Kushman and H. Hajishirzi, "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", pp. 1152–1157 (2016).

Korukonda, M. P., S. R. Mishra, K. Rajawat and L. Behera, "Hybrid adaptive framework for coordinated control of distributed generators in cyber-physical energy systems", IET Cyber-Physical Systems: Theory & Applications **3**, 1, 54–62 (2018).

Korukonda, M. P., S. R. Mishra, A. Shukla and L. Behera, "Improving microgrid voltage stability through cyber-physical control", in "2016 National Power Systems Conference (NPSC)", pp. 1–6 (IEEE, 2016).

Korukonda, M. P., S. R. Mishra, A. Shukla and L. Behera, "Handling multi-parametric variations in distributed control of cyber-physical energy systems through optimal communication design", IET Cyber-Physical Systems: Theory & Applications **2**, 2, 90–100 (2017).

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

Kuznia, K., S. Mishra, M. Parmar and C. Baral, "Less is more: Summary of long instructions is better for program synthesis", in "Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing", pp. 4532–4552 (Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022), URL https://aclanthology.org/2022.emnlp-main.301.

Lake, B. and M. Baroni, "Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks", in "International Conference on Machine Learning", pp. 2873–2882 (PMLR, 2018).

Le Scao, T. and A. M. Rush, "How many data points is a prompt worth?", in "Proceedings of NAACL-HLT", pp. 2627–2636 (2021).

Lester, B., R. Al-Rfou and N. Constant, "The power of scale for parameter-efficient prompt tuning", in "Proceedings of EMNLP", (2021).

Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension", in "Proceedings of ACL", (2019).

Li, W., L. Yu, Y. Wu and L. C. Paulson, "Isarstep: a benchmark for high-level mathematical reasoning", in "International Conference on Learning Representations", (2021), URL https://openreview.net/forum?id=Pzj6fzU6wkj.

Li, Y., D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago *et al.*, "Competition-level code generation with alphacode", arXiv preprint arXiv:2203.07814 (2022).

Lieber, O., O. Sharir, B. Lenz and Y. Shoham, "Jurassic-1: Technical details and evaluation", White Paper. AI21 Labs (2021).

Lin, B. Y., S. Lee, R. Khanna and X. Ren, "Birds have four legs?! numersense: Probing numerical commonsense knowledge of pre-trained language models", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)", pp. 6862–6868 (2020).

Lin, C.-Y., "Rouge: A package for automatic evaluation of summaries", in "Text summarization branches out", pp. 74–81 (2004).

Lin, K., O. Tafjord, P. Clark and M. Gardner, "Reasoning over paragraph effects in situations", in "Proceedings of the 2nd Workshop on Machine Reading for Question Answering", pp. 58–62 (2019).

Lin, W., R. Yangarber and R. Grishman, "Bootstrapped learning of semantic classes from positive and negative examples", in "Proceedings of ICML Workshop on The Continuum from Labeled to Unlabeled Data", vol. 1, p. 21 (2003).

Lin, X. V., C. Wang, L. Zettlemoyer and M. D. Ernst, "Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system", in "Proceedings of LREC", (2018).

Ling, W., D. Yogatama, C. Dyer and P. Blunsom, "Program induction by rationale generation: Learning to solve and explain algebraic word problems", arXiv preprint arXiv:1705.04146 (2017).

Liu, J., D. Shen, Y. Zhang, B. Dolan, L. Carin and W. Chen, "What makes good in-context examples for gpt-3?", arXiv preprint arXiv:2101.06804 (2021a).

Liu, P., W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing”, arXiv preprint arXiv:2107.13586 (2021b).

Liu, X., Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang and J. Tang, “Gpt understands, too”, arXiv preprint arXiv:2103.10385 (2021c).

Liu, Y., M. Ott, N. Goyal, J. Du, M. S. Joshi, D. Chen, O. Levy, M. Lewis, L. S. Zettlemoyer and V. Stoyanov, “RoBERTa: A robustly optimized bert pretraining approach”, arXiv URL http://arxiv.org/abs/1907.11692 (2019).

Logan IV, R. L., I. Balažević, E. Wallace, F. Petroni, S. Singh and S. Riedel, “Cutting down on prompts and parameters: Simple few-shot learning with language models”, arXiv preprint arXiv:2106.13353 (2021).

Lourie, N., R. Le Bras, C. Bhagavatula and Y. Choi, “Unicorn on rainbow: A universal commonsense reasoning model on a new multitask benchmark”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 35, pp. 13480–13488 (2021).

Lu, P., R. Gong, S. Jiang, L. Qiu, S. Huang, X. Liang and S.-C. Zhu, “Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning”, in “The 59th Annual Meeting of the Association for Computational Linguistics (ACL)”, (2021a).

Lu, P., S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark and A. Kalyan, “Learn to explain: Multimodal reasoning via thought chains for science question answering”, in “The 36th Conference on Neural Information Processing Systems (NeurIPS)”, (2022a).

Lu, P., L. Qiu, K.-W. Chang, Y. N. Wu, S.-C. Zhu, T. Rajpurohit, P. Clark and A. Kalyan, “Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning”, arXiv preprint arXiv:2209.14610 (2022b).

Lu, P., L. Qiu, J. Chen, T. Xia, Y. Zhao, W. Zhang, Z. Yu, X. Liang and S.-C. Zhu, “Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning”, in “The 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)”, (2021b).

Lu, S., D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang *et al.*, “Codexglue: A machine learning benchmark dataset for code understanding and generation”, arXiv preprint arXiv:2102.04664 (2021c).

Lu, Y., M. Bartolo, A. Moore, S. Riedel and P. Stenetorp, “Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity”, arXiv preprint arXiv:2104.08786 (2021d).

Luo, M., S. Saxena, S. Mishra, M. Parmar and C. Baral, “Biotabqa: Instruction learning for biomedical table question answering”, arXiv preprint arXiv:2207.02419 (2022).

McCann, B., N. S. Keskar, C. Xiong and R. Socher, “The natural language decathlon: Multitask learning as question answering”, arXiv preprint arXiv:1806.08730 (2018).

Miao, S.-y., C.-C. Liang and K.-Y. 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”, pp. 975–984 (Association for Computational Linguistics, Online, 2020), URL https://aclanthology.org/2020.acl-main.92.

Min, S., V. Zhong, L. Zettlemoyer and H. Hajishirzi, “Multi-hop reading comprehension through question decomposition and rescoring”, arXiv preprint arXiv:1906.02916 (2019).

Mishra, S. and A. Arunkumar, “Front contribution instead of back propagation”, arXiv preprint arXiv:2106.05569 (2021).

Mishra, S. and A. Arunkumar, “A proposal to study" is high quality data all we need?"”, arXiv preprint arXiv:2203.06404 (2022).

Mishra, S., A. Arunkumar and C. Baral, “Investigating the failure modes of the auc metric and exploring alternatives for evaluating systems in safety critical applications”, arXiv preprint arXiv:2210.04466 (2022a).

Mishra, S., A. Arunkumar, C. Bryan and C. Baral, “Our evaluation metric needs an update to encourage generalization”, arXiv preprint arXiv:2007.06898 (2020a).

Mishra, S., A. Arunkumar, C. Bryan and C. Baral, “A survey of parameters associated with the quality of benchmarks in nlp”, arXiv preprint arXiv:2210.07566 (2022b).

Mishra, S., A. Arunkumar, B. Sachdeva, C. Bryan and C. Baral, “Dqi: Measuring data quality in nlp”, arXiv preprint arXiv:2005.00816 (2020b).

Mishra, S., M. Finlayson, P. Lu, L. Tang, S. Welleck, C. Baral, T. Rajpurohit, O. Tafjord, A. Sabharwal, P. Clark and A. Kalyan, “LILA: A unified benchmark for mathematical reasoning”, in “Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing”, pp. 5807–5832 (Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022c), URL https://aclanthology.org/2022.emnlp-main.392.

Mishra, S., M. K. Jena and A. K. Tripathy, “Towards the development of disaster management tailored machine learning systems”, in “2022 IEEE India Council International Subsections Conference (INDISCON)”, pp. 1–6 (IEEE, 2022d).

Mishra, S., D. Khashabi, C. Baral, Y. Choi and H. Hajishirzi, “Reframing instructional prompts to GPTk’s language”, in “Findings of the Association for Computational Linguistics: ACL 2022”, pp. 589–612 (Association for Computational Linguistics, Dublin, Ireland, 2022e), URL https://aclanthology.org/2022.findings-acl.50.

Mishra, S., D. Khashabi, C. Baral and H. Hajishirzi, “Cross-task generalization via natural language crowdsourcing instructions”, in “Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, pp. 3470–3487 (2022f).

Mishra, S., A. Mitra, N. Varshney, B. Sachdeva, P. Clark, C. Baral and A. Kalyan, "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)", pp. 3505–3523 (2022g).

Mishra, S. and E. Nouri, "Help me think: A simple prompting strategy for non-experts to create customized content with models", arXiv preprint arXiv:2208.08232 (2022).

Mishra, S., J. Payan, C. Negreanu, C. Poelitz, C. Baral, S. Roy, R. Chakravarthy, B. V. Durme and E. Nouri, "Instructexcel: A benchmark for natural language instruction in excel", arXiv preprint (2023).

Mishra, S. and B. S. Sachdeva, "Do we need to create big datasets to learn a task?", in "Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing", pp. 169–173 (Association for Computational Linguistics, Online, 2020), URL https://aclanthology.org/2020.sustainlp-1.23.

Mishra, S., B. S. Sachdeva and C. Baral, "Pretrained transformers do not always improve robustness", arXiv preprint arXiv:2210.07663 (2022h).

Mishra, S. R., M. P. Korukonda, L. Behera and A. Shukla, "Enabling cyber-physical demand response in smart grids via conjoint communication and controller design", IET Cyber-Physical Systems: Theory & Applications **4**, 4, 291–303 (2019).

Mishra, S. R., N. V. Srinath, K. M. Preetam and L. Behera, "A generalized novel framework for optimal sensor-controller connection design to guarantee a stable cyber physical smart grid", in "2015 IEEE 13th International Conference on Industrial Informatics (INDIN)", pp. 424–429 (IEEE, 2015).

Nayyar, R. K., P. Verma and S. Srivastava, "Differential assessment of black-box ai agents", in "Proceedings of the AAAI Conference on Artificial Intelligence", vol. 36, pp. 9868–9876 (2022).

Nogueira, R., Z. Jiang and J. Li, "Investigating the limitations of the transformers with simple arithmetic tasks", arXiv preprint arXiv:2102.13019 (2021).

Nye, M., A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz, M. Bosma, D. Luan *et al.*, "Show your work: Scratchpads for intermediate computation with language models", arXiv preprint arXiv:2112.00114 (2021).

Ouyang, L., J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback", Preprint (2022).

Pal, K. K., K. Kashihara, P. Banerjee, S. Mishra, R. Wang and C. Baral, "Constructing flow graphs from procedural cybersecurity texts", in "Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021", pp. 3945–3957 (2021).

Papineni, K., S. Roukos, T. Ward and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation", in "Proceedings of the 40th annual meeting of the Association for Computational Linguistics", pp. 311–318 (2002).

Parmar, M., S. Mishra, M. Geva and C. Baral, “Don’t blame the annotator: Bias already starts in the annotation instructions”, arXiv preprint arXiv:2205.00415 (2022a).

Parmar, M., S. Mishra, M. Purohit, M. Luo, M. Mohammad and C. Baral, “In-BoXBART: Get instructions into biomedical multi-task learning”, in “Findings of the Association for Computational Linguistics: NAACL 2022”, pp. 112–128 (Association for Computational Linguistics, Seattle, United States, 2022b), URL https://aclanthology.org/2022.findings-naacl.10.

Patel, A., S. Bhattamishra and N. Goyal, “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”, pp. 2080–2094 (Association for Computational Linguistics, Online, 2021), URL https://aclanthology.org/2021.naacl-main.168.

Patel, P., S. Mishra, M. Parmar and C. Baral, “Is a question decomposition unit all we need?”, EMNLP 2022, Abu Dhabi (2022).

Perez, E., *Finding and Fixing Undesirable Behaviors in Pretrained Language Models.*, Ph.D. thesis, New York University, USA (2022).

Perez, E., P. Lewis, W.-t. Yih, K. Cho and D. Kiela, “Unsupervised question decomposition for question answering”, arXiv preprint arXiv:2002.09758 (2020).

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee and L. Zettlemoyer, “Deep contextualized word representations”, in “Proceedings of NAACL-HLT”, pp. 2227–2237 (2018).

Puri, R. S., S. Mishra, M. Parmar and C. Baral, “How many data samples is an additional instruction worth?”, arXiv preprint arXiv:2203.09161 (2022).

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners”, OpenAI blog **1**, 8, 9 (2019).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer”, Journal of Machine Learning Research **21**, 140, 1–67, URL http://jmlr.org/papers/v21/20-074.html (2020).

Rajpurkar, P., R. Jia and P. Liang, “Know what you don’t know: Unanswerable questions for squad”, arXiv preprint arXiv:1806.03822 (2018).

Ran, Q., Y. Lin, P. Li, J. Zhou and Z. Liu, “Numnet: Machine reading comprehension with numerical reasoning”, arXiv preprint arXiv:1910.06701 (2019).

Ravichander, A., A. Naik, C. Rose and E. Hovy, “Equate: A benchmark evaluation framework for quantitative reasoning in natural language inference”, arXiv preprint arXiv:1901.03735 (2019).

Reif, E., D. Ippolito, A. Yuan, A. Coenen, C. Callison-Burch and J. Wei, “A recipe for arbitrary text style transfer with large language models”, arXiv preprint arXiv:2109.03910 (2021).

Reynolds, L. and K. McDonell, “Prompt programming for large language models: Beyond the few-shot paradigm”, in “Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems”, pp. 1–7 (2021).

Ribeiro, M. T., T. Wu, C. Guestrin and S. Singh, “Beyond accuracy: Behavioral testing of nlp models with checklist”, in “Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics”, pp. 4902–4912 (2020).

Rogers, A., M. Gardner and I. Augenstein, “Qa dataset explosion: A taxonomy of nlp resources for question answering and reading comprehension”, arXiv preprint arXiv:2107.12708 (2021).

Ross, A., T. Wu, H. Peng, M. E. Peters and M. Gardner, “Tailor: Generating and perturbing text with semantic controls”, arXiv preprint arXiv:2107.07150 (2021).

Roy, S. and D. Roth, “Solving general arithmetic word problems”, in “Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing”, pp. 1743–1752 (2015).

Roy, S. and D. Roth, “Unit dependency graph and its application to arithmetic word problem solving”, in “Thirty-First AAAI Conference on Artificial Intelligence”, (2017).

Roy, S. and D. Roth, “Mapping to declarative knowledge for word problem solving”, Transactions of the Association for Computational Linguistics **6**, 159–172 (2018).

Roy, S., T. Vieira and D. Roth, “Reasoning about quantities in natural language”, Transactions of the Association for Computational Linguistics **3**, 1–13 (2015).

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge”, International journal of computer vision **115**, 3, 211–252 (2015).

Sakaguchi, K., R. Le Bras, C. Bhagavatula and Y. Choi, “Winogrande: An adversarial winograd schema challenge at scale”, in “Proceedings of the AAAI”, (2020).

Sanh, V., A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf and A. M. Rush, “Multitask prompted training enables zero-shot task generalization”, in “Proceedings of ICLR”, (2022).

Sarkar, B. K., *Hindu Achievements in Exact Science: A Study in the History of Scientific Development* (Longmans, Green and Company, 1918).

Saxton, D., E. Grefenstette, F. Hill and P. Kohli, “Analysing mathematical reasoning abilities of neural models”, arXiv preprint arXiv:1904.01557 (2019).

Scaria, K., H. Gupta, S. A. Sawant, S. Mishra and C. Baral, “Instructabsa: Instruction learning for aspect based sentiment analysis”, arXiv preprint arXiv:2302.08624 (2023).

Schick, T. and H. Schütze, “Few-shot text generation with natural language instructions”, in “Proceedings of EMNLP”, (2021).

Sellam, T., D. Das and A. Parikh, “Bleurt: Learning robust metrics for text generation”, in “Proceedings of ACL”, pp. 7881–7892 (2020).

Shao, Y. and N. Nakashole, “Chartdialogs: Plotting from natural language instructions”, in “Proceedings of ACL”, pp. 3559–3574 (2020).

Shridhar, M., J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer and D. Fox, “Alfred: A benchmark for interpreting grounded instructions for everyday tasks”, in “Proceedings of the IEEE/CVF”, pp. 10740–10749 (2020).

Srivastava, A., A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso *et al.*, “Beyond the imitation game: Quantifying and extrapolating the capabilities of language models”, arXiv preprint arXiv:2206.04615 (2022).

Stepputtis, S., J. Campbell, M. Phielipp, S. Lee, C. Baral and H. B. Amor, “Language-conditioned imitation learning for robot manipulation tasks”, arXiv preprint arXiv:2010.12083 (2020).

Swayamdipta, S., R. Schwartz, N. Lourie, Y. Wang, H. Hajishirzi, N. A. Smith and Y. Choi, “Dataset cartography: Mapping and diagnosing datasets with training dynamics”, in “Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)”, pp. 9275–9293 (2020).

Tafjord, O., P. Clark, M. Gardner, W.-t. Yih and A. Sabharwal, “Quarel: A dataset and models for answering questions about qualitative relationships”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 33, pp. 7063–7071 (2019).

Talmor, A. and J. Berant, “The web as a knowledge-base for answering complex questions”, arXiv preprint arXiv:1803.06643 (2018).

Tam, D., R. R. Menon, M. Bansal, S. Srivastava and C. Raffel, “Improving and simplifying pattern exploiting training”, arXiv preprint arXiv:2103.11955 (2021).

Upadhyay, S. and M.-W. Chang, “Draw: A challenging and diverse algebra word problem set”, Tech. rep., Citeseer (2015).

Upadhyay, S., M.-W. Chang, K.-W. Chang and W.-t. Yih, “Learning from explicit and implicit supervision jointly for algebra word problems”, in “Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing”, pp. 297–306 (2016).

Varshney, N., S. Mishra and C. Baral, “Interviewer-candidate role play: Towards developing real-world nlp systems”, arXiv preprint arXiv:2107.00315 (2021).

Varshney, N., S. Mishra and C. Baral, “ILDAE: Instance-level difficulty analysis of evaluation data”, in “Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)”, pp. 3412–3425 (Association for Computational Linguistics, Dublin, Ireland, 2022a), URL https://aclanthology.org/2022.acl-long.240.

Varshney, N., S. Mishra and C. Baral, “Investigating selective prediction approaches across several tasks in iid, ood, and adversarial settings”, in “Findings of the Association for Computational Linguistics: ACL 2022”, pp. 1995–2002 (2022b).

Varshney, N., S. Mishra and C. Baral, “Let the model decide its curriculum for multitask learning”, in “Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing”, pp. 117–125 (2022c).

Varshney, N., S. Mishra and C. Baral, “Towards improving selective prediction ability of nlp systems”, in “Proceedings of the 7th Workshop on Representation Learning for NLP”, pp. 221–226 (2022d).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, “Attention is all you need”, Advances in neural information processing systems **30** (2017).

Verma, P., S. R. Marpally and S. Srivastava, “Asking the right questions: Learning interpretable action models through query answering”, in “Proceedings of the AAAI Conference on Artificial Intelligence”, vol. 35, pp. 12024–12033 (2021).

Verma, P. and S. Srivastava, “Learning causal models of autonomous agents using interventions”, in “IJCAI 2021 Workshop on Generalization in Planning”, (2021).

Wang, A., Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, “Superglue: A stickier benchmark for general-purpose language understanding systems”, Proceedings of NeurIPS **32** (2019).

Wang, A., A. Singh, J. Michael, F. Hill, O. Levy and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding”, arXiv preprint arXiv:1804.07461 (2018).

Wang, L., R. Li, Y. Yan, Y. Yan, S. Wang, W. Wu and W. Xu, “Instructionner: A multi-task instruction-based generative framework for few-shot ner”, arXiv preprint arXiv:2203.03903 (2022a).

Wang, X., J. Wei, D. Schuurmans, Q. Le, E. Chi and D. Zhou, “Self-consistency improves chain of thought reasoning in language models”, arXiv preprint arXiv:2203.11171 (2022b).

Wang, Y., Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi and H. Hajishirzi, “Self-instruct: Aligning language model with self generated instructions”, arXiv preprint arXiv:2212.10560 (2022c).

Wang, Y., S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Arunkumar, A. Ashok, A. S. Dhanasekaran, A. Naik, D. Stap *et al.*, "Benchmarking generalization via in-context instructions on 1,600+ language tasks", arXiv preprint arXiv:2204.07705 (2022d).

Wang, Y., W. Wang, S. Joty and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation", arXiv preprint arXiv:2109.00859 (2021).

Wei, J., M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai and Q. V. Le, "Finetuned language models are zero-shot learners", in "Proceedings of ICLR", (2022a).

Wei, J., X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le and D. Zhou, "Chain of thought prompting elicits reasoning in large language models", arXiv preprint arXiv:2201.11903 (2022b).

Welleck, S., J. Liu, R. L. Bras, H. Hajishirzi, Y. Choi and K. Cho, "Naturalproofs: Mathematical theorem proving in natural language", in "Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)", (2021), URL https://openreview.net/forum?id=Jvxa8adr3iY.

Welleck, S., P. West, J. Cao and Y. Choi, "Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics", in "AAAI", (2022), URL https://arxiv.org/pdf/2109.13986.pdf.

Weller, O., N. Lourie, M. Gardner and M. Peters, "Learning from task descriptions", in "Proceedings of EMNLP", pp. 1361–1375 (2020).

Weston, J., A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks", arXiv preprint arXiv:1502.05698 (2015).

Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing", arXiv preprint arXiv:1910.03771 (2019).

Wolfson, T., M. Geva, A. Gupta, M. Gardner, Y. Goldberg, D. Deutch and J. Berant, "Break it down: A question understanding benchmark", Transactions of the Association for Computational Linguistics **8**, 183–198 (2020).

Wu, Y., A. Jiang, J. Ba and R. B. Grosse, "{INT}: An inequality benchmark for evaluating generalization in theorem proving", in "International Conference on Learning Representations", (2021), URL https://openreview.net/forum?id=O6LPudowNQm.

Xie, K., S. Wiegreffe and M. Riedl, "Calibrating trust of multi-hop question answering systems with decompositional probes", URL https://arxiv.org/abs/2204.07693 (2022).

Xuan, H., A. Stylianou, X. Liu and R. Pless, "Hard negative examples are hard, but useful", in "Proceedings of ECCV", pp. 126–142 (Springer, 2020).

Yang, Z., P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov and C. D. Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering", CoRR **abs/1809.09600**, URL http://arxiv.org/abs/1809.09600 (2018).

Yao, Z., Y. Tang, W.-t. Yih, H. Sun and Y. Su, "An imitation game for learning semantic parsers from user interaction", in "Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)", pp. 6883–6902 (Association for Computational Linguistics, Online, 2020), URL https://aclanthology.org/2020.emnlp-main.559.

Ye, Q., B. Y. Lin and X. Ren, "Crossfit: A few-shot learning challenge for cross-task generalization in nlp", in "Proceedings of EMNLP", (2021).

Ye, Q. and X. Ren, "Zero-shot learning by generating task-specific adapters", arXiv preprint arXiv:2101.00420 (2021).

Yin, P., B. Deng, E. Chen, B. Vasilescu and G. Neubig, "Learning to mine aligned code and natural language pairs from stack overflow", in "International Conference on Mining Software Repositories", MSR, pp. 476–486 (ACM, 2018).

Zamir, A. R., A. Sax, W. Shen, L. J. Guibas, J. Malik and S. Savarese, "Taskonomy: Disentangling task transfer learning", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 3712–3722 (2018).

Zhang, J., Y. Zhao, M. Saleh and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization", (2019).

Zhang, X., D. Ramachandran, I. Tenney, Y. Elazar and D. Roth, "Do language embeddings capture scales?", arXiv preprint arXiv:2010.05345 (2020).

Zhao, W., K. Arkoudas, W. Sun and C. Cardie, "Compositional task-oriented parsing as abstractive question answering", arXiv preprint arXiv:2205.02068 (2022).

Zhao, Z., E. Wallace, S. Feng, D. Klein and S. Singh, "Calibrate before use: Improving few-shot performance of language models", in "Proceedings of ICML", pp. 12697–12706 (2021).

Zheng, K., J. M. Han and S. Polu, "Minif2f: a cross-system benchmark for formal olympiad-level mathematics", arXiv preprint arXiv:2109.00110 (2021).

Zhong, R., K. Lee, Z. Zhang and D. Klein, "Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections", in "Findings of the Association for Computational Linguistics: EMNLP 2021", pp. 2856–2878 (2021).

Zhong, R., C. Snell, D. Klein and J. Eisner, "Active programming by example with a natural language prior", URL https://arxiv.org/abs/2205.12422 (2022).

Zhou, B., D. Khashabi, Q. Ning and D. Roth, ""going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding", in "Proceedings of EMNLP-IJCNLP", pp. 3354–3360 (2019).

Zhou, D., N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le and E. Chi, "Least-to-most prompting enables complex reasoning in large language models", arXiv preprint arXiv:2205.10625 (2022).

APPENDIX A

RELATED PUBLICATION DETAILS

Most ideas in this dissertation have appeared in the following publications, with consent of my co-authors.

- Mishra, S., Khashabi, D., Baral, C., Hajishirzi, H. (2022, May). Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 3470-3487).

- Mishra, S., Mitra, A., Varshney, N., Sachdeva, B., Clark, P., Baral, C., Kalyan, A. (2022, May). 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) (pp. 3505-3523).

- Mishra, S., Khashabi, D., Baral, C., Choi, Y., Hajishirzi, H. (2022, May). Reframing Instructional Prompts to GPTk's Language. In Findings of the Association for Computational Linguistics: ACL 2022 (pp. 589-612).

- Mishra, S., Finlayson, M., Lu, P., Tang, L., Welleck, S., Baral, C., ... Kalyan, A. (2022). 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.

- Kuznia, K., Mishra, S., Parmar, M., Baral, C. (2022). Less is more: Summary of long instructions is better for program synthesis. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4532–4552, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

- Patel, P., Mishra, S., Parmar, M., Baral, C. (2022). Is a Question Decomposition Unit All We Need?. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 4553–4569, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

- Puri, R. S., Mishra, S., Parmar, M., Baral, C. (2022). How Many Data Samples is an Additional Instruction Worth?. arXiv preprint arXiv:2203.09161.

- Mishra, S., Nouri, E. (2022). HELP ME THINK: A Simple Prompting Strategy for Non-experts to Create Customized Content with Models. arXiv preprint arXiv:2208.08232.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA