Chain-of-Instructions: Compositional Instruction Tuning on Large Language Models

Shirley Anugrah Hayati[*1], **Taehee Jung**[2], **Tristan Bodding-Long**[2]
Sudipta Kar[2], **Abhinav Sethy**[†3], **Joo-Kyung Kim**[2], **Dongyeop Kang**[1]
[1]University of Minnesota
[2]Amazon
[3]Grammarly
hayat023@umn.edu, dongyeop@umn.edu

Abstract

Fine-tuning large language models (LLMs) with a collection of large and diverse instructions has improved the model's generalization to different tasks, even for unseen tasks. However, most existing instruction datasets include only single instructions, and they struggle to follow complex instructions composed of multiple subtasks. In this work, we propose a novel concept of compositional instructions called *chain-of-instructions* (CoI), where the output of one instruction becomes an input for the next like a chain. Unlike the conventional practice of solving single instruction tasks, our proposed method encourages a model to solve each subtask step by step until the final answer is reached. CoI-tuning (i.e., fine-tuning with CoI instructions) improves the model's ability to handle instructions composed of multiple subtasks as well as unseen composite tasks such as multilingual summarization. Overall, our study finds that simple CoI tuning of existing instruction data can provide consistent generalization to solve more complex, unseen, and longer chains of instructions.

Code and Datasets — {https://github.com/amazon-science/chain-of-instructions}

Introduction

Large language models (LLMs) have demonstrated impressive performance in various tasks, from conventional NLP downstream tasks, such as machine translation and summarization, to open-ended tasks, such as writing an outline for blog posts and giving tips for presentation, when fine-tuned on human-like instructions (Ouyang et al. 2022; Wang et al. 2022; Conover et al. 2023; Mishra et al. 2022). These models excel at single instruction tasks, but their ability to handle complex and compositional instructions is less explored.

A compositional instruction contains a series of sequential subtasks, as the output of one subtask becomes the input of the next one in a chained manner as shown in Figure 1. We call this problem as *Chain-of-Instructions* or shortly CoI.

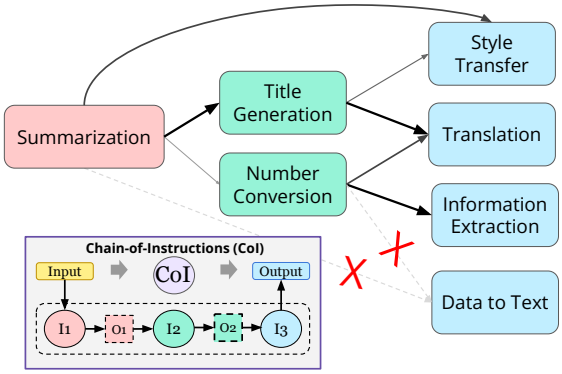

Figure 1: Chain-of-Instructions (CoI) example. The summarization output can be an input for a title generation subtask; the output of the title generation can be an input for style transfer or translation subtasks. Arrow thickness denotes the probability of instruction composability. X means that these subtasks cannot be composed due to format mismatch. I_k is k^{th} instruction and O_k is k^{th} output.

We examine what subtasks can be composed more naturally than others.

In Figure 1, we can see that some tasks can be composed together, such as input a summary to a title generation task, while some tasks cannot be composed together, e.g., a summary as input for a Data-to-Text task. Some tasks have a higher probability of being able to be composed, such as generating a title from a summary compared to converting numbers in a summary since sometimes a summary does not contain a number. Figure 2 illustrates a more detailed example of CoI. The given instruction "*Generate a blog-like title in French*" can be decomposed into three chained sub-instructions:

1. Generate a title for the given text
2. Convert the style of the title to be similar to a blog post title
3. Translate the blog post title into French

When these sub-instructions are composed, we call it a compositional instruction or chain-of-instructions. Our study investigates whether LLMs can handle compositional

[*] Work was partially done during internship at Amazon.
[†] Work was done while at Amazon.

	Instruction	Composed	Data Size	Domain
Chain-of-Instructions (Ours)	✓	✓	18k	NLP tasks
Self-Instruct (Wang et al. 2023)	✓	✗	52k	Daily QA-like tasks
Dolly (Conover et al. 2023)	✓	✗	15k	Daily QA-like tasks
Super-NaturalInstruct (Wang et al. 2022)	✓	✗	1.6k	NLP tasks
Faith and Fate (Dziri et al. 2023)	✗	✓	N/A	Math, logic, programming
Compositional Semantic (Drozdov et al. 2022)	✗	✓	N/A	CFQ, COGS, Parsing
MuSiQue (Trivedi et al. 2022)	✗	✓	24.8k	Multi-hop QA

Table 1: A comparison of our work with existing related works. As some previous works do not contribute a new dataset, the dataset size is shown as N/A. For instruction datasets, data size refers to the number of instructions, not task instances (input-output pair). More prior work is studied in §.

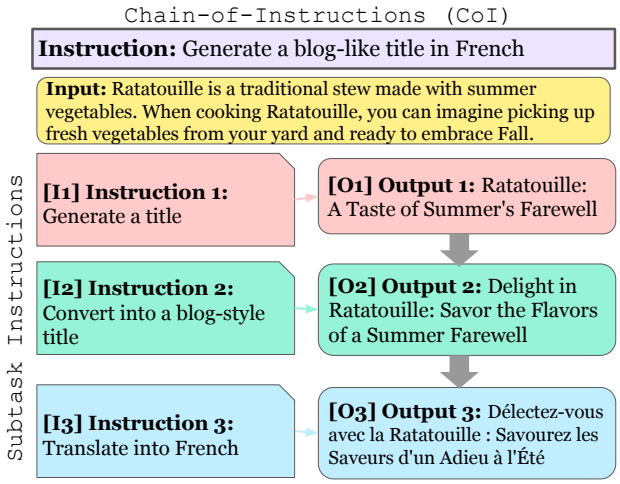

Figure 2: An example of the Chain-of-Instructions task. The last output is the expected output of the CoI.

instructions effectively and whether models tuned with compositional instructions can be generalized to solve more *complex*, *unseen*, or *longer chains of instructions*. We first create a new CoI dataset with our proposed LLM-based compositionality checker, and then evaluate our model's performance in handling (1) traditional single instructions and (2) compositional instructions.

Our work is closely related to other instruction-tuning works and compositional studies in NLP, as summarized in Table 1. Wang et al. (2023); Conover et al. (2023); Wang et al. (2022) propose new instruction datasets, but they only handle single instruction problems. Although our approach draws inspiration from Chain-of-Thought (CoT) prompting (Wei et al. 2022b) or Least-to-Most prompting (Zhou et al. 2022), our CoI is not a prompting technique but a collection of chained instructions validated by an LLM, showing generalization in solving complex and compositional problems. Our contributions are as follows:

- We introduce a novel task called Chain-of-Instructions (CoI) to examine LLMs' capabilities in following compositional instructions by creating a new benchmark dataset.
- We develop a framework to automatically construct composed instruction datasets with minimal human supervision. The framework leverages in-context learning on existing single-instruction datasets to create CoIs.
- We propose a method for enabling LLMs to solve compositional tasks in an explainable way. As an example, a model can generate incremental outputs at each step of a complex task chain. With CoI-tuning, step-by-step instruction following becomes easier, especially when dealing with instructions composed of multiple subtasks.
- We demonstrate through experiments and analysis that the CoI-tuned model outperforms both individual instructions and sequential compositional instructions. By training on CoI data, the model achieves higher performance. This result also generalizes for unseen longer chain test sets and downstream tasks.

Chain-of-Instructions

Formulation

Compositional instructions contain multiple subtask instructions where the output from one subtask becomes the input for the next subtask similarlity to a composition function in math. Thus, we formalize the problem of chain-of-instructions as follows:

Definition 1 (Chain of Instructions)**.** Given a tuple of <instruction I, input X, output Y>, let $I(X) = Y$ refer that an LLM generates output Y with instruction I and input X. A sequence of instructions $\{I_1, ..., I_k\}$ is a chain of instructions with length k if $I_{i+1} \circ I_i(X_i) = Y_{i+1}$, for all $i \in \{\mathbb{N} : 1 \leq i \leq k\}$.

Automatic Dataset Creation Pipeline

Seed Datatsets We curate a new compositional instruction dataset from existing single task instruction dataset: SUPER-NATURALINSTRUCTIONS (SUP-NATINS) (Wang et al. 2022). We select SUP-NATINS as the seed dataset because it contains a wide variety of tasks (1,616 unique tasks) from 76 categories, including text categorization, summarization, and machine translation. Each category contains many different NLP tasks. For example, under the text categorization category, there exist sarcasm detection and politeness classification tasks. Each task in SUP-NATINS con-

tains human-written descriptions that can be considered as instructions and instances as pairs of inputs and outputs. We only select tasks with English (1,341 unique tasks) as their input language to make sure that the chain is connected. For our single-task instruction tuning data (CoI_1), we randomly sample 10 input-output pairs, resulting in 13,410 instances.

Instruction Composition Composing two single instructions poses a challenge due to their lengthy and specific descriptions, and differing output formats. Figure 3 illustrates a two-step process for creating a compositional instruction dataset with the help of an LLM as elaborated in the following paragraphs. Here we use GPT 3.5 Turbo (Ouyang et al. 2022) because of its reasonable price and at the time, we examine that the quality of the result is good enough. However, this data creation procedure can be reproducible with other strong LLMs as well.

Step 1: Single instruction summarization The task instructions in SUP-NATINS are lengthy and detailed, which may deviate from real human-like instructions. With the same dataset (SUP-NATINS), Yin et al. (2023) find that 60% tokens can be removed with comparable, if not better, model performance. Thus, we use the LLM to shorten each instruction in the SUP-NATINS dataset. This step reduces the average number of words in the SUP-NATINS descriptions from 62.19 to 14.33.

Step 2: Composability check To generate compositional instructions from single instructions, we perform a two-step process: (1) validity check and (2) generate the output for the second (or third) subtask. The validity check is performed to examine whether two subtasks are composable. We first filter out non-composable tasks with heuristics developed by the authors' knowledge (the Heuristics for Validity Check section in the Appendix). For example, classification tasks can only be the last subtask when composing a pair of tasks. After applying these heuristics, we additionally check whether LLM can generate the output for the second instruction based on the input of the first instruction. If so, we treat the pair as composable.[1]

For the pairs that pass the validity check, we generate the new output using the first output and second instruction for the second task. This generated output serves as the ground truth for the second subtask in the instruction-tuning phase. Our approach is a variation of distillation from a larger LLM as has been done by previous works for different problems (Gu et al. 2024; Hsieh et al. 2023; West et al. 2022). We define compositional instructions originating from two instructions as CoI_2 and those originating from three instructions CoI_3. CoI_3 is created by chaining two CoI_2s if there exists $I_x \circ I_y$ and $I_y \circ I_z$, resulting in $CoI_3 = I_x \circ I_y \circ I_z$. The same method is applied for creating longer chains such as CoI_4 and CoI_5.

To examine the quality of LLM's composability check, we randomly sampled 100 instances and manually inspected which composed instructions are valid. We find that 75% are valid composed instructions. For CoI_3, similarly we randomly sampled 100 instances and found that 59% are valid compositions. Such error rates are often found in LLM-generated data (Das et al. 2024; Wang et al. 2023).

[1]Prompt for this step is available in the Appendix.

chain length (σ)	train	test
1	13,410	-
2	2,993	588
3	2,187	480
4	-	844
5	-	355

Table 2: Dataset statistics per chain length.

CoI Dataset

Table 2 shows the data statistics of CoI datasets. In chain length 2, we obtain 970 unique category pairs; in chain length 3, we obtain 418 unique category triplets. In each pair or triplet, we randomly select at most three instances and divide them into training and testing sets. For the longer chains (4, 5), we only use them for testing. Please find Appendix **??** for the detailed statistics.

Figure 4 shows a t-SNE plot when we embed subtask instructions of frequent CoI_2 instructions using Sentence-BERT (Reimers and Gurevych 2019) with DistilRoberta (Sanh et al. 2019).[2] We find generation tasks such as paraphrasing and question generation can be compiled as both the first and second subtasks, except for problems involving specific input formats, such as code to text or data to text, which can only be compiled as the second subtask. On the other hand, close-ended problems (e.g., POS tagging or grammar error detection) mostly appear as the second subtask.

Experiment Setup

CoI models We fine-tune the base models of Alpaca-7B (Taori et al. 2023) and Mistral-7B-Instruct (Jiang et al. 2023). Since both models are open-sourced single instruction-tuned models which are widely used, they are suitable to be compared with CoI-tuned models.

Baselines

- Off-the-shelf version of Alpaca-7B (Taori et al. 2023) model and Mistral-7B-Instruct model without fine-tuning (Base).
- The same non-finetuned Alpaca and Mistral with chain-of-thought prompting (Wei et al. 2022b) (CoT) with seven-shot demonstrations and least-to-most prompting (Zhou et al. 2022) (LtM).
- Fine-tuned base models with a subset of single-instruction SUP-NATINS dataset (CoI_1).

Metrics For our evaluation metric, we report ROUGE-L (Lin 2004), following Wang et al. (2022) and LLM (gpt-4o-mini) as a preference judge. ROUGE can be used to assess various text generation tasks and using LLM as a judge has

[2]We only select instruction pairs that appear more than 7 times, and 9 is a max number of occurrences in CoI_2 dataset.

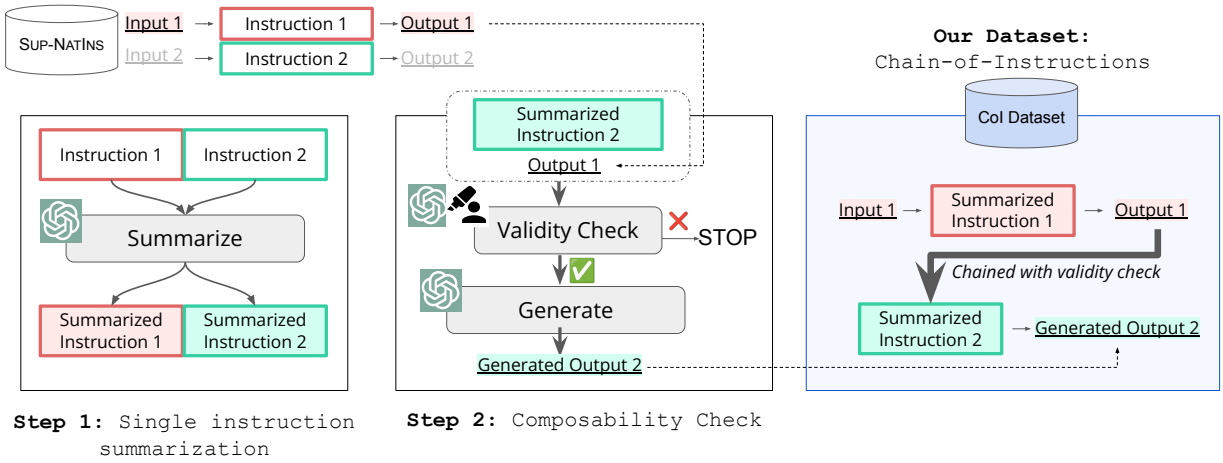

Figure 3: Data creation for CoI_2. We use an LLM for both instruction summarization and composability check. The right column shows an example instance of our chain-of-instruction dataset. Output 1 in Step 2 comes from the original SupNatInst data.

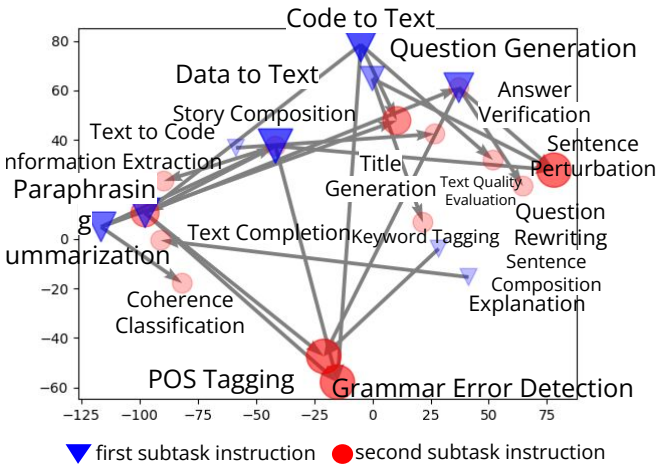

Figure 4: T-SNE of sentence embeddings for most frequent compositional instructions with CoI_2.

been widely adopted in NLP research (Liu et al. 2023; Fu et al. 2024). We also have human evaluation to perform blind pairwise comparison between the outputs from the baseline and from our best CoI models.

Test sets To assess the compositionality of our models, we prepare three types of evaluation suites.

- **CoI Test set** For the compositional instruction evaluation, we tested the models on CoI test sets with $\sigma = \{2, 3, 4, 5\}$ where σ is a chain length.
- **BIG-Bench Hard** For the single instruction test set, we use BIG-Bench Hard (Suzgun et al. 2022), a collection of 27 challenging tasks, such as date understanding and evaluating the truth value of Boolean expressions, and each task has $\leq$ 250 instances. BIG-Bench Hard subset enables us to evaluate the model's performance on diverse and challenging NLP tasks with clear single instructions and associated input-output pairs.
- **Downstream Task** In addition to CoI test sets, we examine the usefulness of CoI on the downstream task of multilingual summarization using WikiLingua (Ladhak et al. 2020), which is a multilingual dataset based on WikiHow[3] for abstractive summarization in 18 languages. WikiHow articles provide step-by-step instructions to complete procedural tasks on different topics, and each step includes a one-sentence summary as well. In our experiment, we select source-target language pairs ; English-to-French (*WikiLingua-en-fr*) and Spanish-to-English (*WikiLingua-es-en*) and randomly sample 300 test instances for each. Given an input content from source language L_{src}, we aim to generate a summary in target language L_{tgt}. This task is similar to a 2-instruction problem as we summarize first and then translate. Note that CoI training data only contains translation tasks from English to Punjabi, German, and Catalan, thus, selected source-target pairs are unseen in CoI training set.

Results

We conduct experiments to measure the performance of CoI-tuned models on our compositional instructions (§), and the generalization capability to difficult single instructions (§), and longer-chain instructions $\sigma = \{4, 5\}$ (§), and the application to an existing downstream task (§). We also conduct an ablation study to see if the correctness of second and third subtask outputs matter in Appendix. We see degrading performance of models fine-tuned with incorrect outputs, showing the importance to have the correct output for the subtasks during training.

Performance on In-domain Composite Tasks ($\text{CoI}_{2,3}$)

Automatic metric As we evaluate our CoI models' performance against the baselines on multi-CoI test sets, we find

[3]https://www.wikihow.com

	Mistral		Alpaca	
	Base	CoI	Base	CoI
Test Set: CoI_2				
Subtask 1	1.32	**90.50**	13.98	**84.16**
Subtask 2	2.40	**49.21**	7.02	**45.57**
Test Set: CoI_3				
Subtask 1	18.04	**81.49**	9.56	**91.77**
Subtask 2	6.82	**68.65**	2.13	**71.67**
Subtask 3	6.93	**32.73**	3.30	**35.52**

Table 3: ROUGE-L results on intermediate tasks. CoI models refer to best models of CoI: CoI_{12} model if the test set=CoI_2 and CoI_{123} model if the test set=CoI_3.

Model	CoI_2-test		CoI_3-test		BBH	
	Mistral	Alpaca	Mistral	Alpaca	Mistral	Alpaca
Baselines						
Base	24.93	24.95	23.66	20.99	8.51	14.36
CoT	16.61	23.82	16.90	20.09	5.84	17.05
LtM	15.07	23.54	16.41	19.79	3.99	3.99
CoI_1	39.72	32.32	29.62	21.75	27.68	28.74
Chain-of-Instructions Models						
CoI_2	60.43	62.04	48.31	48.23	10.65	12.11
CoI_3	33.63	31.62	60.03	47.03	5.78	7.00
CoI_{12}	**70.76**	**67.50**	59.84	50.23	24.44	**28.80**
CoI_{123}	45.16	67.12	**61.61**	**67.49**	**29.39**	27.57

Table 4: ROUGE-L results on compositional instruction test sets and BIG-Bench Hard (BBH). Base refers to the non-fine-tuned base models, CoT = chain-of-thought prompting on base models, LtM = least-to-most prompting on base models. The best scores are marked as **bold**.

that both Mistral and Alpaca fine-tuned on CoI_{12} instructions perform the best for CoI_2-test (Table 4) [4]. Similarly, for CoI_3-test, both CoI_{123} Mistral and Alpaca perform the best. All models fine-tuned on compositional instructions generally outperform the baselines, except for CoI_3-tuned Alpaca. This model performs slightly worse than the CoI_1-tuned Alpaca on CoI_2 test set. We hypothesize that this happens because instructions in CoI_3 become very long, thereby it becomes harder for the model to generalize without CoI_2 and CoI_1 examples. As a result, models only fine-tuned on CoI_3 tend to generate long sentences with hallucinations as in Table 5.

In the LLM-as-a-judge experiment, we evaluate the performance of the best CoI models on CoI_2-test and CoI_3-test against the best baseline, CoI_1. On CoI_2-test, the LLM prefers 69.90% of Alpaca CoI_{12}'s outputs and 70.92% of Mistral CoI_{12}'s outputs over the baseline. Similarly, on CoI_3-test, the LLM favors Alpaca CoI_{123}'s outputs and Mistral CoI_{123}'s outputs over the baseline by 81.04% and 60.00%, respectively.

[4]Fine-tuning details in Appendix.

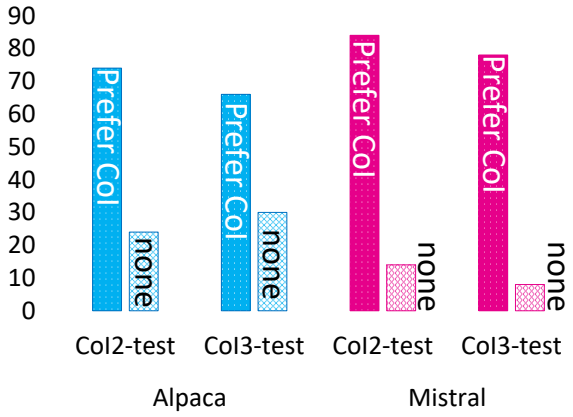

Figure 5: Human evaluation results. "Prefer CoI" refers to the percentage of CoI outputs preferred by humans; "none" refers to when humans think the outputs for both first and second subtasks are incorrect.

CoI Results per Subtasks We examine how CoI models perform for each subtask in the compositional instruction. To do this, we compare the results from the best version of CoI (CoI_{12} for CoI_2 test set, CoI_{123} for CoI_3 test set) against the non-finetuned baseline models. Since there is no clear boundary to distinguish the first subtask output and the second subtask output in the baseline's outputs, we use an LLM to separate the responses. Given the subtask instruction and the output, we ask the LLM to decide which span of the output text responds to the subtask instruction. To remove the possibility of LLM's hallucination being counted as part of the output, we only include LLM's output span when it appears in the baseline's output. When LLM deems that the output is incorrect, we assign ROUGE = 0 because this output could refer to the first subtask or second subtask.

Table 3 and Table 4 show results of CoI models and baseline on CoI_2 and CoI_3 test sets. In general, CoI models outperform the baseline for both test sets, with the best results on the first subtask and then followed by the second subtask and the third subtask. However, it is interesting to see that CoI's performance on the second subtask for CoI_3 test set is higher than CoI's performance on the second subtask for CoI_2. We conjecture that it happens because the previous subtasks can be easier to solve or knowing the subsequent subtask helps improving the performance.

Human evaluation We randomly sample 200 instances from CoI_2-test and CoI_3-test for both Mistral and Alpaca and ask 8 computer science students with NLP knowledge to rate the output. We show 25 sets of instruction, input, baseline output, and CoI output to each annotator. The order of the outputs is randomized. Each annotator then selects the output they prefer. Figure 5 depicts the percentage of CoI outputs that humans prefer vs. when none of the outputs are preferred. Outputs from CoI-tuned models are preferred for all test sets for both models.

Instruction: Simplify the given sentence by paraphrasing it. and then Determine if the paraphrased sentence has proper punctuation with True or False.		
Input: many of the churches work together for town-wide projects under the banner of "churches together in stevenage".		
Correct output: Task 1 output and task 2 input: many of the churches work together for projects across the town under the slogan of "churches together in stevenage". Task 2 output: False		
Model	**Output**	**ROUGE-L**
CoI_1	many of the churches work together for town wide projects under the banner of " churches together in stevenage ".	75.00
CoI_2	Task 1 output and task 2 input: many of the churches work together for projects across the town under the slogan of " churches together in stevenage ". Task 2 output: False	**100**
CoI_3	1 output and 2 input: many of the churches work together for projects across the town under the slogan of " churches together in stevenage ". 2 output and 3 input: The pronoun 'them' refers to the noun phrase 'many of the churches' because the sentence states that 'they' work together for projects across the town. This coreference is justified by the knowledge that the churches are the ones that are working together. 3 output: False	43.75
CoI_{12}	1 output and 1 input: many of the churches work together for projects across the town under the slogan of " churches together in stevenage ". 2 output: False	85.71
CoI_{123}	Task 1 output and task 2 input: many of the churches work together for projects across the town under the slogan of " churches together in stevenage ". Task 2 output: False	**100**

Table 5: Example outputs from different CoI-tuned Alpaca on a 2-instruction test instance.

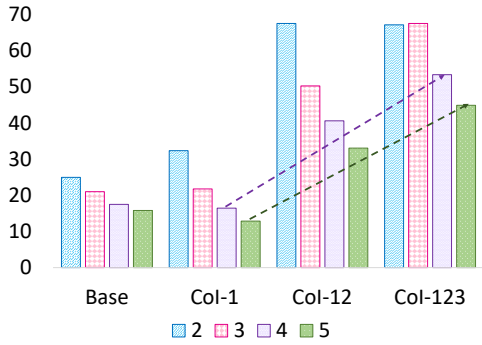

Figure 6: ROUGE-L (x-axis) on CoI test sets $\sigma = 2, 3, 4, 5$ for various Alpaca models (y-axis). Base refers to the non-fine-tuned Alpaca.

Generalization to Unseen Single Tasks

To assess whether adding compositional instructions helps improve the model's performance on unseen and difficult single instruction tasks, we tested CoI-tuned models on BIG-Bench Hard (BBH). CoI_{123}-tuned Mistral performs the best (ROUGE: 29.39) as shown in Table 4. For Alpaca, the model fine-tuned on CoI_{12} is also better than the baseline and achieves a higher ROUGE score of 28.80. This confirms that having compositional instructions helps the model to understand hard single instruction problems as well.

Generalization to Longer Chains ($CoI_{4,5}$)

In this experiment, we examine whether our CoI models can generalize to longer chains. We run inference on CoI_4 and CoI_5 test sets using CoI_1, CoI_{12}, and CoI_{123}-tuned Alpaca. [5] As shown in Figure 6, longer chains ($\sigma = 2, 3$) in the training set help the model to understand unseen longer chain ($\sigma = 4, 5$) in the test set as well. Moreover, the performance does not drop as high as CoI_1 or the baseline non-fine-tuned models that do not learn the chaining reasoning. We posit that the knowledge of compositional instructions in the training set, even though the length of the chain is shorter than 4 or 5, still helps the model to understand the composed tasks.

Generalization to Downstream Composite Tasks

For this experiment, we use CoI_{12} because it shows the highest ROUGE-L on 2-instruction problem. For the baseline, we use non-finetuned Alpaca and Mistral. We evaluate the performance of the models using four metrics below.

- **ROUGE-L (all)** the ROUGE score of the summary of the whole generated output.
- **ROUGE-L (src)** the ROUGE score only from the summary in the source language.
- **ROUGE-L (tgt)** the ROUGE score only from the summary in the target language.
- **#valid outputs** number of valid summaries in the source and the target languages are generated because sometimes the model may not generate them properly.

Table 6 shows the results for our downstream task experiments. For the English-to-French summarization task, CoI_{12} can generate more valid target outputs than the baselines. Moreover, CoI_{12} obtains higher ROUGE for both source and

[5] Mistral results are in Appendix.

Metric	Mistral		Alpaca	
	Base	CoI_{12}	Base	CoI_{12}
English to French				
ROUGE-L (all)	8.03	**10.97**	5.78	**8.90**
ROUGE-L (src)	10.68	**15.66**	3.84	**12.71**
ROUGE-L (tgt)	7.45	**10.93**	5.46	**7.96**
#valid src outputs	206	**295**	126	**228**
#valid tgt outputs	212	**295**	221	**228**
Spanish to English				
ROUGE-L (all)	11.22	**12.43**	7.87	**10.39**
ROUGE-L (src)	0.07	**4.85**	**2.47**	1.87
ROUGE-L (tgt)	11.22	**12.30**	**7.68**	7.13
#valid src outputs	1	**290**	80	**150**
#valid tgt outputs	**300**	290	**240**	150

Table 6: Results of the multilingual summarization task on 300 instances. Base refers to non-fine-tuned baseline, src is source language, and tgt is target language.

target summaries than the baselines. For the Spanish-to-English summarization task, CoI_{12} Mistral outperforms the baseline for all ROUGE-L scores, but Alpaca fails to have better ROUGE-L (src) and ROUGE-L (tgt) against the baseline.

In general, CoI performs better in English-to-French summarization compared to Spanish-to-English summarization because our training instances contain a translation task from English to other languages (Punjabi, German, and Catalan), even though the target language of the translation task in the training set is not French. On the other hand, we see poor performance in Spanish summaries across all models, possibly due to the lack of Spanish as the first subtask in training datasets. We conjecture this issue could be resolved if we add more Spanish tasks during the fine-tuning stage.

Related Work

Instruction tuning There has been a notable surge in research focused on fine-tuning LLMs using human instructions. Efrat and Levy (2020) examined LLMs' ability to follow natural language instructions compared to crowdworkers. Wei et al. (2022a); Sanh et al. (2021) have transformed NLP task descriptions into human-like language instructions and showed that LLMs fine-tuned with those instructions have generalizable capability toward unseen tasks (Chung et al. 2024). Subsequently, many studies have emerged to create new instruction datasets aimed at training models in instruction-tuning paradigm: some instruction datasets are fully written by humans (Wang et al. 2022; Conover et al. 2023), the others are written with the help of LLMs (Honovich et al. 2023; Wang et al. 2023; Taori et al. 2023); some instructions are NLP-specific (Mishra et al. 2022; Wang et al. 2022; Weller et al. 2020), and the others are designed to respond to general-purpose instructions (Ouyang et al. 2022; Wang et al. 2023). These prior studies only work on single instruction datasets, so we construct a new compositional dataset upon Wang et al. (2022)'s SUPER-NATURALINSTRUCTION. Our work is also related to several past works which have leveraged LLMs to generate training data (Schick and Schütze 2021), and some of them specifically use LLMs for generating instruction data (Peng et al. 2023; Shao et al. 2023). Nevertheless, our CoI data generation framework differs from previous works as we use LLMs to determine the composability of individual instructions, and then generate responses for subsequent subtask instructions.

Compositional problems in NLP Several NLP work have investigated the capability of Transformer model on compositional problems including algorithm and math problems (Dziri et al. 2023), compositional semantics (Drozdov et al. 2022), and multi-hop question-answering (QA) tasks (Trivedi et al. 2022). Dziri et al. (2023) highlight how Transformer models often struggle with compositional mathematics computation or program executions (Nye et al. 2022; Saparov and He 2022). Drozdov et al. (2022) introduce a new prompting method which first decomposes the compositional questions or sentences (Keysers et al. 2019; Kim and Linzen 2020), then sequentially predicts the answers to subproblems, and finally generating the final output. Compositionality in NLP is closely related with multi-hop QA problems with compositional questions where the answers from sub-questions are needed to answer the main question (Yang et al. 2018; Ho et al. 2020; Trivedi et al. 2022). Qiu et al. (2022) have shown how a model with compositional latent structure improves large language models' performance on compositional generalization tasks through synthethic data augmentation. CoI is most related to Aksu et al. (2023) as they work on dealing with compositional tasks for dialogue systems. However, their definition of "compositional task" is different from ours as they do not require the output of one subtask is shown to the next subtask. Meanwhile, in our CoI, the outputs from the previous subtasks become the next subtask inputs.

Conclusion and Future Work

In this work, we propose a new task called Chain-of-Instructions and develop a dataset for building models to solve the task. We introduce an automatic pipeline on how to build our dataset and demonstrate the usefulness of our CoI-tuned models on the tasks of the generated dataset and downstream tasks. Since human language is complex and an instruction may actually be composed of subtasks, it is important to have a model that can deal with compositional instructions, especially as we show that models fine-tuned only on single instructions never outperform the CoI-tuned models on multi-instruction tasks. For future work, we consider looking into instruction decomposition in addition to the instruction composition problem. We also recommend trying out more tasks to be composed besides those from SUPERNATURALINSTRUCTION.

Acknowledgements

We would like to thank members of the MinnesotaNLP lab for their feedback and intellectual support.

References

Aksu, T.; Hazarika, D.; Mehri, S.; Kim, S.; Hakkani-Tür, D.; Liu, Y.; and Namazifar, M. 2023. CESAR: Automatic Induction of Compositional Instructions for Multi-turn Dialogs. In *EMNLP*.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Castro-Ros, A.; Pellat, M.; Robinson, K.; Valter, D.; Narang, S.; Mishra, G.; Yu, A.; Zhao, V.; Huang, Y.; Dai, A.; Yu, H.; Petrov, S.; Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; and Wei, J. 2024. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70): 1–53.

Conover, M.; Hayes, M.; Mathur, A.; Xie, J.; Wan, J.; Shah, S.; Ghodsi, A.; Wendell, P.; Zaharia, M.; and Xin, R. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.

Das, D.; De Langis, K.; Martin, A.; Kim, J.; Lee, M.; Kim, Z. M.; Hayati, S. A.; Owan, R.; Hu, B.; Parkar, R.; et al. 2024. Under the Surface: Tracking the Artifactuality of LLM-Generated Data. *arXiv preprint arXiv:2401.14698*.

Drozdov, A.; Schärli, N.; Akyürek, E.; Scales, N.; Song, X.; Chen, X.; Bousquet, O.; and Zhou, D. 2022. Compositional Semantic Parsing with Large Language Models. In *ICLR*.

Dziri, N.; Lu, X.; Sclar, M.; Li, X. L.; Jian, L.; Lin, B. Y.; West, P.; Bhagavatula, C.; Bras, R. L.; Hwang, J. D.; Sanyal, S.; Welleck, S.; Ren, X.; Ettinger, A.; Harchaoui, Z.; and Choi, Y. 2023. Faith and Fate: Limits of Transformers on Compositionality. In *NeurIPS*.

Efrat, A.; and Levy, O. 2020. The turking test: Can language models understand instructions? *arXiv preprint arXiv:2010.11982*.

Fu, J.; Ng, S.-K.; Jiang, Z.; and Liu, P. 2024. GPTScore: Evaluate as You Desire. In Duh, K.; Gomez, H.; and Bethard, S., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 6556–6576. Mexico City, Mexico: Association for Computational Linguistics.

Gu, Y.; Dong, L.; Wei, F.; and Huang, M. 2024. MiniLLM: Knowledge Distillation of Large Language Models. In *ICLR*.

Ho, X.; Duong Nguyen, A.-K.; Sugawara, S.; and Aizawa, A. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In Scott, D.; Bel, N.; and Zong, C., eds., *Proceedings of the 28th International Conference on Computational Linguistics*, 6609–6625. Barcelona, Spain (Online): International Committee on Computational Linguistics.

Honovich, O.; Scialom, T.; Levy, O.; and Schick, T. 2023. Unnatural Instructions: Tuning Language Models with (Almost) No Human Labor. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 14409–14428. Toronto, Canada: Association for Computational Linguistics.

Hsieh, C.-Y.; Li, C.-L.; Yeh, C.-k.; Nakhost, H.; Fujii, Y.; Ratner, A.; Krishna, R.; Lee, C.-Y.; and Pfister, T. 2023. Distilling Step-by-Step! Outperforming Larger Language Models with Less Training Data and Smaller Model Sizes. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 8003–8017. Toronto, Canada: Association for Computational Linguistics.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Keysers, D.; Schärli, N.; Scales, N.; Buisman, H.; Furrer, D.; Kashubin, S.; Momchev, N.; Sinopalnikov, D.; Stafiniak, L.; Tihon, T.; et al. 2019. Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In *International Conference on Learning Representations*.

Kim, N.; and Linzen, T. 2020. COGS: A Compositional Generalization Challenge Based on Semantic Interpretation. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 9087–9105. Online: Association for Computational Linguistics.

Ladhak, F.; Durmus, E.; Cardie, C.; and McKeown, K. 2020. WikiLingua: A New Benchmark Dataset for Cross-Lingual Abstractive Summarization. In Cohn, T.; He, Y.; and Liu, Y., eds., *Findings of the Association for Computational Linguistics: EMNLP 2020*, 4034–4048. Online: Association for Computational Linguistics.

Lin, C.-Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Liu, Y.; Iter, D.; Xu, Y.; Wang, S.; Xu, R.; and Zhu, C. 2023. G-Eval: NLG Evaluation using Gpt-4 with Better Human Alignment. In Bouamor, H.; Pino, J.; and Bali, K., eds., *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2511–2522. Singapore: Association for Computational Linguistics.

Mishra, S.; Khashabi, D.; Baral, C.; and Hajishirzi, H. 2022. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In Muresan, S.; Nakov, P.; and Villavicencio, A., eds., *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3470–3487. Dublin, Ireland: Association for Computational Linguistics.

Nye, M.; Andreassen, A. J.; Gur-Ari, G.; Michalewski, H.; Austin, J.; Bieber, D.; Dohan, D.; Lewkowycz, A.; Bosma, M.; Luan, D.; et al. 2022. Show Your Work: Scratchpads for Intermediate Computation with Language Models. In *Deep Learning for Code Workshop*.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. In *NeurIPS*, 27730–27744.

Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J.

2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Qiu, L.; Shaw, P.; Pasupat, P.; Nowak, P.; Linzen, T.; Sha, F.; and Toutanova, K. 2022. Improving Compositional Generalization with Latent Structure and Data Augmentation. In *NAACL*, 4341–4362.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., eds., *EMNLP-IJCNLP*, 3982–3992. Hong Kong, China: Association for Computational Linguistics.

Sanh, V.; Debut, L.; Chaumond, J.; and Wolf, T. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sanh, V.; Webson, A.; Raffel, C.; Bach, S.; Sutawika, L.; Alyafeai, Z.; Chaffin, A.; Stiegler, A.; Raja, A.; Dey, M.; et al. 2021. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*.

Saparov, A.; and He, H. 2022. Language Models Are Greedy Reasoners: A Systematic Formal Analysis of Chain-of-Thought. In *The Eleventh International Conference on Learning Representations*.

Schick, T.; and Schütze, H. 2021. Generating Datasets with Pretrained Language Models. In Moens, M.-F.; Huang, X.; Specia, L.; and Yih, S. W.-t., eds., *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 6943–6951. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics.

Shao, Z.; Gong, Y.; Shen, Y.; Huang, M.; Duan, N.; and Chen, W. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *ICML*.

Suzgun, M.; Scales, N.; Schärli, N.; Gehrmann, S.; Tay, Y.; Chung, H. W.; Chowdhery, A.; Le, Q. V.; Chi, E. H.; Zhou, D.; et al. 2022. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. In *ACL Findings*, 13003–13051.

Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

Trivedi, H.; Balasubramanian, N.; Khot, T.; and Sabharwal, A. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics*, 10: 539–554.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *ACL*, 13484–13508.

Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Naik, A.; Ashok, A.; Dhanasekaran, A. S.; Arunkumar, A.; Stap, D.; Pathak, E.; Karamanolakis, G.; Lai, H.; Purohit, I.; Mondal, I.; Anderson, J.; Kuznia, K.; Doshi, K.; Pal, K. K.; Patel, M.; Moradshahi, M.; Parmar, M.; Purohit, M.; Varshney, N.; Kaza, P. R.; Verma, P.; Puri, R. S.; Karia, R.; Doshi, S.; Sampat, S. K.; Mishra, S.; Reddy A, S.; Patro, S.; Dixit, T.; and Shen, X. 2022. Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks. In *EMNLP*, 5085–5109.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022a. Finetuned Language Models are Zero-Shot Learners. In *ICLR*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, volume 35, 24824–24837.

Weller, O.; Lourie, N.; Gardner, M.; and Peters, M. E. 2020. Learning from Task Descriptions. In Webber, B.; Cohn, T.; He, Y.; and Liu, Y., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1361–1375. Online: Association for Computational Linguistics.

West, P.; Bhagavatula, C.; Hessel, J.; Hwang, J.; Jiang, L.; Le Bras, R.; Lu, X.; Welleck, S.; and Choi, Y. 2022. Symbolic Knowledge Distillation: from General Language Models to Commonsense Models. In Carpuat, M.; de Marneffe, M.-C.; and Meza Ruiz, I. V., eds., *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4602–4625. Seattle, United States: Association for Computational Linguistics.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. In Riloff, E.; Chiang, D.; Hockenmaier, J.; and Tsujii, J., eds., *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2369–2380. Brussels, Belgium: Association for Computational Linguistics.

Yin, F.; Vig, J.; Laban, P.; Joty, S.; Xiong, C.; and Wu, C.-S. 2023. Did You Read the Instructions? Rethinking the Effectiveness of Task Definitions in Instruction Learning. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3063–3079. Toronto, Canada: Association for Computational Linguistics.

Zhou, D.; Schärli, N.; Hou, L.; Wei, J.; Scales, N.; Wang, X.; Schuurmans, D.; Cui, C.; Bousquet, O.; Le, Q. V.; et al. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *The Eleventh International Conference on Learning Representations*.