Fine-Tuning Large Language Models with Sequential Instructions

Hanxu Hu* **Simon Yu*** **Pinzhen Chen*** **Edoardo M. Ponti**
School of Informatics, University of Edinburgh
{hanxu.hu,eponti}@ed.ac.uk

Abstract

Despite the success of existing instruction-tuned models, we find that they usually struggle to respond to queries with multiple instructions. This impairs their performance in complex problems whose solution consists of multiple intermediate tasks. Thus, we contend that part of the fine-tuning data mixture should be *sequential*—containing a chain of interrelated tasks. We first approach sequential instruction tuning from a task-driven perspective, manually creating interpretable intermediate tasks for multilingual and visual question answering: namely "translate then predict" and "caption then answer". Next, we automate this process by turning instructions in existing datasets (e.g., Alpaca and FlanCoT) into diverse and complex sequential instructions, making our method general-purpose. Models that underwent our sequential instruction tuning show improved 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 fine-tuning method. We hope that our endeavours will open new research avenues on instruction tuning for complex tasks.[1]

1 Introduction

Instruction tuning (IT), or supervised fine-tuning (SFT), gives large language models (LLMs) the ability to execute new 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 suffer from the absence of multi-step instructions. While this dataset design presumably mirrors the properties of natural data, where such instructions rarely occur, we speculate that this hinders the fine-tuned models from navigating a sequence of sub-tasks in a single command, which is arguably crucial for complex tasks requiring reasoning (e.g., coding and maths) or knowledge transfer (e.g., cross-lingual and cross-modal question answering, Shi et al., 2023; Zhang et al., 2023). Moreover, this detracts from user experience as models do not track whether all requests have been fulfilled.

We empirically verify this hypothesis by prompting various versions of state-of-the-art open-source LLMs (e.g. Llama 3 AI@Meta 2024 and Mistral Jiang et al. 2023) with simple two-step instructions—already more than they can shake a stick at. After manually inspecting their answers, we find that not only did their 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 problem, we propose a sequential instruction tuning (SIT) paradigm which uses simple

*Equal contribution.
[1]Our data and code are available at https://seqit.github.io/.

Preprint. Under review.

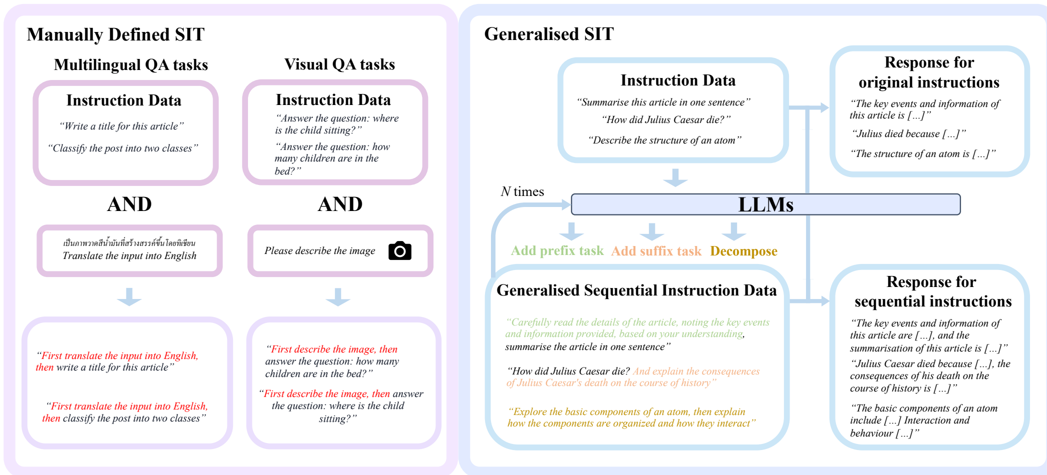

Figure 1: Construction of sequential instruction data via manual and automatic processes.

strategies to automatically augment instructions without the need for additional human annotations. First, we explore an augmentation strategy that is task-focused and interpretable, by introducing pre-defined intermediate steps for multilingual (Artetxe et al., 2020) and visual question answering (Hudson and Manning, 2019), namely "translate then predict" and "caption then answer".

Moreover, to make our method task-agnostic, we generalise the pipeline to automatic instruction augmentation where intermediate tasks are seeded from a single-task instruction. Augmenting instruction mixtures, such as Alpaca (Taori et al., 2023), FlanCoT (Longpre et al., 2023) and Tulu-V2 (Ivison et al., 2023), allows us to construct natural, diverse, and high-quality sequential instruction datasets. Our method stands in contrast with previous automatic augmentation methods, such as WizardLM (Xu et al., 2024) which make instructions more complex or diverse, but not sequential. Comparing LLMs fine-tuned with our new sequential dataset and the original dataset, we observe very significant boosts in factuality (MMLU; Hendrycks et al., 2021), reasoning (GSM8K and HumanEval; Cobbe et al., 2021; Chen et al., 2021), and open-ended generation (length-controlled AlpacaEval 2.0; Li et al., 2023c). Ablation studies confirm SIT's generalizability 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 Alpaca-SIT and FlanCoT-SIT datasets, as well as the *SeqEval* benchmark, will contribute to endowing LLMs with the ability to solve complex tasks.

2 Methodology

2.1 Sequential Instructions

Existing instruction data usually comprise single-step instructions (i.e., each instruction–response pair resembles one task); however, this falls short of equipping models with the ability to handle a query containing (explicitly or implicitly) multiple sub-tasks. 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 k^{th} 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 $\hat{\boldsymbol{y}} = \hat{\boldsymbol{y}}_1 \oplus \hat{\boldsymbol{y}}_2 \oplus \cdots \oplus \hat{\boldsymbol{y}}_n$.

Method	Model	RL	BS	Following		
		R	R	R	A	R+A
Prompt	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-2-7B-Chat	41	87	30	81	30
	Llama-2-13B-Chat	48	87	42	89	42
	Llama-2-70B-Chat	39	89	89	97	89
	Llama-3-8B-Alpaca	42	82	33	69	27
	Llama-3-8B-Instruct	75	93	98	98	97
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: ROUGE-L (RL, %), BERTScore (BS, %), and following rate (%) of prompted pre-trained models, instruction-tuned (IT), and sequential instruction-tuned (SIT) models evaluated on 100 random instances of CommonsenseQA, for the tasks of repeating (R), answering (A), or both (R+A). We highlight in grey the models fine-tuned on non-public SFT data.

2.2 Prompting Existing LLMs with Sequential Instructions

First, we verify our assumption that LLMs instruction-tuned with single instructions from existing public datasets do not generalise well to a sequence of instructions. To this end, we probe state-of-the-art open-source LLMs—such as Llama 2, Llama 3, and Mistral (Touvron et al., 2023; Jiang et al., 2023)—on a subsample from CommonsenseQA by asking these models to repeat the input and then answer the question. We report ROUGE-L (Lin, 2004) and BERTScore (Zhang et al., 2020) for input repetition and we manually inspect whether a model executes the repetition (R) and answering (A) steps. As it emerges from Table 1, LLMs fine-tuned on public datasets, such as Alpaca (Taori et al., 2023), usually complete only one of the tasks and sometimes even struggle with the entire prompt. On the other hand, models fine-tuned on proprietary SFT mixtures (Llama-2-Chat or Llama-3-Instruct), highlighted in grey, perform considerably well.

2.3 Sequential Instruction Tuning

As a solution to mitigate this limitation of public instruction datasets, we propose to include sequential instructions when fine-tuning LLMs. In particular, we present two strategies to create sequential instruction tuning (SIT) data, one manual and one automatic. 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.

2.3.1 Manually defining instructions

Complex tasks involving multiple languages or modalities could be challenging for a model to deal with in a single step. 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 downstream tasks of interest.

Specifically, for multilingual and cross-lingual tasks, the sequential instructions can contain prefix tasks like translation (often into English) (Conneau et al., 2018; Zhang et al., 2023). For multimodal and cross-modal tasks, an intermediate step could be speech-to-text transcription or image-to-text captioning. While this process is manually defined, it is broadly applicable to entire families of tasks and it increases interpretability and control. Whereas previous approaches split an instruction into a translation task and a question-answering task during prompting (Qin et al., 2023; Huang et al., 2023), we apply this idea to instruction tuning by transforming the SFT data themselves.

2.3.2 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 single instructions in existing datasets, such as Alpaca (Taori et al., 2023) and FlanCoT (Longpre et al., 2023). Inspired by self-instruct (Wang et al., 2023), this pipeline is general-purpose and can automatically generate diverse instructions with different intermediate tasks from powerful open-source LLMs (Llama-3-70B-instruct and Command R+; AI@Meta, 2024; Gomez, 2024). We anticipate that models fine-tuned on such data are more robust and versatile in handling complex queries.

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 and natural yet lead to coherent and relevant instructions:

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.

2.4 The *SeqEval* Benchmark

Finally, to measure both the **response quality** and **following ability** of LLMs when queried with sequential instructions, we put forward a novel open-ended generation benchmark named *SeqEval*. We apply the pipeline described in Section 2.3.2 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.

2.5 Evaluation Metrics for Sequential Task

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

- **Following rate** is the proportion of test instances where a model can successfully generate output answers for all tasks in the instruction, regardless of their correctness. For tasks where the intermediate output is known, e.g. "translate-then-predict", we use Rouge-L between the model output and the ground-truth answer to measure whether a model has attempted the task. For other tasks, we rely on 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 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 6.

3 Experiments and Results

In Section 3.1, 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 3.2, 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.

3.1 Task-Driven SIT

3.1.1 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. (2024), who translated the English instruction and input 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.

Evaluation results 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.

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.

3.1.2 Image captioning in 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.

Dataset	Iter	Avg. Input Tokens	Avg. Output Tokens	*Seq-Instruct* Option (§2.3.2)			
				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

Table 3: Statistics for *Seq-Instruct*. We report the average number of input and output tokens, and the percentage (%) of times each option in the *Seq-Instruct* pipeline is selected in each iteration. Iteration 0 is equivalent to the original version of the instruction dataset.

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 context (image). 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*".

Method	VQAv2 (in-D)	GQA (OOD)
prompt	60.7	46.8
IT	61.3	47.0
SIT	**63.4**	**48.9**

Figure 2: VQAv2 and GQA results (accuracy, %) for InstructBLIP-Vicuna-7B prompting, IT, and SIT.

Evaluation results 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 off-the-shelf LLMs and the two instruction tuning methods in Figure 2. 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.

3.2 Generalised SIT

Task construction For *Seq-Instruct* experiments, we select two widely-used instruction datasets: Alpaca (Taori et al., 2023) and the Flan Collection (Longpre et al., 2023), and one mixed collection of high-quality instruction datasets: Tulu-V2 (Ivison et al., 2023). In particular, as a seed data D^0, we start with the 52K Alpaca dataset, a 100K sample of FlanCoT data from the Open-Orca dataset (Kim et al., 2023), or a 100K sample of the Tulu-v2 dataset. We use Llama-3-70B-Instruct to generate new sequential instruction data as described in Section 2.3.2. In particular, we apply *Seq-Instruct* for two iterations. Crucially, the number of examples remains constant. Afterwards, we fully fine-tune Llama-3-8B (AI@Meta, 2024) as a base model with the resulting Alpaca-SIT, FlanCoT-SIT and Tulu-V2-SIT, respectively. The rest of the training details are in Appendix B.3, whereas we report statistics for the SIT datasets in Table 3.

Baseline As a baseline for SIT, we compare it with instruction tuning (IT) on the original datasets without sequential instructions (i.e., Alpaca, FlanCoT and Tulu-V2). In addition, we report the results for WizardLM (Xu et al., 2024), a method that automatically enhances instruction datasets to make

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**

Table 4: *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*.

their instructions more complex ("in-depth evolution") and more diverse ("in-breadth evolution"). Specifically, we report WizardLM results based on its augmentation of Alpaca—it is worth noting that the process of WizardLM does not result in sequential instructions. The output for both baselines is re-generated by the same model as our own *Seq-Instruct*, Llama-3-70B-Instruct, to ensure a fair comparison.

Evaluation results We 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 request the model to *First translate, then perform chain-of-thought reasoning, and lastly answer* the questions. Finally, we evaluate models on our *SeqEval*, using LLM-as-a-Judge to measure the response quality and following rate on answering sequential instructions.

We report all results on the above benchmarks in Table 4. 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 both Alpaca, FlanCoT and Tulu-V2 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.

4 Analysis and Discussion

4.1 Is Sequence Length the Driving Factor Behind Performance

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

- The first is a **data-level** experiment where we keep the total training tokens equal for IT and SIT. This is done by progressively sampling data from SIT data until its total output tokens equal IT's. This reduces the SIT data from 52K to 36K instances.
- Next, at a stricter **instance level**, we control every instance's length between IT and SIT data to be the same. This is done by iteratively adding data points from IT and SIT, with the same length

when tokenized by Llama-3, to sub-training sets for IT and SIT. The final size for both IT and SIT sub-training sets is 40K instances. Intuitively, since each IT and SIT instance pair has a matching length, every sub-task in SIT would be much shorter than the task in IT.

- At the **task level**:
  1. **SIT-split**: We decompose each sequential instruction back into multiple single-task data points and join 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. The training instances can be formulated 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 where a user raises a single-query instruction followed by a model generation in several rounds.

The length and task ablation experiments are listed in Table 5 (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 despite that they have the same task contents. In addition, SIT-multi generally surpasses SIT-split but 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.

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	G=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**
	B=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 5: Ablation experiments and results. TOP: controlled lengths and tasks; BOTTOM: replaced generator (G) and base (B) models. All results are based on Llama 3 fine-tuned on Alpaca-IT/SIT measured by the same metrics as Table 4.

4.2 Generalisation to a Variable Number of Sub-Tasks

Further, we investigate the models' behaviour when the number of intermediate tasks in a sequential instruction grows at inference time. To this end, we evaluate the same models as Section 3.2 on intermediate versions of *SeqEval* at different iterations. Note that these imply different maximum numbers of sub-tasks: as explained in Section 2.4, for each iteration, every instruction is optionally extended with one more task. We report the quality scores (left) and the following rate (right) for different iterations in Figure 3.

In each iteration of the creation of *SeqEval*, SIT methods consistently perform better than their IT counterparts concerning both response quality and following rates. As the iteration number increases, the performance gap widens—indicating the superior ability of "extrapolation" to more tasks of the *Seq-Instruct* procedure. We also establish additional baselines: first, we find that WizardLM has the lowest performance across all iterations, which highlights that SIT is the most competitive data

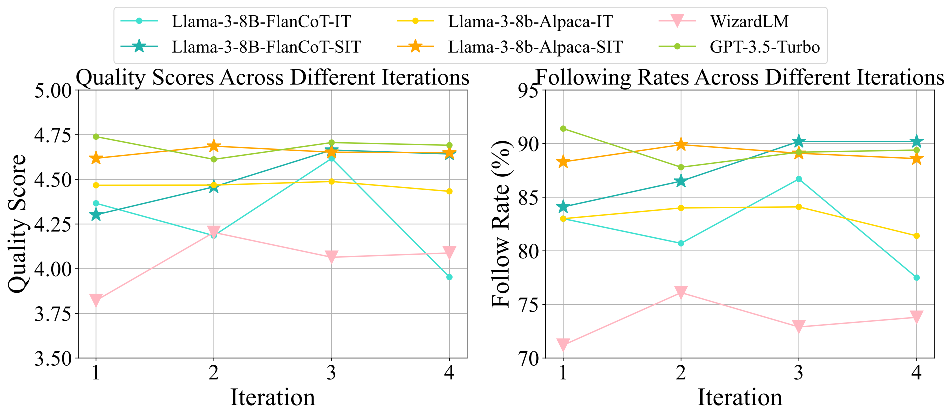

Figure 3: Quality scores and following rates on different iterations of *SeqEval* for Llama-3-8B fine-tuned with Alpaca or FlanCoT under IT or SIT. We also report WizardLM and GPT-3.5-Turbo as baselines, which represent alternative data augmentation methods and proprietary models respectively.

augmentation procedure. Second, SIT methods match the performance of vastly larger proprietary models, such as GPT-3.5-Turbo.

4.3 Generalisation of *Seq-Instruct* to Other Models

We then run the *Seq-Instruct* pipeline with different LLM families. Specifically, we replaced either the generation model (G) or base model (B) to mitigate the potential bias of using the models from the Llama-3 family for both roles. At the top of Table 5, we show the results of replacing the generation models with Command R+ (Gomez, 2024), another powerful open-sourced LLM. Besides, we also display the results when the base models are replaced with Mistral-7B (Jiang et al., 2023) in Table 5 (BOTTOM). Both experiments result in a promising leap from IT to SIT, in all benchmarks except for ARC and HumanEval, demonstrating how our *Seq-Instruct* pipeline generalises to different LLMs.

4.4 Qualitative Study of the SIT Data

Finally, we check the kinds of instructions generated via *Seq-Instruct* and draw potential links to model improvements in different skill types. 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 15 most frequent root verbs and their direct noun objects are plotted in Figure 7 for Alpaca-SIT and Figure 8 for FlanCoT-SIT. Verbs like "use", "analyze" and "identify" are often added as prefix tasks 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" leverage the model's outputs from previous tasks, prompting it to continue generating content relevant to the task. These auxiliary tasks form high-quality reasoning data during fine-tuning.

5 Related Work

Instruction tuning Instruction tuning fine-tunes a foundation model on specially formatted input-output data to make it follow instructions 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 to complete multiple tasks sequentially. We might glean insights into this phenomenon from the composition of instruction datasets: they are 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., 2024). On the other hand, several works have used multi-turn conversational datasets to fine-tune LLMs (Touvron et al., 2023; Chiang et al., 2023), allowing users to interact with the model to complete multiple tasks iteratively. More recently, Xu et al. (2024) propose using off-the-shelf LLMs to generate more complex instructions, and a concurrent work explored training on combinations of existing instruction tasks (Hayati et al., 2024). Distinguishing us from these ideas is that we begin with interpretable intermediate tasks and generalise to creating interrelated sequential tasks automatically.

Knowledge pivoting 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). Zhang et al. (2023) introduced cross-lingual instruction tuning, which can be seen as a task-driven case of our approaches whereas our automatic SIT generalises beyond this.

Chain-of-thought Prompting an LLM to generate a multi-step reasoning process before answering a question yields better outcomes, which is known as Chain-of-Thought (CoT, Wei et al., 2022b; Kojima et al., 2022). CoT only considers the intermediate task of "step-by-step reasoning" before answering the final question in reasoning tasks. This is extended by chained prompting (Wu et al., 2022) and least-to-most-prompting (Zhou et al., 2023). Our work points to the existence of a much broader search space for intermediate tasks, which has only been partially explored. We also underline that this work concerns instruction tuning in addition to (sequential) prompting.

Increased computation Prolonged generation incurs higher inference costs but also has higher computational capacity (Lanham et al., 2023; Goyal et al., 2023; Pfau et al., 2024). Our work might be considered from the perspective of training an LLM with a stretched length. 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.

6 Conclusion

In this work, we unveiled a major drawback in state-of-the-art, open-source models as large as Llama-3-70B and Mixtral-8×7B: they struggle to follow multiple task instructions within a single query. Accordingly, we proposed a new method, sequential instruction tuning (SIT), to equip LLMs with this ability. We systematically explore sequential instructions: from manually constructing sequential instruction data with a pre-defined intermediate task—such as translating or captioning for multilingual and multimodal question answering—to automatically constructing large-scale diverse sequential instructions from existing single-instruction datasets such as Alpaca or FlanCoT. Fine-tuning language models on SIT-enriched data not only helped them follow multiple instructions more faithfully but also recorded a better performance in complex tasks that require multi-step reasoning, such as maths and coding, as well as open-ended generation.

Social Impact

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 which is difficult to filter out.

References

AI@Meta. Llama 3 model card. GitHub, 2024.

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

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

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

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. Evaluating large language models trained on code. *arXiv preprint*, 2021.

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

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

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

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. Training verifiers to solve math word problems. *arXiv preprint*, 2021.

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

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

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

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

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

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

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

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

Haoyang Huang, Tianyi Tang, Dongdong Zhang, Xin Zhao, Ting Song, Yan Xia, and Furu Wei. 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*, 2023.

Drew A Hudson and Christopher D Manning. 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*, 2019.

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. Camels in a changing climate: Enhancing LM adaptation with Tulu 2. *arXiv preprint*, 2023.

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. Mistral 7B. *arXiv preprint*, 2023.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 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*, 2023.

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

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

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

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint*, 2023.

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

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

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

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, 2004.

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

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. The Flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

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

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

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. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

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

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

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

Libo Qin, Qiguang Chen, Fuxuan Wei, Shijue Huang, and Wanxiang Che. 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*, 2023.

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

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. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*, 2023.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 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*, 2019.

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

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

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

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

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

Tongshuang Wu, Michael Terry, and Carrie Jun Cai. 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*, 2022.

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

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. BERTScore: Evaluating text generation with BERT. In *International Conference on Learning Representations*, 2020.

Zhihan Zhang, Dong-Ho Lee, Yuwei Fang, Wenhao Yu, Mengzhao Jia, Meng Jiang, and Francesco Barbieri. PLUG: Leveraging pivot language in cross-lingual instruction tuning. *arXiv preprint*, 2023.

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. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

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. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

A Generalization 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 6, 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.

A.2 Generalisation to Other (Sequential) Instructions

To further understand the characteristics of language models trained on sequential instruction data, we analyzed the **generalization 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 3.1, 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 7, 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.

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

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

Table 7: 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 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 for Multimodal Question Answering

For cross-modal experiments involving both texts and images, we use the LAVIS[2] library for training and evaluation Li et al. (2023a). We fine-tuned InstructBLIP[3] 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 4 for the generation type classification and Figure 5 for the generation process.

[2]https://github.com/salesforce/LAVIS
[3]https://huggingface.co/Salesforce/instructblip-vicuna-7b

```
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 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 6. The prompt follows Zheng et al. (2023)'s design with a distinct feature checking whether all queries are responded to by the model.

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 precision at 10 (P@10).

C Result Breakdown

The complete results for XQuAD are shown in Table 8. Complete results for MGSM8k for models tuned in Alpaca and FlanCoT are shown in Table 9 and Table 10, 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.

```
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 5: 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.

C.1 *SeqEval*

Detailed results from our baselines and SIT models evaluated by our own *SeqEval* are reported in Table 11. In addition, we test these models on each intermediate test set at various iterations through developing the final *SeqEval*—the results are in Table 12

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

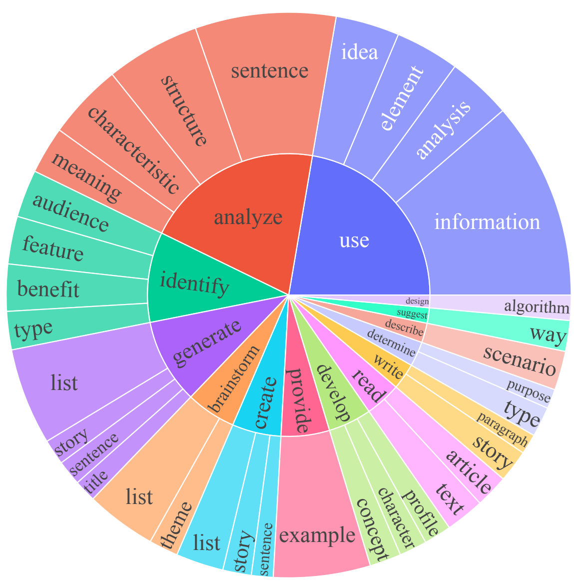

Figure 7: Top 15 root verbs (inner circle) and their top 4 direct nouns (outer circle) in Alpaca-SIT.

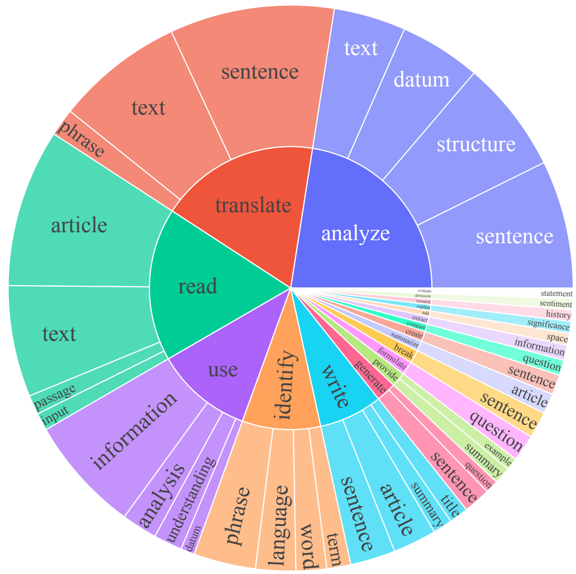

Figure 8: Top 15 root verbs (inner circle) and their top 4 direct nouns (outer circle) in FlanCoT-SIT.

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^G	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^G	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^G	**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 8: Complete breakdown for XQuAD results. SIT^M refers to the task-driven while SIT^G refers to the generalised version.

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 9: Complete results for 8-shots MGSM8k (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 10: Complete results for 8-shot MGSM8k (accuracy, %) fine-tuned on FlanCoT.

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**
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 (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 11: 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.

Iter	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 12: Comprehensive evaluation results on the *intermediate versions* of *SeqEval* with varying numbers of tasks. Same metrics as Table 11.