SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2025, Vol. 68, Iss. 7, 172102:1–172102:15
https://doi.org/10.1007/s11432-023-3911-2

Improving cross-task generalization with step-by-step instructions

Yang WU[1], Yanyan ZHAO[1*], Zhongyang LI[2], Bing QIN[1] & Kai XIONG[1,3]

[1]*Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology, Harbin 150000, China*
[2]*Huawei Cloud, Shenzhen 518100, China*
[3]*School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore*

Received 5 June 2023/Revised 13 October 2023/Accepted 4 December 2023/Published online 18 June 2025

Abstract Instruction tuning aims to improve cross-task generalization of language models. However, it is still challenging for language models to complete the target tasks following the instructions because the instructions are general and lack intermediate steps. To solve this issue, step-by-step instructions are introduced to help language models decompose tasks, which can offer detailed and specific procedures for completing the target tasks. The step-by-step instructions are obtained automatically by prompting ChatGPT, and are further merged with the original instructions to tune the language models. Extensive experiments on Sup-NatInst reveal that high-quality step-by-step instructions can enhance cross-task generalization across different model sizes. Further analysis indicates the significance of the order of steps in the proposed instruction for improvement. To promote future research, step-by-step instructions and human quality evaluation results will be released.

Keywords instruction tuning, generalization, step-by-step instruction, large language model, task decomposition

Citation Wu Y, Zhao Y Y, Li Z Y, et al. Improving cross-task generalization with step-by-step instructions. Sci China Inf Sci, 2025, 68(7): 172102, https://doi.org/10.1007/s11432-023-3911-2

1 Introduction

How to improve cross-task generalization of language models is an important but challenging problem, that has garnered increasing attention from the NLP (natural language processing) community [1–5]. Mishra et al. [2] built the NatInst dataset with 61 various NLP tasks to assess the cross-task generalization of language models, which are trained on some of the tasks and evaluated on other tasks. Wang et al. [3] extended NatInst and built a much larger dataset, Sup-NatInst which includes 1616 NLP tasks. Studies [2,3] conducted on NatInst and Sup-NatInst demonstrated that instruction tuning can enhance the generalization of language models to new tasks.

However, the natural language instructions adopted by an earlier work [3] are primarily task definitions. They are general and lack intermediate steps, which makes it demanding for language models to follow instructions and complete target tasks. Herein, step-by-step instructions are introduced to make the instructions more detailed and specific. The step-by-step instructions can give the task-level intermediate problem-solving steps without depending on any specific example, and are easy to understand and follow.

Because manually acquiring such detailed step-by-step instructions is tedious and laborious, ChatGPT is employed to automatically acquire such instructions. It has strong problem-solving abilities and the ability to follow complex prompts. For the first aspect, ChatGPT has realized superior performance [6] on multiple challenging datasets proposed for holistic assessment of large language models such as MMLU [7] and BBH [8]. For the second aspect, ChatGPT is trained to align with instructions by using reinforcement learning from human feedback [9], enabling accurate understanding of our prompts and generating desirable output. Conversely, other models, such as GPT-3 [10], Bloom [11], and GLM-130B [12] often express unintended behaviors such as not following user's prompts, particularly complex prompts, because the original training objective of LLMs is language modeling such as predicting the next token, which is different from the objective "follow the user's instructions" [9]. Moreover, ChatGPT enables us to refine the generated step-by-step instructions through multiple interactions, which helps in improving

* Corresponding author (email: yyzhao@ir.hit.edu.cn)

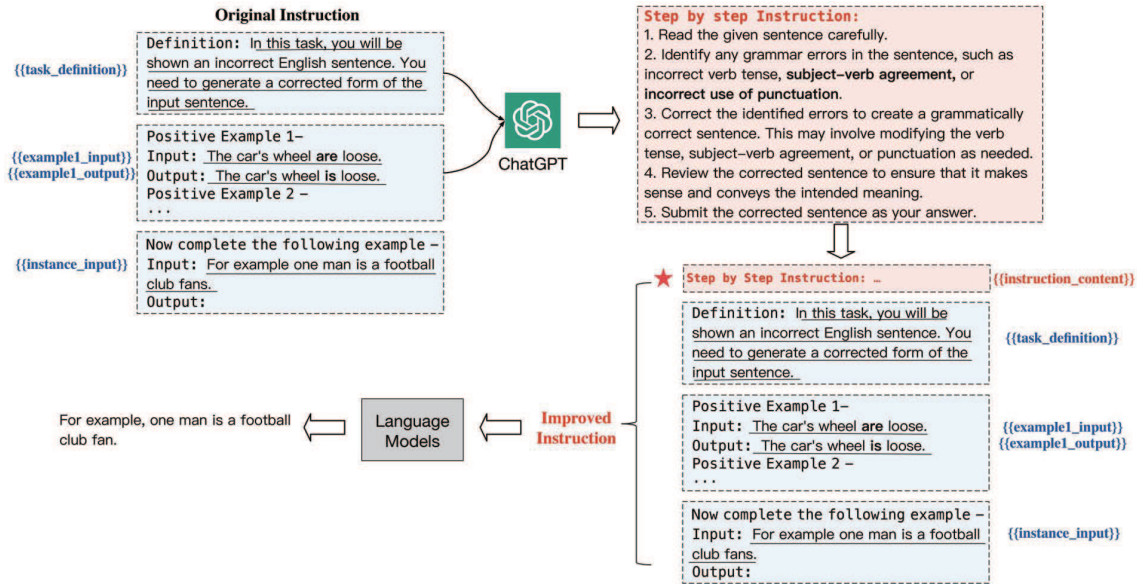

Figure 1 (Color online) Step-by-step instruction tuning consists of two steps: (1) prompt ChatGPT to obtain the step-by-step instruction based on the task definition and the positive examples (Subsections 3.1 and 3.2); (2) combine the step-by-step instruction with the original instruction to tune the language models (Subsection 3.3). The contents of the original instruction elements are marked with underlines and we omit the refining process for simplicity.

the correctness and completeness of such instructions. Particularly, ChatGPT is treated as a translator and asked to first understand the intent behind the original instructions and then write the step-by-step instructions for completing the target tasks. To acquire more desirable results, further progressive refinement of the step-by-step instructions is performed through multiple interactions with ChatGPT. The final step-by-step instructions are integrated with the original instructions to tune the language models.

To gain an intuitive understanding of our approach, namely step-by-step instruction tuning, Figure 1 illustrates our proposed strategy. The original instruction includes a task definition, positive examples, and instances. However, the task definition is short and general. Therefore, we pass the task definition and positive examples to ChatGPT to obtain step-by-step instructions. The obtained step-by-step instructions are detailed and specific, showing the intermediate steps of completing the task and even indicating possible grammar errors such as subject-verb agreement and incorrect use of punctuation in Step 2. We believe that the detailed step-by-step instructions can help language models complete this task.

Extensive experiments were conducted on Sup-NatInst [3] to assess our proposed approach. The experimental results show the effectiveness of step-by-step instruction tuning, enhancing the cross-task generalization of T5-LM with different model sizes. To comprehensively understand our approach, various important factors affecting model performance are analyzed, such as the order of steps. The analysis indicates that shuffling the order of steps results in a performance drop because it corrupts the correctness of the step-by-step instruction. In addition, an attempt to leverage ChatGPT is made to generate positive examples, and the results reveal that ChatGPT can generate useful positive examples to improve the model performance of lower quality than the manually written examples.

In conclusion, (1) this work is the first to incorporate step-by-step instructions to improve cross-task generalization and demonstrate their effectiveness through extensive experiments; (2) this work is also the first to automatically generate and refine the step-by-step instructions with ChatGPT; (3) a comprehensive human evaluation is performed to examine the quality of the step-by-step instructions; and (4) the step-by-step instructions and the results of the human evaluation are published to promote future research on improving generalization with the step-by-step instructions.

2 Related work

Instruction tuning. Instruction tuning is effective for improving language models generalization. Sanh et al. [4] and Wei et al. [13] collected a large dataset of different tasks and split a part of the tasks as the training set and the remaining tasks as the test set. They mixed the data from the training set and

trained the language models using multitask learning. The tuned models were assessed on the test set to estimate their zero-shot performance. Their experimental results revealed that instruction tuning can improve the zero-shot performance of large language models. Wang et al. [3] also built a meta-dataset, Sup-NatInst, which comprises various NLP tasks and explored the few-shot performance of language models given the instruction, in a format different from the previous two studies. In contrast to using the simple manual prompt as the instruction [4, 13], Sup-NatInst instruction includes the task definition, positive examples, and negative examples. Our work focuses on this instruction format and improves the cross-task generalization of language models with step-by-step instructions. Our approach is also markedly different from Mishra et al. [14], because Mishra et al. [14] manually reframed the instructions of the evaluation tasks of NatInst [2] to make them more suitable for prompting GPT models, which is difficult to extend to new tasks. Our approach automatically obtains step-by-step instructions by prompting ChatGPT with a series of task-agnostic prompts, which can be easily used for other tasks.

Chain-of-thought prompting. Recently, chain-of-thought (CoT) prompting [15] has exhibited impressive results on several complicated reasoning tasks, such as the math word problem, which incorporates a series of manually written intermediate reasoning steps for demonstration examples to unlock the reasoning ability of large language models. However, the adopted demonstrations are task-specific and carefully designed, which are difficult to obtain. Conversely, Kojima et al. [16] proposed Zero-shot-CoT, which first obtains the intermediate reasoning steps by prompting the large language models and then incorporates such intermediate reasoning steps to obtain the answer. Zhang et al. [17] introduced Auto-CoT to sample questions with diversity and generate reasoning chains to construct demonstrations. Although both CoTs and our approach decompose the task into multiple steps [18, 19], our approach is fundamentally different from CoTs. First, CoTs are a type of prompting method that provides the reasoning steps in exemplars and elicits reasoning in a large language model during inference. They are not applicable to instruction tuning (post-training phrase) and cannot improve the generalization of models. However, our method can be employed to improve generalization during instruction tuning. Second, CoTs cannot be used for small models (approximately 10 billion parameters) as per the experimental results in FLAN-T5 [5], because small models cannot follow such complex prompts and generate intermediate steps. Third, our proposed step-by-step instruction does not depend on any specific example, which is a general problem-solving procedure for completing the target task, while CoT is a series of intermediate reasoning steps for a specific example.

Annotating data with ChatGPT. Impressed by the remarkable abilities of ChatGPT, some researchers have begun exploring the potential of leveraging ChatGPT to annotate data. Gilardi et al. [20] utilized ChatGPT to label tweets and found that it outperforms crowd-workers for several annotation tasks. Alpaca employed ChatGPT to build a large collection of instructional data. In this paper, we adopt ChatGPT to generate step-by-step instructions for solving the target tasks and undertake meticulous assessments to guarantee data quality. The human evaluation demonstrates that ChatGPT can understand our intents and generate high-quality step-by-step instructions.

3 Method

In this section, we first introduce how to obtain and refine step-by-step instructions automatically by prompting ChatGPT with a series of task-agnostic prompts (Subsections 3.1 and 3.2). We elaborate this process in Figure 2. Then, we further propose step-by-step instruction tuning to incorporate such step-by-step instructions to improve cross-task generalization of language models (Subsection 3.3).

3.1 Step-by-step instruction obtaining

We carefully design the prompt to ask ChatGPT to generate an easy-to-follow step-by-step instruction for the target task based on the task category and task definition. The content of the task category is most often covered by the task definition. It is used here to induce ChatGPT to attend to the task definition at the bottom of the prompt.

In this prompt, we add some constraints to help ChatGPT to generate more desirable outputs. For example, we specify that the step-by-step instruction is used for instructing the generative pre-trained language models to prevent ChatGPT from generating unapplicable intermediate steps, such as using the search engines. The full adopted prompt is as follows.

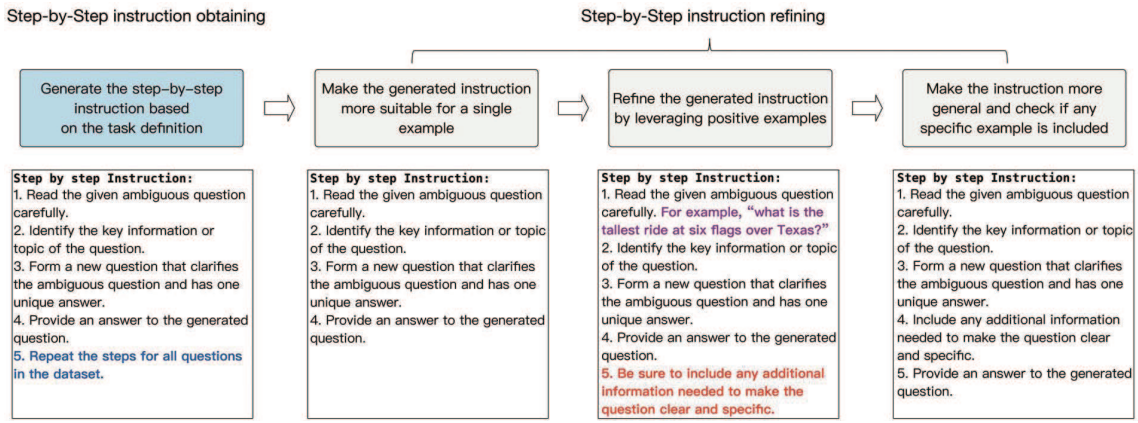

Figure 2 (Color online) Illustration of obtaining and refining the step-by-step instructions. We present a series of examples to demonstrate our method. First, we prompt ChatGPT to generate the step-by-step instruction based on the task definition. Second, we ask ChatGPT to refine the generated step-by-step instruction and remove the step of iterating through the dataset (marked with blue color) as the language models only process one example at each time. Third, we leverage the positive examples to further refine the step-by-step instruction. The refined instruction contains more detailed information (marked with red color) but includes the specific example (marked with purple color). Finally, we ask ChatGPT to remove the specific example to make the instruction more general.

> Please provide a step-by-step instruction for completing the {{task_category}} task. The generated instruction will be directly used as the input of the generative pre-trained language models. The instruction should be simple and easy to understand, without any specific examples.
> Note that: The instruction should only focus on the current example. Do not contain the step of iterating through the dataset.
> {{task_category}}: {{task_definition}}

{{task_category}} and {{task_definition}} will be replaced with the task category and definition of the specific target task.

3.2 Step-by-step instruction refining

Although ChatGPT is asked to generate appropriate step-by-step instruction, ChatGPT sometimes does not follow the prompt. Hence, we propose to refine the step-by-step instruction through multiple interactions with ChatGPT. First, ChatGPT could instruct to iterate the process through the dataset. To address this problem, we design the prompt to make ChatGPT refine the step-by-step instruction to make sure that it is suitable for a single example.

> Refine the instruction to make sure the instruction is applicable for a single example and does not instruct to repeat the process through the dataset.

Second, we utilize the positive examples to help ChatGPT to more comprehensively understand the task leading to better step-by-step instruction. Similarly, {{example1_input}}, {{example1_output}}, {{example2_input}}, and {{example2_output}} will be filled with the contents of the specific examples.

> Refine your instruction according to the given examples and return the full refined instruction. The refined instruction should be simple and easy to understand, without any specific examples.
>
> Example 1:
>
> Input: {{example1_input}}
>
> Output: {{example1_output}}
>
> Example 2:
>
> Input: {{example2_input}}
>
> Output: {{example2_output}}

Third, to avoid the step-by-step instruction containing any specific example and make it general, we further ask ChatGPT to check and refine the step-by-step instruction.

Refine the instruction to make sure the instruction does not contain any specific example.

Lastly, we fetch the final step-by-step instruction from ChatGPT using the following prompt.

Output the refined instruction.

3.3 Step-by-step instruction tuning

To incorporate the step-by-step instructions, we further tune the T5-LM model[1)] [21] from the checkpoint of T*k*-Instruct [3] on the training set. Besides the effective elements used by T*k*-Instruct including the task definition and positive examples, we add the step-by-step instruction and combine it with the definition and positive examples. The negative examples and the explanations are not utilized, as previous work [3] find they could hurt the model performance. We train and test our models with the step-by-step instructions and call our method step-by-step instruction tuning. The full instruction template we adopted is as follows.

Step by Step Instruction: {{instruction_content}}

Definition:{{task_definition}}

Positive Example 1−

Input:{{example1_input}}

Output:{{example1_output}}

Positive Example 2−

...

Now complete the following example−

Input:{{instance_input}}

Output:

{{instruction_content}} and {{instance_input}} represent the step-by-step instruction and the input of the instance respectively. The full example will be fed to the T5-LM model to predict the output.

4 Experiment

4.1 Dataset

We evaluate our method on Sup-NatInst [3], which is a large benchmark of various NLP tasks and their natural language instructions. Each instruction consists of the definition, positive examples, and negative examples. The definition shows how an input text is expected to be mapped to an output text. Each positive/negative example has three elements namely the input text, correct/incorrect output, and corresponding explanation. Because the negative examples and explanations of the positive examples hurt the model's performance [3], previous work does not use them. We also exclude them for a fair comparison. Following its gold split for evaluating English cross-task generalization, the training set consists of 757 different tasks for supervision and the test set contains 119 unseen tasks for evaluation. The evaluation tasks are diverse and can be divided into 12 categories such as question answering, title generation, and cause-effect classification, covering both generation and classification. The statistics of Sup-NatInst (English track) are shown in Table 1. The total number of categories means the number of task types in the dataset. For the task category and task, there is no overlap between the training set and test set. The average word count of the step-by-step instructions is larger than that of the definitions, since our proposed step-by-step instructions provide more detailed procedures for completing the tasks. ROUGE-L [22] is adopted to evaluate the methods following Wang et al. [3] as it aligns well with human evaluation and can be easily applied to various tasks. Specifically, the Python package, namely rouge-score (0.1.2), is adopted, which is also used by T*k*-Instruct [3]. To accurately estimate the performance, we also conduct human evaluation.

1) It is obtained by further tuning T5 with the LM objective.

Table 1 Statistics of Sup-NatInst (English track).

Statistic	Train	Test
Total number of tasks	757	119
Total number of categories	60	12
Average word count per definition	66.2	65.6
Average word count per step-by-step instruction	118.2	91.6
Average number of steps per step-by-step instruction	6.0	5.4

4.2 Training details

We conduct our experiments based on T5-LM. As for T*k*-Instruct, we rerun the public code[2)] released by Wang et al. [3] and report the results. To leverage step-by-step instructions, we continue to train T5-LM models from the checkpoints of T*k*-Instruct with a training epoch of 2. The learning rate is set to 1e−5 for the 11B model and 5e−5 for others. The batch size is set to 16 for the 3B and 11B models and 8 for others. The maximum input length of T*k*-Instruct and our models is set to 1224, and the maximum output length is set to 128. Following the experimental setting adopted by Wang et al. [3], we sample a maximum of 100 instances per task, which results in 75317 and 11810 instances for training and testing. We implement our approach based on the code released by T*k*-Instruct. We train our 3B model on four A100 (40 GB) GPUs and use DeepSpeed (0.7.7)[3)] to reduce the GPU memory. The training process takes 28 h to complete. The 11B models are trained on four A100 (80 GB) GPUs and it takes 120 h to finish. For the hyper-parameters, we mainly follow the hyper-parameters adopted by T*k*-Instruct such as the learning rate and epoch number and we increase the max length of the input to 1224 for adding the step-by-step instruction. Note that, the results of T*k*-Instruct reported in our paper are obtained by re-running T*k*-Instruct with the increased length for a fair comparison. The random seed of all experiments in this paper is set to 42 following T*k*-Instruct.

4.3 Baselines

There are three kinds of baselines. One kind of model is the heuristic baselines including Copying Instance Input and Copying Demo Output. Another kind of model is large language models including T5-LM [21] and GPT-3 [10]. The remaining baselines are instruction-tuned models including T0 [4], InstructGPT [23], and T*k*-Instruct [3]. Copying Instance Input [3] copies the input of the test instance as the output. Copying Demo Output [3] randomly selects one input demonstration example and copies the output of the example as the predicted output. T5-LM [21] as a language model takes the input text and directly generates the output, which is obtained by further tuning T5 with the LM objective. GPT-3 [10] is a large language model, which is trained on massive amounts of unlabeled data using the next token prediction objective. GPT-3 achieves impressive performance on many tasks in the few-shot setting. T0 [4] is trained on many natural language prompted datasets using multi-task learning to improve its generalization to unseen tasks. InstructGPT [23] adopts the RLHF fine-tuning procedure to learn to follow instructions. T*k*-Instruct [3] is the most relevant baseline, which is built on T5-LM and trained using the instructions provided by Wang et al. [3]. We also use such instructions but we leverage the additional step-by-step instructions, which can provide the detailed intermediate steps for solving the tasks. Besides, T*k*-Instruct has four variants: T*k*-Instruct (base), T*k*-Instruct (large), T*k*-Instruct (3B), and T*k*-Instruct (11B). These variants are obtained by initializing T*k*-Instruct from different sizes of pretrained T5-LM checkpoints, including the small, base, large, xl and xxl sizes. We use the prediction results released by Wang et al. [3] to evaluate the performance of GPT-3 and InstructGPT. They accessed GPT-3 ("davinci") and InstructGPT ("text-davinci-001") through the API on May 30, 2022 and shared the prediction results[4)]. We compute ROUGE-L of GPT-3 and InstructGPT based on the prediction results.

4.4 Main results

We show the results of step-by-step instruction tuning in Table 2. There are four key takeaways.

2) https://github.com/yizhongw/Tk-Instruct.
3) https://github.com/microsoft/DeepSpeed.
4) https://github.com/yizhongw/Tk-Instruct/tree/main/output/default.

Table 2 Results on the unseen tasks in the test set of SUP-NATINST. The step-by-step instructions improve cross-task generalization of baseline models with different model sizes (base, large, 3B, and 11B).

Method	ROUGE-L
Copying instance input	14.2
Copying demo output	28.5
T5-LM (11B)	30.2
GPT-3 (175B)	45.0
T0 (11B)	32.3
InstructGPT (175B)	52.1
T*k*-INSTRUCT (base)	42.6
+ Step-by-step instruction tuning	**43.6** (↑ 1.0)
T*k*-INSTRUCT (large)	48.0
+ Step-by-step instruction tuning	**49.7** (↑ 1.7)
T*k*-INSTRUCT (3B)	54.4
+ Step-by-step instruction tuning	**56.3** (↑ 1.9)
T*k*-INSTRUCT (11B)	60.0
+ Step-by-step instruction tuning	**60.9** (↑ 0.9)

• Incorporating the step-by-step instructions can improve the cross-task generalization of languages models. step-by-step instruction tuning consistently improves T*k*-INSTRUCT across different model sizes (Base, Large, 3B, and 11B). We attribute our success to two facts. The first one is that our proposed step-by-step instructions are helpful, effective, and informative, which can provide the detailed and specific procedures for completing the target tasks while the original instructions are general. The language models can easily follow our instructions and solve the problems step by step. In-depth analysis shows that incorporating the step-by-step instructions in either training or testing or both phases is helpful, which further demonstrates the effectiveness of such instructions. The second one is that ChatGPT can generate high-quality step-by-step instructions through prompting. As for this point, we conduct comprehensive analysis and we find that the correctness and completeness of the obtained step-by-step instructions are surprisingly high due to the strong ability of decomposing and solving the tasks of ChatGPT, which is unlocked by our well-designed prompts.

• Overall performance increases with the model size, which can be concluded by comparing our models with different model sizes. This finding is in line with the scaling law. Because larger models could store more knowledge and have stronger learning ability, they can more easily learn how to map the input text to the output text. Moreover, our approach not only enhances the cross-task generalization of smaller models but also proves beneficial for larger models.

• Instruction tuning can improve cross-task generalization of language models. For instance, InstructGPT, T0, and T*k*-INSTRUCT outperform their base models, namely GPT-3, T5-LM, and T5-LM, as they have been trained to follow the instructions, which makes them benefit more from the instructions. Besides, even though our model (11B) is much smaller than InstructGPT, our model surpasses InstructGPT and we attribute it to that our model is trained with more diverse instructional data and more effective instructions.

• The heuristic baselines obtain worse results, which indicates that copying the input of the test instance or the output of the example without fully considering the task definition and the task input is far from enough. To achieve good performance on this challenging dataset, the model should be able to fully understand the diverse objectives of the target tasks and solve the tasks successfully. To help the language models to achieve the goals more easily, we propose to incorporate the step-by-step instructions to decompose the final goals and provide the detailed and specific step-by-step solutions for completing the tasks.

Moreover, to estimate the model performance of ChatGPT[5)], we randomly sample 10 instances for each test task resulting in 1190 instances for evaluation. ChatGPT obtains 61.2 ROUGE-L scores and our model (11B) surpasses ChatGPT and obtains 61.4 ROUGE-L scores on the same evaluation dataset. This result further reveals the effectiveness of our approach.

5) This paper was written in May 2023 and the results of ChatGPT may change if it is updated.

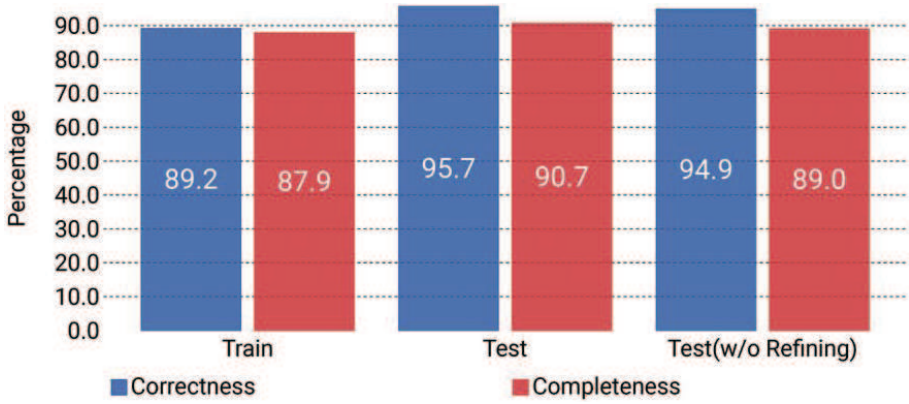

Figure 3 (Color online) Human evaluation of the step-by-step instructions.

5 In-depth analysis

5.1 Analysis of step-by-step instruction

To quantitatively evaluate the quality of the step-by-step instructions, we adopt correctness and completeness as the metrics. Specifically, if both each step of the step-by-step instruction and the order of the steps are right for completing the target task, we consider that it is correct. And the completeness metric indicates whether the step-by-step instruction contains all the necessary information of the original instruction such as the constraints.

We invited three well-educated students including two undergraduates and one graduate student as our annotators to evaluate the quality of the step-by-step instructions. They were fully informed of the purpose of our study and the potential ethical risks involved in the tasks and we paid them a fair and reasonable hourly wage. The instruction for annotating the quality of the step-by-step instruction is as follows: "You are given some samples and each sample consists of a task definition, two positive examples, and a step-by-step instruction generated by ChatGPT. The step-by-step instruction will be combined with the task definition and the positive examples to instruct language models to complete the target task. Please carefully read the given sample and analyze the correctness and completeness of the step-by-step instruction. As for correctness, if each step of the step-by-step instruction and the order of the steps are right for completing the target task, it is considered as correct. As for completeness, if the step-by-step instruction contains all necessary information about the task definition such as the constraints, it is considered complete."

To evaluate the inter-annotator agreement among the three annotators, we sample 60, 20, and 20 examples from the training set, test set, and test set (w/o Refining) as a shared annotation part of all annotators and split the remaining examples into three parts as the independent part for each annotator. We measure inter-annotator agreement among the three annotators in the shared part using Fleiss's kappa [24]. The scores are 0.78 for correctness and 0.73 for completeness indicating substantial agreement among the three annotators. We present the overall evaluation results in Figure 3. According to the evaluation results, we make the following observations.

Our approach can obtain high-quality step-by-step instructions with high correctness and completeness. As shown in Figure 3, the correctness and completeness of the step-by-step instruction are very high, even though all the step-by-step instructions are generated by ChatGPT without any manual modification. Besides, the statistics of Table 1 show that the average word count per step-by-step instruction is larger than the average word count per definition. These findings convince us that the detailed, correct, and complete step-by-step instruction can help language models to complete the target tasks more easily.

Refining can improve the quality of the step-by-step instructions. Both correctness and completeness of the step-by-step instructions of the test tasks increase after refining. This observation indicates that it is possible to further refine the step-by-step instructions by progressively leveraging more relevant information about the target tasks via multiple interactions with ChatGPT. To deeply analyze the quality of the step-by-step instructions, we split the step-by-step instructions into four classes, which are correct and complete, correct but incomplete, complete but incorrect, and incorrect and incomplete.

Table 3 Correct and complete step-by-step instruction for sentiment analysis.

Task category	Sentiment analysis
Task ID	task293_storycommonsense_emotion_text_generation
Task definition	In this task, you're given a context, a sentence, and a character. The sentence describes an action or job of the given character. Also, the context provides more information about the sentence or the character. Your task is to return one of the emotions which are expressed by the Character in the given sentence. For that you can use the Context; however, Context is not provided in all the inputs. Also, in some inputs, there can be zero emotion; for that return "None".
Step-by-step instruction	(1) Begin by reading the context and sentence provided in the input. (2) Identify the emotion or sentiment expressed by the character in the sentence. Look for words or phrases that indicate the character's emotional state, such as "happy", "sad", or "angry". (3) If the context is provided, consider how it may affect the character's emotional state. For example, if the context describes a positive event or experience, it may indicate that the character is feeling happy or content. (4) If you are unable to determine the character's emotion from the sentence or context, return "None". (5) Once you have identified the emotion expressed by the character, return that emotion as your output.

Table 4 Correct but incomplete step-by-step instruction for question answering. The constraint in the task definition, which is marked with bold text, is not included in the step-by-step instruction.

Task category	Coreference resolution
Task ID	task329_gap_classification
Task definition	In this task, you will be presented with a text, a pronoun from the text, and two candidate names. You should determine what the pronoun refers to and classify the answers into A, B, or Neither. A and B here are referring to option A and option B. **Position of the pronoun in the text is showed within two "_"s.**
Step-by-step instruction	(1) Read the given text and pronoun. (2) Identify the two candidate names, referred to as A and B. (3) Consider the context of the pronoun within the text, including the words and phrases surrounding it. (4) Determine which of the two candidate names, A or B, the pronoun most likely refers to based on the context. (5) Classify the answer as A, B, or Neither, depending on your determination.

Table 5 Incorrect but complete step-by-step instruction for program execution. The incorrect statement is marked with bold text as all elements are lowercase. The correct statement should be "e.g., a is at position 1, b is at position 2, etc."

Task category	Program execution
Task ID	task622_replace_alphabets_in_a_list_by_their_position_in_english_alphabet
Task definition	In this task, you are given an input list A. You need to convert all the alphabets in the list with a number representing their position in the English alphabet. E.g., replace A by 1, B by 2, a by 1, b by 2, and so on.
Step-by-step instruction	(1) Initialize an empty output list, which will store the modified elements of the input list. (2) Iterate through each element in the input list. (3) For each element in the input list, check if it is an alphabet. If it is not an alphabet, append it to the output list without modifying it. (4) If the element is an alphabet, convert it to its lowercase form. (5) Calculate the position of the lowercase alphabet in the English alphabet **(e.g., A is at position 1, B is at position 2, etc.)**. (6) Append the calculated position to the output list. (7) Repeat the process for each element in the input list until all elements have been processed. (8) Return the output list as the result of the program execution.

Table 6 Incorrect and incomplete step-by-step instruction for dialogue generation. Language models cannot directly utilize natural language generation techniques, hence, this step-by-step instruction is incorrect and incomplete.

Task category	Dialogue generation
Task ID	task1600_smcalflow_sentence_generation
Task definition	In this task, you have given an input which is agent's reply, based on that you have to identify what will be users' command for that reply. The user generally asks a question related to event scheduling or information of the scheduled events. Agent replies based on the available information or do the task as per user command.
Step-by-step instruction	(1) Identify the input, which is the agent's reply to a user's question or command. (2) Determine the context of the conversation by considering the previous exchange between the user and the agent. (3) Use the context and the content of the agent's reply to generate an appropriate response or command from the user. (4) Consider the user's goals and needs, as well as any constraints or limitations, when generating the response. (5) **Use natural language generation techniques** to produce a response that is coherent and flows naturally with the rest of the conversation. (6) Test and iterate on the generated response to ensure it is accurate and appropriate for the given context.

The percentages of them in the whole dataset including the training set and test set are 87.9%, 2.8%, 0.5%, and 8.8%. We show the representative examples in Tables 3–6, respectively.

5.2 Impact of step-by-step instruction

Incorporating the step-by-step instructions in either training or testing, or both phases

Table 7 Incorporating the step-by-step instructions in different phases.

Method	ROUGE-L
T*k*-INSTRUCT (3B)	54.4
+ Step-by-step instruction (inference)	55.0 (↑ 0.6)
+ Step-by-step instruction (training)	56.0 (↑ 1.6)
+ Step-by-step instruction (training and inference)	56.3 (↑ 1.9)

Table 8 Incorporating the step-by-step instructions at different positions. "Prepend" and "Append" mean prepending and appending the step-by-step instructions to the task definitions respectively.

Method	Position	ROUGE-L
T*k*-INSTRUCT (base)	None	42.6
+ Step-by-step instruction tuning	Prepend	43.6 (↑ 1.0)
+ Step-by-step instruction tuning	Append	43.6 (↑ 1.0)
T*k*-INSTRUCT (large)	None	48.0
+ Step-by-step instruction tuning	Prepend	49.7 (↑ 1.7)
+ Step-by-step instruction tuning	Append	50.3 (↑ 2.3)
T*k*-INSTRUCT (3B)	None	l54.4
+ Step-by-step instruction tuning	Prepend	56.3 (↑ 1.9)
+ Step-by-step instruction tuning	Append	55.3 (↑ 0.9)

helps cross-task generalization. To analyze the effect of incorporating the step-by-step instructions in different phases, we conduct experiments based on T*k*-INSTRUCT (3B) and list the experimental results in Table 7. According to the results, we find that directly incorporating the step-by-step instructions in the inference phase can improve the model performance, even though the model does not see the step-by-step instructions in the training phase. Another observation is that leveraging the step-by-step instructions in the training phase can also boost the model performance and it is not necessary for obtaining improvement to fetch the step-by-step instructions during inference. Moreover, when we utilize the step-by-step instructions in both phases, our model achieves the best performance and surpasses the baseline model by 1.9 ROUGE-L scores. These findings demonstrate our motivation, which is that the step-by-step instructions can complete the original instructions.

Incorporating the step-by-step instructions at different positions. To study the effect of different positions of the step-by-step instructions, we incorporate the step-by-step instructions by prepending and appending them to the task definitions. The experimental results are presented in Table 8. As shown in the table, for both positions, incorporating the step-by-step instructions can improve cross-task generalization across different model sizes, which demonstrates that the step-by-step instructions are useful for instructing the models to solve new tasks. According to the results, we find that prepending the step-by-step instructions to the task definitions is preferred considering the average improvement.

5.3 Effect of the refining process

As we mentioned in the previous section, the original step-by-step instructions generated by ChatGPT could not be good enough. Hence, we further ask ChatGPT to progressively refine the original step-by-step instructions of the test set[6)]. The human evaluation results shown in Figure 3 have demonstrated that the correctness and completeness of the step-by-step instructions are improved through refining. To analyze the contribution of refining on the model, we conduct the ablation study and report the results in Table 9. There are two findings. One is that refining the step-by-step instructions is helpful for improving the model performance since some mistakes could be fixed through refining leading to a better quality of the step-by-step instructions. The other one is that adopting the original step-by-step instructions can improve cross-task generalization, which further demonstrates the effectiveness of such instructions.

5.4 Importance of the order of steps

The order of the steps of the step-by-step instruction is vital for its correctness. For example, the step-by-step instruction for instructing the model to find the longest word in the given sentence is first splitting

6) We only perform the refining process on the test set to save the cost of collecting the step-by-step instructions and we also think the model could overcome the noises during training.

Table 9 Results of ablating the refining process for obtaining the step-by-step instructions. Step-by-step instruction inference means only using the step-by-step instructions in the inference phase.

Method	ROUGE-L
T*k*-Instruct (3B)	54.4
+ Step-by-step instruction inference	55.0
l+ Step-by-step instruction inference (original)	54.8 (↓ 0.2)
+ Step-by-step instruction tuning	56.3
+ Step-by-step instruction tuning (original)	56.2 (↓ 0.1)

Table 10 Results of randomly shuffling the order of steps of the step-by-step instruction.

Method	ROUGE-L
T*k*-Instruct (3B)	54.4
+ Step-by-step instruction inference	55.0
+ Step-by-step instruction inference (shuffled)	54.8 (↓ 0.2)
+ Step-by-step instruction tuning	56.3
+ Step-by-step instruction tuning (shuffled)	54.4 (↓ 1.9)

the sentence, obtaining the length of each word, and returning the longest word. If the order of the three steps is shuffled, the step-by-step instruction becomes: first obtaining the length of each word, splitting the sentence, and returning the longest word. As for this example, the step-by-step instruction is changed and is incorrect, which can hurt the effectiveness of the step-by-step instruction. But there are also some cases that shuffling the order does not hurt the correctness. For example, if we instruct the model to determine the relationship between the given two sentences, there are two steps specifically first reading sentence A and then reading sentence B. Changing the order of these two steps does not affect the correctness. To demonstrate this point, we automatically parse the step-by-step instructions and obtain the steps. Then, we randomly shuffle the steps for each step-by-step instruction. The experimental results are listed in Table 10. As we can see, randomly shuffling the step-by-step instructions hurts the performance especially for step-by-step instruction tuning, as the noisy inputs make it hard to learn useful information through further tuning.

5.5 Model performance for each category

We present the model performance of our model (3B) and T*k*-Instruct (3B) for each category of the test dataset in Figure 4. The results show that with step-by-step instruction tuning, there is a gain in performance for most of the categories, especially for overlap extraction. One possible reason is that this kind of tasks could be solved more easily by following the steps of the step-by-step instructions, such as splitting the sentence, removing the stop words, and outputting the overlapping word. Another observation is that the model performance on word analogy drops and we attribute it to the low correctness of the step-by-step instructions as shown in Figure 5, as such instructions could provide wrong information and confuse the language model.

5.6 Understanding of intermediate steps

To quantitatively analyze whether the model can understand and complete each intermediate step, we construct two intermediate tasks, identifying the positive words/phrases and replacing the positive words/phrases with the negative words/phrases, for task928_yelp_positive_to_negative_style_transfer (sampled from the training set) based on the step-by-step instructions. The samples for evaluation are derived from the original instances. We keep the inputs and obtain the gold outputs by prompting ChatGPT with the new intermediate task definitions and manually constructed exemplars, given our observation that ChatGPT performs well on these intermediate tasks. We evaluate our model (3B) on the intermediate tasks and our model obtains 55.5 and 69.7 ROUGE-L scores, while T*k*-Instruct (3B) obtains 50.2 and 53.1 ROUGE-L scores. These results reveal that our model could implicitly learn to solve the intermediate tasks through step-by-step instruction tuning. Moreover, we conduct a similar analysis on task039_qasc_find_overlapping_words (sampled from the test set) and our model also achieves better performance than T*k*-Instruct.

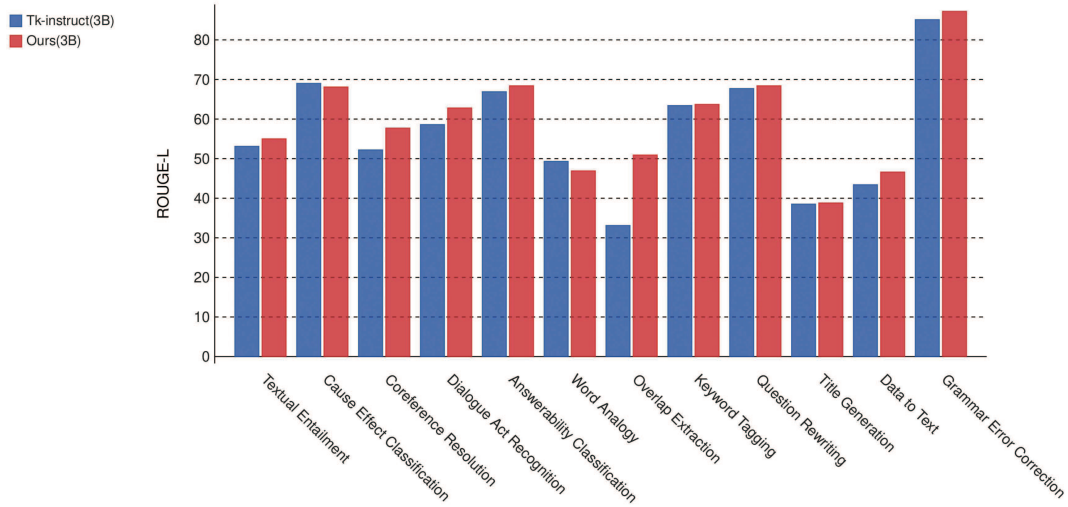

Figure 4 (Color online) Model performance of our model (3B) and T*k*-INSTRUCT (3B) for each category.

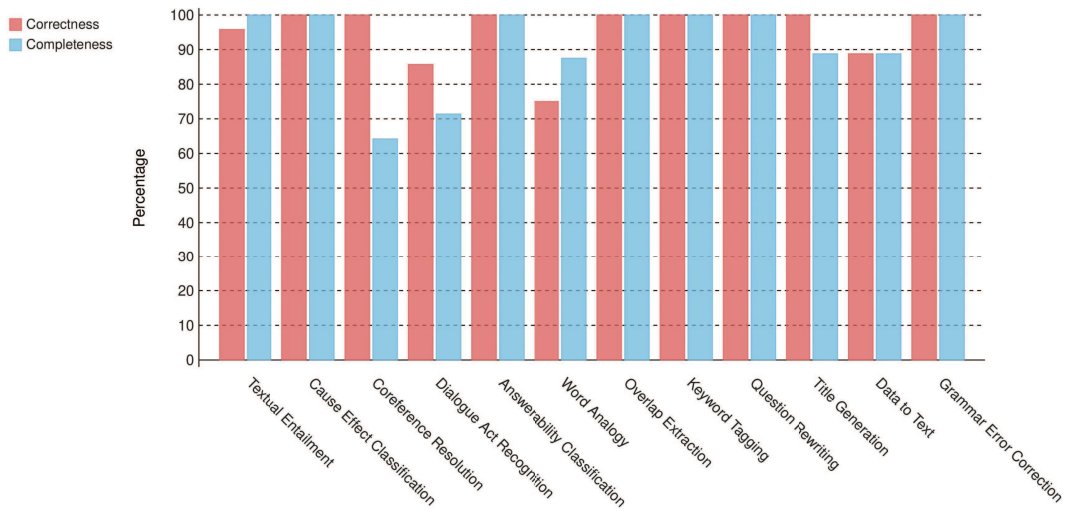

Figure 5 (Color online) Correctness and completeness of the step-by-step instructions of the test dataset for each category.

5.7 Human evaluation

We conduct the human evaluation to further demonstrate the effectiveness of the step-by-step instructions, as the automatic metric is a proxy of the performance of the models. Specifically, we randomly select three test instances for each task resulting in 357 instances. For each annotator, an instance is presented with the task definition, the associated positive samples, and two prediction results generated by our model (3B) and T*k*-INSTRUCT (3B). The annotators do not know where the predictions come from and determine which prediction is better. The win/lose/tie rates are 17.6%, 12.6%, and 69.8%, and the Fleiss's kappa score is 0.72, which indicates our model achieves better performance and further demonstrates that step-by-step instruction tuning can improve cross-task generalization. As for the details of the annotation, we invited three well-educated students including two undergraduates and one graduate student as our annotators to conduct the comparison between our model and the baseline model. The instruction for annotating the comparison results between our model and the baseline model is as follows: You are given some samples and each sample consists of a task definition, two positive examples, and two predictions from two evaluated models (Model A and Model B). Please carefully read the given sample and determine which prediction is better (Model A or Model B). If you can not determine which prediction to select, pick the "Same" option.

Table 11 Example instance from task190_snli_classification in SUP-NATINST benchmark.

Task category	Textual entailment
Task ID	task190_snli_classification
Task definition	In this task, you're given a pair of sentences, sentence 1 and sentence 2. Your job is to choose whether the two sentences clearly agree (entailment)/disagree (contradiction) with each other, or if this cannot be determined (neutral). **Your answer must be in the form of the letters E, C, and N respectively.**
Step-by-step instruction	(1) Read sentence 1 and sentence 2. (2) Compare the two sentences to determine whether they agree or disagree with each other. (3) If the sentences agree with each other, choose the "entailment" option (E). (4) If the sentences disagree with each other, choose the "contradiction" option (C). (5) If it is not possible to determine whether the sentences agree or disagree, choose the "neutral" option (N).
Instance input	Sentence 1: four males in a string quartet perform on an indoor stage. Sentence 2: the pianists put on shows in enormous outdoor arenas.
Output (T*k*-INSTRUCT)	E ✗
Output (Ours)	C ✓

5.8 Case study

We show an example instance from the test set in Table 11. The task definition states that the answer must be in the form of the letters E, C, and N, but the meaning of these three options is not very clear only considering the last sentence. This requires language models should understand the context and find out that the letters E, C, and N represent entailment, contradiction, and neutral respectively. In contrast, the step-by-step instruction clearly explains the meaning of each option and provides relevant useful information about the three options in Steps 3–5. The step-by-step instruction elaborates the procedure of completing the textual entailment task, which enables our model to solve the problem step by step following it. With the help of such clear and detailed step-by-step instruction, our model (3B) completes the task successfully while T*k*-INSTRUCT (3B) fails.

5.9 Quality of the generated examples

We also attempt to ask ChatGPT to automatically generate the positive examples encouraged by the finding that it can generate valuable step-by-step instructions. To help ChatGPT output desirable examples, we carefully design our prompts and introduce the self-ranking process. Specifically, we first ask ChatGPT to generate multiple examples using the following prompt. {{instruction_content}} denotes the content of the step-by-step instruction and {{generated_example_num}} is a hyper-parameter that controls the number of the generated examples.

> Give me {{generated_example_num}} harder examples for the {{task_category}} task following this structure:
>
> Input:
>
> Output:
>
> Explanation:
>
> The instruction for this task is :
>
> {{instruction_content}}

Then ask ChatGPT to rank the generated examples denoted as {{example_content}} and return the Top-2 examples.

> Rank following examples by the correctness of their answers and the relevance and consistency with the task instruction. Return the content of the best two examples and do not need to explain the reason.
>
> {{example_content}}
>
> The instruction for this task is : {{instruction_content}}

To qualify the quality of the obtained examples, we replace the manually written examples with the generated examples and the ranked examples[7]. Note that, the number of all adopted examples is two for a fair comparison and the utilized generated examples are selected randomly from the obtained multiple

7) The explanations are not used following previous work.

Table 12 Results of leveraging the generated demonstration examples by ChatGPT. SSII and SSIT mean step-by-step instruction inference and step-by-step instruction tuning respectively.

Method	ROUGE-L
T*k*-Instruct (3B) w/ SSII w/o Examples	44.4
+ Generated examples	51.6 (↑ 7.2)
+ Ranked generated examples	51.7 (↑ 7.3)
+ Manually written examples	55.0 (↑ 10.6)
T*k*-Instruct (3B) w/ SSIT w/o Examples	44.0
+ Generated examples	53.3 (↑ 9.3)
+ Ranked generated examples	53.4 (↑ 9.4)
+ Manually written examples	56.3 (↑ 12.3)

examples. The experimental results are shown in Table 12. According to the results, we find that generating the examples as better as the manually written examples is still very challenging since it requires the model to be able to fully understand the task and generate informative and right inputs and outputs. While ChatGPT can understand the intents behind task instructions, it still generates trivial examples and provides incorrect answers to the generated questions. Even though the models utilizing the generated examples and ranked examples underperform the model adopting the manually written examples, they surpass the model without examples, which indicates that the generated and ranked examples are useful for improving cross-task generalization. Besides, the comparison between the models utilizing the generated examples and ranked examples shows the self-ranking process can further improve the quality of the obtained examples.

6 Discussion

Standing on the shoulders of ChatGPT. ChatGPT has impressed humans with its ability to provide detailed and well-organized answers to questions. Encouraged by it, we propose to distill its knowledge of how to solve a specific problem to improve the instruction and further enhance the cross-task generalization of language models by instruction tuning. Particularly, we treat ChatGPT as a meta-instructor and ask it to write down its step-by-step problem-solving process by prompting it. The smaller language model serves as an executor to follow the detailed instructions and complete the task.

Why can ChatGPT generate such useful step-by-step instructions? ChatGPT is trained by instruction tuning and reinforcement learning from human feedback (RLHF), which allows it to understand the true intent of the human-written instructions and complete the tasks. Another possible reason is that pre-training on code data teaches it how to perform procedure-oriented and object-oriented programming, which helps it to learn to solve tasks step by step and decompose complex tasks into simpler ones.

Effect of the biases of ChatGPT on the quality of the step-by-step instructions. ChatGPT is tuned to interact with humans. Thus some generated detailed instructions are suitable for humans but not for language models. For instance, ChatGPT instructs to build a deep learning model for completing the sentiment classification task. As a solution, we asked ChatGPT to generate suitable instructions for instructing language models. Another possible solution is to manually refine the step-by-step instructions; however, it is time- and cost-consuming, and it cannot be adopted quickly for new tasks.

Regarding safety issues of the content generated by ChatGPT. ChatGPT can help people label data; however, its use exhibits certain safety challenges, such as the generation of potentially harmful content and the non-transparency of the generation process. To solve the first problem, we manually checked all the generated instructions to ensure that the step-by-step instructions did not have any offensive content or private information. For the second problem, our method distills the knowledge of ChatGPT transparently and we know what we are distilling. If there are other capable open-source models, that can follow our complex prompt to generate step-by-step instructions, we can also obtain such instructions for more transparency. Step-by-step instructions can also be generated by humans, but it is more time-consuming and laborious. Creating human-generated instructions requires 15 min per instruction, while human evaluation requires 3 min.

7 Conclusion

Herein, incorporating step-by-step instructions is proposed to improve the cross-task generalization of language models. Extensive experiments demonstrate the effectiveness of our proposed approach, namely step-by-step instruction tuning. We attribute the success to the fact that the high-quality step-by-step instructions can provide detailed and specific procedures for completing the tasks, which aids language models in decomposing the tasks. We believe that the step-by-step instructions can provide more opportunities for better cross-task generalization. Therefore, the step-by-step instructions and human evaluation results are released to facilitate future research. In future work, we would like to explore explicitly guiding language models to solve the intermediate steps of the step-by-step instructions.

References

1 Ye Q Y, Lin B Y C, Ren X. CrossFit: a few-shot learning challenge for cross-task generalization in NLP. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Punta Cana, 2021. 7163–7189
2 Mishra S, Khashabi D, Baral C, et al. Cross-task generalization via natural language crowdsourcing instructions. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, 2022. 3470–3487
3 Wang Y Z, Mishra S, Alipoormolabashi P, et al. Super-naturalinstructions: generalization via declarative instructions on 1600+ tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, 2022. 5085–5109
4 Sanh V, Webson A, Raffel C, et al. Multitask prompted training enables zero-shot task generalization. In: Proceedings of the International Conference on Learning Representations, 2022
5 Chung W H, Hou L, Longpre S, et al. Scaling instruction-finetuned language models. 2022. ArXiv:221011416
6 Liang P, Bommasani R, Tsipras D, et al. Holistic evaluation of language models. 2022. ArXiv:221109110
7 Hendrycks D, Burns C, Basart S, et al. Measuring massive multitask language understanding. In: Proceedings of the International Conference on Learning Representations, 2021
8 Suzgun M, Scales N, Schärli N, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. 2022. ArXiv:221009261
9 Stiennon N, Ouyang L, Wu J, et al. Learning to summarize with human feedback. In: Proceedings of the Advances in Neural Information Processing Systems, 2022. 3008–3021
10 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. In: Proceedings of the Advances in Neural Information Processing Systems, 2020. 1877–1901
11 Scao T L, Fan A, Akiki C, et al. Bloom: a 176b-parameter open-access multilingual language model. 2022. ArXiv:221105100
12 Zeng A, Liu X, Du Z X, et al. GLM-130b: an open bilingual pre-trained model. In: Proceedings of the International Conference on Learning Representations, 2023
13 Wei J, Bosma M, Zhao V, et al. Finetuned language models are zero-shot learners. In: Proceedings of the International Conference on Learning Representations, 2022
14 Mishra S, Khashabi D, Baral C, et al. Reframing instructional prompts to GPTk's language. In: Proceedings of the Findings of the Association for Computational Linguistics, Dublin, 2022. 589–612
15 Wei J, Wang X Z, Schuurmans D, et al. Chain of thought prompting elicits reasoning in large language models. In: Proceedings of the Advances in Neural Information Processing Systems, 2022. 24824–24837
16 Kojima T, Gu S X S, Reid M, et al. Large language models are zero-shot reasoners. In: Proceedings of the Advances in Neural Information Processing Systems, 2022. 22199–22213
17 Zhang Z S, Zhang A, Li M, et al. Automatic chain of thought prompting in large language models. In: Proceedings of the International Conference on Learning Representations, 2023
18 Zhou D, Schärli N, Hou L, et al. Least-to-most prompting enables complex reasoning in large language models. In: Proceedings of the International Conference on Learning Representations, 2023
19 Khot T, Trivedi H, Finlayson M, et al. Decomposed prompting: a modular approach for solving complex tasks. In: Proceedings of the International Conference on Learning Representations, 2023
20 Gilardi F, Alizadeh M, Kubli M. ChatGPT outperforms crowd-workers for text-annotation tasks. 2023. ArXiv:230315056
21 Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2020. ArXiv:1910.10683
22 Lin C Y. ROUGE: a package for automatic evaluation of summaries. In: Proceedings of the Text Summarization Branches Out, Barcelona, 2004. 74–81
23 Ouyang L, Wu J, Jiang X, et al. Training language models to follow instructions with human feedback. In: Proceedings of the Advances in Neural Information Processing Systems, Barcelona, 2022. 27730–27744
24 Fleiss J L. Measuring nominal scale agreement among many raters. Psycholog Bull, 1971, 76: 378–382