DoG-Instruct: Towards Premium Instruction-Tuning Data via Text-Grounded Instruction Wrapping

Yongrui Chen[1,2], **Haiyun Jiang**[3,*], **Xinting Huang**[3], **Shuming Shi**[3] **& Guilin Qi**[1,2†]
[1]Southeast University
[2]Key Laboratory of New Generation Artificial Intelligence Technology
and Its Interdisciplinary Applications (Southeast University), Ministry of Education
{yrchen,gqi}@seu.edu.cn
[3]Tencent AI Lab
{haiyunjiang,jeffjhhuang,shumingshi}@tencent.com

Abstract

The improvement of LLMs' instruction-following capabilities relies heavily on the availability of high-quality instruction-response pairs. Unfortunately, the current methods used to collect the pairs suffer from either unaffordable labor costs or severe hallucinations in the self-generation of LLM. To tackle these challenges, this paper proposes a scalable solution. It involves training LLMs to generate instruction-response pairs based on human-written documents, rather than relying solely on self-generation without context. Our proposed method not only exploits the advantages of human-written documents in reducing hallucinations but also utilizes an LLM to wrap the expression of documents, which enables us to bridge the gap between various document styles and the standard AI response. Experiments demonstrate that our method outperforms existing typical methods on multiple benchmarks. In particular, compared to the best-performing baseline, the LLM trained using our generated dataset exhibits a 10% relative improvement in performance on AlpacaEval, despite utilizing only 1/5 of its training data. Furthermore, a comprehensive manual evaluation validates the quality of the data we generated. Our trained wrapper is publicly available[1].

1 Introduction

Recent efforts in the NLP community have focused on *instruction-tuning* (Sanh et al., 2022; Mishra et al., 2022; Wei et al., 2022), i.e., improving large language models' (LLMs) capacity to understand and follow instructions (Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023a). Advanced LLMs have been trained to be capable of generating customized outputs when provided with specific instructions (with inputs), enabling them to adapt to new tasks without prior exposure.

As a crucial problem in improving LLMs' instruction-following capability, how to collect high-quality instruction-response pairs is gaining popularity. The majority of existing methods either rely on hiring professionals to write instructions for various NLP tasks (Wang et al., 2022; Conover et al., 2023) or promote the use of LLMs to automatically generate instructions (Wang et al., 2023; Taori et al., 2023; Yin et al., 2023). Unfortunately, these methods have limitations either in terms of scalability due to the labor-intensive nature of the annotation process or in terms of data quality due to the hallucination problem (Zhang et al., 2023; Zheng et al., 2023) associated with LLMs.

Recent research (Köksal et al., 2023; Li et al., 2023a) has provided a more potential idea: first directly using human-written documents as typical responses and then utilizing LLMs to predict the latent user instructions. This method, known as *instruction back-translation* (Li et al., 2023a), is based on the belief that human-written documents are inherently less prone to hallucinations compared to responses generated solely by LLMs.

However, we argue that even if a document is free of hallucinations, it is not always appropriate to employ it directly as a typical response. This is attributed to two main reasons: a) First, not all parts of a document are valuable in constructing a response. For example, the red part of the document (A) in Figure 1 is completely useless for back-translating the resulting instruction (the gold box). Moreover, valuable parts of the document often have fuzzy boundaries. For instance, the red text of (B) aims to create a tense atmosphere, again unsuitable to keep in response, but it also has some relevance to the topic (alien research) and is therefore difficult to be filtered out by simple preprocessing. b) Second, due to the different purposes of writing, there are often gaps in expression between

*Corresponding Author
†Corresponding Author
[1]https://github.com/Bahuia/Dog-Instruct

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 4125–4135
June 16-21, 2024

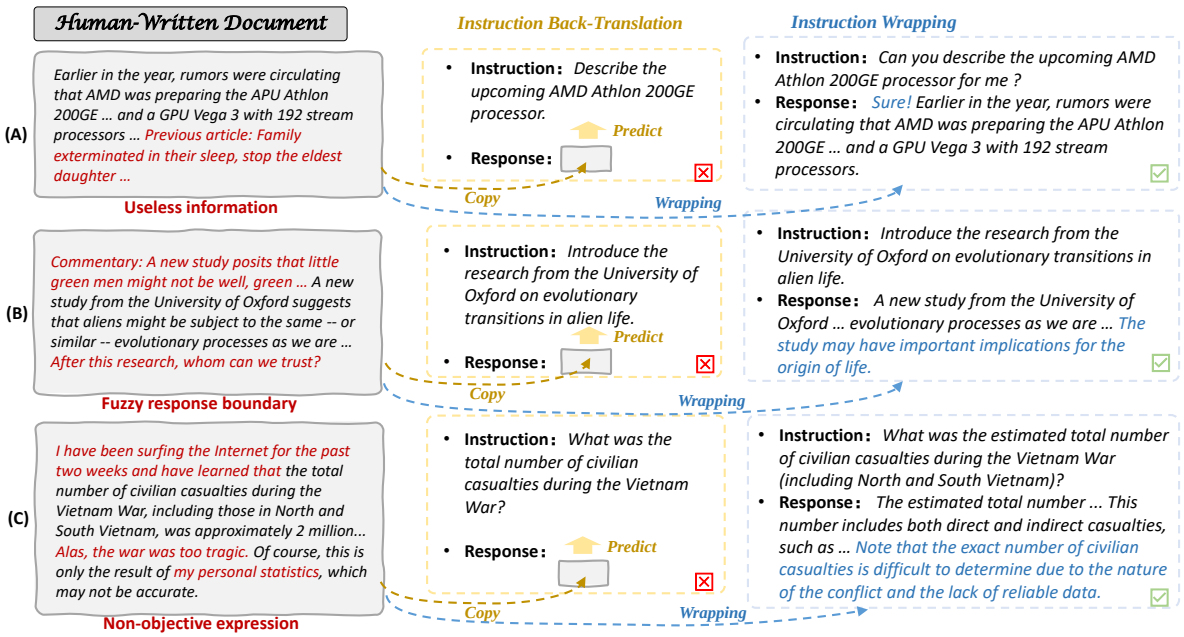

Figure 1: Differences between our proposed *instruction wrapping* with *instruction back-translation* (Köksal et al., 2023; Li et al., 2023a). Red text is not appropriate for responses. Blue text indicates that the original text has been added, deleted, or rewritten by LLM to align more closely with the desired standardized response.

the raw documents and the standard responses. As an illustration, the red portion of (C) contains multiple subjective descriptions, which deviates from the expected objectivity of an AI assistant.

In this paper, we propose a new paradigm for constructing instruction-tuned data, called *instruction wrapping*. It aims to train an open-sourced LLM to identify valuable parts from the original document and further transform them into fluent and objective instruction-response pairs.

Briefly, our proposed method consists of two stages as shown in Figure 2. In stage a), a well-aligned LLM is employed as the teacher to construct a meta-training set Ω for instruction wrapping. Each example in this set comprises a sampled document and its corresponding instruction-response pair, involving one of the following two views. In the alignment view, we employ in-context learning to guide the teacher LLM in generating instruction-response pairs based on human-written documents. It allows for the adaptation of the teacher LLM to various real document styles. In the diversity view, we begin with an existing diverse instruction set and prompt the teacher LLM to reversely generate a pseudo-document for each instruction-response pair. It ensures the training examples maintain instruction diversity. Subsequently, we use the meta-training set to perform supervised fine-tuning on a publicly released LLM, which serves as our instruction wrapper. In stage b), human-written documents from multiple domains are fed into our trained wrapper to generate instruction-response pairs. Then, a simple but efficient post-processing strategy is adopted to filter invalid examples based on the literal similarity. Eventually, we name the resulting dataset DOcument-Grounded Instructions (DoG-Instruct), containing 12.4K instruction-response pairs.

The LLM trained using DoG-Instruct achieves a remarkable 4.8% improvement in performance on AlpacaEval compared to the best-performing baseline, while using only 1/5 of the training data. Furthermore, it achieves state-of-the-art results on the other three widely-used benchmarks. Through further manual evaluation, we illustrate that our DoG-Instruct method effectively mitigates the issue of hallucination while aligning the raw document with the desired response in terms of style. In summary, the contributions of this paper include:

- We propose a novel paradigm that trains LLMs to generate instruction-response pairs based on human-written documents. It not only leverages the document to reduce the hallucinations of responses, but also aligns the style of the raw document with the ideal response using LLM.
- We release a well-trained Llama-based instruction wrapper capable of consistently generating

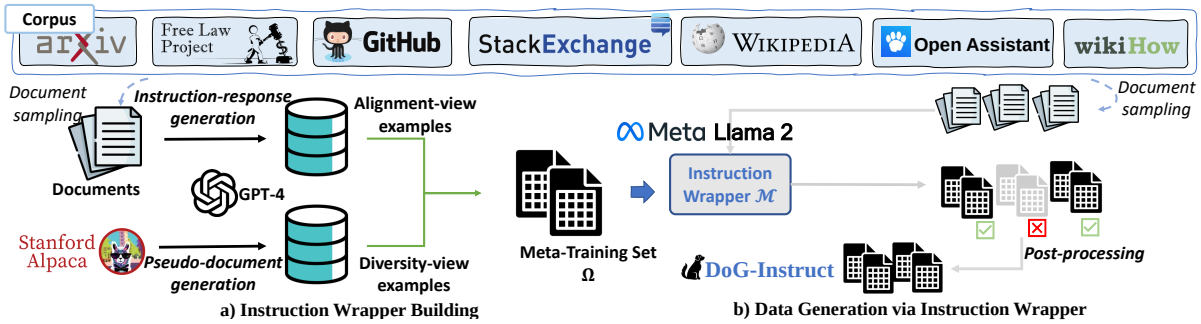

Figure 2: Overview of DoG-INSTRUCT construction process. In stage a), a meta-training set Ω is constructed using GPT-4 and utilized to train the instruction wrapper. In stage b), the wrapper generates instruction-response pairs for each sampled document, and a post-processing strategy is employed to filter out invalid examples.

high-quality instruction-response pairs for documents across multiple domains.

- We conducted a comprehensive evaluation, both automatic and manual, which demonstrates that the LLM trained using our generated data outperforms all the compared baselines.

2 Problem Formulation

Given a set of documents $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$, where each $\mathcal{D}_i$ is a human-written document, our goal is to construct a set of pairs $\{(\mathcal{X}_1, \mathcal{Y}_1), ..., (\mathcal{X}_m, \mathcal{Y}_m)\}$, where $m \leq n$, $\mathcal{X}_i$ and $\mathcal{Y}_i$ denote the instruction and response, respectively, and $(\mathcal{X}_i, \mathcal{Y}_i) := \mathcal{M}(\mathcal{D}_j)$. Here $\mathcal{M}$ is an LLM-based instruction wrapper that transforms $\mathcal{D}_i$ into an instruction-response pair.

3 Collection of DoG-Instruct Data

Figure 2 shows the entire process of our method. a) First, the instruction wrapper $\mathcal{M}$ is trained using the meta-training set Ω, which is constructed by the well-aligned GPT-4. b) Subsequently, $\mathcal{M}$ takes sampled documents $\mathcal{D}$ as inputs to generate $(\mathcal{X}, \mathcal{Y})$ for constructing DoG-INSTRUCT.

3.1 Corpus & Document Sampling

To create a diverse set of documents, we utilize the Pile (Gao et al., 2021) corpus, which is a multi-domain collection of human-written documents. From the Pile, we sample documents from six different domains: ArXiv, FreeLaw, StackExchange, Wikipedia, Github. Following existing work (Li et al., 2023a; Köksal et al., 2023), we also sample documents from Open Assistant[1] and WikiHow[2] to introduce some structured examples. We randomly choose several consecutive paragraphs from each original text in the corpus to serve as our document. To ensure that each document contains enough information to generate at least one instruction-response pair, we only keep the documents that contain a range of 500 to 1000 tokens.

3.2 Instruction Wrapper Building

To empower a general LLM with the capability of instruction wrapping, we need to construct sufficient training examples mapping the document $\mathcal{D}$ to the instruction-response pair $(\mathcal{X}, \mathcal{Y})$. Inspired by (Li et al., 2023a), we leave this job to the well-aligned GPT-4 (OpenAI, 2023) to minimize the cost of human annotations. We hypothesize that an ideal meta-training set Ω should fulfill two essential requirements: *alignment* and *diversity*. *Alignment* guarantees that Ω encompasses a wide range of real human-written documents, enabling the wrapper to comprehend different domains and writing styles. *Diversity*, on the other hand, ensures that Ω contains a variety of instructions, enabling the wrapper to generate diverse instructions effectively after training. Therefore, we collect examples of Ω from the following two views.

Alignment-view Examples. In this section, the examples are constructed by utilizing GPT-4 to directly generate instruction-response pairs for real human-written documents. To accomplish this goal, we harness the power of *in-context learning* (ICL). In particular, for each domain, 30 examples are first manually constructed as the seeds. Then, for each $\mathcal{D}$, the prompt fed to GPT-4 is denoted by $(\mathcal{X}^*, \mathcal{D}_1, \mathcal{P}_1, ..., \mathcal{D}_k, \mathcal{P}_k, \mathcal{D})$, where $\mathcal{X}^*$ is definition of mapping $\mathcal{D}_j$ to the instruction-response pair $\mathcal{P}_j = (\mathcal{X}_j, \mathcal{Y}_j)$. See Appendix A.1 for the full prompt. The resulting examples are denoted by Ω_a.

Diversity-view Examples. Intuitively, it is diffi-

[1] https://huggingface.co/datasets/OpenAssistant/oasst1

[2] https://huggingface.co/datasets/wikihow

cult to generate diverse instructions using just a few dozen manual seeds. Therefore, we start from the publicly released instructions, such as ALPACA, and then inversely fuse their provided instructions and responses to create pseudo-documents. Specifically, for each instruction-response pair $(\mathcal{X}, \mathcal{Y})$ sampled in ALPACA, we write the prompt to employ GPT-4 to integrate $\mathcal{X}$ and $\mathcal{Y}$ into a new document $\tilde{\mathcal{D}}$. We enable $\tilde{\mathcal{D}}$ to encompass all the content from $\mathcal{X}$ and $\mathcal{Y}$, but we intentionally blur their boundaries. This allows for the addition of new content as needed, while ensuring a smooth and coherent flow of information. These pseudo-documents $\tilde{\mathcal{D}}$ and their corresponding $(\mathcal{X}, \mathcal{Y})$ constitute the remaining training examples, denoted by Ω_d. Appendix A.2 gives the detail prompt.

Wrapper Training. we select LLAMA2 (Touvron et al., 2023b), an advanced LLM publicly available as our instruction wrapper $\mathcal{M}$ and perform *supervised fine-tuning* (SFT) on $\mathcal{M}$ using the constructed meta-training set $\Omega = \Omega_a \cup \Omega_d$. For each document D and its instruction-response pair $\mathcal{P} = (\mathcal{X}, \mathcal{Y})$ of Ω, we add a unified instruction $\mathcal{U}$ = "`Convert the given text into a task. Input is a text and Response contains two fields: #instruction# and #output#.`". Then, the training loss is calculated by a log-likelihood,

$$\mathcal{L}(\mathcal{U}, \mathcal{D}, \mathcal{P}) = -\sum_{j=1}^{|\mathcal{P}|} \log P(t_j | \mathcal{U}, \mathcal{D}, t_{<j}), \quad (1)$$

where t_j is the j-th token of $\mathcal{T}$. It is crucial to emphasize that although our meta-training set Ω may include hallucinations, we hypothesize that this does not affect the learning of the wrapper $\mathcal{M}$. This is because our primary objective for $\mathcal{M}$ is to learn the stylistic transformation from documents to instruction-response pairs with semantic consistency. During the inference phase, we exclusively utilize real human-written documents without pseudo-documents, which naturally reduces the occurrence of hallucinations.

3.3 Data Generation via Instruction Wrapper

In this stage, we use the trained $\mathcal{M}$ to generate instruction-response pairs for 20,000 human-written documents, which have been sampled using the method described in Section 3.1. To avoid the hallucination that the wrapper generates too much content unrelated to the original text, we propose a post-processing strategy for each generated task $\mathcal{T}_i$. Concretely, we devise a score $\sigma(\mathcal{T}_i) = \min(\tilde{\sigma}(\mathcal{P}_i, \mathcal{X}_i), \tilde{\sigma}(\mathcal{P}_i, \mathcal{Y}_i))$ to measure the similarity between the text and the instruction-response pair, where $\tilde{\sigma}(\mathcal{P}_i, s) = |t(\mathcal{D}_i) \& t(s)| / |t(s)|$ and $t(s)$ denotes the token set of text s. All examples $(\mathcal{P}_i, \mathcal{T}_i)$ will be removed where $\sigma(\mathcal{T}_i) < \theta$.

Table 1: Statistics of alignment-view examples Ω_a, diversity-view examples Ω_d, the meta-training set Ω and DOG-INSTRUCT. Here $x \pm y$ denotes the average x and standard deviation y.

	Example #	$\mathcal{X}$ **Token #**	$\mathcal{Y}$ **Token #**
Ω_a	306	16 ± 13	134 ± 126
Ω_d	2998	43 ± 35	140 ± 76
Ω	3371	41 ± 34	139 ± 81
DOG-INSTRUCT	12426	32 ± 79	310 ± 152

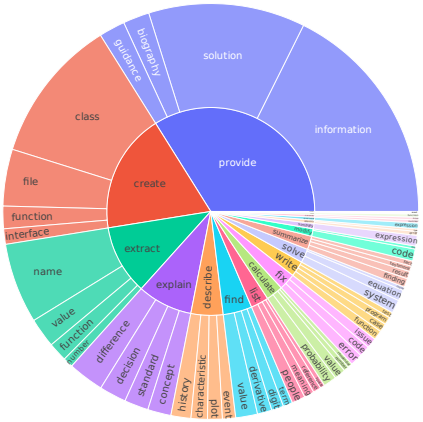

Figure 3: Instruction diversity of DOG-INSTRUCT data. The inner circle shows common root verbs with the corresponding common noun objects in the outer circle.

4 DoG-Instruct Statistics

Data Statistics. Table 1 shows the statistics of alignment-view examples Ω_a, diversity-view examples Ω_d, the meta-training set Ω, and DOG-INSTRUCT dataset. Theoretically, as long as there is a constant stream of text, our method has no upper limit on the amount of data. However, through experimentation, we discovered that competitive results can be achieved by using a mere 12k of our DOG-INSTRUCT. DOG-INSTRUCT tends to have longer responses compared to the examples in Ω. In addition, DOG-INSTRUCT have larger standard deviations regarding the response field than Ω. The top of Table 2 presents the statistical data for different domains in DOG-INSTRUCT.

Diversity of Instructions. We performed a diversity analysis on DOG-INSTRUCT using the method

Table 2: Statistics of different domains in DoG-INSTRUCT. Instruction, input, and output lengths are given as the number of tokens. BS denotes the Bert-Score(Zhang et al., 2019). OASST is short for Open Assistant.

	Wikipedia	FreeLaw	ArXiv	StackExchange	Github	OASST	WikiHow
# of Examples	5371	427	450	475	690	946	4060
Length of $\mathcal{X}$	15 ± 34	201 ± 207	119 ± 187	101 ± 104	54 ± 110	34 ± 51	11 ± 25
Length of $\mathcal{Y}$	347 ± 123	476 ± 123	577 ± 205	328 ± 121	328 ± 132	326 ± 121	326 ± 104
$\tilde{\sigma}(\mathcal{D}, \mathcal{Y})$	0.981	0.949	0.942	0.976	0.978	0.957	0.957
$\text{BS}(\mathcal{D}, \mathcal{Y})$	0.981	0.963	0.942	0.911	0.967	0.946	0.930

Table 3: Performance of the methods on the AlpacaEval benchmark (win rate over text-davinci-003 evaluated by GPT-4). The **Text-Grounded** field indicates whether the instruction generation is based on human-written text. The **Avg. Length** denotes the average token number of the model responses. Our DoG-INSTRUCT achieves the highest win rate (53.1%) with the least training examples (12.4K).

Data Generator	Dataset	Text-Grounded	# of Examples	Win Rate (%)	Avg. Length
text-davinci-003	LONGFORM	✓	23.7K	11.7	268
GPT-3.5-Turbo	SELF-INSTRUCT	×	82K	14.2	284
	ALPACA	×	52K	15.3	271
	DYNOSAUR	×	800K	2.9	142
	EVOL-INSTRUCT	×	70K	48.3	669
GPT-4	ALPACA-GPT-4	×	52K	44.5	653
LLAMA2-7B	HUMPBACK†	✓	18K	41.0	755
	DoG-INSTRUCT	✓	12.4K	**53.1**	**1149**

described by (Wang et al., 2023). Figure 3 illustrates the distribution of the verb-noun structure of instructions, showcasing the diverse range.

Relevance to Raw Documents. Additionally, we computed the relevance of the responses to the raw documents. The average relevance scores are displayed at the bottom of Table 2, with $\tilde{\sigma}$ representing the measure of literal relevance utilized in our post-processing, and BS denoting Bert-Score (Zhang et al., 2019) for evaluating the semantic relevance. Both in terms of literal score and semantic score, the responses exhibit a high level of relevance to the raw documents. However, they are not 100% aligned due to the appropriate rewriting carried out by our instruction wrapper.

5 Experiments

5.1 Experimental Setup

Compared Datasets. We compared our DoG-INSTRUCT with several typical instruction-tuning datasets: SELF-INSTRUCT (Wang et al., 2023), and ALPACA (Taori et al., 2023) are automatically generated by LLMs including GPT-3.5-Turbo and text-davinci-003. DYNOSAUR (Yin et al., 2023) repackages huggingface's existing NLP dataset and regenerates instructions for it using ChatGPT. LONGFORM (Köksal et al., 2023) and HUMPBACK (Li et al., 2023a) are most similar to our work in that they generate tasks by performing the instruction back-translation. Unlike these methods, DoG-INSTRUCT wraps the documents and carefully selects the valuable parts to compose a comprehensive instruction-response pair. Since HUMPBACK hasn't been released yet, we got an unofficial version [3] from HuggingFace, denoted by †.

Implementation Details. All our experiments ran on 8 Tesla V100 GPUs with FP16. We trained $\mathcal{M}$ using LoRA (Hu et al., 2022). The hyper-parameters were set as follows: (1) The batch size was set to 128. (2) The learning rate was set to 1×10^{-4}. (3) The epoch number was 7. (4) The cutoff token number was set to 2048. (5) The temperature and beam size were 0 and 4, respectively. (6) The LoRA target modules consisted of $[\text{q}_{\text{proj}}, \text{k}_{\text{proj}}, \text{v}_{\text{proj}}, \text{o}_{\text{proj}}, \text{up}_{\text{proj}}, \text{down}_{\text{proj}}, \text{gate}_{\text{proj}}, \text{embed}_{\text{tokens}}, \text{lm}_{\text{head}}]$.

[3] https://huggingface.co/datasets/Spico/Humback

Table 4: Rouge-L (R), Meteor (M), and Bert-Score (B) of different methods on the test sets of three benchmarks. All methods follow zero-shot settings.

Data Generator	Dataset	# of Examples	ELI5		LF-Test		Super-NI
			R(%)	M(%)	R(%)	M(%)	B(%)
text-davinci-003	LONGFORM	23.7K	7.5	5.4	24.9	18.1	81.8
GPT-3.5-Turbo	SELF-INSTRUCT	82K	9.8	8.2	22.4	16.5	83.0
	ALPACA	52K	10.1	8.8	23.1	17.3	82.9
	DYNOSAUR	800K	3.1	1.5	15.6	11.0	86.0
	EVOL-INSTRUCT	70K	18.9	18.4	25.2	21.8	85.6
GPT-4	ALPACA-GPT-4	52K	11.1	13.3	25.1	22.4	85.8
LLAMA2-7B	HUMPBACK†	18K	9.3	6.1	25.0	22.2	83.7
	DOG-INSTRUCT	12.4K	**19.0**	**19.7**	**25.9**	**23.6**	**86.1**

Table 5: Experimental results of ablation studies for all benchmarks used.

Stage	Setting	AlpacaEval	ELI5	LF-Test	Super-NI
		Win Rate(%)	M(%)	M(%)	B(%)
	DOG-INSTRUCT	**53.1**	**19.7**	**23.6**	**86.1**
Training	w/o alignment-view	46.7	18.5	20.2	85.2
	w/o diversity-view	12.5	12.1	15.7	83.8
	w instruction back-translation	5.9	9.3	16.6	81.7
Generation	w/o post-processing	32.0	17.2	23.3	85.9

5.2 Automatic Evaluation

To begin with, we conducted an automatic evaluation on multiple benchmarks. For each dataset being compared, We independently fine-tuned an identical baseline LLM using its respective training examples and evaluated its performance in accurately following the instructions.

Baseline LLM. We select LLAMA2-7B (Touvron et al., 2023b) + LORA (Hu et al., 2022) as the baseline LLM. For ease of presentation, we refer to the baseline LLM trained on dataset x as x-model.

Benchmarks. We first used the GPT-4 evaluation from AlpacaEval (Li et al., 2023b) to evaluate response quality on 805 instructions from the Alpaca Leaderboard. AlpacaEval compares the pairwise win rate against the reference model text-davinci-003. In addition, we conducted evaluations on three other NLG benchmarks: Long-Form Question Answering (ELI5) (Fan et al., 2019), Long-Form test set (LF-Test) (Köksal et al., 2023), and Super-NaturalInstructions (Super-NI) (Wang et al., 2022). None of the methods incorporate data from these benchmarks. i.e. zero-shot setting.

Automatic Evaluation Metrics. For AlpacaEval, we ran its scripts directly, using GPT-4 for evaluation. For ELI5 and LF-Test, we followed (Köksal et al., 2023; Yin et al., 2023) to calculate the Rouge-L (Lin, 2004) and Meteor (Banerjee and Lavie, 2005) scores. These scores are computed by comparing the model outputs with the provided references in the respective datasets. For Super-NI, we utilize Bert-Score (Zhang et al., 2019) for evaluation instead of other long-text metrics like Rouge. This choice is made due to the typically short nature of the outputs in this dataset.

5.2.1 AlpacaEval Results.

The win rate and average length of model responses for different methods on AlpacaEval are presented in Table 3. It is worth highlighting that despite utilizing the least amount of data, we achieved the best performance while maintaining the same baseline LLM premise at the 7B model scale. The DYNOSAUR-model demonstrates the lowest performance, potentially due to its output being excessively standardized and concise rather than a detailed reply. By surpassing all non-text-based methods, we demonstrate the effectiveness of human-written text in mitigating LLM hallucinations. In comparison to the text-grounded method Hump-

back, we achieved a substantial improvement by adapting our command wrapper to the AI response style, resulting in significant advancements.

5.2.2 ELI5, LF-Test and Super-NI Results.

Table 4 shows the Rouge-L (R), Meteor (M), and Bert-Score (B) of different models on ELI5, LF-Test, and Super-NI. Our method outperforms all the compared methods across all three benchmarks in terms of Rouge-L, Meteor, and Bert-Score, achieving superior performance in all evaluation metrics. This observation showcases that our dataset enables better alignment between LLM outputs and human annotations, indicating the efficacy of our method in improving the performance of LLM models.

GPT-4 Evaluation. To mitigate any bias introduced by conventional metrics such as Rouge, we employed GPT-4 for evaluation on ELI5, LF-Test, and Super-NI benchmarks. We calculated the win/tie/lose rates by comparing the model responses with the reference responses provided by the benchmarks. The results are shown in Figure 4. Our DoG-INSTRUCT-model consistently achieves the highest win rate across all benchmarks.

5.2.3 Ablation Study.

We compared the LLM performance using different settings to construct DoG-INSTRUCT.

- **w/o alignment-view**: we reconstructed the meta-training set Ω without any examples constructed by real human-written texts;
- **w/o diversity-view**: we reconstructed Ω without any examples fused by the instructions and responses from ALPACA;
- **w instruction back-translation**: we replaced our instruction wrapping with instruction back-translation to reconstruct DoG-INSTRUCT while keeping the input documents unchanged.
- **w/o post-processing**: we removed post-processing when generating DoG-INSTRUCT.

Table 5 shows the experimental results. Our DoG-INSTRUCT equipped with all components performs best in terms of all metrics. Dramatic performance degradation demonstrates that the adaptation of the PLM to the task format is critical to the effectiveness of prompt tuning.

5.3 Human Evaluation

While the automatic evaluation in the previous section provided an overall assessment of model performance, we now aim to specifically evaluate the effectiveness of our DoG-INSTRUCT in reducing hallucinations and aligning model responses with human-like outputs.

Table 6: Human evaluation on dataset qualification. For each dataset, we randomly sampled 50 examples. Here ↓ means the smaller the value, the better.

Dataset	V (%)	H (%) ↓	F (%)
ALPACA-GPT-4	94	22	92
EVOL-INSTRUCT	94	18	94
LONGFORM	76	14	84
HUMPBACK†	48	12	62
Ω	92	20	94
DoG-INSTRUCT	**96**	**12**	**96**

5.3.1 Data Quality

We randomly select 50 examples from each dataset and manually evaluate their quality. Since the generated tasks may involve knowledge from several different domains, we require that the annotator needs to retrieve the corresponding evidence using the search engines and compare them one by one. The entire process of manual evaluation took approximately 8 man-hours.

Human Evaluation Metrics. a) *validation* (V) indicates the percentage of the example whose response follows the instruction. b) *hallucination* (H) measures the percentage of the example whose response contains factual errors. c) *fluency* (F) indicates the percentage of the example that has instructions and responses that are smooth and fluent.

Results. The results are shown in Figure 6. Both ALPACA-GPT-4 and EVOL-INSTRUCT demonstrate higher levels of hallucination due to their complete reliance on using LLMs to generate instruction-response pairs from scratch. By generating tasks from human-written documents, both LONGFORM and HUMPBACK effectively mitigate the hallucinations. Nevertheless, the inclusion of noise in real text leads to lower fluency (F) compared to datasets that are fully generated by LLMs. In contrast, our method combines the use of human-written text as factual support with style modification through LLMs, leading to superior performance across all three metrics.

5.3.2 Text-Grounded Generation Capability

For the same document, we compared the quality of generated instruction-response pairs by employing two different methods: instruction wrapping

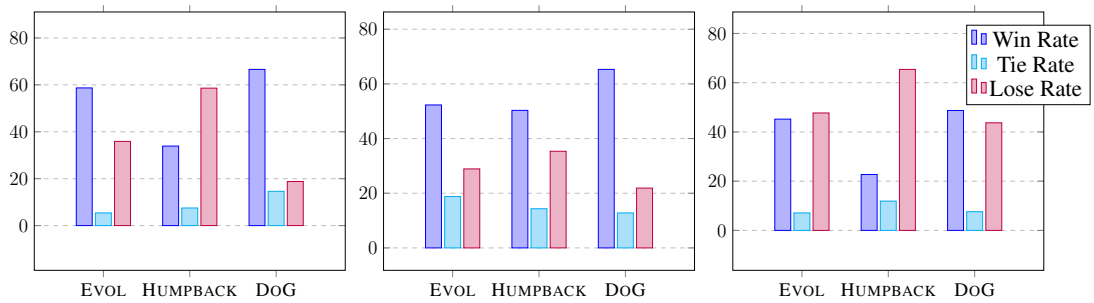

Figure 4: GPT-4 automatic evaluation results on subsets of Eli5 (left), LF-Test (middle), Super-NI (right). To account for the cost of GPT-4, each subset contains 200 examples that randomly sampled from the original test sets. The win/tie/lose rates are computed by comparing the model responses with the given reference responses.

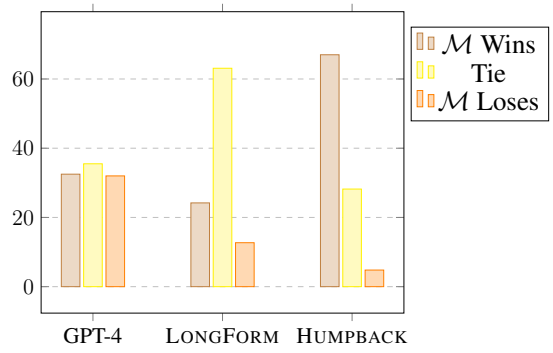

Figure 5: Human evaluation comparing DOG-INSTRUCT with various text-grounded methods. The evaluation was carried out using the same set of human-written documents as input for all methods.

and instruction back-translation. Specifically, we randomly selected 100 documents from the corpus used in LONGFORM and HUMPBACK to feed our instruction wrapper $\mathcal{M}$. The results, depicted in Figure 5, demonstrate that our wrapper yields instruction-response pairs of superior quality for the same document. Furthermore, we randomly sampled approximately 100 documents from our and had GPT-4 to perform instruction wrapping. Figure 5 illustrates that the instruction-response pairs generated by our $\mathcal{M}$ exhibit competitive quality to those produced by GPT-4.

6 Related Work

Instruction Tuning Humans possess the ability to effortlessly comprehend and execute tasks based on verbal instructions (Touvron et al., 2023a; OpenAI, 2023; Touvron et al., 2023b). Likewise, advancements in deep learning have enabled Language Models (LLMs) (Brown et al., 2020; OpenAI, 2023; Chowdhery et al., 2022; Touvron et al., 2023a) to acquire the capability to understand and follow instructions. Instruction tuning serves as a promising method, involving the fine-tuning of LLMs using training data and instructions from a collection of upstream tasks(Sanh et al., 2022; Mishra et al., 2022; Wei et al., 2022; Chung et al., 2022; Longpre et al., 2023; Peng et al., 2023).

Instruction-Tuning Data Collection The collection of high-quality instruction-tuning data (Xu et al., 2023; Yin et al., 2023; Honovich et al., 2023) is a pressing issue in enhancing the capability of instruction-following. Previous methods can be broadly categorized into three main groups. Firstly, methods like SUPER-NI (Wang et al., 2022) and DOLLY (Conover et al., 2023) rely on hiring professionals to create instructions for diverse NLP tasks. Secondly, methods such as SELF-INSTRUCT (Wang et al., 2023) and ALPACA (Taori et al., 2023) advocate for the use of LLMs to automatically generate instruction-tuning data, thus eliminating the need for manual labor. Lastly, Dynosaur (Yin et al., 2023) employs LLMs to convert existing NLP datasets from Huggingface into instruction-tuning data at a reduced cost. The work most related to this paper is (Köksal et al., 2023; Li et al., 2023a). It uses a human-written document as a natural response and leverages an LLM to generate the corresponding instruction based on the response. In contrast, our instruction wrapper selects the valuable parts of the documents for constructing appropriate responses.

7 Conclusion & Limitation

This paper introduces a new method called instruction wrapping, which enables the automatic collection of high-quality instruction-tuning data from human-written documents. Our trained instruction wrapper not only utilizes documents to mitigate response hallucinations but also modifies raw documents to align them with the standard response

style. Through comprehensive evaluations, we demonstrate that our method achieves remarkable results on various widely used benchmarks while utilizing the fewest training examples. The limitations of our method are that it cannot handle excessively long documents and can only generate a single task for a document. In future work, we will explore generating more complicated instructions that involve multiple longer documents.

Acknowledgement

This work is partially supported by National Nature Science Foundation of China under No. U21A20488. We thank the Big Data Computing Center of Southeast University for providing the facility support on the numerical calculations in this paper.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

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.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: long form question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3558–3567. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2021. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14409–14428. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. Longform: Optimizing instruction tuning for long text generation with corpus extraction. *CoRR*, abs/2304.08460.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Luke Zettlemoyer, Omer Levy, Jason Weston, and Mike Lewis. 2023a. Self-alignment with instruction backtranslation. *CoRR*, abs/2308.06259.

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

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

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 *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3470–3487. Association for Computational Linguistics.

OpenAI. 2023. Chatgpt.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *CoRR*, abs/2304.03277.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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. `https://github.com/tatsu-lab/stanford_alpaca`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

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 (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5085–5109. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned

language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244.

Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. *CoRR*, abs/2305.14327.

Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. 2023. How language model hallucinations can snowball. *CoRR*, abs/2305.13534.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. 2023. Why does chatgpt fall short in answering questions faithfully? *arXiv preprint arXiv:2304.10513*.

A Full Prompt to Construct the Meta-Training Set

A.1 Prompt for Constructing Ω_a

The full prompt for building the meta-training set is as follows.

```
  For the given text, design a task.
Each task contains three fields,
instruction, input, and output.
instruction defines a task in natural
language.
Instruction is a complete definition of
how an input text (e.g., a sentence or a
document) is expected to be mapped to an
output text.
Requiring instruction, input, and output
are derived from text wherever possible.
Input can be empty to indicate that the
task has no input.
Instruction must be in imperative
sentences formal.
Here are demonstrations where your
response should be as different from
theirs as possible.
{}
#text#:  "{}"
```

A.2 Prompt for Constructing Ω_d

The full prompt for building the meta-training set is as follows.

```
  Combine the following instruction and
output into a single coherent text.
You can add, delete, or modify some
content as appropriate to make the
combined text logically sound.
#instruction#:  "{}"
#output#:  "{}"
```