Improving Cross-task Generalization of Unified Table-to-text Models with Compositional Task Configurations

Jifan Chen[1*] **Yuhao Zhang**[2] **Lan Liu**[2] **Rui Dong**[2]
Xinchi Chen[2] **Patrick Ng**[2] **William Yang Wang**[2] **Zhiheng Huang**[2]
[1]The University of Texas at Austin
[2]AWS AI Labs
jfchen@cs.utexas.edu {yhzhang, liuall, ruidong}@amazon.com
{xcc, patricng, wyw, zhiheng}@amazon.com

Abstract

There has been great progress in unifying various table-to-text tasks using a single encoder-decoder model trained via multi-task learning (Xie et al., 2022). However, existing methods typically encode task information with a simple dataset name as a prefix to the encoder. This not only limits the effectiveness of multi-task learning, but also hinders the model's ability to generalize to new domains or tasks that were not seen during training, which is crucial for real-world applications. In this paper, we propose *compositional task configurations*, a set of prompts prepended to the encoder to improve cross-task generalization of unified models. We design the task configurations to explicitly specify the task type, dataset name, as well as its input and output types. We show that this not only allows the model to better learn shared knowledge across different tasks at training, but also allows us to control the model by composing new configurations that apply novel input-output combinations in a zero-shot manner. We demonstrate via experiments over ten table-to-text tasks that our method outperforms the UnifiedSKG baseline by noticeable margins in both in-domain and zero-shot settings, with average improvements of +0.5 and +12.6 from using a T5-large backbone, respectively.

1 Introduction

Table-to-text tasks, such as table-based question answering (Pasupat and Liang, 2015; Herzig et al., 2020), summarization (Parikh et al., 2020), or fact verification (Chen et al., 2019), are of high interest to the NLP community and have been applied in many real-world applications. Traditionally, these tasks have been studied individually, with methods commonly optimized for one or a few tasks (Liu et al., 2021; Shi et al., 2021). However, with the recent popularity of pre-trained transformer models (Raffel et al., 2020; Lewis et al., 2020; Xue et al., 2021), there has been a paradigm towards unifying multiple NLP tasks with a single encoder-decoder model (Khashabi et al., 2020; Sanh et al., 2022). More recently, UnifiedSKG (Xie et al., 2022) extended this paradigm to table-to-text tasks by flattening the structured input (e.g., tables) into text format, and unifying all tasks with a T5 model (Raffel et al., 2020). By training the model over 21 datasets with structured input, it has established new state-of-the-art results for most of these tasks.

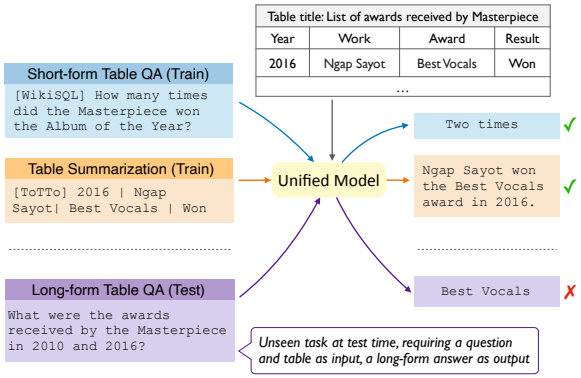

Figure 1: An example of unifying different tasks with a single encoder-decoder model with dataset name as a prefix. The model is trained on short-form table QA and table summarization tasks, and tested on a new long-form table QA task. As there is a mismatch between the training and test tasks, the model is unable to generalize.

Despite the success, existing work often rely on a simple trick to encode task information: the name of the dataset is often used as a prefix to the encoder at both training and test time. We argue that this overly simplified design has at least two major limitations. First, since no detailed information about the task is provided, any sharable knowledge between tasks is learned in a latent manner. Second, with this design, models are trained and evaluated on their abilities to solve specific *datasets*, rather than *tasks*. As a result, we may see substantial performance degradation when we apply the model to an unseen task at test time.

[*]Work done during an internship at AWS AI Labs.

Findings of the Association for Computational Linguistics: ACL 2023, pages 5523–5539
July 9-14, 2023

Figure 1 illustrates the aforementioned limitations: a unified model (such as UnifiedSKG) is trained on *short-form table QA* (Zhong et al., 2017) and *table-based summarization* (Parikh et al., 2020), and we want to test the trained model on *long-form table QA* (Nan et al., 2022a), where the model should take a question and a table as input and output an abstractive sentence as the answer. As there is no way to instruct the model about the information of the new task, the model can only make an educated guess by generating the most plausible text "Best Vocals" according to the training datasets, which fails to serve as a good long-form answer. We therefore argue that it is critical to test a table-to-text model's *cross-task generalizability*, which is captured in neither the training methods nor the evaluation setup in existing work.

In this paper, we propose the use of *compositional task configurations*, a set of text prompts prepended to the encoder to improve the cross-task generalizability of unified table-to-text models. For a given task, we design its configuration prompt to be compositional, describing the task type, dataset name, input type, and output type. This design offers at least two key advantages. First, the task configurations explicitly inform the model what is shared between different tasks. For example, the model is able to learn from the configurations that table-based fact checking and table-based QA share the same inputs but different outputs. Second and more importantly, using task configurations allows us to have explicit control over the model's behaviors. For the example in Figure 1, we can now compose a new configuration for long-form table QA at test time to instruct the model to first produce a set of relevant cells and synthesize them to produce a long-form answer, which is within the capabilities of the two training tasks. We discuss this further in the next section.

Our evaluation focuses model's cross-task generalizability. Specifically, we train our model on 5 table-to-text datasets and test it on an additional set of 5 new datasets that cover either a new domain of an existing task or a new task of which the capabilities can be composed by the ones learned through the 5 training datasets. Our main findings can be summarized as follows:

- Our method not only outperforms the strong UnifiedSKG baseline consistently on the 5 in-domain datasets, but also demonstrates much stronger cross-task generalization.
- In zero-shot evaluation on the 5 test-only tasks, our model outperforms UnifiedSKG by a substantial margin of +6.5 and +12.6 average scores from using T5-base and T5-large, respectively. Notably, we find that in zero-shot evaluation on FeTaQA (Nan et al., 2022a), a long-form table QA task, while the baseline completely fails with a 0.6 F1 score, our method leads to much better generalization, achieving a 21.2 F1 score.
- We also show that using the compositional task configurations allows the model to output *supporting table cells* that supplement its final prediction in a zero-shot manner. Human evaluation of the generated supporting cells for the TabFact dataset reveals that more than 80% of the generated cells have high relevance to the task.

2 Method & Tasks

Prompting is a natural and feasible way to impose explicit control over the behaviors of pre-trained language models (Wei et al., 2021; Chung et al., 2022; Sanh et al., 2022). In this work, we implement the task configurations as prompts of an encoder-decoder model. Each task configuration contains the following four aspects: task type, input type, output type, and dataset name. The task type is the end goal of a task, e.g., QA and summarization, as shown in Figure 2. Input and output types specify the inputs of the encoder and the outputs of the decoder of table-to-text models, respectively. These types can be compositional, for example, both long-form and short-form table QA in Figure 2 require the decoder to output a set of relevant cells and the final answer. The dataset name specifies the dataset used for training. As different datasets can share the same task type, input and output types, we assume that the model is able to learn the shared and the unique knowledge across different datasets by adding the dataset names as configurations. When testing the model on a new dataset, we can simply omit the dataset name since it is not trained.

One of the major advantages of having explicit task configurations is that it enables the model to learn the mapping between a configuration and its behavior. At test time, we can compose a new set of configurations which suits best for an unseen task using the trained configurations. Figure 2 demon-

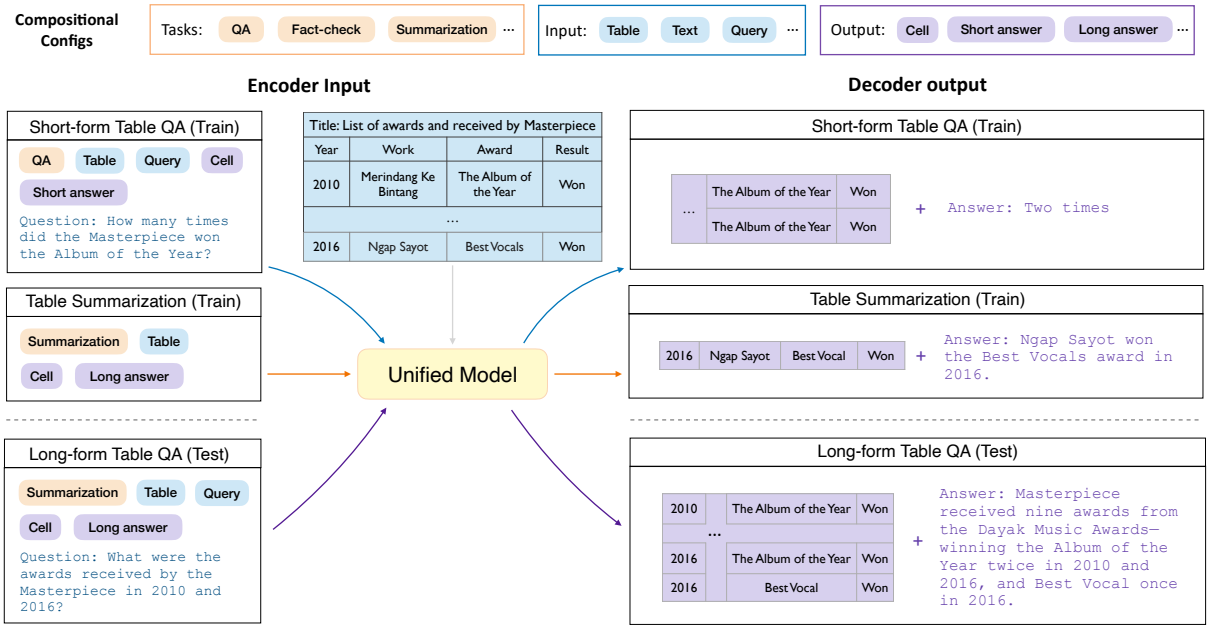

Figure 2: Our method allows us to reformulate the unseen task of long-form table QA as a query-based summarization task by composing a new configuration using the existing ones from the two training tasks. At test time, this composed configuration allows the model to first identify relevant cells in the table based on the input question using the cell generation skills learned from short-form table QA, and then generates a long-form answer by utilizing the knowledge acquired from table summarization. For all three tasks, the model also takes the linearized table as input; for simplicity, we omit the linearized table and dataset name configuration in the figure.

strates that by training on short-form table QA, the model learns the ability to generate a set of relevant table cells according to the question and then derive the answer based on those cells. By training on table summarization, the model learns to produce a summary based on a set of table cells.[1] At test time, by reformulating the long-form table QA as a query-based summarization task, our model is able to first generate a set of relevant cells (learned through short-form table QA) and then synthesize those cells to yield a long-form answer (learned from table summarization).

Note that our method is an efficient extension to the original UnifiedSKG model. It only requires a small input prefix, comprising less than 5% of the total sequence length, making it flexible for generalization to more tasks and datasets.

2.1 Datasets and Task Configurations

A detailed list of our datasets with their task information is shown in Table 1. We consider 5 datasets as in-domain datasets for both training and testing: WIKISQL (Zhong et al., 2017), WIKITQ (Pasupat and Liang, 2015), SQUAD (Rajpurkar et al., 2016), TOTTO (Parikh et al., 2020) and TABFACT (Chen et al., 2019). For SQUAD and TOTTO, since no official test set is released, we follow UnifiedSKG (Xie et al., 2022) and report results on the official development sets. In addition, we consider 5 datasets for test only: NQ-TABLES (Kwiatkowski et al., 2019; Herzig et al., 2021), HYBRIDQA (Chen et al., 2020), TAT-QA (Zhu et al., 2021), FETAQA (Nan et al., 2022b) and FEVEROUS (Aly et al., 2021).

The test-only evaluation setup aims to assess the effectiveness of our method in enabling the model to generalize to unseen tasks with new compositional configurations, as well as to a new dataset with existing configurations. Specifically, we test if the model can benefit from a combination of input configurations for tasks such as HYBRIDQA, TAT-QA, and FEVROUS, which involve both passages and tables as inputs, despite the model is only trained on one or the other during training. Similarly, we examine if the configuration for FETAQA, a combination of WIKISQL and TOTTO as shown in Figure 2, allows for explicit control over the model's behaviors, resulting in improved generalization. Finally, we assess the

[1] Conventionally, when using ToTTo, it is common to feed the cells to the encoder and decode the summary. To make the task more compatible with other table-to-text tasks, we feed the relevant cells as starting inputs to the decoder and the model's loss is not calculated on those cells.

	Dataset	Task Type	Input	Output	Unseen?
Train + Test	WIKISQL	QA (table)	query + table	cells + short-form answer	-
	WIKITQ	QA (table)	query + table	short-form answer	-
	SQUAD	QA (text)	query + passage	short-form answer	-
	TOTTO	Summarization	table	cells + summary	-
	TABFACT	Fact-check	query + table	binary answer	-
Test-only	NQ-TABLES	QA (table)	query + table	short-form answer	-
	HYBRIDQA	QA (hybrid)	query + passage + table	short-form answer	✓
	TAT-QA	QA (hybrid)	query + passage + table	short-form answer	✓
	FETAQA	QA (abstractive)	query + table	cells + long-form answer	✓
	FEVEROUS	Fact-check	query + passage + table	binary answer	✓

Table 1: Datasets and tasks considered in our experiments. Tasks that have an input/output combination unseen at training time are marked with ✓in the "Unseen?" column. We include detailed statistics and task configurations applied for each dataset in Appendix A & B due to space limitation.

model's ability to generalize to a new dataset, NQ-TABLES, which has the same configurations as WIKISQL and WIKITQ.

For all of these datasets, we linearize the tables following the strategy used in UnifiedSKG (Xie et al., 2022). By inserting several special tokens like vertical bars to indicates the boundaries between cells and rows, a table can be linearized as: "Headers: $h_1|...|h_m$, Row 1: $c_{11}|...|c_{1m}$ ... Row n: $c_{n1}|...|c_{nm}$". Here, h_i denotes the ith header of a table and c_{ij} denotes the cell content in the ith row and the jth column. For simplicity, we fix the order of the task configurations to be task type, dataset name, input type, and output type. We prepend the task configuration to the original input of a dataset and feed it to the model. To make our input and output better aligned with the configurations, we also introduce some special markups to separate different parts of inputs and outputs: Figure 3 illustrates the actual model's input and output of the short-form table QA example from Figure 2. See Appendix B for inputs and outputs constructed for all of the datasets and Appendix A for preprocessing details of each dataset.

Encoder Input

```
[Task: QA] [Input: query] [Input: table]
[Output: cells] [Output: short answer] [query]
How many times did the Masterpiece won the Album
of the year? [/query] [table] Headers: Year |
Work | Award | Result | Row1: 2010 | Masterpiece
| Best New Artist | Nominated | Row2 … [/table]
```

Decoder Output

```
[cell] 2016 | Merindang Ke Bintang | the Album
of the year | Won | 2016 | Ngarap Ka Nuan Kikal
Pulai | the Album of the year | Won | [/cell]
[answer] Two times [/answer]
```

Figure 3: An example of the input and output given to the model. After training, the model is able to establish a correspondence between the task configurations and the input/output format. Dataset name is omitted here for simplicity.

3 Experiments

Experimental settings We evaluate our method by following the experimental setup shown in Table 1. We follow the experimental settings of UnifiedSKG (Xie et al., 2022) and use T5 (Raffel et al., 2020) as the backbone of our table-to-text model. Our implementation is based on the publicly released code of UnifiedSKG which is developed based on the `transformer` library (Wolf et al., 2019). To balance the size of different datasets during training, we use the temperature upsampling method proposed in the original T5 paper and set the temperature to 2. For all experiments, we use a batch size of 128 and AdamW (Loshchilov and Hutter, 2018) as the optimizer with the initial learning rate set to 5e-5. We limit the maximum length of the input, including task configuration and the actual inputs, to be 1080 sentence-piece tokens. We train both the T5-base and T5-large models on the training set for 20 epochs and we use early stopping with the patience set to 2. We use deepspeed (Rasley et al., 2020) to reduce the GPU memory loads when training the T5-large model. The approximate GPU hours for T5-base and T5-large are 250 and 650 respectively on A100 GPUs with 40G memory.

Baseline We mainly compare our method against UnifiedSKG (Xie et al., 2022), a strong baseline that was shown to achieve state-of-the-art results on many table-to-text tasks via multi-task training. In UnifiedSKG, for each task, a dataset name is prepended to the encoder during multi-task fine-

	Zero-shot Test-only Tasks						
	Models	NQ-TABLES BLEU	FETAQA EM	HYBRIDQA EM	TAT-QA EM	FEVEROUS Acc.	Avg. –
T5-base	Single Task	*51.6*	*29.9*	*54.3*	*34.5*	*81.3*	*50.3*
	UnifiedSKG	37.8	0.6	22.5	18.2	67.5	29.3
	Task Configs (Ours)	**39.4**	**21.0**	**28.9**	**20.8**	**68.9**	**35.8**
T5-large	Single Task	*52.2*	*33.0*	*56.6*	*36.2*	*82.1*	*52.0*
	UnifiedSKG	42.6	0.7	34.1	20.4	41.4	27.8
	Task Configs (Ours)	**43.0**	**25.2**	**38.0**	**20.8**	**75.0**	**40.4**
	In-domain Tasks						
	Models	WIKISQL EM	WIKITQ EM	TOTTO BLEU (dev.)	SQUAD EM (dev.)	TABFACT Acc.	Avg. –
T5-base	Single Task	*81.6*	*35.8*	*36.7*	*83.6*	*76.1*	*62.8*
	UnifiedSKG	82.9	41.1	37.2	82.5	77.1	64.2
	Task Configs (Ours)	**83.5**	**42.5**	**37.4**	**83.0**	**77.5**	**64.8**
T5-large	Single Task	*85.5*	*43.4*	*37.8*	*86.0*	*81.0*	66.7
	UnifiedSKG	86.0	48.5	**38.7**	86.1	83.0	68.5
	Task Configs (Ours)	**86.7**	**50.0**	**38.7**	**86.2**	**83.3**	**69.0**

Table 2: Zero-shot and in-domain performance of our proposed method (Task Configs) vs. baselines for both T5-base and T5-large. Here, "EM" denotes exact match accuracy. For all tasks we also include the results from *single-task finetuning* as references. Note that a direct comparison with single-task result is not fair as the latter has access to training data in each task. Higher numbers among our method and UnifiedSKG are highlighted in bold.

tuning as a pseudo-task configuration. For fair comparisons, we re-trained UnifiedSKG models on the five in-domain datasets by using the authors' implementation.[2]

Evaluation For the in-domain tasks, we simply train on their training sets and evaluate on their test sets. For the test-only tasks, we evaluate our method in two settings: 1) a **zero-shot** setting, where we directly apply the model trained on in-domain datasets and use a new set of task configs designed for each test dataset; 2) a **few-shot** setting, where for each test dataset, we further fine-tune the model using n randomly sampled training examples (where n is small). Since we observed that the few-shot training is unstable and heavily depends on the sampled examples, we report average performance from 5 different random seeds (each with a different set of few-shot examples).

[2]We in fact found that our version of the UnifiedSKG model fine-tuned over the five in-domain datasets outperforms the original authors' version on several datasets, establishing a more competitive baseline. For example, our version achieves 82.9 on WikiSQL and 77.1 on TabFact, whereas the original model by Xie et al. (2022) achieves 81.9 and 71.2, respectively.

4 Results

4.1 Main Results

We present the in-domain and zero-shot evaluation results for all datasets in Table 2 and the few-shot evaluation results for OOD datasets in Figure 4. We have the following observations:

First, **using compositional task configs shows much stronger performance on zero-shot datasets unseen at training time** (Table 2). For example, the UnifiedSKG baseline fails to generalize at test time to FETAQA, a long-form table QA task where the input is a question-table pair and the output is a long-form abstractive answer. This is due to the baseline model having no clue of what format of output should be produced and what knowledge learned through the training datasets should be leveraged for this task. In contrast, by reformulating the long-form table QA as a query-based summarization task and composing the input configurations to be *table* and *query* as well as the output task configs to be *relevant cell* and *summary*, our method notably improves the zero-shot performance and closes the gap between zero-shot and single-task finetuning results. Note that among the zero-shot datasets, NQ-TABLES represents a new dataset for an existing task (short-form table QA), whereas others represent new tasks unseen at train-

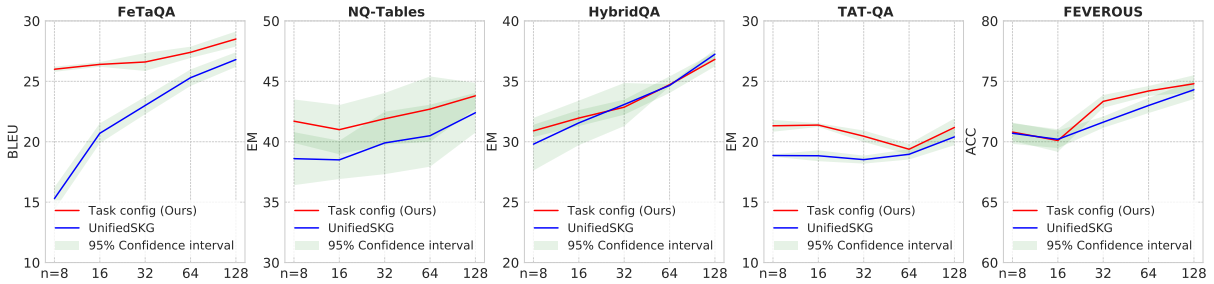

Figure 4: Few-shot performance on out-of-domain tasks, with results aggregated from 5 random training runs. Our method demonstrates better few-shot performance than the baseline in most settings, with the gap reduced as more supervised examples are added.

ing. Nevertheless, we found the improvements to be consistent for all zero-shot datasets, with average improvements of +6.5 and +12.6 for base and large models, respectively.

Second, in most cases, **using compositional task configs consistently improves the in-domain performance** over the UnifiedSKG baseline and single-task training (Table 2). The observation is consistent for both base and large model sizes, with average improvements of +0.6 and +0.5 over UnifiedSKG, respectively. The improvement over single-task fine-tuning is even greater for all datasets. One explanation to this improvement is that by adding task configs we explicitly encourage the model to learn the shared knowledge between different tasks and datasets.

Last but not the least, in few-shot evaluation (as depicted in Figure 4), we find that **using task configurations has improved few-shot learning performance for most test-time tasks**. Overall the difference between our method and the UnifiedSKG baseline is particularly notable when the number of supervised examples (n) is small; and the performance gap diminishes for HybridQA, TAT-QA, and FEVEROUS as n gets larger. One possible explanation is that the prior captured by the task configurations during training is not closely aligned with these three datasets, when n getting larger, the prior introduced by the task configurations is gradually overridden by knowledge learned from the supervised data.

4.2 Ablation of Task Configs at Training time

The impact of individual configurations on model performance was evaluated by removing one configuration at a time during training. The results, presented in Table 3, indicate that the removal of the **output type** resulted in the largest performance drop, as the model was only able to guess the desired output type based on learned parameters. The removal of the **input type** had the least impact on performance. This is likely due to the fact that learning the representation of the two input types was not difficult for the model, and explicitly informing the model about the input type does not provide significant benefit, as observed in the previous section. The removal of the **dataset name** also results in a performance drop, particularly on the NQ-Tables dataset, indicating that even when the task type, input, and output are the same, including the dataset name helps the model learn dataset-independent knowledge more effectively. The removal of the **task type** results in a complete failure on the FeTaQA dataset, demonstrating that all configurations are needed to produce the correct form of output. A more detailed discussion of these findings can be found in Section 7.

Models	FeTaQA	NQ	Hybrid	TAT	FEVR	Avg.
Full configs	21.0	39.4	28.9	20.8	68.9	**35.8**
- dataset	21.2	36.1	25.3	19.8	68.1	34.1
- task type	0.4	38.4	28.3	21.3	68.5	31.4
- input	20.3	39.5	29.1	19.4	68.3	35.3
- output	17.3	34.8	17.9	16.1	68.2	30.9

Table 3: Ablation of task configurations during training. We only report zero-shot performance here. We see removing any one of the configurations causes a performance drop on average.

4.3 Ablation of Task Configs at Test time

While our method demonstrates much stronger zero-shot task performance, it is crucial to understand the extent to which input and output configurations contribute to this success, particularly for tasks involving hybrid input or output types that are not present during training. To examine the contributions of input configurations, we remove each configuration from the hybrid tasks (HybridQA,

TAT-QA and FEVEROUS) at test time, with results shown in Table 4. We found that deleting either of the input configurations results in a performance drop in most cases, and the drop is quite notable when the table and passage input configurations are removed together. This suggests that the input configuration captures useful priors about the input during training, and **different configurations can be combined to yield better performance in the zero-shot transfer to hybrid tasks.** We also observe a similar trend in Figure 5 where we test the model performance by removing the *cell* output configurations for FeTaQA (thereby skipping cell generation). We see that in both zero-shot and few-shot settings, model performance drops by a large margin. This shows not only that the model can generate different outputs by combining the output configurations, but also that it can better utilize the prior captured by the configurations to improve task performance.

Models	HybridQA	TAT-QA	FEVR	Avg.
Full Configs	28.9	**20.8**	**68.9**	**39.5**
- input:passage	28.8	19.8	68.6	39.1
- input:table	**29.4**	20.3	66.7	38.8
- input:all	27.3	19.3	66.1	37.6

Table 4: Ablation of the input configurations at test time. The inputs of the three datasets include both table and passage. We show the model can achieve better performance by combining two input configurations.

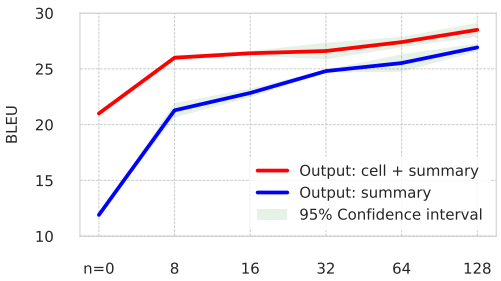

Figure 5: Ablation of the output configurations on FeTaQA. The model has better cross-task performance by generating a set of relevant cells in both zero-shot and few-shot settings. "n=0" denotes zero-shot setting.

4.4 Human Evaluation of Generated Cells

In addition to the strong task generalizability, a key advantage of applying the proposed task configurations to table-to-text tasks is that we can modify the task configurations to output more results for improved explainability, even when such a configuration combination is never seen at training time. An example of this is for the table-based fact verification task, TabFact, instead of only generating a binary label, we can extend the output configuration to include a *cell* component that can serve as supporting evidence of the binary prediction. We include two examples of this setting in Figure 6.

To understand how well our model can generate supporting cells without ever being trained for it, we conduct a human evaluation over 50 randomly sampled outputs from the TabFact dataset. We ask annotators to manually evaluate the generated cells based on their level of **relevance** and **completeness**. Relevance denotes the usefulness of the generated cells in verifying a claim (precision) and completeness refers to the extent to which all of the relevant cells are generated (recall). Detailed annotation instructions are shown in Appendix C. For each aspect, we ask the annotators to select between three labels that characterize its degree: "full", "partial" or "none". Three of the authors conduct the annotations, achieving 0.72 and 0.80 Fleiss Kappa (Fleiss, 1971) for relevance and completeness, respectively. We conduct majority vote to get the consensus label and the results are shown in Table 5. Overall we found that the model is able to generate cells with high relevance (with 72% examples being fully relevant generations), but struggle with full completeness (with 34% fully complete).

5 Related Work

Table-to-text tasks Table-based tasks, including table-based question answering (Pasupat and Liang, 2015; Zhong et al., 2017; Chen et al., 2020; Cheng et al., 2022; Zhao et al., 2022), table-based fact-checking (Chen et al., 2019; Aly et al., 2021), table summarization (Parikh et al., 2020; Suadaa et al., 2021; Moosavi et al., 2021), have gained increasing attention in recent years. A flurry of work using transformer-based structure explored modeling table structure via pretraining, for example, TabTransformer (Huang et al., 2020), VIME (Yoon et al., 2020), TABBIE (Iida et al., 2021), TaBERT (Yin et al., 2020), TUTA (Wang et al., 2021), TabT5 (Andrejczuk et al., 2022), and TableFormer (Yang et al., 2022).

Our work mainly focuses on table-to-text tasks but the ideal neural architecture for encoding table structures is not our focus. Instead, we emphasize multi-task knowledge-sharing (similar to UnifiedSKG (Xie et al., 2022)) and cross-task gen-

Caption: 1976 world junior figure skating championships				
rank	name	nation	points	places
1	sherri baier / robin cowan	canada	128.39	9
2	lorene mitchell / donald mitchell	united states	124.94	16
3	elizabeth cain / peter cain	australia	116.67	33
4	jana bláhová / ludek	czechoslovakia	113.74	36

Task configs:
[Task: Fact-checking] [Input: table] [Output: cells] [Output: binary answer]

Input claim:
the top 2 ranked team be in north America

Output:
[cell] 1 | sherri baier / robin cowan | canada | 2 | lorene mitchell / donald mitchell | united states [/cell] [answer] entailed [/answer]

Caption: Charl Schwartzel					
tournament	wins	top - 5	...	top - 25	events
masters tournament	1	1		2	4
us open	0	0		3	7
the open championship	0	0		4	9
pga championship	0	0		2	8
totals	1	1		11	28

Task configs:
[Task: Fact-checking] [Input: table] [Output: cells] [Output: binary answer]

Input claim:
there be a total of 28 event that make up the 4 tournament

Output:
[cell] 4 [/cell] [answer] refuted [/answer]

Figure 6: Two examples from the development set of TABFACT where we force the model to produce relevant cells by changing the output configurations. For the left example, the model produces a set of complete and relevant cells that help understand its "entailed" prediction. For the right example, the model is misled by the number "4" when generating the cells, and we are able to check that the final answer "refuted" is wrong.

	Full	Partial	None
Relevance	72%	10%	18%
Completeness	34%	48%	18%

Table 5: Human evaluation results of the zero-shot cell generation quality for the TABFACT task.

eralization in table-to-text tasks. Also, the proposed framework is capable of generalizing to a broader range of tasks and datasets.

Task unification There have been a vein of work that tries to solve various NLP tasks using a single model. This includes encoder-decoder models like T5 (Raffel et al., 2020), UnifiedQA (Khashabi et al., 2020), UnifiedQA2 (Khashabi et al., 2022), UnifiedSKG (Xie et al., 2022); decoder-only models driven by prompts, for example, GPT3 (Brown et al., 2020), Codex (Chen et al., 2021), PaLM (Chowdhery et al., 2022). Our work extends UnifiedSKG by using an encoder-decoder model as the backbone and designing prompts to encourage better knowledge sharing between different tasks and enable control over the model's behaviors.

Cross-task generalization with pretrained models Various efforts have been made to improve the ability of unified models to generalize to new tasks and datasets, including instruction-tuning using a wide range of natural language instructions (Chung et al., 2022; Sanh et al., 2022; Wei et al., 2021; Zhong et al., 2021), better design of prompts in zero-shot and few-shot setting (Wei et al., 2022; Zhou et al., 2022; Kojima et al., 2022). Our proposed method differs from instruction-tuning models like FLAN (Wei et al., 2021) in that we use a more symbolized prompt structure and it is possible to attribute cross-task generalizability to specific tasks and configurations. Also, instruction-tuning models like FLAN achieve behavior control and cross-task generalizability through costly large-scale instruction tuning. In contrast, our approach demonstrates that within a specific task domain with **limited datasets like table-to-text**, this can be achieved by utilizing a compositional prompt structure.

Our method is also relevant to Macaw (Tafjord and Clark, 2021), ProQA (Zhong et al., 2022b), and SchemaPro (Zhong et al., 2022a), which also utilize explicit task descriptions to facilitate knowledge sharing between various NLP tasks. Our work differs in two main aspects: (1) Our work focuses on compositional generalization at test time, examining whether the model can combine different configurations from multiple tasks during training to generalize to unseen tasks at test time. (2) Our work focuses on table-to-text tasks.

6 Conclusion

We introduced compositional task configurations for unified table-to-text models. Compared to existing unified encoder-decoder models that sim-

ply use dataset names as input prefix, compositional task configurations allow us to specify the task type, input, and output types at a finer level, which improve multi-task learning effectiveness and cross-task generalization. Further, we showed that our method allows fine-grained control over the model's generation at test time, enabling the model to generalize to unseen tasks and improving explainability via generating high-quality supporting table cells.

7 Limitations

Task configurations are entangled with the full model parameters. In our ablation study of task configurations at training time (Table 3), we see that when training without **task type**, the model fails to generalize to FeTaQA. Upon examining the model output, we find that although we change the output configuration to "long answer", the model still produces a short-form answer. This indicates that model behaviors are not always aligned with a single configuration, leading us to question the extent to which each individual configuration influences the model. In order to have better and more interpretable control over the models, one potential avenue for future research is to develop pluggable task configurations, where each configuration controls a more atomic function of the model and can be plugged, unplugged, and combined to yield different model behaviors.

Our exploration scope is limited to table-to-text tasks. Due to the constraints of the computational resources, we haven't explored joint training with a broader range of other NLP tasks. We think with some modifications, such as the inclusion of dataset domains in the configuration set, it would be possible to extend our approach to additional datasets and tasks.

8 Ethics Statement

The authors of this paper are committed to conducting research ethically. Data used in this work has been collected from public sources and used in accordance with all applicable laws and regulations. The only area of work that involves human annotation of data is described in Section 4.4, where authors of this paper annotated a group of samples for analyzing models' behaviors. We ensure that no external human subject was involved or harmed. In addition, this work uses language models, for which the risks and potential harms are discussed in numerous previous works (Bender et al., 2021; Weidinger et al., 2021). The authors strive to ensure that the research and its results do not cause harm.

9 Acknowledgement

We would like to thanks the anonymous reviewers for their insightful comments and suggestions.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In *Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER)*, pages 1–13, Dominican Republic. Association for Computational Linguistics.

Ewa Andrejczuk, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. 2022. Table-to-text generation and pre-training with tabt5. *arXiv preprint arXiv:2210.09162*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. HybridQA: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036, Online. Association for Computational Linguistics.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2022. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland. Association for Computational Linguistics.

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

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Jonathan Herzig, Thomas Mueller, Syrine Krichene, and Julian Eisenschlos. 2021. Open domain question answering over tables via dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TAPAS: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. 2020. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained representations of tabular data. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3446–3456, Online. Association for Computational Linguistics.

Daniel Khashabi, Yeganeh Kordi, and Hannaneh Hajishirzi. 2022. Unifiedqa-v2: Stronger generalization via broader cross-format training. *arXiv preprint arXiv:2202.12359*.

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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. Learning to reason for text generation from scientific tables. *arXiv preprint arXiv:2104.08296*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Nick Schoelkopf, Riley Kong, Xiangru Tang, et al. 2022a. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, Mutethia Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, Dragomir Radev, and Dragomir Radev. 2022b. FeTaQA: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13806–13814.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. 2021. Towards table-to-text generation with numerical reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1451–1465, Online. Association for Computational Linguistics.

Oyvind Tafjord and Peter Clark. 2021. General-purpose question-answering with macaw. *arXiv preprint arXiv:2109.02593*.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: tree-based transformers for generally structured table pre-training. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

Jason Wei, Maarten Bosma, Vincent 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. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

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

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 528–537, Dublin, Ireland. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. 2020. Vime: Extending the success

of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2856–2878, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Wanjun Zhong, Yifan Gao, Ning Ding, Zhiyuan Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022a. Improving task generalization via unified schema prompt. *arXiv preprint arXiv:2208.03229*.

Wanjun Zhong, Yifan Gao, Ning Ding, Yujia Qin, Zhiyuan Liu, Ming Zhou, Jiahai Wang, Jian Yin, and Nan Duan. 2022b. ProQA: Structural prompt-based pre-training for unified question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4230–4243, Seattle, United States. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Dataset Statistics and Preprocessing

The statistics and the licenses of the datasets we used in this paper are shown in Table 6. All datasets are English-based and all of them are based on Wikipedia except for TAT-QA which is based on financial reports. To fit the data into the encoder, for all datasets, we limit the max length of each cell to be 15 sentence-piece tokens. If the length (measured by sentence-piece tokens) of the linearized table is longer than 1024, we truncate random rows to reduce the table size.

The original annotation of WIKISQL dataset does not include the relevant cells. We extract the relevant cells by executing the accompanied SQL query annotations. In most cases, the relevant cells equal to the final answer annotations; for the rest of the cases, aggregations or numerical operations need to be run to obtain the final answer. During training, we also create another version of the WIKISQL dataset, in which we exclude the relevant cells and only use the final answer as supervision to improve output diversity. We use both versions at training time.

NQ-TABLES is a table-based QA dataset derived from the NaturalQuestions dataset (Kwiatkowski et al., 2019) and was originally released by Herzig et al. (2021). The original test set of NQ-TABLES contains 966 unique examples. In our experiments, to make the dataset more compatible with other table-based QA tasks, we evaluate on a customized version of NQ-TABLES where we only include an example if the answer is uniquely locatable as one or more table cells. This filtering step results in 549 unique triples of table, question and answers.

To make TOTTO more compatible with other table-to-text tasks, we feed the selected cells as inputs to the decoder, as we mentioned in Section 2. Also, we find it helpful to create a reversed version of the TOTTO dataset, where we treat the annotated summary and the table as input and let the model predict the relevant cells. We add both versions of TOTTO to the training of all models, including the baseline.

B Task Configurations Applied for Each Dataset

Below we list the task configurations applied to all datasets. For each dataset, we present input to the encoder and output from the decoder separately. For encoder, we include the template of the full input, including task configurations as well as how the dataset input is structured (with actual data replaced by "..."). For decoder, we include how we structure the annotated output during training and how we parse the output during testing.

	Dataset	Train	Dev	Test	License	Domain	Language
Train+Test	WIKISQL	56,355	8,421	15,878	BSD 3-Clause	Wikipedia	English
	WIKITQ	11,321	2,810	4,344	CC BY-SA 4.0	Wikipedia	English
	SQUAD	87,599	10,570	–	CC BY-SA 4.0	Wikipedia	English
	TOTTO	120,761	7,700	–	CC BY-SA 3.0	Wikipedia	English
	TABFACT	92,283	12,792	9,750	CC BY 4.0	Wikipedia	English
Test-only	NQ-TABLES	–	–	549	Apache License 2.0	Wikipedia	English
	HYBRIDQA	–	3,466	–	CC BY 4.0	Wikipedia	English
	TAT-QA	–	–	1,669	MIT License	Finance	English
	FETAQA	–	–	2,003	CC BY-SA 4.0	Wikipedia	English
	FEVEROUS	–	7,890	–	CC BY-SA 3.0	Wikipedia	English

Table 6: Dataset statistics of the datasets we used in the paper. Note that for the test-only datasets, except for few-shot experiments, only the test splits of the original datasets are used. For SQUAD, HYBRIDQA, FEVEROUS, and TOTTO, as no public test set is offered, we evaluate the model on the original development sets following UnifiedSKG (Xie et al., 2022). For NQ-TABLES, we use a modified version of it in our experiments as described in Appendix A.

B.1 WikiSQL

Encoder:

```
[Task: QA] [Dataset: WikiSQL] [Input: query] [Input: table] [Output: cells] [Output: short answer] [query] ... [/query] [table] ... [/table]
```

Decoder:

```
[cell] ... [/cell] [answer] ... [/answer]
```

B.2 WikiTQ

Encoder:

```
[Task: QA] [Dataset: WikiTQ] [Input: query] [Input: table] [Output: short answer] [query] ... [/query] [table] ... [/table]
```

Decoder:

```
[answer] ... [/answer]
```

B.3 SQuAD

Encoder:

```
[Task: QA] [Dataset: SQuAD] [Input: query] [Input: passage] [Output: short answer] [query] ... [/query] [passage] ... [/passage]
```

Decoder:

```
[answer] ... [/answer]
```

B.4 ToTTo

Encoder:

```
[Task: Summarization] [Dataset: ToTTo] [Output: cells] [Output: long answer]
```

Decoder:

```
[cell] ... [/cell] [answer] ... [/answer]
```

B.5 TabFact

Encoder:

```
[Task: Fact-checking] [Dataset: TabFact] [Input: query] [Input: table] [Output: binary answer] [query] ... [/query] [table] ... [/table]
```

Decoder:

```
[answer] ... [/answer]
```

B.6 NQ-Tables

Encoder:

```
[Task: QA] [Input: query] [Input: table] [Output: short answer] [query] ... [/query] [table] ... [/table]
```

Decoder:

```
[answer] ... [/answer]
```

B.7 HybridQA

Encoder:

```
[Task: QA] [Input: query] [Input: table] [Input: passage] [Output: short answer] [query] ... [/query] [table] ... [/table] [passage] ... [/passage]
```

Decoder:

```
[answer] ... [/answer]
```

B.8 TAT-QA

Encoder:

```
[Task: QA] [Input: query] [Input: table] [Input: passage] [Output: short answer] [query] ... [/query] [table] ... [/table] [passage] ... [/passage]
```

Decoder:

```
[answer] ... [/answer]
```

B.9 FeTaQA

Encoder:

```
[Task: Summarization] [Input: query] [
Input: table] [Output: cells] [Output:
long answer] [query] ... [/query] [table
] ... [/table]
```

Decoder:

```
[cell] ... [/cell] [answer] ... [/answer
]
```

B.10 FEVEROUS

Encoder:

```
[Task: Fact-checking] [Input: query] [
Input: table] [Input: passage] [Output:
binary answer] [query] ... [/query] [
table] ... [/table] [passage] ... [/
passage]
```

Decoder:

```
[answer] ... [/answer]
```

B.11 WikiSQL-Answer-only

Encoder:

```
[Task: QA] [Dataset: WikiSQL] [Input:
query] [Input: table] [Output: short
answer] [query] ... [/query] [table] ...
 [/table]
```

Decoder:

```
[answer] ... [/answer]
```

B.12 ToTTo-Reverse

Encoder:

```
[Task: Cell-generation] [Input: query] [
Input: table] [Output: cell] [query] ...
 [/query] [table] ... [/table]
```

Decoder:

```
[cell] ... [/cell]
```

C Annotation Interface

The annotation interface we used for our human study in this paper is shown in Figure 7

Zero-shot cell generation (TabFact)

Annotation guideline:

For each claim, the corresponding table, and the generated cells, our main goal is to evaluate the generated cells under the following two aspects:

1. **cell relevance:** whether the predicted cells are relevant to check the claim (precision). We have the following three labels: (1) relevant: all cells are relevant (precision == 1) (2) partially relevant: some cells are relevant (0 < precision < 1) (3) irelevant: none of the cells are relevant (precision == 0)

2. **cell completeness:** whether the predicted cells contain all information needed. For example, claim mentions two entities but only one is predicted. We have the following three labels: (1) complete: all necessary information is contained by the cells. (2) partially complete: only part of the information is covered by the cells. (3) incomplete: none of the information is covered by the cells.

Example ID: 17

Submit

Input claim:

the top 2 ranked team be in north america

Output:

[cell] 1 | sherri baier / robin cowan | canada | 2 | lorene mitchell / donald mitchell | united states [/cell] [answer] entailed [/answer]

rank	name	nation	points	places
1	sherri baier / robin cowan	canada	128.39	9
2	lorene mitchell / donald mitchell	united states	124.94	16
3	elizabeth cain / peter cain	australia	116.67	33
4	jana bláhová / ludek	czechoslovakia	113.74	36
5	sabine fuchs / xavier vide	france	114.12	39
6	karen wood / stephen baker	united kingdom	100.33	55
7	catherine brunet / philippe brunet	france	94.27	62

table caption:

1976 world junior figure skating championships

Figure 7: Instruction and the annotation interface we used for the human study in section 4.4.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☑ A2. Did you discuss any potential risks of your work?
Section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Last paragraph of Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 2.1

☑ B1. Did you cite the creators of artifacts you used?
Section 2.1

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix A

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We checked the data-collecting procedure of the datasets we used in this paper and found no specific offensive content is included. However, we did not take further action to conduct a more comprehensive review of each dataset, as they are widely used within the community and the large volume of datasets we employed made this impractical.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Appendix A

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Appendix A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

Section 3

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 3

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 4.4

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Section 4.4 Appendix C

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.