Toward Zero-Shot Instruction Following

Renze Lou and **Wenpeng Yin**
The Pennsylvania State University
{renze.lou, wenpeng}@psu.edu

Abstract

This work proposes a challenging yet more realistic setting for zero-shot cross-task generalization: *zero-shot instruction following*, presuming the existence of a paragraph-style task definition while no demonstrations exist. To better learn the task supervision from the definition, we propose two strategies: first, to automatically find out the critical sentences in the definition; second, a ranking objective to force the model to generate the gold outputs with higher probabilities when those critical parts are highlighted in the definition. The joint efforts of the two strategies yield state-of-the-art performance on the SUPER-NATURALINSTRU (Wang et al., 2022b).[1]

1 Introduction

With the rapid evolutions of the pre-training techniques, large language models (LLMs), such as GPT-3 (Brown et al., 2020) and ChatGPT (OpenAI, 2022), are found to be capable of handling various novel NLP tasks by following in-context instructions (Radford et al., 2019).[2] Typically, a formal task instruction consists of two components: (1) a task definition that describes the task intent; (2) a few labeled examples to demonstrate this intent (i.e., demonstrations). Then the problem is often named as "*k-shot instruction following*", where k is the example size. Due to the performance superiority of the in-context examples (Lampinen et al., 2022; Gu et al., 2023a), prior research has predominantly relied on demonstrations, allocating relatively limited attention toward effectively utilizing task definitions; we refer to this setting as "demonstration-driven instruction following" (Min et al., 2022a,b; Hu et al., 2022).

Notwithstanding the surprising results, this phenomenon could manifest as an instance of overestimated progress. Two reasons: firstly, **demonstrations are usually hard to be crafted in real-world applications**. Since LLMs are becoming helpful daily-task assistants and most end-users are non-experts (Chiang et al., 2023; Xie et al., 2023, 2024), it is usually exhausting and unrealistic for users to design *concrete* demonstrations for every daily task, especially for those tasks that require specific domain knowledge. Secondly, as Gu et al. (2023a) concluded, so far, **we still lack a method to effectively learn from instructions to solve tasks without demonstrations** for various reasons. For example, Khashabi et al. (2022) showed that the models constantly ignored the crucial information emphasized in the definition (e.g., an output constraint that asks models to "*generate no more than five words*"); Webson and Pavlick (2022) found that the models always struggled to truly understand the content of the definition.

To more effectively utilize the task definition, this work studies a more challenging setting: *zero-shot instruction following*. Technically, our approach consists of two strategies.[3] (i) Strategy I: automatically learn the critical task-relevant information from the lengthy definition to help the model better grasp the instruction. (ii) Strategy II: to make the model truly distinguish instructions that are specified by the critical information or not, we set a ranking-based training objective. Given instructions with critical information highlighted, this ranking strategy forces the model to generate ground-truth outputs with higher probabilities than instructions otherwise. Our system, PICK&RANK, achieves state-of-the-art on the benchmark, SUPER-NATURALINSTRU (Wang et al., 2022b).

[1] Code: https://github.com/RenzeLou/Pick-Rank

[2] Task instructions can be any textual expressions, e.g., task names, short sentences, or paragraphs, that describe the task semantics; prompts are the special case of instructions (Lou et al., 2023).

[3] In the rest of the paper, we use the terms "definition" and "instruction" alternately, when examples are unavailable.

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 50–60
March 21-22, 2024

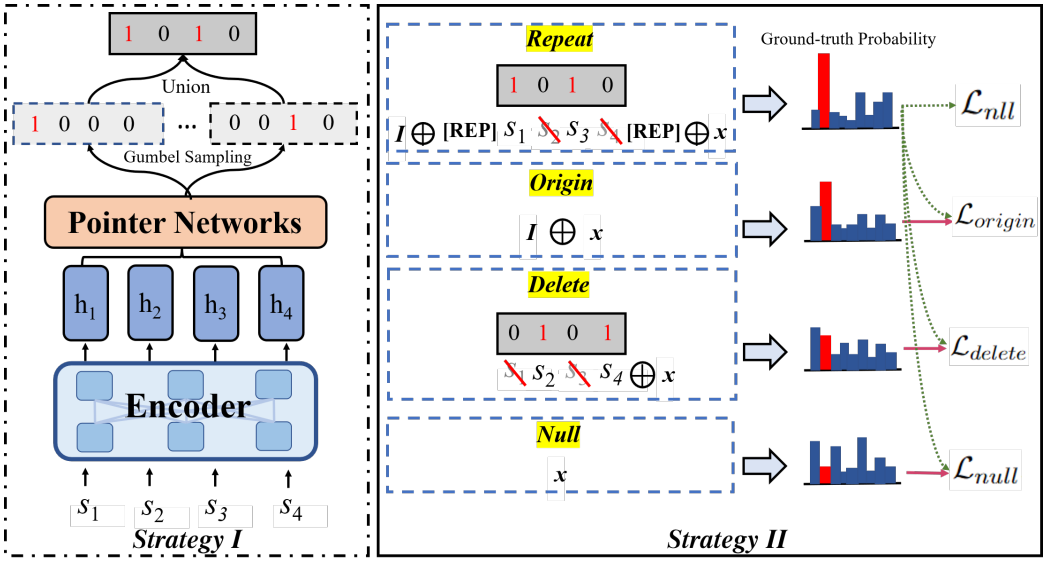

Figure 1: The illustration of our PICK&RANK. Two main components: **Strategy I** (PICK) and **Strategy II** (RANK). Strategy I aims to predict a binary value for each sentence in a definition, indicating whether a sentence is crucial. The outputs of Strategy I are used to construct instructions with different sufficiencies, e.g., "*Repeat*" represents the most beneficial instructions where the crucial sentences are repeated. Strategy II then drives the LMs to generate higher ground-truth probabilities on the more beneficial instructions. The whole system is optimized end-to-end.

2 Related Work

Prompt & In-context Learning. Prompting techniques usually acquire answers from large language models (LLMs) after rewriting the original task input into a LLM-oriented format. Impressive progress has been observed in various NLP tasks, such as question answering (Radford et al., 2019), text generation (Schick and Schütze, 2021), information extraction (Wang et al., 2022a; Sun et al., 2024), etc. Brown et al. (2020) further developed in-context learning (ICL): GPT-3 achieved competitive few-shot results without parameter tuning by prepending a prompt with a few demonstrations to new inputs. Follow-up work delved into improving ICL, including how to choose better demonstrations (Rubin et al., 2021; Lu et al., 2022), how to formulate the tasks (Zhao et al., 2021; Min et al., 2022a), etc. However, the short and simplistic nature of the prompts makes it difficult to express NLP tasks of diverse complexities (Chen et al., 2022). Our work tries to learn from instructions that describe the task semantics in more detail, such as Amazon MTurk instructions.

Follow Human-annotation Instructions. Prompts are more friendly for LLM to emit outputs. In the real world, humans describe tasks using paragraph-style instructions, such as crowd-sourcing guidelines. This type of instruction has recently attracted much attention, including increasingly larger datasets (Mishra et al., 2022; Wang et al., 2022b), new learning problems (Yin et al., 2022) and applications (Zhang et al., 2023), etc. To achieve cross-task generalization given instructions, prior systems trained a text-to-text model on a long sequence of text, i.e., concatenating task definition, demonstrations, and all other resources (Lou et al., 2023). We ignore demonstrations and focus on the supervision extraction from task definitions.

3 Problem Definition & Our Approach

We study zero-shot instruction following in a cross-task generalization setting, where evaluation tasks are unseen in training.

Zero-Shot Instruction Following: Three task sets: TRAINING TASKS, DEV TASKS, and TEST TASKS. There are no overlapping tasks among them. Each task T has its instruction I and a collection of labeled examples D ={(x, y)}. x: input; y: gold output of x under I. I is a short paragraph consisting of n sentences, i.e., $I = \{s_1, s_2, \cdots, s_n\}$. No examples exist in I. D of DEV TASKS and TEST TASKS are only used for evaluation. As shown in Figure 1, we adopt two strategies to better leverage the supervision in I.

Strategy I: picking critical sentences of instructions. Given the instruction $I = \{s_1, \cdots, s_n\}$, the goal of this phase is to learn a binary value for each s_i, indicating that if s_i is critical for the task T. We expect to select k most critical sentences.

As shown in Figure 1, we train a Pointer Network (Vinyals et al., 2015) to select critical sentences from the input automatically. First, we concatenate all $\{s_i\}$ in I as the encoder input to learn a hidden vector h_i for each s_i as: $h_i = \text{Encoder}(s_i|I)$, where $h_i \in \mathcal{R}^d$, and is average-pooled from all token-level vectors of s_i.

Second, we concatenate all sentence-level vectors $\{h_i\}$. Then a one-hot vector m^t of length n, indicating which sentence is critical, is derived by:

$$m^t \sim \text{Gumbel}(W[h_1, h_2, \ldots, h_n]) \quad (1)$$

where $W \in \mathcal{R}^{n \times nd}$, "Gumbel" is Gumbel-Softmax (Maddison et al., 2016), calculating a Gumbel distribution over the linear model predictions and samples categorical one-hot value from it. We use Gumbel-Softmax because it enables gradient back-propagation to help train the system end-to-end.

Since m^t is n-dimensional one-hot vector; it only picks a single critical sentence. To aggregate more potentially useful information from I, we do the Gumbel sampling procedure k times (where set k as 2 in our experiments) and take the element-wise union of $\{m^t\}$, $t = [1, \cdots, k]$. Accordingly, the final mask m is a $k^{'}$-hot vector ($k^{'} \leq k$) with each m_i as:

$$m_i = \cup_{t=1}^{k} m_i^t \quad (2)$$

Therefore, m enables the model to pick at most k critical sentences in I. As shown in Figure 1, $I = \{s_1, s_2, s_3, s_4\}$, and $\{s_1, s_3\}$ are critical sentences.

Strategy II: ranking-based objective. In a conventional text-to-text generation, we mainly optimize the probability, through negative log-likelihood ($\mathcal{L}_{nll}$), of generating the gold output. In zero-shot instruction following, when we are aware of which sentences in the I are crucial, in addition to applying the standard loss $\mathcal{L}_{nll}$, we can further take a ranking loss to make sure more informative instructions (I^+) lead to gold outputs with higher probabilities than less informative ones (I^-).[4] Specifically, we can build (I^+, I^-) pairs in three ways:

- **Repeat vs. Origin** (origin): I^+ is [s_1, s_2, s_3, s_4, [REP], s_1, s_3, [REP]]. This means $\{s_1, s_3\}$ will be repeated in the input instruction, and the special token [REP] can help tell the model which part is highlighted. I^- is [s_1, s_2, s_3, s_4];
- **Repeat vs. Delete** (delete): I^+ is [s_1, s_2, s_3, s_4, [REP], s_1, s_3, [REP]], I^- is I when those critical sentences are masked, i.e., [s_2, s_4];
- **Repeat vs. Null** (null): I^+ is [s_1, s_2, s_3, s_4, [REP], s_1, s_3, [REP]], and I^- is an empty string.

Let's use $f_{I^+}(y|x)$ and $f_{I^-}(y|x)$ to denote the probabilities of generating the gold output y given the input x and the instruction. Then our ranking loss $\mathcal{L}_{rank}$ is implemented as:

$$\mathcal{L}_{rank} = \max(0, \alpha - f_{I^+}(y|x) + f_{I^-}(y|x)) \quad (3)$$

where α controls the probability margin, and $f_*(y|x)$ is the average of word-level probabilities on the decoder side. The final loss of our model PICK&RANK is $\mathcal{L} = \mathcal{L}_{nll} + \beta \cdot \mathcal{L}_{rank}$. Different approaches to generating (I^+, I^-) pairs can specify the $\mathcal{L}_{rank}$ as: $\mathcal{L}_{origin}$, $\mathcal{L}_{delete}$, or $\mathcal{L}_{null}$ (as shown in Figure 1). We will study their individual and joint contributions in experiments. When testing, we generate the final prediction on "*Repeat*".

4 Experiments

Dataset. We work on the benchmark SUPER-NATURALINSTRU (Wang et al., 2022b), which contains 1,040 diverse English tasks (921 in *train* and 119 unseen tasks in *test*). We follow Wang et al. (2022b) only using 756 tasks in *train* to train the final model. Each task is expressed by an instruction, originally consisting of a paragraph-level task definition and a couple of positive&negative examples, and a large set of input-output instances. To satisfy our setting, we only use definitions as instruction I. The average definition length is 65.73 by words (4.09 by sentences). Those classification and generation tasks are respectively evaluated by EXACTMATCH and ROUGE-L (Lin, 2004). We also report ROUGE-L (overall), which calculates the ROUGE-L on both classification and generation tasks, to reflect an overall estimation. More dataset and metric details can be found in Appendix and Table 5.

Baselines. Since prior systems for few-shot instruction following need examples in instructions, in order to apply them to a zero-shot setting, we

[4] The motivation is that, given the informative I^+, the models can still ignore the beneficial parts selected by Strategy I (cf. Mishra et al., 2022). Thus, Strategy II further forces the models to pay attention to those crucial parts (textual differences between I^+ and I^-) by producing different probabilities.

			EXACTMATCH	ROUGE-L	ROUGE-L(overall)
GPT-4 (OpenAI, 2023)			64.51(±2.56)	59.36(±2.24)	62.96(±2.08)
ChatGPT (OpenAI, 2022)			46.90(±2.23)	56.82(±3.10)	52.41(±2.30)
SeqGAN (Yu et al., 2017)			24.50(±1.13)	31.19(±2.09)	27.55(±1.32)
ReCross (Lin et al., 2022)			28.95(±0.45)	38.81(±0.92)	33.88(±0.58)
MetaICL (SeqGAN) (Min et al., 2022b)			24.28(±0.98)	33.65(±1.87)	28.14(±1.22)
MetaICL (ReCross) (Min et al., 2022b)			14.98(±0.42)	21.63(±0.83)	20.74(±0.40)
TK-INSTRUCT (Wang et al., 2022b)			28.56(±0.39)	39.35(±0.85)	33.64(±0.47)
PICK&RANK	Strategy I		29.67(±0.43)	39.54(±0.90)	34.98(±0.57)
	w/ Strategy II	*ranking ori*	29.98(±0.87)	41.79(±1.08)	35.62(±0.76)
		ranking del	28.68(±1.04)	41.86(±1.21)	34.46(±0.89)
		ranking null	29.34(±0.92)	42.13(±1.13)	35.10(±0.93)
		ranking all	**30.58**(±0.83)	**43.55**(±1.02)	**36.70**(±1.14)

Table 1: Main results. Numbers of different methods were calculated from three random runs. We also put LLMs' performances (GPT-4, etc.) here for reference (i.e., upper bound). Please see the appendix for the baselines' details.

try to generate silver examples for them. For this thread, our baselines include (i) *SeqGAN* (Yu et al., 2017): Using GAN to generate silver y by utilizing task definition and x; (ii) *ReCross* (Lin et al., 2022): Retrieving similar examples from the training set using task definition and x; (iii) *MetaICL* (Min et al., 2022b): Meta-learning given task definition and a few examples. Due to the different resources of examples, *MetaICL* is specified to *MetaICL* (SeqGAN) and *MetaICL* (ReCross). Another baseline concatenates task definition, examples, and x in the encoder to decode y, namely the prior state-of-the-art system Tk-INSTRUCT (Wang et al., 2022b). More details about baselines are in the Appendix.

Our model implementation. We follow Wang et al. (2022b) using T5-base (Raffel et al., 2020) for all experiments. Please refer to Appendix and Table 4 for more experimental settings (e.g., hyper-parameters and computational cost).

Results. Table 1 summarizes the results on zero-shot instruction following. Overall, our approach shows successive performance improvements by adding the two proposed strategies and gains state-of-the-art results by adopting them jointly, proving the effectiveness of our method. Worth noting that the Tk-INSTRUCT can be regarded as our backbone, and after adding strategy I, our method has already improved by 1.34 ROUGE-L (overall) score, indicating the benefits of highlighting crucial sentences. Moreover, we gain further performance improvements by adding strategy II, because the ranking objective trains the model to discriminate the differences in the inputs, thus it drives the model to understand the highlighted information rather than simply ignoring them (Webson and Pavlick, 2022). As

I: The answer will be "yes" if the provided sentence contains an explicit mention that answers the given question. Otherwise, the answer should be "no". Instances where the answer is implied from the sentence using "instinct" or "common sense" [· · ·] should be labeled as "no". **y**: Yes. **TK-INSTRUCT** $\hat{\mathbf{y}}$: March **PICK&RANK** $\hat{\mathbf{y}}$: Yes
I: Given a text passage, come up with an appropriate title for it. [· · ·] The title should be 1-5 words long. **y**: Nobel Peace Prize **TK-INSTRUCT** $\hat{\mathbf{y}}$: The Nobel Peace Prize is one of the five Nobel Prizes created by the Swedish industrialist, inventor, and armaments manufacturer Alfred Nobel. **PICK&RANK** $\hat{\mathbf{y}}$: Nobel Peace Prize
I: In this task, you're given an ambiguous question (which can be answered in more than one way). Your task is to write a question that clarifies the given question in such a way that the generated question has one unique answer. **y**: When was the National World War II memorial officially established? **TK-INSTRUCT** $\hat{\mathbf{y}}$: 1830 **PICK&RANK** $\hat{\mathbf{y}}$: When was the memorial built?

Table 2: Effect of Strategy I. $\hat{\mathbf{y}}$: system output. The detected crucial sentences are highlighted in blue.

for MetaICL, due to the huge task differences between *train* and *test* (as shown in Table 5), those silver examples generated or retrieved by using the *train* do not provide the in-distribution patterns (Min et al., 2022c),[5] leading to sub-optimal or even worse performances, cf. MetaICL (ReCross) vs. ReCross. Note that, ReCross directly retrains the model with the retrieved examples and obtains relatively better results, however, it is still suffering from the drawbacks of few-shot instruction following in such a strict cross-task setting, so as SeqGAN.

Analysis. We try to clear up three concerns.

[5]We also observed the low instance similarities predicted by ReCross between *train* and *test*.

I: Generate an overlapping word between the given two sentences. [· · ·] You must generate significant words which **are not** the stop words like "the" or "of", etc. **x**: s_1: Amphibians have permeable skin that easily absorbs substances from the environment. s_2: Amphibians begin their lives in the water. **y**: Amphibians ‖ $\hat{\mathbf{y}}$:the Error type: negation
I: Two analogies that relate items to whether they are trash or treasure is given in the form "A : B. C : ?" [· · ·] "A : B" relates item A to whether it is trash or treasure, as specified by B. [· · ·] **x**: baby : treasure. leaf : ? **y**: trash ‖ $\hat{\mathbf{y}}$: relates item A to whether it is trash or treasure Error type: pattern copy
I: [· · ·] If it is about requesting something, generate 'REQUEST'. [· · ·] If it is about informing something, generate "INFORM". **x**: Please tell me do you have any particular date for the event? **y**: REQUEST ‖ $\hat{\mathbf{y}}$: INFORM Error type: incomplete critical sent. detection

Table 3: The error patterns by our system. We highlight the crucial sentences in the instructions with blue, and mark the error type as red.

$\mathcal{Q}_1$: Did the detected critical sentences really contribute to the generation of gold outputs? To answer $\mathcal{Q}_1$, we checked some examples where our system improves over the strongest baseline TK-INSTRUCT. As shown in Table 2, our approach can generally point out those crucial task-relevant sentences that are hardly encoded by the TK-INSTRUCT, such as *output space* (the first example), *length constraint* (the second example), and *types of output* (the last example). With the help of such highlights, our system can produce outputs that are better aligned with the task definitions, while TK-INSTRUCT often violates the requirements of instructions.

$\mathcal{Q}_2$: Could ranking objective really improve the probability of gold outputs? Regarding $\mathcal{Q}_2$, we test our model on all TEST TASKS with two versions of task instructions: repeat vs. origin. For each version, we calculate the corresponding probability of the ground truth output by averaging token-level probabilities in the output string. Our model can produce a higher ground-truth probability once "repeat" instruction is adopted (score: 0.59) than the "origin" definition (score: 0.11),[6] demonstrating the effectiveness of our Strategy II.

$\mathcal{Q}_3$: Error patterns of our systems. We randomly pick up 200 instances from the *test* and summarize three main error patterns of PICK&RANK, as shown in Table 3. (i) *Negation*. As the first example in Table 3 shows, even though the model is able to detect the sentence that has a specific requirement "generate significant words which are not the stop words · · · ", the negation "are not" was not successfully comprehended by the system. Unfortunately, negation understanding has increasingly been a challenge in NLP (AL-Khawaldeh, 2019; Yin et al., 2022; Khashabi et al., 2022). (ii) *Pattern copy*. The second example shows the system sometimes copies a span from the definition, especially when the definition string, e.g., "'A : B' relates item A to whether it is trash or treasure, as specified by B.", matches the format of x, e.g., "baby : treasure. leaf: ?". This resembles demonstration-driven in-context learning, where researchers found pattern match is a key factor of success (Min et al., 2022c). (iii) *Incomplete critical sentence detection*. It is possible that our system detects partial sentences that are critical. As a result, the system is biased toward the requirement of highlighted sentences. Rather than using a hard masking scheme, our future work will explore a soft-masking technique so that no instruction parts will be clearly ignored.

[6]Average from three random seeds experiments.

5 Conclusion

In this paper, we focused on zero-shot instruction following, where we only adopted the task definitions as the instructions to help the model perform cross-task generalization. Expressly, our method pointed the critical sentences out of the lengthy definitions and highlighted them explicitly. In addition, we further designed a ranking objective to improve the instruction grasp of the LMs. We also conducted thorough analyses to help future research on zero-shot instruction following.

References

Fatima T. AL-Khawaldeh. 2019. A Study of the Effect of Resolving Negation and Sentiment Analysis in Recognizing Text Entailment for Arabic. *CoRR*, abs/1907.03871.

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

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2022. Knowprompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. In *Proceedings of the ACM Web Conference 2022*, pages 2778–2788.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. 2021. Towards Interpreting and Mitigating Shortcut Learning Behavior of NLU Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 915–929.

Jun Gao, Yuhan Liu, Haolin Deng, Wei Wang, Yu Cao, Jiachen Du, and Ruifeng Xu. 2021. Improving Empathetic Response Generation by Recognizing Emotion Cause in Conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 807–819.

Jiasheng Gu, Hongyu Zhao, Hanzi Xu, Liangyu Nie, Hongyuan Mei, and Wenpeng Yin. 2023a. Robustness of learning from task instructions. In *Proceedings of ACL Findings*. Association for Computational Linguistics.

Xiaojie Gu, Renze Lou, Lin Sun, and Shangxin Li. 2023b. Page: A position-aware graph-based model for emotion cause entailment in conversation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-context Learning for Few-shot Dialogue State Tracking. *arXiv preprint arXiv:2203.08568*.

Daniel Khashabi, Chitta Baral, Yejin Choi, and Hannaneh Hajishirzi. 2022. Reframing Instructional Prompts to GPTk's Language. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 589–612.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Andrew K Lampinen, Ishita Dasgupta, Stephanie CY Chan, Kory Matthewson, Michael Henry Tessler, Antonia Creswell, James L McClelland, Jane X Wang, and Felix Hill. 2022. Can language models learn from explanations in context? *arXiv preprint arXiv:2204.02329*.

Bill Yuchen Lin, Kangmin Tan, Chris Miller, Beiwen Tian, and Xiang Ren. 2022. Unsupervised Cross-Task Generalization via Retrieval Augmentation. *arXiv preprint arXiv:2204.07937*.

Chin-Yew Lin. 2004. Rouge: A Package for Automatic Evaluation of Summaries. In *Text summarization branches out*, pages 74–81.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized Bert Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Renze Lou, Kai Zhang, Jian Xie, Yuxuan Sun, Janice Ahn, Hanzi Xu, Yu su, and Wenpeng Yin. 2024. MUFFIN: Curating multi-faceted instructions for improving instruction following. In *The Twelfth International Conference on Learning Representations*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2023. Is prompt all you need? no. a comprehensive and broader view of instruction learning. *arXiv preprint arXiv:2303.10475*.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Weicheng Ma, Renze Lou, Kai Zhang, Lili Wang, and Soroush Vosoughi. 2021a. Gradts: A gradient-based automatic auxiliary task selection method based on transformer networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5621–5632.

Weicheng Ma, Kai Zhang, Renze Lou, Lili Wang, and Soroush Vosoughi. 2021b. Contributions of transformer attention heads in multi-and cross-lingual tasks. 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 1956–1966.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *arXiv preprint arXiv:1611.00712*.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022a. Noisy Channel Language Model Prompting for Few-shot Text Classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022b. MetaICL: Learning to Learn In Context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022c. Rethinking the Role of Demonstrations: What Makes In-context Learning Work? *arXiv preprint arXiv:2202.12837*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

OpenAI. 2022. Chatgpt.

OpenAI. 2023. Gpt-4 technical report. *ArXiv preprint*.

Soujanya Poria, Navonil Majumder, Devamanyu Hazarika, Deepanway Ghosal, Rishabh Bhardwaj, Samson Yu Bai Jian, Pengfei Hong, Romila Ghosh, Abhinaba Roy, Niyati Chhaya, et al. 2021. Recognizing emotion cause in conversations. *Cognitive Computation*, 13:1317–1332.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*.

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.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, page arXiv:1606.05250.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to Retrieve Prompts for In-context Learning. *arXiv preprint arXiv:2112.08633*.

Timo Schick and Hinrich Schütze. 2021. Few-shot Text Generation with Natural Language Instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Lin Sun, Kai Zhang, Qingyuan Li, and Renze Lou. 2024. Umie: Unified multimodal information extraction with instruction tuning. *arXiv preprint arXiv:2401.03082*.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in neural information processing systems*, 12.

Yuanhe Tian, Renze Lou, Xiangyu Pang, Lianxi Wang, Shengyi Jiang, and Yan Song. 2022. Improving English-Arabic transliteration with phonemic memories. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3262–3272, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. *Advances in neural information processing systems*, 28.

Liwen Wang, Rumei Li, Yang Yan, Yuanmeng Yan, Sirui Wang, Wei Wu, and Weiran Xu. 2022a. InstructionNER: A Multi-task Instruction-based Generative Framework for Few-shot NER. *arXiv preprint arXiv:2203.03903*.

Xinshao Wang, Yang Hua, Elyor Kodirov, Guosheng Hu, Romain Garnier, and Neil M Robertson. 2019. Ranked List Loss for Deep Metric Learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5207–5216.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022b. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yutong Wang, Renze Lou, Kai Zhang, Mao Yan Chen, and Yujiu Yang. 2021. More: A Metric Learning Based Framework for Open-Domain Relation Extraction. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7698–7702.

Albert Webson and Ellie Pavlick. 2022. Do Prompt-based Models Really Understand the Meaning of

Their Prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. 2023. Adaptive chameleon or stubborn sloth: Unraveling the behavior of large language models in knowledge conflicts. *arXiv preprint arXiv:2305.13300*.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: Toward real-world planning with language agents.

Wenpeng Yin, Jia Li, and Caiming Xiong. 2022. ConTinTin: Continual Learning from Task Instructions. In *ACL*, pages 3062–3072.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. 2023. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *International Conference on Machine Learning*, pages 12697–12706.

Hyper-parameters	Range
lr for T5	[5e-6, 1e-5, **5e-5**, 1e-4]
lr for Pointer Networks	[5e-5, 1e-4, **3e-4**, 5e-4]
lr for Encoder	[1e-6, **5e-6**, 1e-5, 5e-5]
α_{origin}	[0.001, 0.003, **0.01**, 0.03, 0.1]
α_{delete}	[0.001, 0.003, 0.01, **0.03**, 0.1]
α_{null}	[0.01, 0.03, **0.1**, 0.3]
β	[0.01, 0.05, 0.1, 0.5, **1**]
k	[1, **2**, 3, 4, 5]
Pooling Function	[**average**, max]

Table 4: The hyper-parameters trialed in tuning our models. The best ones adopted in our final experiments are highlighted in boldface. Here, "lr" denotes the learning rate; α is the probability margin in equation 3, there are three different α according to three ranking losses; β is a coefficient that controls the influence of the ranking losses; and k is the sampling times in equation 2.

Appendix A. Expanded Technique Details

Due to the length limitation, we have to elaborate on some other important details of our approach in this section, including four different instructions in Figure 1 and how we enable end-to-end optimization. As we have illustrated in Figure 1, our approach consists of two parts, corresponding to **Strategy I** and **Strategy II** in Section 3, respectively.

Strategy I (the left dashed box in Figure 1) first encodes and converts all the sentences in a definition to sentence-level representations. Then, we adopt pointer networks followed by a Gumbel-Softmax layer to predict a binary vector for these representations, where "1" means the corresponding sentence contains crucial task-relevant information and should be attended by the LMs. In order to pick up more potentially useful information, we repeat the Gumbel sampling several times and take the element-wise union of the sampling results as the final decision of strategy I. It is worth noting that the encoder of this phase shares the same model structure as the encoder of the LMs to keep similar internal features of the downstream procedure (Lin et al., 2022). However, they are optimized individually.

Strategy II (the right solid box in Figure 1) regards the output binary vector of strategy I as a sentence-level mask matrix and constructs four different instructions accordingly: (1). ***Repeat*** indicates the definition in which the critical parts are repeated and highlighted. Practically, we repeat the whole definition once (surrounded by a special token "[REP]") and use the binary vector from the strategy I as the attention mask matrix in the Transformers (Vaswani et al., 2017); (2). ***Origin*** is the original definition without any modifications; (3). ***Delete*** denotes the definition where the critical parts are masked. Similar to *Repeat*, we actually encode the whole definition and use the invert of the binary vector to mask the critical information; (4). ***Null*** means that there are no instructions provided. Intuitively, if the model can truly understand the prefixed instructions, it shall discriminate these text differences and produce better results on the inputs with informative instructions (i.e., *Repeat*) than the others (i.e., *Origin*, *Delete*, and *Null*).[7] Therefore, besides the standard negative log-likelihood $\mathcal{L}_{nll}$, there are three additional ranking losses in total, namely $\mathcal{L}_{origin}$, $\mathcal{L}_{delete}$, and $\mathcal{L}_{null}$.

Notably, our system can be optimized end-to-end because we incorporate the decision of strategy I by utilizing the attention mask mechanism in the LMs of strategy II.

Appendix B. Experimental Details

For hyper-parameters, we use segmented learning rate (5e-5 for T5, 3e-4 and 5e-6 for the pointer networks and encoder, respectively) optimized with Adam (Kingma and Ba, 2014). As for the margins of ranking losses, we follow previous works employing structured margins to obtain a better representation space in LMs (Wang et al., 2019, 2021). Following Wang et al. (2022b), after two epochs training on $train$, we evaluate our model on $test$ with the beam size equal to 1 (greedy decoding). We present our hyper-parameters selection in Table 4. All the ranges of these hyper-parameters are decided empirically, and we search for the best combination greedily by observing the ROUGE-L score on the development set. We use Hugging Face T5-base for all the experiments [8] and utilize Spacy for sentence segmentation.[9] It is notable that the definition length can be diverse, and it will extremely increase the computational burden if we let the pointer networks consider all the sentences in a definition. According to Table 5, we randomly select 5 sentences from the definition of each task as the candidates.

All of our code is implemented by using Python

[7] Unlike the *Repeat*, we do not use any special tokens in the other instructions ("[DEL]", "[NULL]", etc.) to avoid introducing shortcuts to the model (Du et al., 2021).

[8] https://huggingface.co/t5-base

[9] https://github.com/explosion/spacy-models/releases/tag/en_core_web_sm-3.4.1

	Train	Dev	Test
# of tasks	756	100	119
# of instances	75,317	9,958	11,810
# of task types	60	23	12
# of domain types	101	24	35
# of sources	243	46	75
sources overlap with test set	0.0%	80.4%	/
avg def. length (words per task)	66.41	65.58	61.55
avg def. length (sentences per task)	4.11	4.12	3.92

Table 5: The dataset statistics.

3.8.0 and PyTorch 1.12.1 [10] with CUDA 11.6, and we utilize Hugging Face Transformers 4.18.0 [11] to train and evaluate our models. We conduct all our experiments on Ubuntu 18.04 LTS using Intel(R) Core(TM) i9-10900KF CPU with 32 GB of memory, and employing NVIDIA RTX A5000 GPU with 24 GB of memory. On the whole, there are about 332 million parameters in our models. It takes about 12 hours to train and evaluate our models (2 epochs with batch size equal to 1). At the same time, the peak of GPU usage is 23GB.

Appendix C. Dataset and Metrics

We show the statistics of the benchmark dataset in Table 5. We only focus on the English tasks and use the same data split policy as previous work (Wang et al., 2022b), where all those tasks coming from the same sources as the test set are excluded from the training set (as shown in Table 5). However, because no official development set is provided, we randomly select 100 tasks from those excluded tasks with a maximum of 100 instances per task, as the development set used in our experiments. Similarly, we follow Wang et al. (2022b) to use the first 100 instances per testing task and randomly choose 100 instances per training task.

As for the evaluation metrics, we follow Wang et al. (2022b) utilizing ROUGE-L (Lin, 2004) and EXACTMATCH (Rajpurkar et al., 2016) to evaluate the cross-task generalization performance of the text-to-text LMs. To be specific, the ROUGE-L reflects the string overlap between the answers and the predictions, while EXACTMATCH measures the ratio of the number of correctly predicted examples. Both of these metrics are widely adopted by previous works (Rajpurkar et al., 2016; Poria et al., 2021; Gu et al., 2023b). Since the EXACTMATCH calculates the ratio of how many ground truth labels

[10] https://pypi.org/project/torch/

[11] https://github.com/huggingface/transformers/releases

I: You are given two sentences and have to find if there is entailment or agreement of the Hypothesis by the Premise. [· · ·] Your task is to return "entails" if the premise supports hypothesis else return "neutral". **y**: entails **TK-INSTRUCT** $\hat{\mathbf{y}}$: calorie **PICK&RANK** $\hat{\mathbf{y}}$: entails
I: Generate an appropriate title for the given text. The generated title must be short and include the main topic of the text. The preferred titles are under fifteen words. **y**: Case Logic Laptop roller bag **TK-INSTRUCT** $\hat{\mathbf{y}}$: This bag is great for carrying laptop, HP Printer, portable scanner, cables and supplies **PICK&RANK** $\hat{\mathbf{y}}$: bag for laptop
I: In this task, you are given two questions about a domain. Your task is to combine the main subjects of the questions to write a new, natural-sounding question. For example, [· · ·]. **y**: Did this president go to college in the state he was born in? **TK-INSTRUCT** $\hat{\mathbf{y}}$: this president **PICK&RANK** $\hat{\mathbf{y}}$: this president was born on the east coast?
I: Given a document, generate a short title of the document. The title should convey the main idea/event/topic about which the document is being written. Note that URLs in the text have been replaced with [Link]. **y**: Dutch politician on trial on hate speech charges **TK-INSTRUCT** $\hat{\mathbf{y}}$: Geert Wilders **PICK&RANK** $\hat{\mathbf{y}}$: Geert Wilders is on trial for hate speech

Table 6: More cases. The crucial sentences are in blue.

are generated, it is similar to the accuracy score. Thus, we report the EXACTMATCH score for those classification tasks in Table 1. What's more, we use the same evaluation script as Wang et al. (2022b) to compute these metrics.[12]

Appendix D. Baselines

As mentioned in Section 4, we implement four baselines for a comprehensive comparison. As follows, we provide detailed implementation information. Worth noting that we tune all the hyperparameters of the baselines on the development set or use the default settings reported by the original paper.

SeqGAN It regards the generation as a sequential decision procedure and uses the Reinforcement Learning (RL) rewards of an additional classifier to optimize the generator. The original SeqGAN is based on LSTM (Hochreiter and Schmidhuber, 1997). In order to fair compare with the other models, we change the backbone to T5-base. For training the SeqGAN, including the generator and classifier, we use the following steps: (1). Pre-training: we first pre-train the T5-base on the benchmark dataset as the generator, that is, we concatenate

[12] https://github.com/yizhongw/Tk-Instruct/blob/main/src/compute_metrics.py

the original definition with the task input (i.e., x) and drive the model to predict the output (i.e., y). As for the classifier, we use Hugging Face bert-large-cased [13] to perform a sequence classification, namely predicting the binary label (i.e., "0" or "1") by encoding the task definition and the (x, y) pair produced by the generator; (2). Adversarial training: We follow Yu et al. (2017) training the generator and classifier alternately. Specifically, when generating each token, we employ Monte Carlo (MC) search to complete the whole sequence and use policy gradient (Sutton et al., 1999) to optimize the generator. After 20 steps of training on the generator (batch size equals 4), we use the silver answers predicted by the generator as the negative examples to train the classifier. After adversarial training the generator with 5 epochs, we then use it to predict the instances of the unseen tasks in the test set (i.e., $\hat{y}$). Meanwhile, these (x, $\hat{y}$) pairs can also serve as examples for in-context learning (see MetaICL for more details).

ReCross This is a retrieve-based method that utilizes the unlabeled examples of an unseen task to retrieve similar labeled examples from the training set. These retrieved examples can be further used for retraining the model. Similarly, they can also be used for in-context learning (i.e., MetaICL). We follow the official implementation of Lin et al. (2022).[14] However, there are several differences between the original algorithm and our usage: (1). We use the concatenation of definition and task input as the query and index for a fair comparison. We also believe the task definition can provide valuable semantics for the retrieval procedure; (2). Instead of using RoBERTa (Liu et al., 2019), we train a Hugging Face bert-base-cased model as the Reranker,[15] which has relatively better results in our experiments; (3). We use T5-base as the backend of ReCross.

MetaICL Following Min et al. (2022b) and Wang et al. (2022b), we use task definition and two positive examples as instructions to train and test the T5-base model. While the test set examples are those silver examples produced by SeqGAN and ReCross, namely MetaICL (SeqGAN) and MetaICL (ReCross). All the other hyperparameters are the same as what we use in the TK-INSTRUCT.

[13] https://huggingface.co/bert-large-cased
[14] https://inklab.usc.edu/ReCross/
[15] https://huggingface.co/bert-base-cased

TK-INSTRUCT We use the official code and hyper-parameters of Wang et al. (2022b).[16] The only difference is that we use T5-base instead of T5-3B reported in their paper, due to the limited computational resources. It is also worth noting that the original Tk-INSTRUCT is trained with positive demonstrations as additional instructions; in this paper, we solely use the task definition as the instruction of Tk-INSTRUCT to ensure a fair comparison.

ChatGPT & GPT-4 For LLMs' performances, we use the scores reported by Lou et al. (2024) in Table 1, where they concatenate the task instruction with input as a whole query of APIs. Please refer to Lou et al. (2024) for more details.

Appendix E. More Cases

We display more intuitive cases in Table 6.

Appendix F. Limitations

In this section, we summarize several limitations and broader impacts of this paper. (1) As mentioned in Section 4, one limitation of this paper is that our approach is still difficult to fully encode the crucial information in the definitions, even if they are well highlighted, such as the negation expresses. Potential solutions include adopting an additional weighting strategy on the decisions of the pointer networks (See et al., 2017), adding a soft fusion mechanism in the LMs (Gao et al., 2021; Tian et al., 2022), or proposing an automatic instruction reframing technology (Khashabi et al., 2022). (2) Meanwhile, since the task definition is usually a paragraph consisting of several sentences, this paper mainly focuses on detecting crucial sentence-level information. However, in some cases, task-relevant information should be better represented in a word-level or span-level format, such as the *output space*. Therefore, our strategy can be further improved by using a hybrid-level pointer to satisfy the diverse real-world scenarios. (3) Another potential future investigation is to analyze how LMs utilize the highlighted information in the instructions through human intuition, such as visualizing the multi-head attention score distribution of the transformers (Ma et al., 2021b,a), or probing the conflict between the in-context instruction and model's parametric knowledge (Xie et al., 2023). We leave them as our future work.

[16] https://github.com/yizhongw/Tk-Instruct