Cross-Task Generalization via Natural Language Crowdsourcing Instructions

Swaroop Mishra[1] **Daniel Khashabi**[2] **Chitta Baral**[1] **Hannaneh Hajishirzi**[2,3]

[1]Arizona State University, [2]Allen Institute for AI, [3]University of Washington

Abstract

Humans (e.g., crowdworkers) have a remarkable ability in solving different tasks, by simply reading textual *instructions* that define them and looking at a few examples. NLP models built with the conventional paradigm, however, often struggle with generalization across tasks (e.g., a question-answering system cannot solve classification tasks). 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 introduce NATURAL-INSTRUCTIONS, a dataset of 61 distinct tasks, their human-authored instructions and 193k task instances. The instructions are obtained from crowdsourcing instructions used to create existing NLP datasets and mapped to a unified schema. We adopt generative pre-trained language models to encode task-specific instructions along with input and generate task output. Our results indicate that models *benefit from instructions* when evaluated in terms of generalization to unseen tasks. These models, however, are far behind supervised task-specific models, indicating significant room for more progress in this direction.[1]

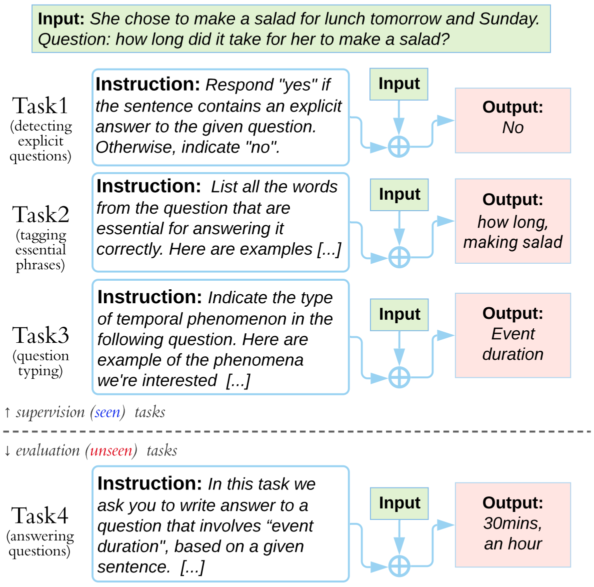

Input: She chose to make a salad for lunch tomorrow and Sunday. Question: how long did it take for her to make a salad?
Task1 (detecting explicit questions)
Instruction: Respond "yes" if the sentence contains an explicit answer to the given question. Otherwise, indicate "no".
Input
Output: No
Task2 (tagging essential phrases)
Instruction: List all the words from the question that are essential for answering it correctly. Here are examples [...]
Input
Output: how long, making salad
Task3 (question typing)
Instruction: Indicate the type of temporal phenomenon in the following question. Here are example of the phenomena we're interested [...]
Input
Output: Event duration
↑ supervision (seen) tasks
↓ evaluation (unseen) tasks
Task4 (answering questions)
Instruction: In this task we ask you to write answer to a question that involves "event duration", based on a given sentence. [...]
Input
Output: 30mins, an hour

Figure 1: NATURAL-INSTRUCTIONS contains a variety of tasks, each accompanied by their natural language instructions. A model is allowed to utilize *seen* tasks to get familiar with language *instructions* and use them to map a given input to its corresponding output. The model is consequently evaluated on *unseen* tasks which requires a successful comprehension of their instructions.

1 Introduction

Recent advancements with large pre-trained language models (LM) (Brown et al., 2020) have shown massive performance gains in solving NLP benchmarks (Wang et al., 2019; Clark et al., 2020). Furthermore, in the context of multi-task learning, recent studies have shown tremendous promise in showing generalization to instances of *similar* tasks (Khashabi et al., 2020; Aghajanyan et al., 2021) (Table 2a; 1st column). However, cross-task generalization, i.e., *generalization* to *unseen* tasks (Table 2a; 2nd column), has generally remained under-explored. Meanwhile, average humans can follow natural language *instructions* to solve a variety of problems, as evident by the success of crowdsourcing platforms.

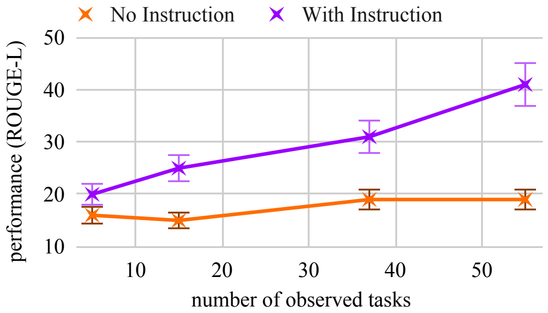
In this paper, we study if models equipped with the understanding of natural language instructions can generalize to *unseen* tasks (Fig. 1). We build a dataset (called NATURAL-INSTRUCTIONS) consisting of crowdsourcing instructions for various tasks and their instances. We use our dataset to build models that learn to encode instructions and systematically evaluate if the models can follow new tasks just from their instructions and without any task-specific labeled data.

NATURAL-INSTRUCTIONS consists of 61 distinct NLP tasks and 193k instances collected

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

Task	Instance-Level Generalization	Task-Level Generalization
Training data	$X^{\text{train}}, Y^{\text{train}}$	$(I_t, X_t^{\text{train}}, Y_t^{\text{train}})$ $t \in \mathcal{T}_{\text{seen}}$
Evaluation	$x \to y$ where: $(x, y) \in (X^{\text{test}}, Y^{\text{test}})$	$(x, I_t) \to y$ where: $(x, y) \in (X_t^{\text{test}}, Y_t^{\text{test}})$ $t \in \mathcal{T}_{\text{unseen}}$

(a) A comparison of *task* vs *instance*-level generalization I_t, X_t and Y_t indicate natural language instructions, input and output sets respectively for task t. In the conventional setup, training and evaluation is done on the instances of the same task. However, in task-level generalization, a model is expected to generalize to unseen tasks, where $\mathcal{T}_{\text{unseen}} \cap \mathcal{T}_{\text{seen}} = \varnothing$.



(b) BART evaluation on *unseen* tasks (y-axis is perf. on $\mathcal{T}_{\text{unseen}}$) when supervised with *seen* tasks (x-axis is $|\mathcal{T}_{\text{seen}}|$). A model using instructions (I_t) consistently improves with more observed tasks. In contrast, a model with no access the instructions show no sign of improved generalization. Details in §5.3.

Figure 2: The formal definition of generalization to unseen tasks (a) and a summary of its empirical outcome (b).

from crowsdsourcing instructions of existing NLP datasets, inspired by Efrat and Levy (2020). Different from this work, NATURAL-INSTRUCTIONS includes a diverse set of tasks whose instructions are all mapped to a unified schema, which enables building models that generalize across different tasks. Moreover, tasks in NATURAL-INSTRUCTIONS are defined as minimal stand-alone steps provided to crowdworkers to complete a complex task. For example, the tasks in Quoref (Dasigi et al., 2019) is divided into two tasks of question generation and answer generation, mimicking its data collection process by humans.

Having collected a dataset of 61 tasks, we are able to train models on a subset of the tasks and generalize them to the remaining ones. We adopt BART (Lewis et al., 2019) and GPT3 (Brown et al., 2020) to encode task-specific instruction and input into textual formats and decode the task output. Our experimental results indicate that these models leverage natural language instructions to improve the generalization to new tasks. For example, BART and GPT3 models achieve 75% and 47% relative gains to models that only use task-specific prompts (§5). Importantly, as Fig. 2b shows, models that use task instructions generalize better to new tasks, as they get to observe more tasks (§5.3).

This observation suggests that scaling up our dataset would provide an exciting future opportunity for building stronger instruction-following systems. Despite the benefits of instructions, we find that there is a sizable gap between models' generalization performance and their estimated upperbounds, indicating substantial room to progress. We hope this gap, as well as the availability of NATURAL-INSTRUCTIONS will encourage the development of stronger models of language.

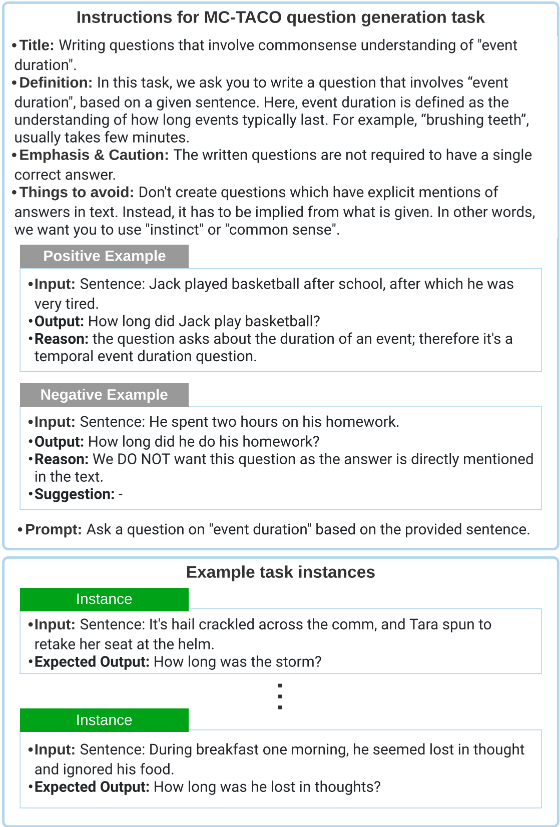
Contributions: In summary, the contributions of this work are as follows: (a) we introduce NATURAL-INSTRUCTIONS, a dataset of human-authored instructions curated from existing well-known datasets mapped to a unified schema, providing training and evaluation data for learning from instructions; (b) we build models that can encode instructions and show: (b.1) the benefit of cross-task generalization by leveraging instructions; (b.2) the importance of different elements of instructions in the performance; (b.3) noteworthy headroom for improvement on our benchmark, which hopefully will motivate further work in this direction.

2 Defining Task-Level Generalization

Here we formally define the problem setup for generalization across tasks. Each task t is defined as the union of input/output instances (X_t, Y_t). Additionally, each task is described in terms of its natural language instructions I_t.

Task-specific models. Standard supervised learning uses task-specific training instances to train a model that learns a mapping between input and output: $M(x) = y$ for $(x, y) \in (X_t^{\text{train}}, Y_t^{\text{train}})$ and evaluated on the test instances of the same (or similar) task $(X_t^{\text{test}}, Y_t^{\text{test}})$. We refer to this as *instance-level* generalization (Table 2a, 1st column).

Cross-task models. In this setup, the goal is to learn a model M that at inference obtains the output y given the input x and the task instruction I_t: $M(I_t, x) = y$, for $(x, y) \in (X_t, Y_t)$. In contrast to the task-specific models, no task-specific training data is used to learn the mapping M. We use NATURAL-INSTRUCTIONS to study this question:



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: An example from our dataset. Note that it follows the schema provided in Fig. 4. See Fig .13 for more examples.

can a model be trained to follow instructions via training tasks $\mathcal{T}_{\text{seen}}$ and be generalized to follow instructions for a task $t' \in \mathcal{T}_{\text{unseen}}$. We refer to this as *task*-level generalization (Table 2a, 2nd column).

3 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 shows a sample instruction that describes the task of 'generating questions that require an understanding of event duration' accompanied with a few positive and negative examples to better guide a crowd worker. Here, we first introduce a unified schema for representing instructions (§3.1), and then describe how existing datasets and their crowdsourcing templates are mapped into our schema (§3.2).

3.1 Instruction Schema

Instructions are used in crowdsourcing various datasets, are written by distinct authors for different purposes, and they are different in a variety of ways (see Appendix A.2 for their differences.)

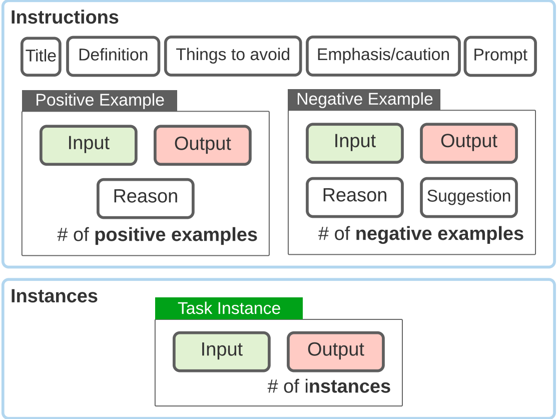


Figure 4: The schema used for representing instruction in Natural-Instructions (§3.1), shown in plate notation.

We introduce a unified schema (Fig. 4) 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 ingredients of this schema:

- Title provides 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 avoided. These are helpful in defining 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 map-

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

Table 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. The complete list is in Table 9 in Appendix.

ping the raw instructions (designed for crowdworkers) to our instruction schema.

3.2 Constructing Natural-Instructions

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 in order to get their crowdsourcing templates and raw data. This yields the following datasets: CosmosQA (Huang et al., 2019), DROP (Dua et al., 2019), 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).[2]

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.[3] Table 1 shows subtasks extracted for Quoref and QASQ. 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 instructions as-is.

In total, there are 61 tasks, which are categorized into 6 semantic categories (Table 2). We assigned these broad categories to the tasks in order 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 are distinct. For example, while *question generation* is part of Quoref, CosmosQA and QASC, each has its own separate variant of the question generation task (see Fig. 12 in Appendix).

3.2.2 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.[4]

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

[2] 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.

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

[4] On average, the process of data curation for each task takes around 5 hrs-34 hrs (details in Appendix; Table 7).

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 2: Task categories and their statistics.

in a slightly different format, the extraction of intermediate annotations is often non-trivial, requiring parsing HTML tags.

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 §4.1). 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.

3.2.3 NATURAL-INSTRUCTIONS Statistics

In summary, NATURAL-INSTRUCTIONS consists of subtasks each with a set of instructions and input/output instances (Fig. 3 and 4). The complete list of instructions is included in the appendix. In total, the dataset includes 61 tasks and $193k$ instances. Table 2 shows data statistics for each task category.[5] 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 8).

4 Problem Setup and Models

Based our problem setup (§2), we define models (§4.2) under different cross-task generalization settings (§4.1).

4.1 Task Splits and Generalizations Types

Random split. This setup follows the common practice in benchmarking NLP models with random data splits. Here, two tasks from each task category (Table 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 $\mathcal{T}_{\text{unseen}}$ and 49 tasks in $\mathcal{T}_{\text{seen}}$.[6]

$\texttt{Prompt}: I_t^{\text{prompt}}$

$\texttt{Definition}: I_t^{\text{Definition}}$

$\texttt{Things to Avoid}: I_t^{\text{avoid.}}$

$\texttt{Emphasis\&Caution}: I_t^{\text{emph.}}$

$\texttt{NegativeExample1}-$

$\quad \texttt{input}: I_t^{\text{pos. ex.}}, \texttt{output}: I_t^{\text{pos. ex.}}, \texttt{reason}: I_t^{\text{pos. ex.}}$

$\texttt{PositiveExample1}-$

$\quad \texttt{input}: I_t^{\text{pos. ex.}}, \texttt{output}: I_t^{\text{pos. ex.}} \texttt{reason}: I_t^{\text{pos. ex.}}$

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

Figure 5: Encoding instruction I_t, where I_t^c refers to the textual of a component c in the instruction schema.

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

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 $\mathcal{T}_{\text{seen}}$.

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 $\mathcal{T}_{\text{seen}}$. 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.

4.2 Models

We build models using pre-trained LMs with encoder-decoder architectures (GPT3 (Brown et al., 2020) and BART (Lewis et al., 2019)).

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 (§3.1). Here, we map each element of the instruction to a textual format and append it before the input instance. Fig. 5 shows how we encode the full instruction.

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

[6] Those tasks that do not accept a relatively reliable automatic evaluation are excluded from $\mathcal{T}_{\text{unseen}}$.

In our experiments, we study the impact of encoding each element of the instruction for cross-task generalization. In particular, we study these encodings: (1)PROMPT (2) POS. EXAMPLES, (3) PROMPT + DEFINITION, (4) PROMPT + THINGS TO AVOID, (5) POSITIVE EXAMPLES, (6) PROMPT + DEF + POS. EXAMP., (7) FULL INSTRUCTION, and (8) FULL INSTRUCTION-NEGATIVE EXAMPLES.[7] Here PROMPT and POS. EXAMPLES correspond to prompting setups in the recent literature (Le Scao and Rush, 2021; Lu et al., 2021).

BART. We use BART (base) (Lewis et al., 2019) which allows us to fine-tune its model parameters. BART (base) is an encoder-decoder architecture with 140 million parameters (roughly $1.2k$ times smaller than GPT3). For each setup, the input is encoded using different instruction elements, trained on all $\mathcal{T}_{\text{seen}}$ tasks, and evaluated on $\mathcal{T}_{\text{unseen}}$ (§4.1).

GPT3. GPT3 (Brown et al., 2020) is an autoregressive LM with 175 billion parameters and has shown successful results in mimicking demonstrations provided in its prompt. Since we are not able to fine-tune the parameters of this model, we use it under its default setting (Brown et al., 2020) by encoding the input given different elements of instructions, and evaluate it on $\mathcal{T}_{\text{unseen}}$ (§4.1).

5 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.[8]

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.

5.1 Generalization Under Random Split

Table 3 reports the results of both BART and GPT3 models on the evaluation tasks obtained from our random split (§4.1) with a variety of encodings that incorporate different elements of the instructions(§4.2).[9]

Note that BART is fine-tuned with the $\mathcal{T}_{\text{seen}}$ tasks, while GPT3 uses no fine-tuning.[10]

Instructions benefit cross-task generalization. Table 3 (avg column) shows that instructions improve the performance of both BART and GPT3, as observed by comparing rows that include instruction components with PROMPT and POSITIVE EXAMPLE rows. For instance, PROMPT+DEFINITION + POS. EXAMPLES results in +65% and +47% relative gains over the PROMPT encoding, in BART and GPT3, respectively. The gains observed for GPT3 are promising since the model is not fine-tuned to training tasks.

Best results correspond to FULL INSTRUCTION - NEG EXAMPLES, indicating all instruction elements are helpful except the NEGATIVE EXAMPLES, which remains a hurdle for NLP models as also discussed in previous work (Xuan et al., 2020; Lin et al., 2003). Finally, it is interesting to observe that adding more instruction elements (e.g., DEFINITIONS) significantly outperforms the PROMPT + POS. EXAMP. setup which resembles similar studies in GPT3 prompting (Zhao et al., 2021).

Note that, these findings are in contrast with those of (Efrat and Levy, 2020), where they observe that large LMs "fail to follow a series of gradually simpler instructions". We hypothesize that our results are due to dividing the crowdsourcing instructions into minimal tasks and mapping them into a coherent schema, which makes it easier for LMs to exploit the prevalence and the diversity of training tasks.

Results on task categories. Table 3 shows the performance of our models on different task categories. For most tasks, PROMPT+DEF+POS. EXAMP. encoding outperforms other encodings, however, these gains are not uniform across task categories. For example, 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.

[7] Refer to Appendix C for our study on the impact of other instruction elements.

[8] 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.

[9] Refer to Appendix C for comprehensive ablation of different instruction elements.

[10] We cannot report results for NO INSTRUCTIONS for GPT3 since the task is under-defined.

	BART							GPT3						
task category →	QG	AG	CF	IAG	MM	VF	avg	QG	AG	CF	IAG	MM	VF	avg
No Instruction	26	6	0	21	33	7	13	-	-	-	-	-	-	-
PROMPT	27	22	7	22	34	**9**	20	33	32	14	13	**73**	16	30
+DEFINITION	35	24	50	**25**	36	7	30↑ (+50)	36	35	40	14	70	16	35↑ (+17)
+THINGS TO AVOID	33	24	4	24	**58**	**9**	25↑ (+25)	28	33	11	16	68	14	28↓ (-7)
+POS. EXAMP.	53	22	14	**25**	17	7	23↑ (+15)	**43**	49	29	21	70	**36**	41↑ (+37)
+DEFINITION+POS. EXAMP.	51	23	**56**	**25**	37	6	33↑ (+65)	**43**	50	**45**	**23**	70	32	**44**↑(+47)
POS. EXAMP.	**55**	6	18	**25**	8	6	20	30	32	15	16	68	23	31↑(+3)
Full Instruction	46	25	52	25	35	7	32↑ (+60)	33	18	8	12	60	11	24↓(-20)
- NEG. EXAMP.	52	**30**	50	**25**	47	8	**35**↑ (+75)	**43**	**54**	44	21	70	32	**44**↑(+47)

Table 3: BART and GPT3 results with various input encodings for different task categories, under random split (§4.1). Both models show improved results when encoded with instructions, comparing relative gains indicated in the 'avg' columns (in percentage compared to 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.

	leave-one-category		leave-one-dataset-		leave-one-task	
evaluation set $\mathcal{T}_{\text{unseen}}$ →	AG	QG	QASC	Quoref	Winogrande AG	QASC QG
No instructions	11	6	37	10	11	20
PROMPT+DEFINITION	18	10	43	**39**	11	22
PROMPT+POS. EXAMP.	18	**20**	47	33	16	55
PROMPT+DEFINITION+POS. EXAMP.	20	17	47	36	13	51
Full instructions	19	17	**51**	37	19	56
- NEG. EXAMP.	**21**	19	50	37	**27**	**57**

Table 4: BART generalization under various leave-one-out splits (§4.1). Encoding instructions improve cross-task generalization across all settings. All numbers are ROUGE-L.

BART and GPT3 show little improvements on *verification* (VF) and *minimal modifications* (MM) categories , respectively, which is reminiscent of the negative results in Efrat and Levy (2020). 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.

We conduct a survey among human annotators to find out the value of instruction elements to humans. Interestingly, we observe that human judgments about the importance of the fields (Appendix C.3; Table 13) are quite similar to their contributions to the model performance. 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., on both models PROMPT + DEFINITION has the highest score on CF category in Table 3).

5.2 Generalization in Leave-one-out Splits

Table 4 reports cross-task generalization results of the BART model under leave-one-out splits (§4.1). For leave-one-category, we evaluate BART for two categories (*answer-generation* and *question-generation*), which are not observed during training. For leave-one-dataset, we evaluate BART for tasks of the two datasets (QASC and Quoref), where no task from these datasets are observed in training. For leave-one-task, we evaluate BART for two tasks (Winogrande answer generation and QASC question generation). We report results with a few main encodings.

The results indicate that BART benefits from instructions in generalizing to new tasks, regardless of task splits – confirming our earlier findings for the random split setting (§5.1). This is particularly interesting for leave-one-category-out setup since the trained model can generalize to new tasks even when it is not exposed to any tasks in that particular semantic category. Note that the absolute values, across different encodings, are lower than the numbers in Table 3 which is likely due to the difficulty of this setup compared to the random split.

error type	GPT3	BART
generates a nonsensical/vague question	4	47
explains the question after generating it	6	0
generates a yes/no question	12	4
generates generic questions independent of the given context	6	0

Table 5: Percentage of errors on QASC QG task. The numbers do not sum to 100 since the error types are not mutually exclusive.

5.3 Generalization vs. Size of Observed Tasks

Fig. 2b shows how the number of observed tasks affects cross-task generalization. We fix 6 tasks (one from each category) as our evaluation tasks. For supervision, we random sample 5, 15, 37, 55 tasks $\mathcal{T}_{\text{seen}}$ (each point in the figure is averaged over 5 random subsamples.)

The figure shows that for the NO-INSTRUCTION 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.

5.4 Estimated Upperbounds from Task-specific Models

For each task, we obtain a task-specific model by training BART separately on each task's annotated training data – i.e., the conventional supervised setting. We evaluate these task-specific models to obtain an loose estimate of upperbounds for each task.

NATURAL-INSTRUCTIONS has a wide margin to be solved. On average, task-specific models score 66% which is considerably higher than our BART models' generalization to the same unseen tasks (35%; Table 3). This indicates that there is considerable room for improving generalization-based models that use instructions.

Table 5 shows the breakdown of most common error types for the QASC question generation task by analyzing 30 errors (more error analyses can be found in Appendix C.2; Table 11).

5.5 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). 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. The results in Table 6 indicate that GPT3 leads to higher performance with our encoding (2nd row) compared to raw instructions (first row). Weak performance of LMs on raw instructions aligns with (Efrat and Levy, 2020)'s finding that "language model performs poorly".

	Quoref	MCTaco	CosmosQA	QASC
raw instructions	12.5	5.00	6.9	3.7
our schema	25.8	42.6	17.7	51.3

Table 6: Comparing performance of GPT3 on raw instructions vs. our encoding. All numbers are ROUGE-L.

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.

6 Related Works

Instructions in NLP. Previous work in NLP has studied "instructions" (Goldwasser and Roth, 2014) in different domains such as robotic instructions (Shridhar et al., 2020; Stepputtis et al., 2020), database commands (Kim et al., 2020), programming instructions (Lin et al., 2018; Shao and Nakashole, 2020), *inter alia*. Such instructions are inherently different from ours; they are often short and intended to be mapped to symbolic forms (e.g., SQL commands). Conversely, our instructions are intended to describe general NLP tasks and have no underlying grammar.

Similar to ours, recent works study the impact of language instructions in modifying a model behavior (Hase and Bansal, 2021; Ye and Ren, 2021; Gupta et al., 2021; Zhong et al., 2021). 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 these instructions were originally targeted for laypeople. The closest work to ours is Efrat and Levy (2020) who also examine models' ability to follow the crowdsourcing instructions that were used to build existing datasets. Different from this work, we map the instructions to a unified schema (§3.1) which makes them more structured. Additionally, we decompose the original crowdsourcing instructions (shown to human workers) into self-contained tasks (§3.2) which leads to more informative and concise instructions allowing models to focus on each annotation step separately. Finally, having collected a dataset of 60+ tasks, we are able to experiments for cross-task generalization by training models on a subset of our tasks and measuring generalization to the remaining ones (§5). Two concurrent work similar to ours are Wei et al. (2021); Anonymous (2022) which explore a goal similar to ours. Unlike our elaborate instructions, these works were conducted on much short language prompts. Additionally, since we represent the instructions in a structured format, we are able to ablate various elements of the instructions and quantify their contributions.

Our work is closely related to the recent literature in prompting LMs (Schick and Schütze, 2020; Le Scao and Rush, 2021; Reynolds and McDonell, 2021; Tam et al., 2021). Such prompts are often overly-simple and do not include detailed descriptions for complex tasks. In contrast, instructions encode richer information about the task definition, things to avoid, and how to correct bad examples. Therefore, we view our evaluation setup in NATURAL-INSTRUCTIONS as a strict generalization of the few-shot prompting paradigm.

Beyond conventional multi-task learning. Multi-task learning is a long-standing goal for AI (Caruana, 1997) and has led to successful model that can support a wider range of tasks (McCann et al., 2018; Khashabi et al., 2020; Aghajanyan et al., 2021; Ye et al., 2021; Raffel et al., 2020). 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 (1st column of Table 2a). We augment this setup by introducing natural language instructions which enables our models to bridge to tasks that were not seen during training.

7 Conclusion

In this paper, 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 evaluate 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.

Future extensions. The observations made in §5.3 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 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).

Acknowledgements

We thank OpenAI for providing academic access to the GPT3 API, the authors who generously shared their dataset templates/annotations with us, Matt Peters and the Aristo team at AI2 for helpful input, and the Beaker team for their support with experiments. We also thank Nicholas Lourie for the discussions that inspired this work.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

Ali M Ali. 1981. The use of positive and negative examples during instruction. *Journal of instructional development*, 5(1):2–7.

Anonymous. 2022. Multitask prompted training enables zero-shot task generalization. In *Submitted to The Tenth International Conference on Learning Representations*. Under review.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Peter Clark, Oren Etzioni, Tushar Khot, Daniel Khashabi, Bhavana Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, et al. 2020. From 'f'to 'a'on the ny regents science exams: An overview of the aristo project. *AI Magazine*, 41(4):39–53.

Pradeep Dasigi, Nelson F Liu, Ana Marasovic, Noah A Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of EMNLP-IJCNLP*, pages 5927–5934.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of NAACL*, pages 2368–2378.

Avia Efrat and Omer Levy. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine learning*, 94(2):205–232.

Tanmay Gupta, A. Kamath, Aniruddha Kembhavi, and Derek Hoiem. 2021. Towards general purpose vision systems. *ArXiv*, abs/2104.00743.

Peter Hase and Mohit Bansal. 2021. When can models learn from explanations? a formal framework for understanding the roles of explanation data. *arXiv preprint arXiv:2102.02201*.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of EMNLP-IJCNLP*, pages 2391–2401.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of NAACL*, pages 252–262.

Daniel Khashabi, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. Learning what is essential in questions. In *Proceedings of CoNLL*, pages 80–89.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UnifiedQA: crossing format boundaries with a single qa system. In *Proceedings of EMNLP: Findings*, pages 1896–1907.

Daniel Khashabi, Gabriel Stanovsky, Jonathan Bragg, Nicholas Lourie, Jungo Kasai, Yejin Choi, Noah A Smith, and Daniel S Weld. 2021. GENIE: A leaderboard for human-in-the-loop evaluation of text generation. *arXiv preprint arXiv:2101.06561*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *Proceedings of AAAI*.

Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to sql: Where are we today? *Proceedings of the VLDB Endowment*, 13(10):1737–1750.

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? In *Proceedings of NAACL-HLT*, pages 2627–2636.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of ACL*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Kevin Lin, Oyvind Tafjord, Peter Clark, and Matt Gardner. 2019. Reasoning over paragraph effects in situations. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 58–62.

Winston Lin, Roman Yangarber, and Ralph Grishman. 2003. Bootstrapped learning of semantic classes from positive and negative examples. In *Proceedings of ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data*, volume 1, page 21.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D Ernst. 2018. Nl2bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of LREC*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

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

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. *arXiv preprint arXiv:2102.07350*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI*.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*, pages 7881–7892.

Yutong Shao and Ndapandula Nakashole. 2020. Chartdialogs: Plotting from natural language instructions. In *Proceedings of ACL*, pages 3559–3574.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF*, pages 10740–10749.

Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. 2020. Language-conditioned imitation learning for robot manipulation tasks. *arXiv preprint arXiv:2010.12083*.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Proceedings of NeurIPS*, 32.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew Peters. 2020. Learning from task descriptions. In *Proceedings of EMNLP*, pages 1361–1375.

Hong Xuan, Abby Stylianou, Xiaotong Liu, and Robert Pless. 2020. Hard negative examples are hard, but useful. In *European Conference on Computer Vision*, pages 126–142. Springer.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp. *arXiv preprint arXiv:2104.08835*.

Qinyuan Ye and Xiang Ren. 2021. Zero-shot learning by generating task-specific adapters. *arXiv preprint arXiv:2101.00420*.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. “going on a vacation” takes longer than “going for a walk”: A study of temporal commonsense understanding. In *Proceedings of EMNLP-IJCNLP*, pages 3354–3360.

Supplemental Material

A Datasets and their Templates

A.1 Division of Crowdsourcing Instructions into Minimal Tasks

Fig. 6 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. 7 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.

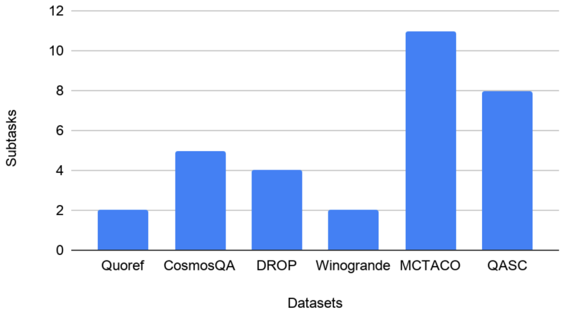


Figure 7: Variations in the number of subtasks

A.2 Analysis of Crowdsourcing Templates

We analyzed crowdsourcing templates of 6 datasets: CosmosQA (Huang et al., 2019), DROP (Dua et al., 2019), 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 collection of more templates.

Size of the instructions. We observe significant variation in size across the 6 datasets (Fig. 9). 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.

Number of positive/negative examples. Variation in the occurrence of POSITIVE and NEGATIVE Examples across datasets has been illustrated in Fig. 8. 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.

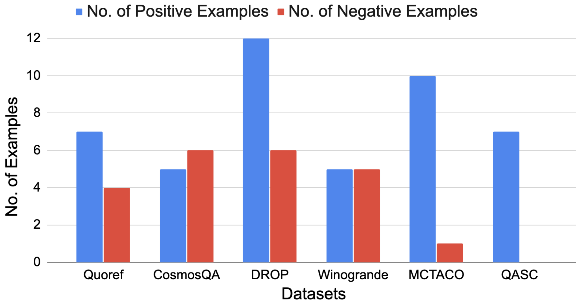


Figure 8: Variation in the number of positive and negative examples

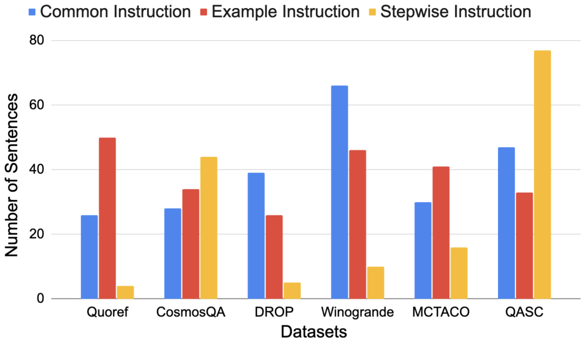
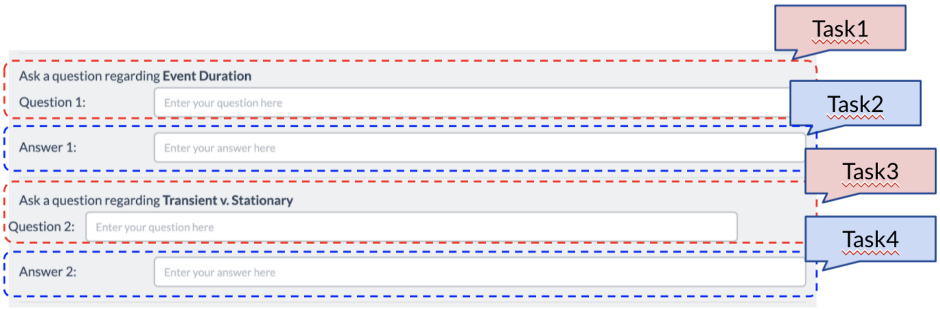


Figure 9: Variation in the number of sentences in the crowdsourcing instructions across datasets

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



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 6: Dividing a data creation task into multiple subtasks for the MC-TACO dataset.

datasets. Finally, Quoref is the only dataset to provide SUGGESTIONS along with the REASONS associated with the NEGATIVE Examples.

A.3 Qualitative Analysis

Writing Style. There exists significant variation in writing style across Instructions of the 6 datasets. For instance, though DROP, Quoref and QASC have the common objective of fooling an AI model, the instructions are stated differently across them. 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 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. To help you do so, we provided an AI system running in the background that will try to answer the questions you write. You can consider any question it can answer to be too easy. However, please note that the AI system incorrectly answering a question does not necessarily mean that it is good."* In QASC, variations are as follows: *"Two AI systems will try to answer your question. Make sure you fool at least on 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 which 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 relevant to Quoref and DROP, but is not mentioned in their respective instructions.

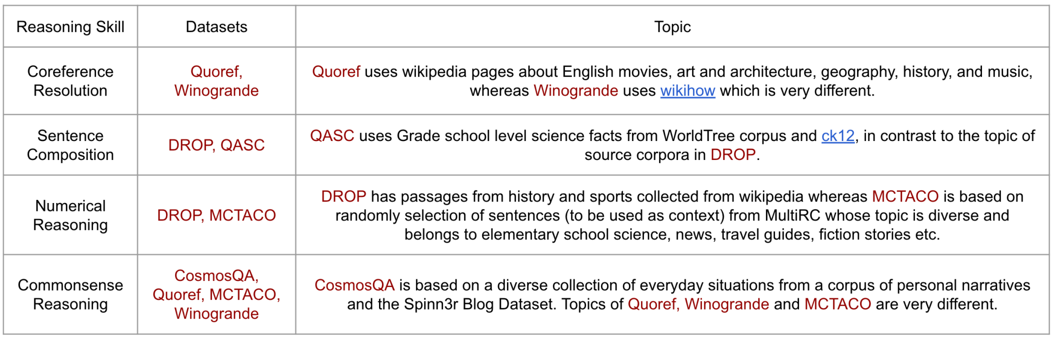
Topic. Fig. 10 illustrates some examples where the reasoning skill associated with the datasets is the same, but the topic varies. The experience gained creating data for one topic may help with understanding instructions and creating data for another dataset with the same underlying reasoning skill.

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 12 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.

Associated reasoning skill. Finally there are similarities among datasets in terms of their underlying skill requirements. Fig. 11 illustrates datasets clustered based on similarity in their associated reasoning class.

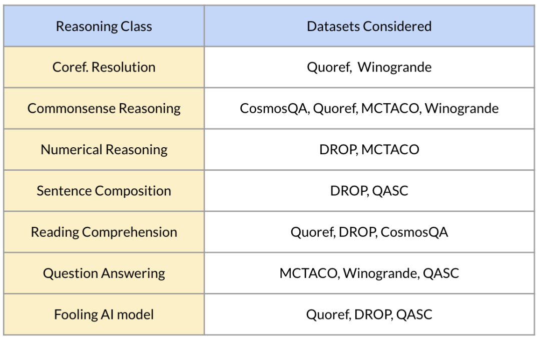
A.4 Data Curation Effort

Table 7 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 9 shows the detailed structure of tasks in NATURAL-INSTRUCTIONS. Fig. 13 shows examples of four different tasks in NATURAL-INSTRUCTIONS.



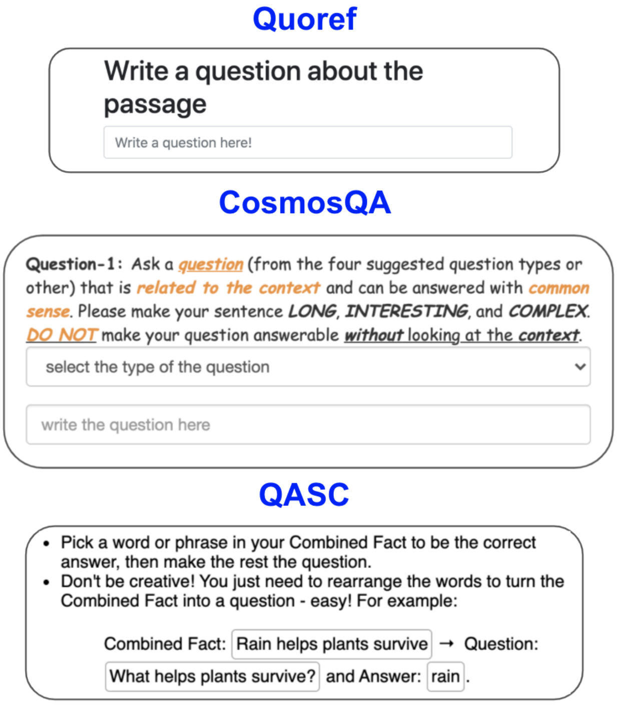
Reasoning Skill	Datasets	Topic
Coreference Resolution	Quoref, Winogrande	Quoref uses wikipedia pages about English movies, art and architecture, geography, history, and music, whereas Winogrande uses wikihow which is very different.
Sentence Composition	DROP, QASC	QASC uses Grade school level science 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 10: Variation in topics



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 11: Variation in reasoning skills



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 12: Variation in Task Specification: Quoref contains a single line instruction whereas the CosomosQA contains a detailed instruction. QASC on the other hand, contains examples along with instruction.

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 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 issue 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

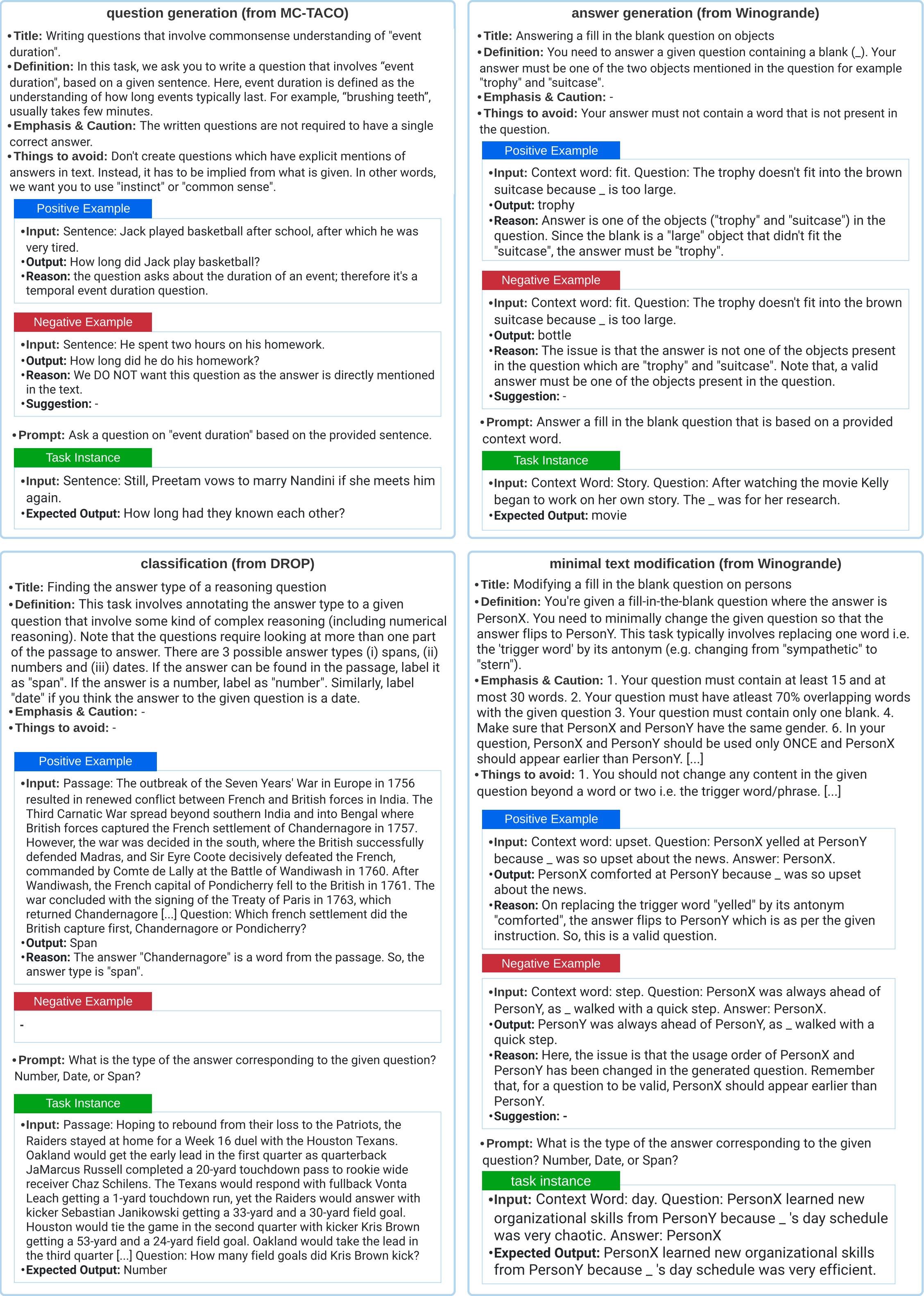
Table 7: Steps taken to curate each task in NATURAL-INSTRUCTIONS and their estimated times.

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
num of positive examples	4.9
num of negative examples	2.2

Table 8: Statistics of NATURAL-INSTRUCTIONS

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
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
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 9: Detailed set of tasks included in Natural-Instructions



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 13: Examples from NATURAL-INSTRUCTIONS. Each task follows the schema provided in Fig. 4.

B Building Baselines for NATURAL-INSTRUCTIONS

In this section, we provide several details on the baselines included in our work.

B.1 Encoding of the instructions

According to our schema (§3.1), each instruction I_t for the t-th task is a set that contains the following fields:

$$I_t = \left\{ I_t^{\text{title}}, I_t^{\text{def.}}, I_t^{\text{avoid}}, I_t^{\text{emph.}}, I_t^{\text{prompt}}, I_t^{\text{pos. ex.}}, I_t^{\text{neg. ex.}} \right\}$$

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

$$\begin{aligned} enc(I_t, x) := & \texttt{input} : x \\ & \texttt{output} :\text{''} \end{aligned} \tag{1}$$

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

$$\begin{aligned} enc(I_t, x) := & \texttt{Prompt} : I_t^{\text{prompt}} \\ & \texttt{input} : x \\ & \texttt{output} :\text{''} \end{aligned} \tag{2}$$

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

$$\begin{aligned} enc(I_t, x) := & \text{``}\texttt{Definition} : I_t^{\text{def.}} \\ & \texttt{Prompt} : I_t^{\text{prompt}} \\ & \texttt{input} : x \\ & \texttt{output} :\text{''} \end{aligned} \tag{3}$$

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

FULL INSTRUCTIONS encoding. This encoding contains all the instruction content:

$$\begin{aligned} enc(I_t, x) := & \text{``}\texttt{Definition} : I_t^{\text{def.}} \\ & \texttt{Prompt} : I_t^{\text{prompt}} \\ & \texttt{Things to Avoid} : I_t^{\text{avoid.}} \\ & \texttt{Emphasis\&Caution} : I_t^{\text{emph.}} \\ & \text{``}\texttt{NegativeExample1}- \\ & \quad \texttt{input} : I_t^{\text{pos. ex.}}(\texttt{input}) \\ & \quad \texttt{output} : I_t^{\text{pos. ex.}}(\texttt{output}) \\ & \quad \texttt{reason} : I_t^{\text{pos. ex.}}(\texttt{reason}) \\ & \texttt{NegativeExample2}- \\ & \quad \ldots \\ & \text{``}\texttt{PositiveExample1}- \\ & \quad \texttt{input} : I_t^{\text{pos. ex.}}(\texttt{input}) \\ & \quad \texttt{output} : I_t^{\text{pos. ex.}}(\texttt{output}) \\ & \quad \texttt{reason} : I_t^{\text{pos. ex.}}(\texttt{reason}) \\ & \texttt{PositiveExample2}- \\ & \quad \ldots \\ & \texttt{input} : x \\ & \texttt{output} :\text{''} \end{aligned} \tag{4}$$

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).

$$\begin{aligned} enc(I_t, x) := & \quad \texttt{input} : I_t^{\text{pos. ex.}}(\texttt{input}) \\ & \quad \texttt{output} : I_t^{\text{pos. ex.}}(\texttt{output}) \\ & \quad \ldots \\ & \texttt{input} : x \\ & \texttt{output} :\text{''} \end{aligned} \tag{5}$$

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

C Analysis on Baseline Results

C.1 An Ablation Study of Instructional Elements

We conduct an ablation study with GPT3 on 3 distinct tasks (answer generation from Winogrande; question generation from QASC; verifying temporal reasoning category of a given question from MC-TACO). Table 10 (top) shows the effect of eliminating various fields in the encoding while Table 10 (bottom) indicates the gains from adding each field. Table 14 further details the ablation results across various tasks. The overall observation is that GPT3 benefits the most from *positive examples*, mildly from *definition*, and deteriorates with *negative examples*. We hypothesize it is easier for GPT3 to mimic the patterns in positive examples while utilizing *negative examples* requires deeper understanding.

encoding ↓	*avg score (R-L)*	*relative change (%)*
all instructions	0.18	-
- definition	0.18	1.9%
- emphasis	0.20	15.1%
- things to avoid	0.19	9.4%
- things to avoid and emphasis	0.19	5.7%
- things to avoid, emphasis, definition	0.19	9.4%
- things to avoid, emphasis, def., prompt	0.18	1.9%
- examples' explanations	0.20	11.3%
- negative examples	0.23	28.3%
- positive examples	0.13	-24.5%
- positive examples, negative examples	0.14	-22.6%

encoding ↓	*avg score (R-L)*	*relative change (%)*
prompt	0.13	-
+ definition	0.16	23.1%
+ emphasis	0.14	10.3%
+ things to avoid	0.15	15.4%
+ things to avoid and emphasis	0.15	15.4%
+ negative examples (no explanation)	0.11	-12.8%
+ positive examples (no explanation)	0.21	59.0%
+ positive or negative ex. (no explanation)	0.20	53.8%
+ definition, positive ex. (no explanation)	0.20	56.4%
+ definition, negative ex. (no explanation)	0.13	0.0%

Table 10: An ablation study of the different fields included in NATURAL-INSTRUCTIONS based on GPT3. This model benefits the most from POSITIVE examples and the least from NEGATIVE examples.

C.2 Error Analysis

We conduct error analysis on 3 distinct tasks (answer generation from Winogrande; question generation from QASC; incorrect answer generation for MC-TACO event-duration questions). We randomly select 30 samples from each of these tasks and categorize the associated errors. Table 11 summarizes our analysis. We observe that GPT3 mainly suffers from generating redundant content and ignoring the instructions provided. On the other hand, our BART model provides more control, however, it fails to generate proper output.

Here are example generations from each category of errors:

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 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?

error type	GPT3	BART
does not follow instruction and generate an invalid question	14	8
generates a nonsensical/vague question	4	47
copies the given fact or a subset of it	8	3
explains the question after generating it	6	0
generates a yes/no question	12	4
generates candidate answers as output	4	0
generates questions whose answer does not exist	4	3
generates generic questions independent of the given context	6	0

Table 11: Percentage of errors on QASC QG task (§C.2). The numbers do not sum to 100 since the error types are not mutually exclusive.

	BART							GPT3						
task category →	QG	AG	CF	IAG	MM	VF	avg	QG	AG	CF	IAG	MM	VF	avg
NO INSTRUCTION	26	6	0	21	33	7	13	-	-	-	-	-	-	-
PROMPT	27	22	7	22	34	**9**	20	33	32	14	13	**73**	16	30
+DEFINITION	35	24	50	**25**	36	7	30↑ (+50)	36	35	40	14	70	16	35↑ (+17)
+THINGS TO AVOID	33	24	4	24	**58**	**9**	25↑ (+25)	28	33	11	16	68	14	28↓ (-7)
+EMPHASIS	38	23	16	**26**	49	3	26↑ (+30)	29	28	18	16	72	16	30
+POS. EXAMP.	53	22	14	**25**	17	7	23↑ (+15)	**43**	49	29	21	70	**36**	41↑ (+37)
+DEFINITION+POS. EXAMP.	51	23	**56**	**25**	37	6	33↑ (+65)	**43**	50	**45**	**23**	70	32	**44**↑(+47)
+POS, NEG EX+ EXPLAN.	50	21	27	25	50	7	30 ↑ (+50)	32	19	8	12	61	13	24↓(-20)
POS. EXAMP.	**55**	6	18	**25**	8	6	20	30	32	15	16	68	23	31↑(+3)
FULL INSTRUCTION	46	25	52	25	35	7	32↑ (+60)	33	18	8	12	60	11	24↓(-20)
- EXAMPLES	40	24	36	25	55	8	31↑ (+55)	31	34	39	14	69	13	33↑(+10)
- NEG. EXAMP.	52	**30**	50	**25**	47	8	**35**↑ (+75)	**43**	**54**	44	21	70	32	**44**↑(+47)

Table 12: Full BART and GPT3 results with various input encodings for different task categories, under random split (§4.1). Both models show improved results when encoded with instructions, comparing relative gains indicated in the 'avg' columns (in percentage compared to PROMPT encoding.) Category names: QG: Question Generation, AG: Answer Generation, CF: Classification, IAG: Incorrect Answer Generation, MM: Minimal Text Modification, VF: Verification.

C.3 User Study to Find Important Task-Specific Instruction Fields

Table 12 shows full BART and GPT3 results for all encodings. In addition to the quality estimation (§3.2.2) of NATURAL-INSTRUCTIONS, we ask our quality assessment annotators to also specify which instruction fields help them understand the task and answer prompts. For each of the 12 tasks in our evaluation set, we ask– *Which instruction field helps you the most to understand the task and answer questions and why? Remember, on removing this field significant major information should get lost.* We compile these results category-wise, and present them in Table 13. In particular, there are two tasks (i) Classification (CF) and (ii) Minimal Text Modification (MM) for which humans find only a single instruction field to be important. Interestingly, this is compatible with our results in Table §3). We find that models also find the same fields to be most important, as evinced in Table §3), where performance of models with these fields are higher than the rest.

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

Table 13: User study to find out importance of various fields in our instruction schema (§C.3). Interestingly, our model also finds DEFINITION and THING TO AVOID helpful for Classification and Minimal Text Modification task, respectively (Table §3).

encoding ↓	*answer generation (Winogrande)*		*question generation (QASC)*		*verifying temporal reasonng of a given question (MCTACO; classification)*		*avg score across the three tasks*		*relative gain by adding the instructions (%)*	
	R-L	*BT*	*R-L*	*BT*	*R-L*	*BT*	*R-L*	*BT*	*R-L*	*BT*
all instructions	0.14	-1.59	0.29	-0.95	0.10	-1.62	0.18	-1.39	-	-
- definition	0.12	-1.58	0.31	-0.93	0.11	-1.65	0.18	-1.39	1.9%	0.0%
- emphasis	0.11	-1.53	0.4	-0.78	0.1	-1.66	0.20	-1.32	15.1%	4.6%
- things to avoid	0.15	-1.58	0.32	-0.89	0.11	-1.67	0.19	-1.38	9.4%	0.5%
- things to avoid and emphasis	0.13	-1.56	0.31	-0.95	0.12	-1.67	0.19	-1.39	5.7%	-0.5%
- things to avoid, emphasis, definition	0.11	-1.60	0.36	-0.99	0.11	-1.67	0.19	-1.42	9.4%	-2.4%
- things to avoid, emphasis, definition, prompt	0.11	-1.60	0.32	-0.91	0.11	-1.67	0.18	-1.39	1.9%	-0.5%
- examples' explanations	0.14	-1.55	0.36	-0.94	0.09	-1.7	0.20	-1.40	11.3%	-0.7%
- negative examples	0.12	-1.57	0.35	-0.85	0.21	-1.62	0.23	-1.35	28.3%	2.9%
- positive examples	0.07	-1.57	0.27	-0.98	0.06	-1.69	0.13	-1.41	-24.5%	-1.9%
- positive examples, negative examples	0.07	-1.59	0.31	-0.96	0.03	-1.69	0.14	-1.41	-22.6%	-1.9%
encoding ↓	*answer generation (Winogrande)*		*question generation (QASC)*		*verifying temporal reasonng of a given question (MCTACO; classification)*		*avg score across the three tasks*		*relative gain by adding the instructions (%)*	
	R-L	*BT*	*R-L*	*BT*	*R-L*	*BT*	*R-L*	*BT*	*R-L*	*BT*
prompt	0.08	-1.62	0.28	-0.94	0.03	-1.82	0.13	-1.46	-	-
+ definition	0.06	-1.58	0.37	-0.84	0.05	-1.79	0.16	-1.40	23.1%	3.9%
+ emphasis	0.06	-1.54	0.35	-0.96	0.02	-1.85	0.14	-1.45	10.3%	0.7%
+ things to avoid	0.05	-1.62	0.31	-0.85	0.09	-1.69	0.15	-1.39	15.4%	5.0%
+ things to avoid and emphasis	0.06	-1.64	0.33	-0.81	0.06	-1.71	0.15	-1.39	15.4%	5.0%
+ negative examples (no explanation)	0.03	-1.63	0.25	-1.03	0.06	-1.79	0.11	-1.48	-12.8%	-1.6%
+ positive examples (no explanation)	0.12	-1.69	0.37	-1.04	0.13	-1.9	0.21	-1.54	59.0%	-5.7%
+ positive examples, negative examples (no explanation)	0.11	-1.66	0.31	-1.03	0.18	-1.69	0.20	-1.46	53.8%	0.0%
+ definition, positive examples (no explanation)	0.11	-1.77	0.37	-1.09	0.13	-1.9	0.20	-1.59	56.4%	-8.7%
+ definition, negative examples (no explanation)	0.06	-1.57	0.25	-1.07	0.08	-1.92	0.13	-1.52	0.0%	-4.1%

Table 14: Detailed results of the encoding ablation performed on three distinct subtasks.