Cross-Task Generalization via Natural Language Crowdsourcing Instructions

Anonymous *ACL submission

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. Despite the success of the conventional supervised learning on individual datasets, such models 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 (input-output pairs). The instructions are obtained from crowdsourcing instructions used to create existing NLP datasets and mapped to a unified schema. Using this meta-dataset, we measure cross-task generalization by training models on *seen* tasks and measuring generalization to the remaining *unseen* ones. 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 (19% better for models utilizing instructions). These models, however, are far behind an estimated performance upperbound, indicating significant room for more progress in this direction.[1]

1 Introduction

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). 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)). In this paper, we study if models can generalize to *unseen* tasks given their crowdsourcing instructions (Fig.1).

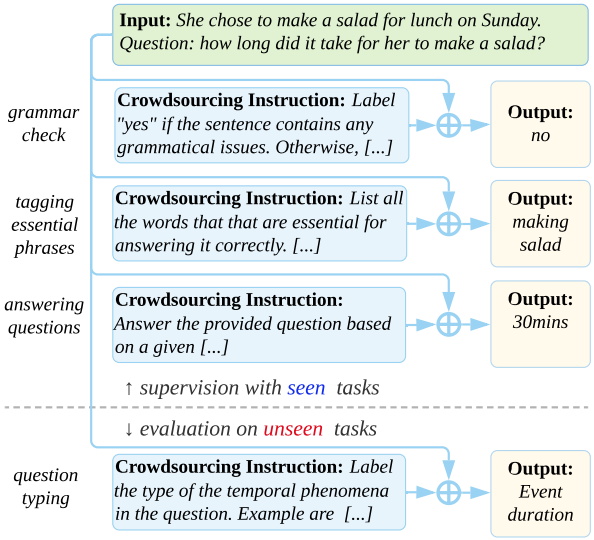


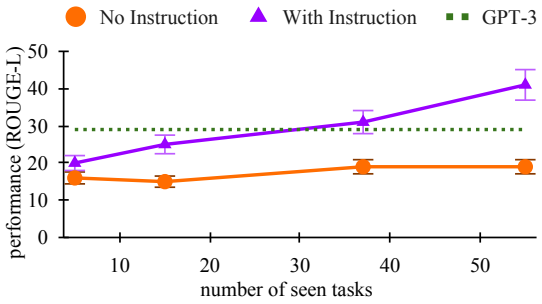
Figure 1: We construct the NATURAL-INSTRUCTIONS dataset from crowdsourcing instructions and instances of different NLP datasets. We study if models can learn from *seen* tasks and generalize to *unseen* tasks given their natural crowdsourcing instructions.

We build NATURAL-INSTRUCTIONS, a dataset consisting of *natural* crowdsourcing instructions for various tasks and their instances. Training on *seen* tasks $\mathcal{T}_{\text{seen}}$ 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 $\mathcal{T}_{\text{unseen}}$, we evaluate if the model can perform *unseen* tasks solely from

[1]Dataset is available at https://git.io/JXg9Z

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	$x \rightarrow y$ where: $(x, y) \in (X^{\text{test}}, Y^{\text{test}})$	$(x, I_t) \rightarrow y$ 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 are 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, models with no access to the instructions show no sign of improved generalization. Details in §6.3.

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

their instructions and without any task-specific labeled data (Table 2a; right). In contrast to the instance-level generalization (Table 2a; left), our model uses instruction as additional input, and evaluations are done on tasks that were not observed in the training stage.

We compile NATURAL-INSTRUCTIONS from task instructions written by researchers for crowdsourcing 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 (§6). Importantly, LMs can generalize better to unseen tasks if they observe more tasks in training (Fig.2b). 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 upper-bounds (6.4), encouraging the community to work on this challenging problem.

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 Related Works

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., 2021; 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., 2021; Sanh et al., 2021) 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 §A.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, 2020; Reynolds and McDonell, 2021; Liu et al., 2021). 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; 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 enable our models to bridge to tasks that were not seen during training.

3 Defining Cross-Task Generalization

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

Task-specific models. Standard supervised learning algorithms use task-specific labeled instances to learn a mapping from input x to output y: $M(x) = y$ for $(x, y) \in (X_t^{\text{train}}, Y_t^{\text{train}})$ and is evaluated on the test instances of the same (or similar) task $(X_t^{\text{test}}, Y_t^{\text{test}})$. We refer to this as the *instance-level* generalization (Table 2a; left).

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 collect NATURAL-INSTRUCTIONS (§4) to study this question: 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 a *task*-level generalization (Table 2a; right).

4 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 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 (§4.1) and then describe how existing datasets (their crowdsourcing templates) are mapped into our schema (§4.2).

4.1 Instruction Schema

Instructions 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.) 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:

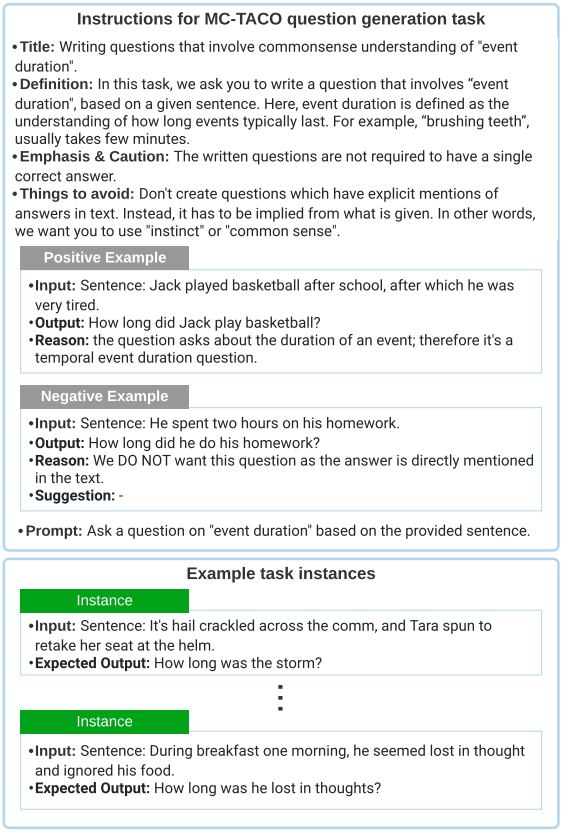


Figure 3: An example from our dataset. Note that it follows the schema provided in Fig.4. See Fig .13 for more examples.

- 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 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.

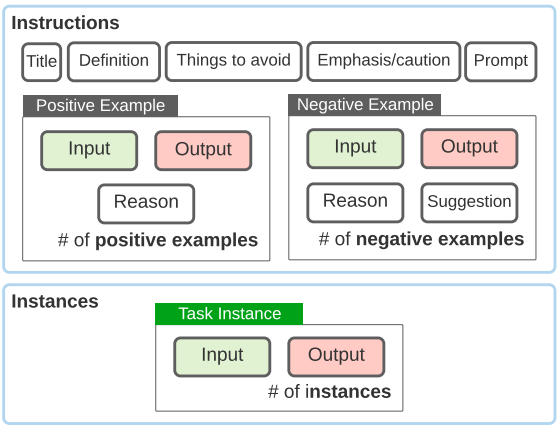


Figure 4: The schema used for representing instruction in NATURAL-INSTRUCTIONS (§4.1), shown in plate notation.

- 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.

4.2 Constructing NATURAL-INSTRUCTIONS

4.2.1 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., 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

[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.

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 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 8 in Appendix.

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.

steps and generate multiple subtasks that are minimal and standalone.[3] Table 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 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 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).

4.2.2 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 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 §5.1). 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.

4.2.3 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

[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 6).

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 7).

5 Problem Setup and Models

Here we define different cross-task generalization settings (§5.1) and the models (§5.2).

5.1 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]

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.

5.2 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: $M : enc(I_t, x) \rightarrow y$.

[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 $\mathcal{T}_{\text{unseen}}$.

Prompt : I_t^{prompt}
Definition : $I_t^{\text{Definition}}$
Things to Avoid : $I_t^{\text{avoid.}}$
Emphasis&Caution : $I_t^{\text{emph.}}$
NegativeExample1–
input : $I_t^{\text{pos. ex.}}$, output : $I_t^{\text{pos. ex.}}$, reason : $I_t^{\text{pos. ex.}}$
PositiveExample1–
input : $I_t^{\text{pos. ex.}}$, output : $I_t^{\text{pos. ex.}}$ reason : $I_t^{\text{pos. ex.}}$
input : x, output :"

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

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 (§4.1). 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.

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) POSITIVE EXAMPLES, and (6) 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., 2021). Refer to Appendix C for our study on the impact of other instruction elements.

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, the input is encoded using different instruction elements, trained on all $\mathcal{T}_{\text{seen}}$ tasks, and evaluated on $\mathcal{T}_{\text{unseen}}$ (§5.1).

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 $\mathcal{T}_{\text{unseen}}$ (§5.1) using the encoding introduced earlier.

6 Experiments

Evaluation metrics. We treat all of our tasks as text generation problems and evaluate them with automated evaluation metrics for text generation.

model ↓	task category →	QG	AG	CF	IAG	MM	VF	avg
BART	NO INSTRUCTION	26	6	0	21	33	7	13
	PROMPT	27	22	7	22	34	9	20 (+7)
	+DEFINITION	35	24	50	**25**	36	7	30 (+17)
	+THINGS TO AVOID	33	24	4	24	58	9	25 (+12)
	+POS. EXAMP.	53	22	14	**25**	17	7	23 (+10)
	POS. EXAMP.	**55**	6	18	**25**	8	6	20 (+7)
	FULL INSTRUCTION	46	**25**	**52**	25	35	7	**32** (+19)
GPT3	FULL INSTRUCTION	33	18	8	12	**60**	**11**	24 (+11)

Table 3: Cross-task generalization with various input encodings under random split (§5.1). Models show improved results when provided with instructions. The numbers in parenthesis indicate absolute gains compared to 'NO INSTRUCTIONS' baseline. BART archives 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).

leave-one-x split →	x = category		x = dataset		x = 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
FULL INSTRUCTIONS	**19**	17	**51**	37	**19**	**56**

Table 4: BART generalization under various leave-one-out splits (§5.1). Encoding instructions improve cross-task generalization across all settings. All numbers are ROUGE-L.

In particular, we use ROUGE-L (Lin, 2004) to automatically evaluate the generated outputs.[7]

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.

6.1 Generalization Under Random Split

Table 3 reports the results of the BART model trained on seen tasks and evaluated on a random split of the tasks (§5.1) with a variety of encodings that incorporate different elements of the instructions (§5.2).[8] For comparison, we evaluate GPT3 which uses no fine-tuning, unlike BART that is fine-tuned with the $\mathcal{T}_{\text{seen}}$ tasks.

Instructions benefit cross-task generalization. Table 3 (avg column) shows that instructions improve the generalization of BART. It additionally shows that encoding more elements of the instructions achieves better results than just using PROMPT or POSITIVE EXAMPLE. Specifically, FULL INSTRUCTIONS results in +19% gains over a model that is not using instructions for BART and +11% for GPT3. In comparison to GPT3, the BART model trained on seen tasks achieves stronger performance despite being $\times 1k$ smaller than GPT3.

Results on task categories. Table 3 shows the performance of our models on different task categories. 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.

6.2 Generalization in Leave-one-out Splits

Table 4 reports cross-task generalization results of the BART model under leave-one-x splits (§5.1).

[7] 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.

[8] See Appendix C for an ablation.

For x = category, we evaluate two categories (*answer-generation*, *question-generation*) which are not observed during training. For x = dataset, we evaluate on all tasks of the two datasets (QASC, Quoref) where no task from these datasets are observed in training. For x = task, we evaluate two tasks (Winogrande answer generation, QASC question generation). We report results with several 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 (§6.1). This is particularly interesting for x = category since the trained model can generalize to the tasks of a particular semantic category, without being exposed to it. 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.

6.3 Generalization vs. Number of Seen Tasks

Fig.2b compares the impact of the number of seen tasks for cross-task generalization. For supervision, we randomly sample a few tasks as $\mathcal{T}_{\text{seen}}$ 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.

6.4 Analyses

Upperbound: Task-specific Models For each task, we obtain a task-specific model (§ 3) 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). This indicates that there is considerable room for improving generalization-based models that use instructions.

Case Study: 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. In a cases study via several models (Table 5) we observe that they all works better *without* (w/o) negative examples, contrary to the previously-observed benefits of other instructional elements (e.g., definition, pos. 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.

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 5: Effect of excluding negative examples from FULL INSTRUCTION encoding. Negative instructions are surprisingly difficult for the models to learn from.

Perceived Impact of Instruction Elements We conduct a survey among 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 (Appendix C.4; Table 15) and their contributions to the model performance (Table 3). 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 14).

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.

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.

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.

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*.

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.

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*.

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 pre-training 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.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

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*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

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*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

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.

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. 9 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. 6 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.

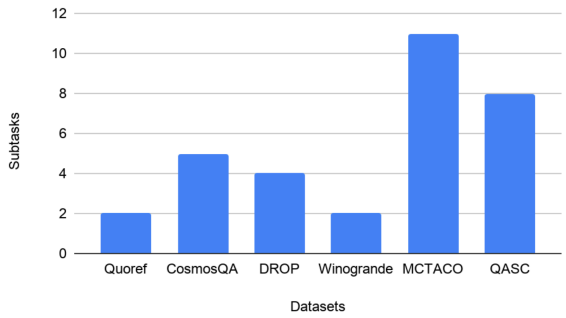


Figure 6: Variations in the number of subtasks

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 the collection of more templates.

Size of the instructions. We observe significant variation in size across the 6 datasets (Fig. 8). 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. 7. 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.

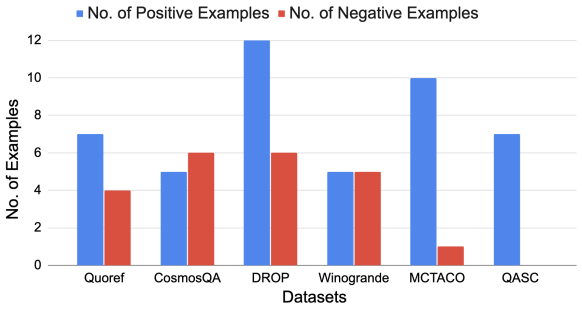


Figure 7: Variation in the number of positive and negative examples

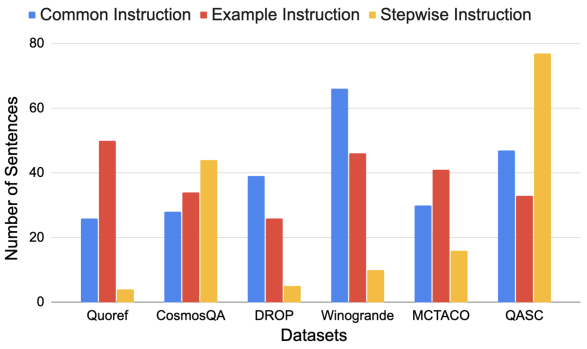


Figure 8: Variation in the number of sentences in the crowdsourcing instructions across datasets

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.

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 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 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 6 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 8 shows the detailed structure of tasks in NATURAL-INSTRUCTIONS. Fig. 13 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 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 6: Steps taken to curate each task in NATURAL-INSTRUCTIONS and their estimated times.

Figure 9: 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 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

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 7: Statistics of NATURAL-INSTRUCTIONS

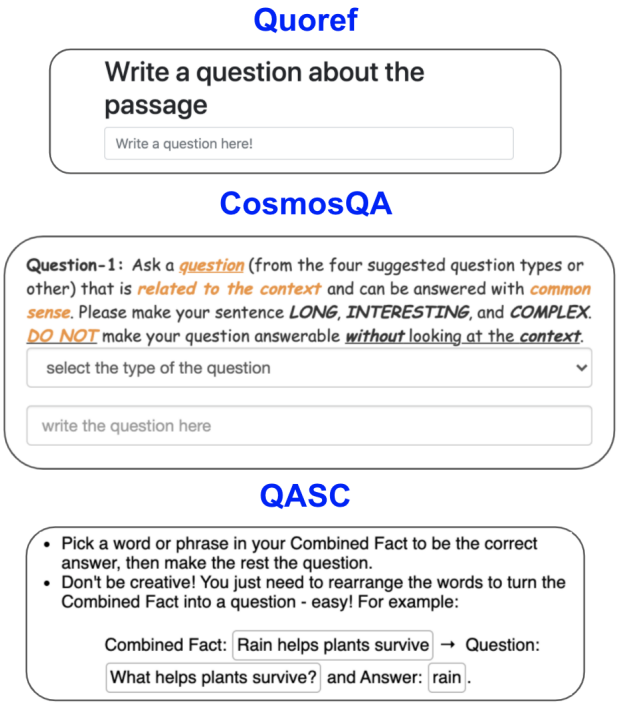


Figure 12: Variation in Task Specification: Quoref contains a single line instruction whereas the CosmosQA contains a detailed instruction. QASC on the other hand, contains examples along with instruction.

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 8: Detailed set of tasks included in NATURAL-INSTRUCTIONS

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

A.5 Qualitative Comparison to PromptSource

We provide a comparison between our proposed dataset and PromptSource (Sanh et al., 2021). 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 broader variety of tasks. For example, tasks for chaining facts (task 38; Table 8), question typing (task 27; Table 8) or detecting inappropriate content (task 22; Table 8) are unique additions in NATURAL-INSTRUCTIONS. Additionally, since our instructions were originally written by various researchers targeted for 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 9). 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 (§6).

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 9: Comparing zero-shot performance of GPT3 on our instructions vs. PromptSource. The instructions curated in this work, despite being lengthier, lead to higher performance.

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 10: Qualitative comparison of the task instructions for several shared tasks among NATURAL-INSTRUCTIONS and PromptSource (Sanh et al., 2021).

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 (§4.1), 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 Comparison to Raw Instructions

We seek to understand the value of breaking the tasks into sub-tasks and mapping them into our proposed schema (§4.2). 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 11 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 11: Comparing GPT3 performance 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.

C.2 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 12 (top) shows the effect of eliminating various fields in the encoding while Table 12 (bottom) indicates the gains from adding each field. 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 12: 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.3 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 13 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 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?

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 13: Percentage of errors on QASC QG task (§C.3). 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 14: Full BART and GPT3 results with various input encodings for different task categories, under random split (§5.1). 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.4 User Study to Find Important Task-Specific Instruction Fields

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 15. In particular, there are two tasks Classification (CF) and Minimal Text Modification (MM) for which humans find only a single instruction field to be important. We find that models also find the same fields to be most important, as evinced in Table §3), where the performance of models with these fields is higher than the rest. Interestingly, this is similar to the patterns observed in the model performance (Table §3).

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 15: User study to find out importance of various fields in our instruction schema (§C.4). Human annotators (similar to model predictions in (Table 3) find DEFINITION and THING TO AVOID helpful for Classification and Minimal Text Modification task, respectively.