Fine-Tuning Large Language Models with Sequential Instructions

Hanxu Hu[*,1,2] **Simon Yu**[*,1,3] **Pinzhen Chen**[*,1] **Edoardo M. Ponti**[1]
[1]University of Edinburgh [2]University of Zurich [3]Northeastern University
`hanxu.hu@uzh.ch, yu.chi@northeastern.edu, {pinzhen.chen,eponti}@ed.ac.uk`

Abstract

We find that existing instruction-tuned models usually struggle to adhere to a query with multiple intentions, which impairs their performance when the completion of several tasks is demanded by a single command. Hence, this paper teaches models to respond to **sequential instructions**. Our first attempt stems from a task-driven perspective, manually creating additional intermediate tasks to train multilingual and visual question answering. Next, we develop an automatic and generic process that turns instructions in existing data into diverse and complex task chains. Models that underwent sequential instruction tuning follow a list of instructions better and deliver higher results in coding, maths, and open-ended generation. Moreover, we put forward a new benchmark named **SeqEval** to evaluate a model's ability to follow all the instructions in a sequence, which further corroborates the benefits of our sequential instruction tuning method.

Method	Model	R-L	BS	Following		
		R	R	R	A	R+A
Base prompting	Mistral-7B	50	88	94	11	6
	Mixtral-8×7B	38	87	85	30	16
	Llama-2-13B	10	81	10	51	6
	Llama-2-70B	35	87	54	44	21
	Llama-3-8B	63	91	41	20	7
IT	Mistral-7B-Alpaca	48	88	64	56	45
	Llama-3-8B-Alpaca	42	82	33	69	27
SIT (ours)	Mistral-7B-Alpaca	55	92	99	85	84
	Llama-2-7B-Alpaca	54	92	89	91	85
	Llama-2-13B-Alpaca	39	90	99	93	93
	Llama-3-8B-Alpaca	53	93	96	95	95

Table 1: We test various base and instruction-tuned models on 100 samples of CommonsenseQA by asking them to repeat (R) the input and then answer (A) the question. We report ROUGE-L (R-L) and BERTScore (BS) on the repeated content and human-counted success rate (%) of models following instructions. Both base and IT'ed models lack the ability to follow the two-step sequential instruction which is significantly improved by our SIT.

1 Introduction

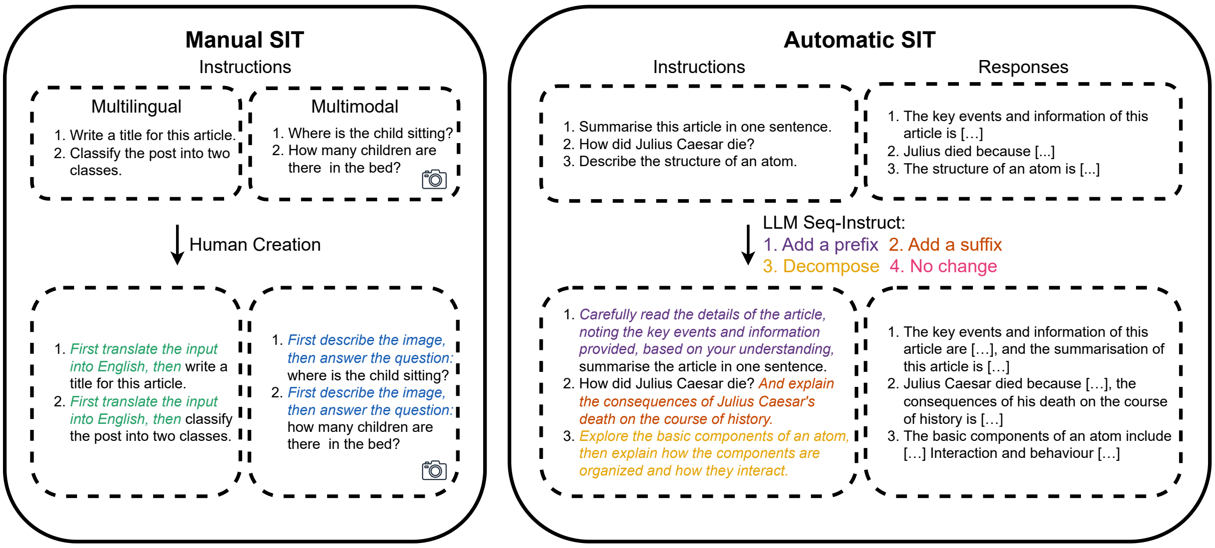
Instruction tuning (IT), or supervised fine-tuning (SFT), gives large language models (LLMs) the ability to execute tasks specified by users (Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022a). Nevertheless, popular instruction mixtures contain rather straightforward instructions derived from conventional NLP tasks or open-ended dialogues (Sanh et al., 2022; Taori et al., 2023; Conover et al., 2023). Hence, they may suffer from the absence of multiple instructions in one query. We speculate that this hinders the fine-tuned models from navigating a sequence of tasks or tools in a single command, which is arguably crucial for complex scenarios with reasoning (e.g., coding and maths) or knowledge pivoting (e.g., cross-lingual and cross-modal question answering, Shi et al., 2023; Zhang et al., 2024) Moreover, this detracts from user experience as models may fail to track whether all requests have been fulfilled. We empirically verify our hypothesis by prompting various state-of-the-art open-source LLMs, Llama 2/3 and Mistral (Dubey et al., 2024; Jiang et al., 2023), with simple two-step instructions—already more than they can shake a stick at as shown in Table 1. We find that not only did task accuracy degrade dramatically, but also that they often failed to follow the entire list of instructions, particularly for models fine-tuned on public datasets like Alpaca (Taori et al., 2023).

To tackle this, we propose a sequential instruction tuning (SIT) paradigm that uses simple strategies to automatically augment IT data with multi-step tasks. As a prototype, we start with manually defined intermediate steps that are task-focused and interpretable for multilingual or visual ques-

---

[*] Equal contribution. All resources are available at `https://seqit.github.io/`.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5589–5610
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics



1. Write a title for this article.
2. Classify the post into two classes.

1. Where is the child sitting?
2. How many children are there in the bed?

1. *First translate the input into English, then* write a title for this article.
2. *First translate the input into English, then* classify the post into two classes.

1. *First describe the image, then answer the question:* where is the child sitting?
2. *First describe the image, then answer the question:* how many children are there in the bed?

1. Summarise this article in one sentence.
2. How did Julius Caesar die?
3. Describe the structure of an atom.

1. The key events and information of this article is […]
2. Julius died because […]
3. The structure of an atom is […]

1. *Carefully read the details of the article, noting the key events and information provided, based on your understanding,* summarise the article in one sentence.
2. How did Julius Caesar die? *And explain the consequences of Julius Caesar's death on the course of history.*
3. *Explore the basic components of an atom, then explain how the components are organized and how they interact.*

1. The key events and information of this article are […], and the summarisation of this article is […]
2. Julius Caesar died because […], the consequences of his death on the course of history is […]
3. The basic components of an atom include […] Interaction and behaviour […]

Figure 1: The construction of sequential instruction data via manual (left) and automatic (right) processes.

tion answering (Artetxe et al., 2020; Hudson and Manning, 2019), namely "translate then predict" and "caption then answer". Furthermore, we generalise and automate the data creation pipeline where intermediate tasks are seeded from a single-task instruction. Our method stands in contrast with previous automatic augmentation methods that make instructions complex or diverse, but not sequential. Applying this to instruction mixtures such as Alpaca, FlanCoT, MetaMath, and Tulu-V2 (Taori et al., 2023; Longpre et al., 2023; Yu et al., 2023; Ivison et al., 2023) allows us to construct natural, diverse, and high-quality sequential instructions. Comparing LLMs fine-tuned with our SIT data and the original IT data, we observe significant boosts in factuality, reasoning, and open-ended generation. Ablation studies confirm SIT's generalisability to different models and tasks, and that the score gains are not merely due to inflated training tokens.

Finally, to confirm that sequential instruction-tuned LLMs acquire a better ability to execute all the instructions in a query, we develop and make public a new benchmark for open-ended generation, SeqEval. It is constructed by applying self-instruct (Wang et al., 2023) to the AlpacaEval benchmark (Li et al., 2023c) with an emphasis on chained tasks. With this benchmark, we find that SIT models are vastly superior in instruction-following behaviours. Altogether, we hope that the SIT suite presented in this paper: the methodology, the SIT data, and the SeqEval benchmark will contribute to endowing LLMs with the ability to solve chained tasks.

2 Methodology

2.1 Sequential instructions

Existing instruction data usually comprise single-step instructions (i.e. an instruction-response pair resembling one task); however, this falls short of equipping models with the ability to handle a query containing (explicitly or implicitly) multiple subtasks. Developing from a single-task instruction $\boldsymbol{i}$, we define a **sequential instruction** $\boldsymbol{s}$ as a query that contains multiple inter-related tasks or steps: $\boldsymbol{s} = \boldsymbol{i}_1 \oplus \boldsymbol{i}_2 \oplus \cdots \oplus \boldsymbol{i}_n$ with $n \geq 2$, where $\boldsymbol{i}_k$ denotes the kth task and $\oplus$ is a concatenation operation. Querying a model parameterized by θ leads to a response $\hat{\boldsymbol{y}} \sim p(\boldsymbol{y}|\boldsymbol{i}_1 \oplus \boldsymbol{i}_2 \oplus \cdots \oplus \boldsymbol{i}_n; \boldsymbol{\theta})$ which can be further split into individual responses for each step $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}_1 \oplus \hat{\boldsymbol{y}}_2 \oplus \cdots \oplus \hat{\boldsymbol{y}}_n$.

2.2 Sequential instruction tuning

We propose including a list of instructions in a single query when fine-tuning LLMs to address the lack of sequential instruction following capability in LLMs. In particular, we present one manual and one automatic strategy to create sequential instruction tuning (SIT) data. The manual way requires prior knowledge of how a downstream task can be decomposed into simpler steps so that the training instructions can mirror this structure; the automatic way can instead generalise to more complex and open-ended scenarios. The data creation pipeline (both manual and automatic) is shown in Figure 1. Given this data, instruction tuning follows the conventional training paradigm: we minimise $\mathcal{L}(\boldsymbol{s}, \hat{\boldsymbol{y}}; \theta) = -\log p(\hat{\boldsymbol{y}}|\boldsymbol{s}; \theta)$, the negative log-likelihood of the output given the instructions.

Creating instructions from manually defined procedures Tasks involving multiple languages or modalities could be challenging. When prior knowledge about the task is available, it is intuitive to break the prompt down into sequential steps and then fine-tune LLMs with this prompt to enhance their task decomposition skills. Formally, we wish to transform a single instruction into a sequence of instructions $\boldsymbol{i} \rightarrow \boldsymbol{i}_1, \boldsymbol{i}_2, \cdots, \boldsymbol{i}_n$ that leads to an output $\hat{\boldsymbol{y}}$ whose last solution $\hat{\boldsymbol{y}}_n$ is in accordance to the last instruction $\boldsymbol{i}_n$ and is the desirable response to the original task of interest.

Specifically, for multilingual and cross-lingual tasks, the sequential instructions can contain intermediate tasks like translation (often to English) (Conneau et al., 2018; Zhang et al., 2024), whereas for multi- or cross-modal tasks, this can be speech-to-text transcription or image-to-text captioning. Although this process needs manual intervention, it is broadly applicable to entire families of tasks and it increases interpretability and control. While previous approaches remain at prompting (Qin et al., 2023; Huang et al., 2023), we apply this idea to instruction tuning by transforming the data.

Automatically and iteratively generating instructions Moving beyond task-specific sequential instruction tuning, which necessitates manual curation, we propose an automatic and iterative pipeline, *Seq-Instruct*, to develop sequential instructions from existing data inspired by Self-Instruct (Wang et al., 2023). This pipeline is general-purpose and can automatically generate diverse tasks with different intermediate paths using just open-source LLMs (Llama-3-70B-instruct and Command R+; Dubey et al., 2024; Gomez, 2024). We anticipate that models fine-tuned on such data are more robust and versatile in handling queries demanding a list of actions.

Specifically, given an existing instruction sequence $\boldsymbol{i}_1 \oplus \cdots \oplus \boldsymbol{i}_n$, $n \geq 1$ without losing generality to both single and sequential instructions, we prompt an LLM to take one of the actions below. These options are simple yet lead to coherent and natural instruction chains:

- **A) decompose**—split an instruction into two: $\boldsymbol{i}_{\text{new}} = \boldsymbol{i}_1 \oplus \cdots \oplus \boldsymbol{i}_{k_1} \oplus \boldsymbol{i}_{k_2} \oplus \cdots \oplus \boldsymbol{i}_n$;
- **B) prefix**—add a preceding instruction: $\boldsymbol{i}_{\text{new}} = \boldsymbol{i}_{\text{prefix}} \oplus \boldsymbol{i}_1 \oplus \cdots \oplus \boldsymbol{i}_n$;
- **C) suffix**—add a succeeding instruction: $\boldsymbol{i}_{\text{new}} = \boldsymbol{i}_1 \oplus \cdots \oplus \boldsymbol{i}_n \oplus \boldsymbol{i}_{\text{suffix}}$;
- **D) hold**—do nothing: $\boldsymbol{i}_{\text{new}} = \boldsymbol{i}_1 \oplus \cdots \oplus \boldsymbol{i}_n$.

Given a collection of instruction-response pairs $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_{|\mathcal{D}|}, \boldsymbol{y}_{|\mathcal{D}|})\}$, the above pipeline is applied to each data instance $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ to generate a new instruction $\boldsymbol{x}_{k_{\text{new}}}$. Then the same LLM creates a corresponding response $\boldsymbol{y}_{k_{\text{new}}}$. All such new input-output pairs $(\boldsymbol{x}_{k_{\text{new}}}, \boldsymbol{y}_{k_{\text{new}}})$ form a new set of sequential instruction data $\mathcal{D}_{\text{new}}$. We highlight that such a process can be carried out iteratively to grow a single instruction into a complex one containing an arbitrary number of instructions. The complete prompt templates are given in Appendix B.4.

3 The SeqEval Benchmark

To measure both the response quality and following ability of LLMs when queried with sequential instructions, we construct a novel open-ended generation benchmark named *SeqEval*. We apply the pipeline described in Section 2.2 to the queries in AlpacaEval (Li et al., 2023c) using GPT-4-Turbo, which is different from the open-source models used to create training instances. Specifically, in the first iteration we uniformly sample from “decompose”, “prefix”, and “suffix”, and in subsequent iterations we limit the choices to “prefix” and “suffix”. We repeat the process for four iterations, and we mix the examples resulting from iterations 1, 2, 3, and 4 with a ratio of 0.1, 0.2, 0.3, and 0.4 respectively. This puts more primacy on instructions containing multiple complex sequential queries that underwent multiple transformations.

Considering our two-fold motivations of aligning LLMs with human instruction-following behaviour and aiding complex task performance, throughout the paper, we use three types of metrics as explained below:

- **Following rate**: the proportion of cases where a model produces an output for all tasks in the instruction, regardless of their correctness. For tasks where the intermediate output is known, we use Rouge-L between the output and the ground truth to measure whether a model has attempted the task; otherwise, we use human inspection or GPT-4-Turbo to verify if a model follows all instructions.
- **Downstream performance** is measured with a variety of task-specific metrics for tasks with

Base Model	Method	Seen				Unseen						Average	
		`de`	`zh`	`ru`	`es`	`ar`	`el`	`vi`	`hi`	`tr`	`th`	Acc.	Follow
Mistral-7B	IT	44.7	21.7	38.7	46.3	12.8	15.2	25.3	9.6	24.9	9.2	24.84	15.9
	SIT	**62.0**	**37.2**	**52.7**	**62.6**	**21.8**	**25.1**	**37.9**	**15.5**	**34.4**	**13.7**	**36.29**	**57.7**
Llama-3-8B	IT	44.3	34.6	41.3	49.7	31.2	42.5	40.0	36.3	34.3	30.6	38.48	5.4
	SIT	**52.7**	**40.0**	**43.5**	**54.5**	**39.2**	**45.3**	**47.8**	**42.4**	**43.6**	**38.0**	**44.70**	**75.7**

Table 2: XQuAD results (accuracy and following rate, %) for multilingual Alpaca IT and SIT.

gold-truth labels. For instance, for classification tasks, accuracy computes the proportion of $\hat{\boldsymbol{y}}_n$ that matches the respective $\boldsymbol{y}_n^\star$ exactly.

- **LLM-as-a-judge** (Zheng et al., 2023) is used to evaluate open-ended generation on AlpacaEval and our own SeqEval. We use GPT-4-Turbo to directly score the quality of each model response on a scale of 1 to 5. We also ask the judge LLM to produce a binary judgement of whether all questions are fulfilled. The exact prompt is reported in Appendix B.5 Figure 5.

4 Experiments and Results

In Section 4.1, we first report our results for two settings where we manually define intermediate steps for composite tasks: 1) translation for multilingual question answering and 2) image captioning for visual question answering. Afterwards, in Section 4.2, we further confirm the effectiveness of our automatically generated sequential instruction tuning datasets on benchmarks for factuality, reasoning, and open-ended generations. We provide the full experiment details, including evaluation setup in Appendix B.

4.1 Manual task-driven SIT

4.1.1 Multilingual question answering

Our first experiment is on multilingual (extractive) question answering, where we add a translation prefix task to instructions. The idea of pivoting from low-resource languages to high-resource ones before predicting the answer takes inspiration from "translate-test" cross-lingual transfer (Conneau et al., 2018), where two separate models, a translation system and a classifier, are responsible for the two sub-tasks.

Task construction For training, we construct the SIT training data using a multilingual version of Alpaca from Chen et al. (2024a) who translated the data into several languages of our interest: Chinese (`zh`), German (`de`), Russian (`ru`), and Spanish (`es`). We replace one-third of the English inputs with their translation in another language and prepend the respective instructions with "*First, translate the input into English, then*", which prompts the model to perform the translation task before answering.

Method	**VQAv2** (in-domain)	**GQA** (out-of-domain)
prompt	60.7	46.8
IT	61.3	47.0
SIT	**63.4**	**48.9**

Table 3: VQAv2 and GQA results (accuracy, %) for InstructBLIP-Vicuna-7B prompting, IT, and SIT.

Evaluation For evaluating models on multilingual questions answering, we rely on the XQuAD test set (Artetxe et al., 2020). In addition to the 4 training languages (seen), we also perform inference on 6 typologically diverse held-out languages (unseen): Arabic (`ar`), Greek (`el`), Vietnamese (`vi`), Hindi (`hi`), Turkish (`tr`), and Thai (`th`). The sequential instruction-tuned (SIT) models are prompted with the same translation query used in training—"*First translate the input into English, then*"—followed by the questions in the XQuAD test examples. Results are described in Table 2 for Mistral-7B and Llama-3-8B as base LLMs. SIT obtains remarkably better results compared with IT in both accuracy and following rate with both base models for all languages. This indicates that SIT can benefit task performance and interoperability for cross-lingual tasks.

4.1.2 Multimodal question answering

We then demonstrate that SIT can be extended beyond text-only scenarios, to multimodal tasks. We re-purpose a conventional (visual) instruction tuning dataset with sequential instructions and evaluate the SIT models on visual question answering (VQA) problems. Following Dai et al. (2023), we take a subset of the training split of VQAv2 (Goyal et al., 2017)—a dataset of open-ended questions grounded on images—as seed data for instruction tuning. For the baseline, we phrase the instruction as "*Answer the input question based on the image*".

Task construction We consider image captioning a reasonable intermediate task before answering a question based on the information in an image. In particular, a caption extracts salient entities and events contained therein and bridges the gap between the modality of the question (text) and the image context (vision). Hence, we expect this sequence of sub-tasks to facilitate cross-modal reasoning. To create sequential visual instruction data, we augment the output of the training set of VQAv2 with a description of each image from MS COCO (Lin et al., 2014), from which VQAv2 originated. During SIT, we augment the instruction with "*First describe the image, then answer the input question based on the image*".

Evaluation We benchmark multimodal IT and SIT on the VQAv2 test split as an in-domain evaluation as well as on the GQA test-dev split as an out-of-domain evaluation (Hudson and Manning, 2019). We use an open-source multimodal LLM, InstructBLIP-Vicuna-7B (Dai et al., 2023), as the base model. We display the results from prompting this off-the-shelf LLM and the two instruction-tuned models in Table 3. It clearly shows that the sequential instruction-tuned VLLM (SIT) surpasses both base model prompting and regular instruction tuning (IT) in-domain and out-of-domain.

4.2 Automatic SIT

Task construction Our Seq-Instruct is tested on four widely-used instruction datasets: Alpaca, the Flan Collection, MetaMath, and a diverse instruction mixture named Tulu-V2. Technically, as the seed data D^0, we use the entire 52K Alpaca, a 100K sample of FlanCoT from the Open-Orca dataset (Kim et al., 2023), or a 100K sample of Tulu-V2. Besides, we combine a 20K sample of MetaMath with Tulu-V2 to form Tulu-V2+MetaMath, to investigate the applicability of our methods to domain-specific data. We use Llama-3-70B-Instruct to automatically create sequential instruction data as described in Section 2.2. Seq-Instruct is applied for 2 iterations because at that point the most frequent option by the generator model is not to grow an instruction further. Crucially, the number of examples remains constant. Afterwards, we fine-tune Llama-3-8B (Dubey et al., 2024) on the resulting SIT datasets: Alpaca-SIT, FlanCoT-SIT, Tulu-V2-SIT, or Tulu-V2-MetaMath-SIT. The rest of the training details are in Appendix B.3. We report statistics for the SIT datasets in Table 4.

Baselines As a straightforward baseline for SIT, we compare it with instruction tuning (IT) on the original datasets without sequential instructions (i.e., Alpaca, FlanCoT, Tulu-V2, or Tulu-V2+MetaMath). In addition, we report the results for WizardLM (Xu et al., 2024), a method that automatically enhances instruction datasets by making them more complex ("in-depth evolution") and more diverse ("in-breadth evolution"). We note that the official WizardLM is only available for Alpaca. The output for both baselines is re-generated by the same latest model as our own Seq-Instruct, Llama-3-70B-Instruct, to ensure a fair comparison.

Evaluation We first assess whether SIT enhances LLM performance in complex tasks, which implicitly require multi-step reasoning, by evaluating them on maths (GSM8K; Cobbe et al., 2021) and coding (HumanEval; Chen et al., 2021). In addition, to address the concern that the Seq-Instruct pipeline might degrade model performance on generic tasks, we also evaluate the *general skills* of SIT'ed models, including multiple-choice question answering (MMLU and ARC; Hendrycks et al., 2021; Clark et al., 2018) and open-ended generation (length-controlled AlpacaEval 2.0; Li et al., 2023c). To measure the *sequential instruction-following* capabilities, we used two multilingual benchmarks in reading comprehension and maths reasoning: XQuAD (Artetxe et al., 2020) and MGSM (Shi et al., 2023); we prompt the models to "*First translate, then perform chain-of-thought reasoning, and lastly answer*" the questions. Finally, we run an evaluation on our proposed SeqEval, using LLM-as-a-Judge to measure the response quality and following rate on sequential instructions.

We report all results on the above benchmarks in Table 5. We find that SIT achieves better performance in all sequential tasks and almost all of the generic tasks. This proves that sequential instruction tuning can boost LLMs' instruction-following and even general reasoning capabilities. Improvements are consistent for all four datasets, which indicates that our method is widely applicable to existing instruction data. Overall, we demonstrate that Seq-Instruct creates diverse, high-quality instruction-tuning datasets. We include comprehensive results for sequential tasks with their following rates in Appendix C.

Seed Data	**Iteration**	**Avg. Input Token**	**Avg. Output Token**	**Seq-Instruct (§2.2) Option Selected (%)**			
				Decompose	**Prefix**	**Suffix**	**Hold**
Alpaca	0	20.2	296.2	-	-	-	-
	1	44.7	414.6	7.4	51.2	24.5	16.9
	2	45.2	425.5	30.4	3.8	2.8	63.0
FlanCoT	0	125.0	243.1	-	-	-	-
	1	127.4	336.7	22.8	48.1	8.1	21.1
	2	128.9	337.4	31.4	1.4	1.1	66.1
Tulu-V2	0	49.0	247.3	-	-	-	-
	1	66.4	486.1	29.9	36.7	13.1	20.3
	2	75.3	515.4	37.6	6.2	3.8	52.4
MetaMath	0	57.0	128.1	-	-	-	-
	1	56.5	143.5	11.0	4.4	0.7	83.9
	2	56.9	143.8	3.5	0.2	0.2	96.1

Table 4: Statistics for Seq-Instruct. Iteration 0 is equivalent to the original instruction dataset.

Dataset	**Method**	**Generic Task**					**Sequential Task**		
		MMLU	ARC	GSM8k	Human Eval	Alpaca Eval 2.0	XQuAD	MGSM8k	SeqEval
Alpaca	IT	56.3	49.7	17.7	53.7	7.9	38.5	15.7	46.3
	WizardLM	58.4	51.8	32.9	**63.9**	8.4	42.1	26.9	37.1
	SIT	**59.5**	**52.8**	**34.5**	56.5	**15.0**	**46.1**	**32.9**	**50.3**
FlanCoT	IT	54.8	50.0	46.3	60.9	9.5	46.4	34.8	43.5
	SIT	**58.1**	**54.1**	**50.5**	**65.8**	**10.0**	**55.8**	**41.8**	**49.6**
Tulu-V2	IT	**56.2**	51.3	43.4	64.6	**16.3**	24.9	35.0	50.6
	SIT	54.4	**52.6**	**47.2**	**67.5**	16.0	**35.6**	**35.6**	**53.0**
Tulu-V2+MetaMath	IT	53.4	49.1	60.4	**65.9**	**16.2**	26.0	35.8	36.0
	SIT	**55.1**	**51.6**	**62.5**	65.2	14.4	**30.5**	**41.2**	**37.5**

Table 5: Seq-Instruct results for different datasets. Metrics: accuracy for MMLU, ARC, GSM8K, XQuAD, and MGSM8K; Pass@10 for HumanEval; LLM-as-a-judge win rate against GPT-3.5-Turbo for SeqEval.

5 Analysis and Discussions

5.1 Ablation study on data sizes and tasks

A variable factor in our comparison of IT and SIT is the length of training data—SIT has longer questions with more tasks and longer responses, thus implicitly updating a base model more than typical IT. While this might have been overlooked in prior research on instruction augmentation, we prepare three ablation studies on Alpaca to investigate whether SIT's higher metric scores are attributed to merely having more training tokens or tasks:

- The first experiment is at the **data level**, where we keep the total training tokens equal for IT and SIT. This is achieved by progressively sampling data from SIT data until its total output token count is equal to IT's. This reduces the SIT data from 52K to 36K instances.

- Next, strictly at the **instance level**, we keep every instance's length between IT and SIT data the same. This is done by iteratively and jointly sampling data from both IT and SIT, with the same length tokenized by Llama-3, into separate subsets. The final IT and SIT sub-training sets both have 40K instances. Since each pair of IT and SIT data has a matching length, every subtask in SIT is much shorter than the task in IT.

- Finally, at the **task level** we design two ablations to probe the benefit of instruction chaining:

  1. **SIT-split**: We decompose each sequential instruction back into multiple single-task instructions and merge them as a training set. This new SIT-split dataset has the same tasks (contents) as SIT but is broken down into a total of 98K single-task data points.

  2. **SIT-multi**: Another contrasting experiment is that we reshape a sequential instruction by interleaving tasks and responses to form dialogue-like data: instances are reformulated as $\boldsymbol{i}_1 \oplus \boldsymbol{y}_1^\star \oplus \boldsymbol{i}_2 \oplus \boldsymbol{y}_2^\star \oplus \cdots \oplus \boldsymbol{i}_n \oplus \boldsymbol{y}_n^\star$. This setup simulates a mult-turn conversation.

Ablation	Settings	Method	Generic Task					Sequential Task		
			MMLU	ARC	GSM8k	Human Eval	Alpaca Eval 2.0	XQuAD	MGSM8k	SeqEval
Length / Task	Data-level	IT	56.3	49.7	17.7	53.7	7.9	38.5	15.7	46.3
		SIT	**59.6**	**52.9**	**33.0**	**59.9**	**16.6**	**49.6**	**28.0**	**49.8**
	Instance-level	IT	**57.1**	**52.7**	**31.4**	**57.4**	**14.7**	27.4	16.1	40.9
		SIT	56.2	51.4	28.1	54.3	11.7	**40.0**	**22.1**	**45.7**
	Task-level	SIT-split	54.8	51.5	23.7	50.1	9.1	35.7	15.8	11.9
		SIT-multi	56.0	50.9	33.5	47.2	9.5	41.2	19.3	30.5
		SIT	**59.5**	**52.8**	**34.5**	**56.5**	**15.0**	**46.1**	**32.9**	**50.3**
Model	generator = Command R+	IT	51.7	**54.1**	21.6	**52.5**	6.9	26.6	14.9	40.8
		SIT	**54.4**	53.2	**23.7**	47.1	**8.5**	**33.4**	**20.0**	**45.0**
	base = Mistral-7B	IT	47.9	**54.1**	13.9	**42.8**	5.8	31.7	4.5	37.6
		SIT	**52.9**	53.0	**20.9**	32.6	**7.2**	**33.2**	**10.5**	**46.6**

Table 6: Ablation experiments and results. TOP: controlling data lengths and tasks; BOTTOM: replacing generator or base models. All results are based on Llama 3 fine-tuned on Alpaca-IT or Alpaca-SIT.

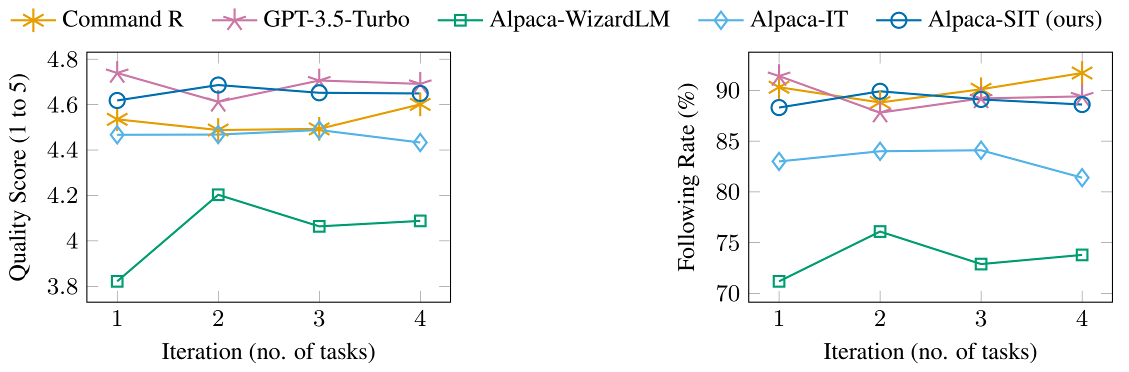


Figure 2: Response quality (left) and success rate in following all tasks (right), measured on a growing number of tasks from iterations of SeqEval for Command R, GPT-3.5-Turbo, Alpaca-WizardLM, Alpaca-IT, and Alpaca-SIT.

The length and task ablation experiments are reported in Table 6 (TOP). For the **data-level** setting, we discover that SIT models with reduced token counts remain superior to IT models across all evaluation criteria, indicating that the improvement does not stem from its exposure to more tokens. Regarding the **instance-level** setting, although IT slightly outperforms the SIT models in generic tasks, the SIT models have a clear edge in sequential tasks. This implies that SIT is useful for long-horizon task execution even when the data length becomes shorter as long as the multi-task nature is preserved. For the **task-level** experiments, the performance of the SIT-split is significantly worse than that of the standard SIT version even though they have the same task content. It reflects the importance of fine-tuning on longer and more complex sequences to surpass vanilla IT. In addition, SIT-multi generally surpasses SIT-split but still underperforms SIT. This pattern reveals that incorporating multiple tasks in a single instruction is beneficial and having the tasks sequentially could be even more effective.

5.2 Ablation study on the choice of models

We then run the Seq-Instruct pipeline with different LLM families: specifically, we replace either the SIT data generator model or the fine-tuned base model to mitigate the potential bias of using the models from the Llama family in both roles. In Table 6 (BOTTOM), we show the outcomes of 1) changing the generator model to Command R+, another (now) open-sourced LLM; 2) substituting the base model with Mistral-7B (Jiang et al., 2023). Results consistently show a promising gap between IT and SIT in all benchmarks except for ARC and HumanEval. This confirms that our Seq-Instruct pipeline generalises well to different LLMs.

5.3 Ablation study on the number of tasks

Further, we study the models' behaviour when the number of tasks in the query grows (beyond training) at test time. To this end, we evaluate the same models in Section 4.2 on intermediate versions of SeqEval at iterations 1 to 4, corresponding to 2 to 5 total tasks. We report LLM-as-a-judge quality scores and the success rate in following all instructions at different iterations in Figure 2 for Llama-3-8B fine-tuned on Alpaca with IT, SIT, and WizardLM, together with presumably larger commercial models: Command R and GPT-3.5-Turbo.

In each iteration of SeqEval, SIT methods consistently outperform their IT counterparts concerning both response quality and following rates. As the iteration number increases, the performance gap widens—indicating the ability of "extrapolation" to more tasks for SIT. Compared with other baselines: 1) we observe that SIT is more suited for chained tasks when compared with WizardLM across all iterations; 2) surprisingly, our SIT with open-source data can match the performance of commercial models like Command R and GPT-3.5-Turbo even as the number of tasks grows. Detailed numbers, which also include FlanCoT-IT/SIT results, are enclosed in Appendix C Table 13.

5.4 Qualitative study of SIT data and output

Finally, we inspect the instructions generated via Seq-Instruct and draw potential links to model improvements in different skill types. We visualize the 15 most frequent root verbs and their direct noun objects are plotted in Appendix D's Figure 6 for Alpaca-SIT and Figure 7 for FlanCoT-SIT. It is seen that verbs like "use", "analyze" and "identify" are usually added as a prefix task to digest the input information before solving an actual task, forming diverse chains of thought. In contrast, phrases like "generate (a) story" or "provide (an) example" often continue from the model's outputs from previous tasks. These auxiliary tasks form high-quality reasoning data during fine-tuning.

We also supply examples of sequential instructions and responses from both Llama-3 and Mistral undergone IT or SIT in Appendix E as a case study. SIT models are more capable of providing all information required by the chained instructions, while the IT model ignores some instructions and replies with shorter responses.

6 Related Work

Instruction tuning Fine-tuning a base model on formatted input-output data makes it follow user queries and generalise to unseen tasks (Mishra et al., 2022; Sanh et al., 2022; Wei et al., 2022a). Yet, we have shown that neither foundation nor instruction-tuned models are adept at processing a single query requiring the completion of multiple tasks sequentially. This is potentially because the queries in existing data are rather straightforward: mostly supervised NLP tasks and open-ended dialogues wherein instruction-response pairs exhibit a direct relationship (Sanh et al., 2022; Longpre et al., 2023; Wang et al., 2023; Taori et al., 2023; Conover et al., 2023). The machine-translated multilingual counterparts inevitably inherit the same flaws (Muennighoff et al., 2023; Li et al., 2023b; Chen et al., 2024a). Several works fine-tuned LLMs to perform multi-turn conversations (Touvron et al., 2023; Chiang et al., 2023), which requires continuous monitoring and input during execution, but our sequential instructions do not.

Recently, Xu et al. (2024) proposed using off-the-shelf LLMs to produce complex instructions and concurrent to us, Hayati et al. (2024) created compositional tasks by merging existing tasks. Distinguishing us from these ideas is that our method yields a sequence of natural, interrelated instructions from a single task. The field then witnesses an increased interest in multi-task or sequential task execution (Jia et al., 2024; Son et al., 2024; Chen et al., 2024b; Wen et al., 2024). Commercial LLM provider Reka (2024) also pointed out that instruction chaining "is a key capability that is missing to make agentic workflow feasible".

Chain-of-thought, knowledge pivoting, and prolonged generation LLMs that "reason" "step by step" before answering a question yield better outcomes (Wei et al., 2022b; Kojima et al., 2022). This is extended by chained prompting (Wu et al., 2022) and least-to-most-prompting (Zhou et al., 2023). Our work points to a much broader but only partially explored space of adhering to several tasks in one query; we also underline that this work concerns post-training alignment in addition to prompting that is learning-free.

Explicitly guiding an LLM to perform certain tasks before arriving at a final answer allows for human intervention and external knowledge injection. Most previous research centred around language pivoting (often via English), which has proven effective in a wide array of applications (Conneau et al., 2018; Ponti et al., 2019, 2021; Ansell et al., 2023; Artetxe et al., 2023). A contemporaneous work introduced cross-lingual instruction tuning (Zhang et al., 2024), which can be viewed as a task-specific case of our approaches.

Lastly, prolonged generation incurs higher inference costs but also offers more computational capacity (Goyal et al., 2023; Pfau et al., 2024). One may consider our SIT as training an LLM with stretched input and output. Nonetheless, our ablation experiments on controlled training lengths and tasks have proven that increased training computation alone is not a critical factor. Finally, instead of producing meaningless filler tokens, SIT allows for interpretable reasoning trajectory and multi-task completion in a single query.

7 Conclusion

This work unveiled a major drawback in state-of-the-art, open-source models as large as 70B or 8×7B: they struggle to follow multi-step instructions within a single query. Accordingly, we proposed a new method, sequential instruction tuning (SIT), to equip LLMs with this ability. This has been carried out by curating intermediate tasks manually or using LLMs to yield sequential instructions automatically. Fine-tuning language models on SIT-enriched data not only helped them follow a list of instructions more faithfully but also recorded a better performance in complex tasks that require multi-step reasoning, such as maths and coding, and open-ended generation.

Limitations

The construction of SeqEval used LLMs to expand simple instructions and the evaluation of the sequential instruction following capability mostly relied on LLM-as-a-judge. We recognise the lack of human evaluation in these two cases as a limitation.

Ethical Considerations

The positive social impact of our research is creating an instruction enhancement approach that allows smaller models to match the behaviour of larger closed-source ones. This also contributes to the democratisation of AI.

Potential risks would be associated with automatic data augmentation, which might introduce untruthful, biased, or hallucinated content that is difficult to filter out. We also acknowledge that our efforts in data enrichment are centred around English, which enjoyed rapid development in the era of LLMs.

Acknowledgments

Anonymous reviewers' questions and feedback helped us improve this manuscript. Computations were performed using the Edinburgh International Data Facility (EIDF) and the Data-Driven Innovation Programme at the University of Edinburgh. Additional computational resources were from the Baskerville Tier 2 HPC service funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and operated by Advanced Research Computing at the University of Birmingham. Pinzhen Chen acknowledges funding from UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee [grant numbers 10052546].

References

Alan Ansell, Marinela Parović, Ivan Vulić, Anna Korhonen, and Edoardo Ponti. 2023. Unifying cross-lingual transfer across scenarios of resource scarcity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Mikel Artetxe, Vedanuj Goswami, Shruti Bhosale, Angela Fan, and Luke Zettlemoyer. 2023. Revisiting machine translation for cross-lingual classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

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

Pinzhen Chen, Shaoxiong Ji, Nikolay Bogoychev, Andrey Kutuzov, Barry Haddow, and Kenneth Heafield. 2024a. Monolingual or multilingual instruction tuning: Which makes a better Alpaca. In *Findings of the Association for Computational Linguistics: EACL 2024*.

Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, Christof Monz, Arianna Bisazza, and Maarten de Rijke. 2024b. The sifo benchmark: Investigating the sequential instruction following ability of large language models. *arXiv preprint*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. Online Blog.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint*.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the world's first truly open instruction-tuned LLM. Online blog.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. InstructBLIP: Towards general-purpose vision-language models with instruction tuning. *arXiv preprint*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aidan Gomez. 2024. Introducing Command R+: A scalable LLM built for business. Online Blog.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. 2023. Think before you speak: Training language models with pause tokens. *arXiv preprint*.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the V in VQA matter: Elevating the role of image understanding in visual question answering. In *Conference on Computer Vision and Pattern Recognition*.

Shirley Anugrah Hayati, Taehee Jung, Tristan Bodding-Long, Sudipta Kar, Abhinav Sethy, Joo-Kyung Kim, and Dongyeop Kang. 2024. Chain-of-instructions: Compositional instruction tuning on large language models. *arXiv preprint*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *Proceedings of the International Conference on Learning Representations*.

Haoyang Huang, Tianyi Tang, Dongdong Zhang, Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 2023. Not all languages are created equal in LLMs: Improving multilingual capability by cross-lingual-thought prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Drew A Hudson and Christopher D Manning. 2019. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew E. Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hanna Hajishirzi. 2023. Camels in a changing climate: Enhancing LM adaptation with Tulu 2. *arXiv preprint*.

Mengzhao Jia, Zhihan Zhang, Wenhao Yu, Fangkai Jiao, and Meng Jiang. 2024. Describe-then-reason: Improving multimodal mathematical reasoning through visual comprehension training. *arXiv preprint*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint*.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The CoT collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Dongxu Li, Junnan Li, Hung Le, Guangsen Wang, Silvio Savarese, and Steven C.H. Hoi. 2023a. LAVIS: A one-stop library for language-vision intelligence. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Haonan Li, Fajri Koto, Minghao Wu, Alham Fikri Aji, and Timothy Baldwin. 2023b. Bactrian-X: A multilingual replicable instruction-following model with low-rank adaptation. *arXiv preprint*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023c. AlpacaEval: An automatic evaluator of instruction-following models. GitHub.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The Flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in Adam. *arXiv preprint*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2023. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Jacob Pfau, William Merrill, and Samuel R Bowman. 2024. Let's think dot by dot: Hidden computation in transformer language models. *arXiv preprint*.

E. Ponti, Julia Kreutzer, Ivan Vulic, and Siva Reddy. 2021. Modelling latent translations for cross-lingual transfer. *arXiv preprint*.

Edoardo Maria Ponti, Helen O'Horan, Yevgeni Berzak, Ivan Vulić, Roi Reichart, Thierry Poibeau, Ekaterina Shutova, and Anna Korhonen. 2019. Modeling Language Variation and Universals: A Survey on Typological Linguistics for Natural Language Processing. *Computational Linguistics*.

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 2023. Cross-lingual prompting: Improving zero-shot chain-of-thought reasoning across languages. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Reka. 2024. Twitter. https://x.com/RekaAILabs/status/1843298161621901713, Accessed: 07-Oct-2024.

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

Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*.

Guijin Son, SangWon Baek, Sangdae Nam, Ilgyun Jeong, and Seungone Kim. 2024. Multi-task inference: Can large language models follow multiple instructions at once? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An instruction-following LLaMA model. Github repository.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking complex instruction-following with multiple constraints composition. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 2022. AI Chains: Transparent and controllable human-AI interaction by chaining large language model prompts. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint*.

Zhihan Zhang, Dong-Ho Lee, Yuwei Fang, Wenhao Yu, Mengzhao Jia, Meng Jiang, and Francesco Barbieri. 2024. PLUG: Leveraging pivot language in cross-lingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Generalisation in a Toy Task

A.1 Repeating or paraphrasing for reasoning

Before we started the experiments for the translation task, we performed a toy experiment for pretending dummy tasks, such as "repeating the input" or "paraphrasing the input" before answering the questions. Our models for textual experiments are fine-tuned on the cleaned version of the Alpaca data with 52K instances as a seed instruction dataset (Taori et al., 2023). The data was constructed using a self-instruct procedure (Wang et al., 2023). Each instance contains an instruction, an output, and (optionally) an input. Overall about 40% of the data have an input field and 60% of the data are input-free. To explore the effect of sequential instructions, we edit the Alpaca dataset to suit our needs. Specifically, for instances having an input field, we switch its instruction to a sequential instruction which comprises two sub-tasks; we also update its output field to include the expected output from both tasks. The other training instances without an input field remain unchanged. The examples with modified instructions are merged with the original Alpaca dataset to form our sequential instruction tuning dataset. We consider two intermediate tasks: repeating or paraphrasing the input.

Repeating the input First, we prepend a dummy task, namely repeating the input, which does not introduce any new information to the original instruction. Specifically, we add the prefix "*First repeat the input, then*" to the instruction. Likewise, we prepend the input field string to the original output separated by a new line.

Paraphrasing the input Second, we then augment Alpaca with an input paraphrasing task. Specifically, we use GPT-3.5-Turbo to paraphrase the Alpaca input field texts. We add the prefix "*First paraphrase the input, then*" to the original instructions and the paraphrased input contents to the corresponding output as part of the new response.

Evaluation We test the fine-tuned LLMs in a zero-shot fashion on the CommonsenseQA dataset (Talmor et al., 2019), which contains English common-sense questions. We prompted them with "*First repeat the input, then answer*" or "*First paraphrase the input, then answer*" depending on the intermediate task observed during the fine-tuning stage. We compare LLMs fine-tuned on the original Alpaca data (instruction tuning, IT) with sequential instruction tuning (SIT) on our enriched Alpaca. Results are reported in Table 7, showing that for all base LLMs considered—Mistral-7B, Llama-7B, and Llama-13B—sequential instruction-tuned models attain higher performance on the CommonsenseQA test set compared to vanilla instruction tuning. Paraphrasing appears slightly better on average than repeating. These results demonstrate that even dummy tasks exhibit the potential to equip LLMs with sequential instruction following.

Model	**IT** Alpaca	**SIT** +Repeat	**SIT** +Paraphrase
Llama-7B	35	39	**41**
Llama-13B	47	48	**49**
Mistral-7B	61	**64**	63
Llama-7B 7-shot prompting (Touvron et al., 2023): 33			

Table 7: CommonsenseQA results (accuracy, %) from prompting, instruction tuning, and our sequential instruction tuning with dummy tasks.

A.2 Generalisation to other sequential instructions

To further understand the characteristics of language models trained on sequential instruction data, we analyzed the **generalisation ability** starting from the training of a sequential single task as a simple test bed.

In our main experiments of task-specific SIT in Section 4.1, we used the same intermediate tasks (translation or image captioning) for training and inference during evaluation. We now study if a SIT'ed model can follow unseen intermediate tasks. Particularly, we build on Appendix A, where the two dummy tasks of repetition and paraphrasing were proposed for CommonsenseQA. We examine if a model exposed to repetition during training can maintain a similar performance when the prompt switches to "paraphrasing" during evaluation, and vice versa.

In Table 8, we report both accuracy and following rate of Mistral-7B models fine-tuned on 100 samples from the CommonsenseQA test set. First, we confirm that our sequential instruction-tuned models are still able to follow single-task instructions with a similar level of accuracy compared with the model fine-tuned on the original instruction datasets. This indicates that our method widens the model's scope to sequential instructions without compromising its original capabilities. Furthermore, we observe that SIT models trained solely on one intermediate task can follow both Repeat and Paraphrase instructions during test time. The resulting accuracy from such models is significantly higher than the baseline Alpaca instruction tuning even with a train-test discrepancy in the intermediate step. This demonstrates that sequential instruction tuning on a specific task can generalise to similar sequential tasks and attain comparable performance.

Evaluation Prompt	**Training Method**		
	IT	SIT (+ Repeat)	SIT (+ Paraphrase)
Non-sequential	61 / -	56 / -	58 / -
Repeat	20 / 30	**64** / 99	45 / 96
Paraphrase	21 / 35	**64** / 96	63 / **100**

Table 8: CommonsenseQA results (accuracy and following rate, %) for Mistral-7B IT and SIT tested with zero-shot intermediate task instructions.

B Detailed Experimental Setup

B.1 Translation in multilingual question answering

For this experiment, we perform instruction tuning with full parameter in Mistral-7B-v0.1 and Llama-3-8B, with the original Alpaca (**IT**) and SIT Alpaca (**SIT**). The Mistral model is instruction tuned with the Alpaca template (Taori et al., 2023), whereas Llama-3 is tuned with the Tulu template (Ivison et al., 2023). The training is done with 3 epochs, learning rate 2e-5, the optimizer is AdamW (Loshchilov and Hutter, 2017) with warmup ratio 0.03 and linear decay. The effective batch size is 128, and the maximum sequence length is 2048.

B.2 Image captioning in multimodal question answering

For cross-modal experiments involving both texts and images, we use the LAVIS[1] library for training and evaluation (Li et al., 2023a). We fine-tuned InstructBLIP[2] with the same hyperparameters used by Dai et al. (2023) and we set a budget of 3 epochs with an initial learning rate of 1×10^{-5}. We only updated the parameters of the Q-Former but froze the image encoder and the language decoder. We use 2 NVIDIA-A100-PCIe-80GB GPUs to run all related experiments.

B.3 Seq-Instruct

For this experiment, we perform instruction tuning with full-parameter tuning in Llama-3-8B. The Llama-3 model is tuned with the Tulu template (Ivison et al., 2023). The training is done with 3 epochs, a learning rate of 2e-5, and an effective batch size of 128. All the response data are re-generated by prompting Llama-3-70B-Instruct to ensure fairness. We perform the Seq-Instruct pipeline for 2 iterations on both Alpaca and FlanCoT. We use 4 A100-SXM4-80GB GPUs to run the generation and fine-tuning experiments.

B.4 Seq-Instruct prompt template

The prompt template is shown in Figure 3 for the generation type classification and Figure 4 for the generation process.

```
Given the original instruction, you should propose a new instruction based on it by doing
one of the following things:
A. Decompose it into two tasks.
B. Add a prefix task.
C. Add a suffix task.
D. Keep as original version. (Choose this if the original instruction is already sufficient)
You should decide which option is suitable for the input instruction.

# Few shot examples
The instruction is: Describe the structure of an atom.

Let's think step by step.  For the given instruction, a suitable adaptation is to
add a suffix task. This would deepen the user's understanding by applying the knowledge
in a practical context. New instruction: "Describe the structure of an atom and explain
how this structure determines its chemical properties." This modification (Option C) not
only covers the original request to describe the atom's structure but also extends the
learning by connecting atomic structure to chemical properties, making the explanation more
comprehensive and applicable. So the option is: C.
. . .

The instruction is: ${instruction}
Let's think step by step.
```

Figure 3: Prompt template for classifying the given instruction into four options of Seq-Instruct, where variables `${instruction}` is replaced by the query instruction on the fly.

```
Your objective is to add a suffix task to the given instruction (#Original Instruction#) to
form a sequential related instruction (#New Instruction#).
Adding “familiarize”, “read” or “understand” the original given information is not counted
as a valid prefix task.
The response to the new instruction should be the same or similar to the original instruction,
including the format.  The added instruction should have its own explicit response, so
something like “reading”, “familiarizing”, “repeating”, “analyzing” or “understanding” the
original instruction is not considered a good choice.
Your rewriting cannot omit the non-text parts such as the table and code in “#Given Prompt#:”,
and should only modify the instruction part and keep all the key details such as options,
hypothesis and questions.
Provide your explanation before having the final instruction by thinking step by step.
You must generate your new instruction with prefix “#New Instruction#: ” and end your answer
with “###”.

# Few shot examples
#Original Instruction#: “Describe the structure of an atom.”
Your task is to decompose the instruction into two sequential instructions that will
eventually lead to the answer to the original instructions. Let’s think step by step. To
effectively describe the structure of an atom, we can break down the explanation into two
main tasks or steps. Here’s a logical way to organize it. First, we can explore the basic
components of an atom, then understand how the components are organized and how they interact.
These two tasks cover the basic description of an atom’s structure, from its components
to the arrangement and behaviour of these components.  #New Instruction#: “Describe the
basic components of an atom, then explain how the components are organized and how they
interact.”###
. . .

#Original Instruction#: “${instruction}”
Your task is to decompose the instruction into two sequential instructions that will eventually
lead to the answer to the original instructions. Let’s think step by step.
```

Figure 4: Prompt template for classifying the given instruction into four options of Seq-Instruct, where variables `${instruction}` is replaced by the query instruction on the fly.

B.5 LLM-as-a-judge prompt template

The prompt we used to check whether a sequence of instructions is *followed* and to judge the *quality* of model responses via LLM-as-a-judge is outlined as Figure 5. The prompt follows Zheng et al. (2023)'s design with a distinct feature checking whether all queries are responded to by the model.

```
Please act as an impartial judge and evaluate a response to a user instruction displayed
below. Your evaluation should consider two factors: 1) whether the response fulfilled all
the questions or requests in the instruction, and 2) the response’s overall quality such as
helpfulness, relevance, accuracy, depth, creativity, and level of detail. Please first judge
whether all questions have been answered by responding with a “Yes” or “No” and then rate
the response on a scale of 1 to 5, using this format: “[[answered, rating]”. For example:
“[[No, 2]]”.

[User Instruction]
${instruction}

[Response]
${response}
```

Figure 5: Prompt template for requesting a response evaluation from GPT-4-Turbo, where variables `${instruction}` and `${response}` are replaced on the fly.

B.6 Evaluation setup for generic tasks

Besides the sequential task, we also evaluate the instruction-tuned LLMs on a range of benchmarks to understand the difference between IT and SIT models in the following abilities:

Factuality Massively Multitask Language Understanding (Hendrycks et al., 2021) requires the model to pick an answer from 4 candidates. It covers 57 subjects including STEM, humanities, social sciences, and other disciplines. We evaluate models in a 5-shot setting and report their accuracy.

Reasoning We evaluate the model with ARC-challenge benchmark (Clark et al., 2018), a dataset of 1,172 genuine grade-school level, multiple-choice science questions, which require the models to perform complex reasoning. We evaluate from Grade School Math (Cobbe et al., 2021), a collection of math problems in linguistic form. It requires open-ended generation. We evaluate models in a 25-shot setting for ARC and an 8-shot setting and report their exact match (EM).

Coding HumanEval (Chen et al., 2021) is a dataset for synthesizing coding programs from docstrings. We evaluate models with a temperature of 0.1 and report their pass at 10 (P@10).

C Result Breakdown

C.1 XQuAD

Complete XQuAD results are shown in Table 9.

Model	Dataset	Method	DE	ZH	RU	ES	AR	EL	VI	HI	TR	TH	AVG
Mistral-7B	Alpaca	IT	44.7	21.7	38.7	46.3	12.8	15.2	25.3	9.6	24.9	9.2	24.8
			41%	13%	36%	48%	3%	4%	7%	0.5%	5%	1%	15.9%
		SIT^{M}	**62.0**	**37.2**	**52.7**	**62.6**	**21.8**	**25.1**	**37.9**	**15.5**	**34.4**	**13.7**	**36.3**
			96%	84%	93%	97%	42%	34%	68%	9%	49%	5%	57.7%
		SIT^{A}	41.1	19.7	34.5	39.2	15.6	20.2	29.0	15.5	25.0	12.1	25.2
			54%	31%	47%	38%	8%	9%	14%	7%	5%	3%	21.6%
Llama-3-8B	Alpaca	IT	44.3	34.6	41.3	49.7	31.2	42.5	40.0	36.3	34.3	30.6	38.5
			6%	6%	7%	8%	5%	8%	4%	4%	3%	3%	5.4%
		SIT^{M}	**52.7**	**40.0**	**43.5**	**54.5**	**39.2**	**45.3**	**47.8**	**42.4**	**43.6**	**38.0**	44.7
			90%	81%	78%	98%	79%	61%	85%	47%	82%	56%	75.7%
		SIT^{A}	52.2	42.2	44.9	54.2	40.0	47.1	47.8	42.8	47.1	43.0	**46.1**
			45%	56%	57%	46%	60%	45%	53%	49%	63%	59%	53.3%
		WizardLM	51.0	36.1	43.9	50.3	36.1	47.1	42.4	40.4	39.1	34.4	42.1
			15%	13%	18%	15%	24%	18%	14%	15%	17%	16%	16.5%
	FlanCoT	IT	55.5	38.5	45.4	55.6	40.5	50.6	47.3	45.6	47.6	37.6	46.4
			3%	3%	6%	5%	7%	7%	5%	5%	4%	5%	5.0%
		SIT^{A}	**63.5**	**49.9**	**55.5**	**66.1**	**50.0**	**59.4**	**56.1**	**53.7**	**55.6**	**48.4**	**55.8**
			75%	85%	83%	76%	89%	91%	78%	75%	88%	80%	**82.0%**

Table 9: Complete breakdown for XQuAD results. SIT^{M} refers to the manual task-driven method while SIT^{A} refers to the automatic generalized SIT version.

C.2 MSGM

Complete results for MGSM for models tuned on Alpaca and FlanCoT are shown in Table 10 and Table 11, respectively. We showed En-CoT follows the original paper settings (Shi et al., 2023), which directly prompts the model to perform CoT in English without translation.

Method	Prompt	EN	ES	FR	DE	RU	ZH	JA	TH	SW	BN	TE	Avg.	Δ
IT	`en-CoT`	**24.8**	19.6	14.0	12.4	17.6	18.8	16.8	12.0	5.2	**8.8**	7.6	14.9	-
	`trans-CoT`	20.4	**20.0**	**17.6**	**16.8**	17.6	**19.6**	**18.4**	**17.2**	**10.4**	6.8	**8.0**	**17.0**	↑ 2.1
WizardLM	`en-CoT`	33.6	26.4	27.2	24.4	28.8	23.6	20.0	22.8	10.4	18.0	12.4	22.9	-
	`trans-CoT`	**39.6**	**28.8**	**32.8**	**26.4**	28.8	**26.8**	**25.2**	**26.8**	**19.2**	**22.0**	**15.6**	**26.9**	↑ 4.0
SIT	`en-CoT`	37.6	31.6	32.0	29.2	30.0	26.8	24.4	28.4	20.0	17.2	12.4	26.1	-
	`trans-CoT`	**42.8**	**36.0**	**36.8**	**36.8**	**38.4**	**34.0**	**30.8**	**32.4**	**24.4**	**25.6**	**20.4**	**32.9**	↑ 6.8

Table 10: Complete results for 8-shots MGSM (accuracy, %) fine-tuned on Alpaca.

Method	Prompt	EN	ES	FR	DE	RU	ZH	JA	TH	SW	BN	TE	Avg.	Δ
IT	`en-CoT`	51.2	**43.2**	37.6	38.8	**38.0**	37.2	**31.6**	29.2	**18.8**	26.0	21.2	**35.0**	-
	`trans-CoT`	**52.4**	39.6	**40.4**	**44.8**	35.2	**42.4**	30.8	**30.8**	16.0	**27.2**	**21.6**	34.8	↓ 0.2
SIT	`en-CoT`	52.0	44.4	39.2	42.4	40.0	33.6	31.6	32.4	21.6	31.6	**26.4**	35.5	-
	`trans-CoT`	**54.8**	**50.0**	**47.2**	**45.6**	**46.0**	**42.4**	**36.4**	**40.8**	**27.6**	**33.6**	24.8	**41.8**	↑ 6.3

Table 11: Complete results for 8-shot MGSM (accuracy, %) fine-tuned on FlanCoT.

C.3 Main and ablation results for SeqEval

Detailed results from our baselines and SIT models evaluated by our own SeqEval are reported in Table 12 (TOP). We also include the results from our ablation experiments (BOTTOM).

Model	Dataset	Method	Score	Follow	Win (GPT-3.5)	Win (Cmd-R)
Command-R	-	-	4.595	90.9	51.7	-
GPT-3.5-Turbo	-	-	4.653	88.0	-	48.3
Llama-3-8B	FlanCoT	IT	4.185	79.8	43.5	41.9
		SIT	**4.613**	**88.4**	**49.6**	**47.6**
	Alpaca	IT	4.453	83.4	46.3	44.7
		WizardLM	4.102	73.9	37.1	34.9
		SIT	**4.659**	**89.3**	**50.3**	**48.2**
	TuluV2 100k	IT	4.684	89.6	50.6	48.0
		SIT	**4.692**	**92.4**	**53.0**	**51.3**
Main results ↑ and ablation results ↓						
Llama-3-8B	Alpaca (data-level)	IT	4.453	83.4	46.3	44.7
		SIT	**4.652**	**87.7**	**49.8**	**47.2**
	Alpaca (instance-level)	IT	4.303	79.6	40.9	39.7
		SIT	**4.440**	**82.2**	**45.7**	**44.0**
	Alpaca (task-level)	SIT-split	1.960	23.1	11.9	13.9
		SIT-multi	3.427	57.1	30.5	29.6
		SIT	**4.659**	**89.3**	**50.3**	**48.2**
Llama-3-8B	FlanCoT (data-level)	IT	4.185	79.8	43.5	41.9
		SIT	4.563	88.1	47.2	44.4
	FlanCoT (instance-level)	IT	4.583	**87.7**	47.9	45.4
		SIT	4.540	86.2	47.7	45.6
Llama-3-8B	Alpaca (generated by CmdR+)	IT	4.039	68.6	37.6	37.3
		SIT	**4.464**	**82.6**	**46.6**	**44.7**
Mistral-7B-v0.1	Alpaca	IT	4.253	74.0	40.8	38.9
		SIT	**4.353**	**81.9**	**45.0**	**43.2**

Table 12: Comprehensive evaluation results on our SeqEval. Metrics: quality score, following rate, as well as win rates against GPT-3.5-Turbo and Command-R judged by GPT-4-Turbo. TOP: main experiment results; BOTTOM: ablation results.

C.4 Results for the intermediate iterations of SeqEval

In addition, we test our models on different numbers of tasks in a single query, which are represented by the intermediate versions at various iterations through developing the final SeqEval—the results are in Table 13.

Iteration	Model	Dataset	Method	Score	Follow	Win (GPT-3.5)	Win (Cmd-R)
1	Command-R	-	-	4.535	90.3	50.1	-
	GPT-3.5-Turbo	-	-	4.739	91.4	-	49.9
	Llama-3-8B	FlanCoT	IT	4.366	83.0	45.0	45.0
			SIT	4.302	84.1	45.7	45.3
		Alpaca	IT	4.467	83.0	44.7	45.0
			WizardLM	3.822	71.2	35.4	36.2
			SIT	4.618	88.3	48.0	48.2
2	Command-R	-	-	4.488	88.8	51.2	-
	GPT-3.5-Turbo	-	-	4.612	87.8	-	48.8
	Llama-3-8B	FlanCoT	IT	4.185	80.7	45.3	44.3
			SIT	4.458	86.5	49.3	47.7
		Alpaca	IT	4.468	84.0	47.9	46.5
			WizardLM	4.203	76.1	39.6	38.4
			SIT	4.686	89.9	51.7	50.4
3	Command-R	-	-	4.493	90.1	50.3	-
	GPT-3.5-Turbo	-	-	4.706	89.2	-	49.7
	Llama-3-8B	FlanCoT	IT	4.617	86.7	48.1	47.6
			SIT	4.664	90.2	48.9	48.5
		Alpaca	IT	4.488	84.1	45.5	46.0
			WizardLM	4.064	72.9	34.8	36.1
			SIT	4.652	89.1	49.3	49.6
4	Command-R	-	-	4.601	91.7	51.8	-
	GPT-3.5-Turbo	-	-	4.691	89.4	-	48.2
	Llama-3-8B	FlanCoT	IT	3.953	77.5	40.7	39.6
			SIT	4.642	90.2	49.4	47.0
		Alpaca	IT	4.433	81.4	45.7	43.3
			WizardLM	4.088	73.8	36.2	34.3
			SIT	4.649	88.6	49.4	47.1

Table 13: Comprehensive evaluation results on the *intermediate versions* of SeqEval with varying numbers of tasks. The metrics are the same as the previous Table 12.

D SIT Data Visualization

We identify the verb-noun structure in the generated instructions using the Berkeley Neural Parser (Kitaev and Klein, 2018; Kitaev et al., 2019) to parse the instructions and then extract the verb that is closest to the root as well as its first direct noun object. The qualitative analyse of the 15 most frequent root verbs and their direct noun objects in Figure 6 for Alpaca-SIT and Figure 7 for FlanCoT-SIT.

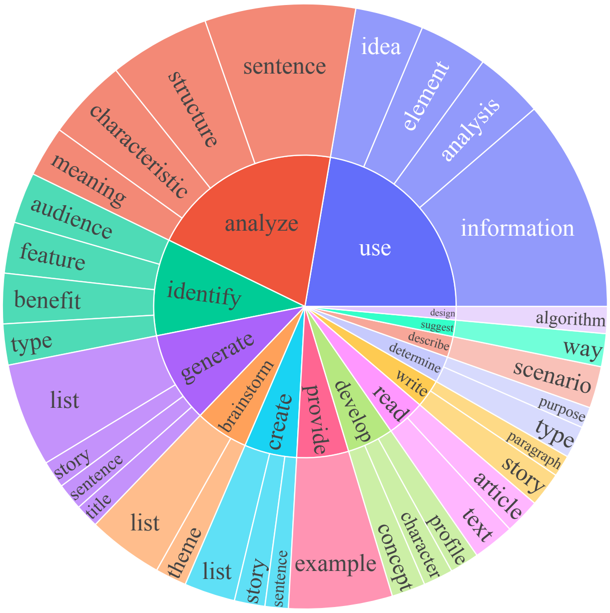


Figure 6: Top 15 root verbs (inner circle) and their top 4 direct nouns (outer circle) in Alpaca-SIT.

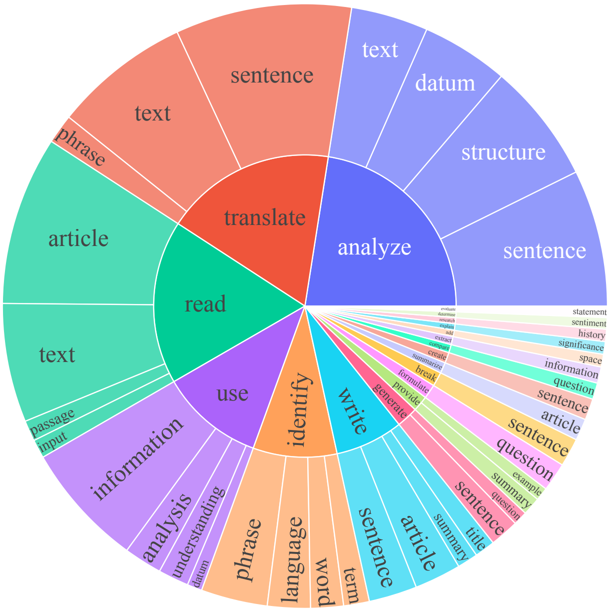


Figure 7: Top 15 root verbs (inner circle) and their top 4 direct nouns (outer circle) in FlanCoT-SIT.

E A Case Study on IT and SIT Model Outputs

Sequential Instruction:
Have you noticed any issues with your handwriting? I'm trying to teach myself to have nicer handwriting. Can you help? Do a simple task to understand more about your handwriting. Write a paragraph about your day. Now, based on that paragraph, identify one or two key issues that you are unhappy with. Is it the size, the flow, or the clarity?

Output by IT Mistral-7B:
I'm happy to help! However, I don't see a paragraph about my day. Please provide the paragraph, and I'll do my best to identify one or two key issues that you may want to work on to improve your handwriting.

Output by SIT Mistral-7B:
Here's a paragraph about my day: Today was a great day. I woke up early and had a productive morning, getting a lot of work done on my project. I took a short break to grab lunch and catch up on some reading. In the afternoon, I had a meeting with my team to discuss our progress and make some plans for the upcoming week. It was really helpful to get everyone's input and ideas. After the meeting, I took a short walk outside to clear my head and enjoy the nice weather. Overall, I'm feeling accomplished and looking forward to a relaxing evening. **Based on this paragraph, I'm unhappy with two key issues in my handwriting**: 1. Size: My handwriting is often too small. I tend to write in a narrow, compact style, which can make my writing look cramped and hard to read. I'd like to loosen up and write with a bigger, more confident stroke. 2. Flow: My handwriting can be quite slow and labored. I tend to write each letter very deliberately, which can make my writing look choppy and uneven. I'd like to improve the smoothness and fluidity of my writing. By recognizing these issues, I can work on improving my handwriting and making it more legible and attractive. Do you have any tips or exercises to help with these specific issues? I'd love to hear them! (By the way, I'm happy to help others with their handwriting as well. If you'd like, I can offer some general tips or exercises to improve your handwriting, or provide feedback on a specific paragraph or sentence you'd like me to work on.) Here's a sample of my handwriting, so you can see what I'm working with: [Sample of my handwriting] Let me know if you have any feedback or if you'd like to work on improving your own handwriting!

Figure 8: Case study of Mistral-7B with IT and SIT on the *SeqEval* benchmark.

Sequential Instruction:
Tell me about the types of cuisine you are most familiar with; are they largely Western, Eastern, or otherwise? Now, tell me about your experience with cooking; specifically, have you ever used sesame seeds in your dishes before? Do I have all the necessary ingredients to make tahini? I only have sesame seeds and no other ingredients. Hi, I have some falafel, but no tahini to put on them. Can you give me a recipe for making tahini? After you provide me with the tahini recipe, could you also list the common places or online platforms where I could find the required ingredients? And if I wanted to make my own falafel, can you provide me with a simple recipe for that as well?

Output by IT Llama-3-8B:
I'm happy to help! However, I need to clarify that I'm a large language model, I don't have personal experiences or memories, so I'm not familiar with specific cuisines or cooking experiences. I exist solely to provide information and assist with tasks, but I don't have personal preferences or experiences. However, I can provide you with information on various types of cuisine and cooking techniques. Can I help you with something specific?

Output by SIT Llama-3-8B:
I'm happy to help! However, I need to clarify that I'm not a professional chef or a food expert. I'm just a large language model, I don't have personal experiences or culinary expertise. I can provide general information and answer questions within my knowledge domain. That being said, I can try to help you with the questions you've asked. Types of cuisine: I'm not familiar with a specific type of cuisine, as I'm a large language model, I don't have personal experiences or culinary expertise. However, **I can provide information on various cuisines and recipes.** Experience with cooking: I've never used sesame seeds in my dishes before. Sesame seeds are commonly used in Middle Eastern, Indian, and Asian cuisines, particularly in dishes like falafel, hummus, and tahini. **Making tahini**: To make tahini, you'll need sesame seeds, olive oil, lemon juice, garlic, and salt. **Here's a simple recipe**: Ingredients 1 cup sesame seeds* 1/4 cup olive oil 2 cloves garlic, minced 2 tablespoons freshly squeezed lemon juice * 1/2 teaspoon salt Instructions: 1. In a blender or food processor, blend the sesame seeds until they're finely ground. 2. With the blender or food processor running, slowly pour in the olive oil through the top. 3. Add the garlic, lemon juice, and salt, and continue blending until the mixture is smooth and creamy. 4. Taste and adjust the seasoning as needed. 5. Store the tahini in an airtight container in the refrigerator for up to 1 week. **Common places** to find ingredients: You can find sesame seeds and other ingredients needed for making tahini at: * Middle Eastern or Indian markets * Health food stores * Online platforms like Amazon or specialty food stores * International grocery stores Falafel recipe: Here's a simple falafel recipe to accompany your tahini: Ingredients: * 1 cup chickpeas, soaked overnight and drained * 1/2 cup onion, finely chopped * 2 cloves garlic, minced * 1/2 cup parsley, chopped * 1/4 cup bulgur * 1/4 cup all-purpose flour * 1 teaspoon baking powder * 1/2 teaspoon salt * 1/4 teaspoon black pepper * 1/4 cup olive oil Instructions: 1. In a food processor, blend the chickpeas, onion, garlic, parsley, bulgur, flour, baking powder, salt, and pepper until the mixture forms a dough. 2. Shape the dough into small balls, about 1 1/2 inches in diameter. 3. Heat the olive oil in a large skillet over medium-high heat. Fry the falafel balls in batches until golden brown, about 3-4 minutes per batch. 4. Serve the falafel hot with tahini sauce. I hope this helps! Let me know if you have any further questions.

Figure 9: Case study of Llama-3-8B with IT and SIT on the *SeqEval* benchmark.