Addressing Format Faithfulness Challenges in Language Models

Anonymous ACL submission

Abstract

The zero- and few-shot prompting paradigms in large language models (LLMs) have significantly improved the accessibility and flexibility in language-related tasks, where the need for task-specific architecture design or supervision is eliminated. Along with the convenience, these paradigms also introduce the requirements of output format specifications, thereby mandating users to devise an output format and include it in the prompt as a request for LLMs to faithfully adhere to. To study the ability of LLMs to comply with format specifications, we identify the concept of format faithfulness. Based on the formal definition and the detailed taxonomization of format faithfulness, we present FORMATBENCH, a benchmark that covers full categories of format faithfulness in our taxonomy and a wide range of LLM application scenarios. Extensive experiments on FORMATBENCH reveal that state-of-the-art LLMs can still have difficulties in generating basic structured output as instructed. To improve the format faithfulness of LLMs, we design and implement three adaptation approaches, namely format regulation, format tuning, and format refinement. Detailed analyses of these approaches validate their effectiveness in improving format faithfulness rate by up to 9.8%. Our codes and datasets are publicly available at Anonymous_Link.

1 Introduction

Recent years have witnessed a significant upsurge in the development and deployment of large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023b; Achiam et al., 2023). With their exceptional zero-shot and few-shot capabilities, LLMs have revolutionized the paradigm of language-related tasks, where a question can be understood and solved to the best without task-specific architecture or supervision (Radford et al.).

The zero- and few-shot prompting paradigms have introduced a new problem in task solving procedure, namely, the specification of the output format. To elaborate, LLMs' task solving paradigm mandates that users must devise an output format and include it in the prompt as a request for LLMs to adhere to. The format specification holds significant importance in tasks of wide concern as illustrated in Figure 1:

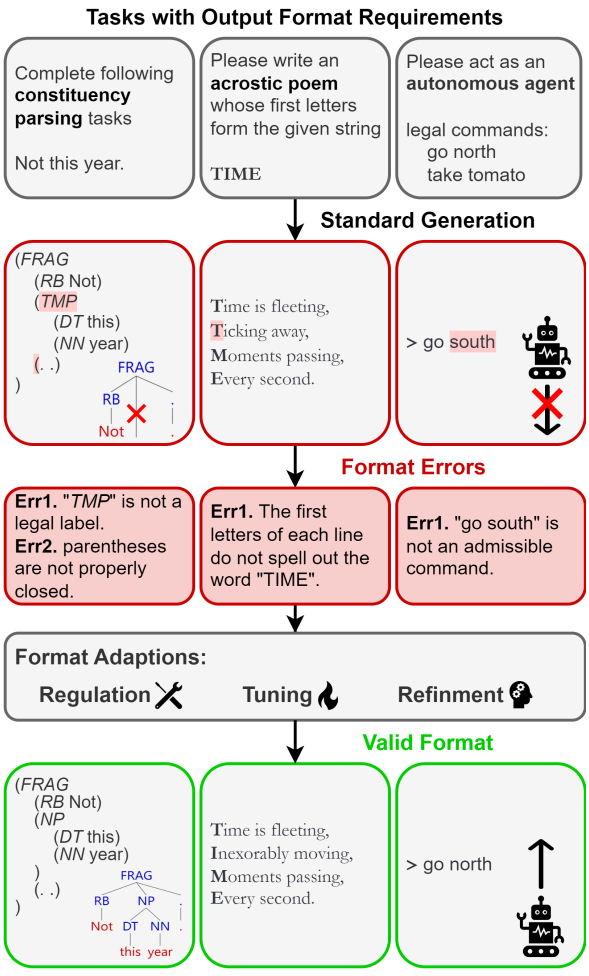

Figure 1: LLMs have difficulties in adhering to formatting instructions in NLP tasks (left), artistic creation (mid), and agent simulation (right). Consequently, this inability of LLMs results in format errors.

- The rigorous output format is necessary for

various natural language processing (NLP) tasks, such as named entity recognition, text-to-data conversion, and syntactic parsing.
- Certain creative works, such as poems, intrinsically possess rigorous forms, including acrostic, sonnet, and numerous others.
- LLM-based autonomous agents need to adhere a pre-defined format to interact with internal and external environments.

To summarize, the ability to adhere to pre-defined format specifications is of utmost importance in the deployment of LLMs. This ability, which we refer to as **format faithfulness**, is a crucial aspect to consider in many real-world tasks.

However, a comprehensive study of format faithfulness is still lacking. Firstly, there is a significant gap in the literature regarding the establishment of benchmarks to evaluate this capability. While a few works have introduced datasets related to formatting, they have primarily focused on one specific task, such as tool using (Qin et al., 2023), text-to-data (Tang et al., 2023), and code generation (Skreta et al., 2023). Secondly, there is still a lack of holistic evaluation and comparison of adaptation approaches aimed at improving format faithfulness.

To address the gap in comprehensive benchmarks, we offer a formal definition and a detailed taxonomy of format faithfulness. Based on the taxonomy, we propose FORMATBENCH, a benchmark consisting of a wide range of tasks to comprehensively evaluate the format faithfulness of LLMs. FORMATBENCH fully covers the categories in our format faithfulness taxonomy, thus ensuring the measurement completeness. Extensive experiments on the benchmark reveal that FORMATBENCH poses significant challenges to even the most capable models with simple format requirements, such as selecting among legal options.

To fill the gap in adaptation approaches, we further design and implement format regulation, format tuning, and format refinement to improve format faithfulness. We compare these three methods on FORMATBENCH, reveal their strength in enhancing LLMs' format following capability, and also point out their weakness in stability or robustness. We also show that the improvement of format faithfulness leads to improved general output quality. These insightful analyses can not only pave the path for future studies on format faithfulness, but also offer useful references in practical deployment.

Our contributions are summarized as follows:

- We introduce the concept of format faithfulness and its taxonomy, as a means to investigate the ability of LLMs to adhere to format specifications, which holds significant importance within the zero-shot and few-shot prompting paradigms.
- To evaluate format faithfulness, we develop FORMATBENCH, a comprehensive benchmark covering a diverse range of tasks, including two datasets of our own construction. Experiments show FORMATBENCH is challenging for even the most capable LLMs.
- We design and implement three format adaptation approaches, namely format regulation, format tuning, and format refinement. Analyses of these approaches validate their effectiveness in improving format faithfulness.

2 Related Work

Controllable Text Generation Controllable text generation (CTG) aims to steer a generative model to generate desired text according to given control conditions (Prabhumoye et al., 2020; Zhang et al., 2023a). Both our proposed format faithfulness and CTG involve improving controllability in LLMs, but the two concepts are different in tasks and methods. In the task aspect, CTG usually introduces soft control conditions like sentiment, topic, and attributes (Keskar et al., 2019; Xu et al., 2020), with a few exceptions (Li et al., 2020; Lin et al., 2020). However, format faithfulness is focused on hard control conditions relating to format, which will be elaborated in Section 3. In the method aspect, common CTG methods either adopt specially designed fine-tuning schema or modify the sampling procedure in decoding steps (Miao et al., 2019; Qin et al., 2022; Kumar et al., 2022), but format adaptations mainly considers techniques that allow for easy adaptation and quick deployment.

LLM Benchmarks In recent years, there has been significant attention paid to benchmarks and evaluation metrics in language modeling fields. Several notable benchmarks have been developed to evaluate the holistic effectiveness of LLMs (Wang et al., 2018; Wang et al.; Liang et al., 2022; Srivastava et al., 2023). Additionally, a few benchmarks have been proposed to evaluate format-related aspects (Qin et al., 2023; Tang et al., 2023). However, all previous format-related benchmarks are task-specific, failing to provide a comprehensive evaluation of overall format faithfulness. Un-

like previous works, we develop an LLM-centric benchmark covering all categories of format faithfulness in our proposed taxonomy, thus ensuring the completeness of the evaluation.

LLM Adaptations Foundation models can acquire general language understanding and problem solving abilities, but on specific tasks, further adaptation of LLMs can be beneficial (Zhao et al., 2023). One way to adapt LLMs to specific goals is to augment LLMs with reasoning or tools, and another is to conduct adaptation tuning (Mialon et al., 2023). To adapt LLMs for better understanding and obedience of format instructions, we utilize these LLM adaptation techniques to design format regulation, format tuning, and format refinement.

3 Definition and Taxonomization

In this section, we delve into a comprehensive discussion on the definition and the taxonomization of format faithfulness.

3.1 Definition

Although both format requirements and other conditioned generation requirements (e.g. emotion and topic constraints) necessitate a generative model to produce text with specified features, there is a fundamental distinction to consider, namely, discriminability. To be specific, the fulfillment of a format requirement can be ascertained by determining if the output can be recognized by the corresponding formal computational model, such as a deterministic finite automaton.

Therefore, format faithfulness, as the ability to adhere to format requirements, can be defined as the extent to which generative models can understand format-related instructions, and ensure the generated text are within the corresponding decidable language. Formally, the format faithfulness of a language model $\mathcal{M}$ can be formulated as,

$$\mathbb{E}_{t \in \mathcal{T}}[\mathcal{M}(t) \in \mathcal{L}_t] \quad (1)$$

where $\mathbb{E}$ is the expectation symbol, $\mathcal{T}$ denotes the set of all input instructions, $\mathcal{M}(t)$ is the model output, and $\mathcal{L}_t$ represents the language of a computational model corresponding to instruction t.

3.2 Taxonomy

In addition to providing a theoretical definition, we provide a taxonomy of format faithfulness in a comprehensive manner inspired by previous work (Schopf et al., 2023). The categorization of format faithfulness requirements is firstly conducted based on task types including classification, sequence labelling, and generation. Subsequently, we further refine the categorization by format requirements within each task, as depicted in Figure 2.

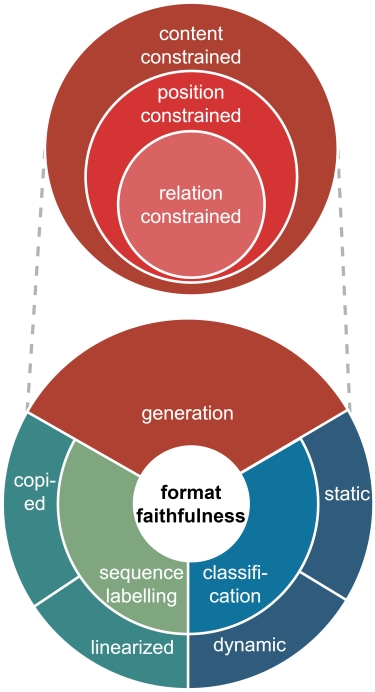

Figure 2: Taxonomy of format faithfulness. The constraints in the sub-types of generation tasks (upper part) form a containment relationship, i.e., the inner constraints contain the outer ones.

Classification tasks refer to the process of categorizing given input into pre-defined classes based on specific semantics. The common format requirement in classification tasks is to ensure that the generated answer is among the legal options. We further categorize these tasks into static classification and dynamic classification, where the legal option set remains the same throughout the processing period in the static ones, but can change from step to step in the dynamic ones.

Sequence labelling tasks involve assigning labels to each word within the input text. Generally, encoder-based models, such as BERT (Kenton and Toutanova, 2019), are well-suited for such tasks. However, generative language models need adapted output formats to effectively handle these tasks. We classify sequence labelling tasks based on the adapted format, including the copied format and the linearized format. The copied format directly replicates the span in the input that requires the label assignment, while the linearized format transcribes the entire input and incorporates tags to

represent the labels. Both methods should adhere to the original input without any modifications, while the linearized format additionally requires the inclusion of tags to be valid in terms of format.

Generation tasks involve the generation of new text using given input instructions and pre-defined rules. The basic constraints in generation tasks are about generated content (the outermost circle in Figure 2), such as what should be included or excluded. Further constraints (the middle circle) additionally specify the position of the constrained content. Even further, in addition to the content and position, the relation constraints (the innermost circle) ensure a certain level of consistency among the constrained parts, rather than treating them as independent elements.

4 FORMATBENCH

We endeavor to conduct a comprehensive evaluation of format faithfulness. To this end, we include various tasks in FORMATBENCH to cover all categories of the format faithfulness taxonomy. The tasks included in the benchmark are either adapted from established NLP datasets, or built on our own as novel datasets. The aggregated statistics of FORMATBENCH are presented in Table 1. In this section, we provide an introduction to FORMATBENCH and outline the specific format requirements other than common ones for each task.

Task	Category	|Train|	|Test|
QC	C:static	5,452	500
Agent	C:dynamic	4,440	514
QA	S:copied	86,821	5,928
NER	S:linearized	14,041	3,453
CapGen	S:linearized	229,703	542
MTT	G:content	-	2,951
AcroP	G:position	-	987
FTime	G:position	-	5,036
Parse	G:relation	-	3,912
XDLGen	G:relation	-	660

Table 1: The statistics of FORMATBENCH. The "Category" column indicates the format faithfulness category to which the task belongs, where "C", "S" and "G" represent classification, sequence labelling and generation tasks respectively. Tasks shown in bold use our own collected or labeled data.

4.1 Classification Tasks

QC The Text TRtrieval Conference (TREC) Question Classification (QC) (Li and Roth, 2002; Hovy et al., 2001) is a task that, given a question, maps it to one of the given classes, which provides a semantic constraint on the sought-after answer. QC is a static classification task, where the legal option set remains the same in all questions.

Agent The First TextWorld Problems (FTWP) (Adam et al., 2019) is to build an AI agent that moves automatically and efficiently in a text simulated world according to text feedback, and finally completes the given goal. Agent task offers a dynamic classification scenario where the admissible choice options can vary from one turn to another as the agent interacts with the environment.

4.2 Sequence Labelling Tasks

QA Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a reading comprehension dataset, where the answer to every question is a segment of text, or span, from the corresponding reading passage. We use a copied format in this task, i.e., requiring LLMs to directly copy the span of the passage without modification.

NER CoNLL-2003 (Sang and De Meulder, 2003) is a named entity recognition (NER) task to detect and categorize named entities. Under the linearized format, an LLM should strictly copy the given sentence, with added part-of-speech tags forming a legal flat NER schema.

CapGen We adapt MuST-Cinema (Karakanta et al., 2020) dataset, a multilingual speech translation corpus built from TED subtitles, to construct the Caption Generation (CapGen) task. CapGen involves inserting end-of-block and end-of-line tags in the raw English text to represent the split of captions in videos, thus simulating the generation of English video captions. Apart from common requirements of linearized format, two simple but meaningful heuristic format requirements are posed, including (1) each subtitle block must contain no more than two lines, and (2) each subtitle line must contain no more than 42 characters.

4.3 Generation Tasks

MTT WMT 2023 Terminology Shared Task (Semenov et al., 2023) is a Germany-English machine terminology translation (MTT) task that challenges machine translation systems to accurately and effectively translate technical terms and specialized vocabulary. MTT belongs to the content-constraints-only category, where we examine whether a target

term is included in the translation if the corresponding source term appears in the source sentence.

AcroP An acrostic poem is a literary form in which the initial letter of each line is arranged to spell out a hidden message. We combine existing datasets and acrostic poems crawled from the Internet as detailed in Appendix C, introducing the task of Acrostic Poetry Generation (AcroP). In this task, an LLM is challenged to compose an acrostic poem, adhering to the format of having the first letter of each line spell out the intended message. AcroP falls under the position constrained category, as both the content and position of generated text should be subject to constraints.

FTime Formatted Time Generation (FTime) task is to generate formatted time representations based on natural language instructions. This task holds particular relevance in reminder applications, which involve the translation of natural languages into formatted time strings. Examples of FTime task are provided in Table 2, where the illustrations of both single-trigger and repetitive-trigger time format are displayed. Detailed requirements, construction procedures, and quality control of FTime are shown in Appendix C. FTime also poses a position constraint to generated time representations.

Ref.	`20021019T140000:Saturday`
Inst.	... soup will be ready in 20 minutes ...
Res.	`20021019T142000`
Ref.	`20121215T090000:Saturday`
Inst.	... walk my dog at 10 a.m. every Monday.
Res.	`R-1/20121217T100000/P0Y0M7DT0H0M0S`

Table 2: In FTime, an LLM is to generate a time string (Res.) referred to by the instruction (Inst.), assuming that the instruction is issued at the reference time (Ref.). The output should be in accordance with the single-trigger (top) or repetitive-trigger (bottom) format.

Parse We conduct constituency parsing on the open source subset of the Penn Treebank (PTB) (Marcus et al., 1993) using the bracket sequence representation of a constituency tree. Our implementation requires the output to form a legal bracket sequence, and use legal labels in each node of the tree. The requirement constitutes a relation constraint, as it involves considering bracket matching as a whole instead of independent components.

XDLGen XDL (Chemical Description Language) (Seifrid et al., 2022) is an XML-based programming language used in chemical synthesis specification and experimental procedure transfer among robots and laboratories. Following previous work (Skreta et al., 2023), we conduct XDL Generation (XDLGen) task by examining the ability of LLMs to generate compilable XDL programs given the description of XDL. The format requirement of XDLGen task also constitutes a relation constraint.

5 Format Adaptation Methods

In this section, we introduce the adaptation methods employed to improve the format faithfulness.

5.1 Format Regulation

Format regulation refers to a non-parametric process based on prescribed rules aiming at steering LLM generation for format requirements, and it can be divided into generation-time intervention and post-generation editing.

Generation-time intervention incorporates logits processors into decoding modules to directly modify the prediction scores of LLMs. This method can be easily adopted for classification tasks including QC and Agent by masking all the illegal generation tokens and only allowing for admissible options. It can also applied to Acrop task by forcing an acrostic character after each line break.

Post-generation editing approaches directly modify the text generated by LLMs to ensure format faithfulness (De Cao et al., 2020; Zhang et al., 2023b). In Parse task, we use the same strategy as Bai et al. (2023) to guarantee that the bracket sequence achieves a balance by adding left brackets to the beginning of the output or appending right brackets to the right.

5.2 Format Tuning

The abilities of LLMs can be further adapted according to specific goals by fine-tuning (Zhao et al., 2023). For example, instruction tuning (Wei et al., 2021; Ouyang et al., 2022; Chung et al., 2022) improves zero-shot performance on unseen tasks. More relevantly, recent work (Tang et al., 2023) fine-tunes LLMs to generate well-structured data on specific tasks.

Inspired by these works, we propose to conduct format tuning on LLMs to improve the overall format faithfulness. Format tuning is to organize format-related instructions and their corresponding answers into formatted training data, and to train LLMs with given data. It is expected that models

will gain better format faithfulness within trained tasks, and the format understanding and obeying ability can generalize to unseen tasks.

We sample the training set of QC, QA, NER, and CapGen to construct our format tuning data. After fine-tuning, we evaluate the models on all tasks in FORMATBENCH, including both in-training and out-of-training tasks.

5.3 Format Refinement

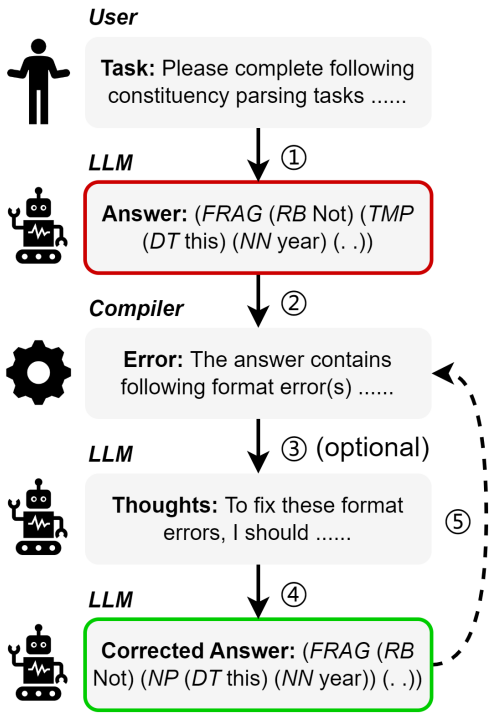

Figure 3: The workflow of format refinement. Given an instruction containing format requirements (①), an LLM often generates incorrectly formatted content, whose format errors can be detected by a non-parametric format compiler (②). Optionally, the LLM can reflect based on the compilation errors and generate revising thoughts (③). Finally, the corrected answer is given based on all the information (④). The iteration can repeat until no compilation error is detected, or a certain stopping criterion is reached (⑤).

There have been many impressive works on augmenting LLMs with internal reflection (Wei et al., 2022; Madaan et al., 2023) and external tools (Peng et al., 2023; Gou et al., 2023) to refine their initial content, and a few works among them focus on generating well-structured codes (Skreta et al., 2023). However, none of these works endeavor to improve the format faithfulness of LLMs on all tasks.

To fill this gap, we propose format refinement as a general prompt schema for improving format faithfulness on all tasks. Format refinement is designed guided by the decidable nature of the format. Specifically, the format faithfulness of any task is defined by a formal language, enabling the implementation of a format compiler that can detect format errors. The detailed implementation is shown in Figure 3, where an LLM iteratively polishes the output format according to error information from a format compiler. Optionally, we further augment the refinement process with LLM internal thoughts.

6 Experiments

6.1 Settings

We conduct evaluation experiments on FORMATBENCH with LLaMA (Touvron et al., 2023a), LLaMA2 (Touvron et al., 2023b), and GPT-3.5 (`gpt-3.5-turbo-instruct`) (OpenAI, 2021). We implement format regulation and format refinement on all three models whenever applicable, and additionally conduct format tuning on locally deployed LLaMA and LLaMA2. The generation configuration and the detailed implementation of three approaches are shown in Appendix A.

The evaluation metric employed in the experiment is format faithfulness rate. This metric measures the percentage of the generated text that adheres faithfully to the specified format. The format correctness depends on the specific task, as is broadly outlined in Section 4, and comprehensively defined in Appendix B.1.

6.2 Results

The main results are presented in Table 3.

Models Comparing the three models, GPT-3.5 exhibits significantly better faithfulness to format specifications than others, as the average format faithfulness rate of GPT-3.5 (63.3) surpasses that of LLaMA (45.9) and LLaMA2 (50.8) by a considerable margin. Moreover, a format faithfulness improvement is observed among three format adaptation methods. A detailed comparison among the three methods is conducted in Section 7.

Tasks It can be observed that some format tasks are still highly challenging for even the most capable models like GPT-3.5. The limited performance of LLMs to adhere to format specifications, which appear straightforward and do not require expert-level language understanding and manipulation ability, indicates the current LLMs' deficiency in maintaining format faithfulness.

Models	QC	Agent	QA	NER	CapGen	MTT	AcroP	FTime	Parse	xdlGen	avg.
LLaMA	89.6	81.7	88.5	74.1	22.1	29.7	0.0	72.8	0.3	0.0	45.9
+ regulation	100.0	100.0	-	-	-	-	85.9	-	2.5	-	-
+ tuning	81.4	94.2	95.0	76.0	22.3	48.9	0.1	30.6	1.3	0.0	45.0
+ refinement	94.4	87.2	89.0	74.2	22.1	33.8	0.0	73.4	0.6	0.0	47.5
+ refinement*	98.8	87.7	89.2	74.1	22.1	37.7	0.0	83.6	0.6	0.0	49.4
LLaMA2	97.4	73.7	86.9	83.9	25.5	39.9	0.1	99.8	0.3	0.0	50.8
+ regulation	100.0	100.0	-	-	-	-	89.8	-	3.5	-	-
+ tuning	98.6	83.4	95.7	93.1	22.3	51.6	0.1	88.2	5.1	0.0	53.8
+ refinement	98.0	85.2	87.4	84.0	25.5	44.9	0.1	99.9	1.1	0.0	52.6
+ refinement*	99.8	83.7	89.6	84.0	25.5	46.3	0.1	99.9	1.1	0.0	53.0
GPT-3.5	99.0	71.0	89.7	95.3	45.8	56.0	44.5	95.4	36.2	0.0	63.3
+ regulation	-	-	-	-	-	-	-	-	69.0	-	-
+ refinement	99.8	88.1	97.8	96.1	62.5	73.1	46.5	95.4	38.8	0.0	69.8
+ refinement*	100.0	90.1	97.1	96.7	72.3	83.8	50.1	95.5	41.2	4.4	73.1

Table 3: Format faithfulness rate (%) on FORMATBENCH. Format refinement with thoughts are noted with an asterisk (*). The dash symbol (-) signifies that format regulation is not applicable to the task.

In classification tasks, while LLMs succeed in selecting legal options in static classification (QC), they still suffer from a format faithfulness drop in dynamic scenarios (Agent). In sequence labelling tasks, although capable LLMs can easily adapt to both linearized and copied strategies, they may face challenges when dealing with tasks related to text length (CapGen). Finally, in the most flexible generation tasks, even the best LLMs obtain only limited format faithfulness when dealing with unfamiliar formats (MTT, AcroP, XDLGen) or number-related tasks (Parse), due to the lack of format instruction understanding and obedience.

Outlier Discussion There is an intriguing exception found in Agent task, where smaller models demonstrate superior faithfulness compared to larger ones. It is important to note that this does not necessarily imply that smaller models produce better planning and action, as is demonstrated in Section 7.2. In fact, smaller models can choose legal but futile action in the agent process, resulting in a action trace that adheres faithfully to format specifications but lacks practical utility, as the example given in Appendix D.

7 Analysis

In this section, we comprehensively compare format regulation, format tuning, and format refinement with respect to their effectiveness in improving format faithfulness and general quality.

7.1 Format Faithfulness Perspective

	Format Regulation	Format Tuning	Format Refinement
magnitude	●	◑	◑
stability	◑	○	●
robustness	○	◑	●

Table 4: Comparison among format adaptation methods in terms of magnitude, stability, and robustness in improving format faithfulness. Symbols ●, ◑, and ○ refer to high, medium and low performance respectively.

We assess and compare the adaptation methods for improving format faithfulness, considering three key aspects: (1) magnitude, which pertains to the level of improvement; (2) stability, referring to the consistency of the improvement; and (3) generalization, indicating the extent to which the method can be applied to unseen tasks. The summarized findings are presented in Table 4.

Format regulation is proven to be highly effective in applied tasks. However, this approach needs specific design and implementation for each task and is not applicable in all tasks. Moreover, logits are unavailable for API calling models like GPT-3.5, thus restricting the application of generation-time intervention approaches.

Format tuning demonstrates effectiveness in both fine-tuned tasks (such as QA and NER) and other tasks (such as Agent and MTT), validating the

Models	QC	Agent	QA	NER	CapGen	MTT	AcroP	FTime	Parse
	Acc	Score	F1	F1	F1	BLEU	Score	Acc	F1
LLaMA	25.0	2.1	61.8	53.2	10.7	14.5	0.0	40.6	0.2
+ regulation	14.0	4.3	-	-	-	-	53.9	-	3.8
+ tuning	11.2	1.4	68.3	55.7	55.3	8.5	0.1	6.1	0.3
+ refinement	25.4	2.2	61.8	53.3	10.7	14.7	0.0	40.7	0.3
+ refinement*	26.0	2.1	61.8	53.2	10.7	14.9	0.0	46.8	0.3
LLaMA2	35.8	5.8	61.3	62.0	54.9	17.0	0.1	54.1	0.6
+ regulation	34.6	3.4	-	-	-	-	57.1	-	4.7
+ tuning	69.6	2.7	73.9	78.1	58.7	14.6	0.1	47.5	0.3
+ refinement	36.0	6.0	61.4	62.0	54.9	17.2	0.1	54.1	0.8
+ refinement*	36.0	6.0	61.3	62.0	54.9	17.3	0.1	54.1	0.8
GPT-3.5	71.6	11.5	70.0	83.6	53.0	17.7	39.1	77.9	18.7
+ regulation	-	-	-	-	-	-	-	-	30.3
+ refinement	72.2	13.4	72.1	84.3	54.7	18.5	40.8	77.9	19.8
+ refinement*	71.8	13.6	71.6	84.7	53.3	19.1	43.9	77.9	20.8

Table 5: General quality on FORMATBENCH. All values are scaled by 100. Format refinement with thoughts are noted with an asterisk (*). The dash symbol (-) signifies that format regulation is not applicable to the task.

hypothesis that fine-tuning on certain tasks can improve the holistic format faithfulness of LLMs. However, it is noted that the improvement is not consistent due to the task feature variance and the catastrophic forgetting phenomenon.

Format refinement exhibits stable and favorable effects in improving format faithfulness, and introducing reflecting thoughts does have a significant positive effect. It is worth noting that, the effectiveness of refinement is closely related to the capabilities of LLMs, as GPT-3.5 benefits significantly from refinement, while the improvement is relatively modest for LLaMA and LLaMA2. This phenomenon is consistent with the observations in related work (Madaan et al., 2023), that smaller models struggle significantly in the refinement process. Specifically, LLaMA and LLaMA2 often fail to generate meaningful critics feedback (thoughts), and tend to repeat the same output as previous ones.

7.2 General Quality Perspective

In practical scenarios, both format faithfulness and general quality should be taken into consideration. For example, a commercial machine translation system with a terminology module should not only comply with prescribed rules for accurately translating trademarks and terminologies, but also provide precise translations for general language. Consequently, it is crucial to assess the efficacy of adaptation methods in ensuring the general quality.

The evaluation of general quality on FORMATBENCH is presented in Table 5. Comparing the results with the format faithfulness rate in Table 3, it can be observed that format refinement leads to an enhancement in both general quality and format faithfulness. The improvement is especially significant with GPT-3.5 which exhibits better reflection and refinement ability. Meanwhile, the improvement in general quality is not consistent for format tuning, as fine-tuned models tend to suffer from catastrophic forgetting on unfine-tuned tasks.

8 Conclusion

In this paper, we introduce the concept of format faithfulness for the first time, thereby examining the capacity of LLMs in adhering to format specifications. A formal definition and a comprehensive taxonomy of format faithfulness are provided. Based on the taxonomy, we develop FORMATBENCH, a collection of extensive tasks for LLM format faithfulness evaluation. In the construction of FORMATBENCH, apart from adapting existing datasets, we introduce two novel tasks to fully cover the categories in the taxonomy. Experimental results on FORMATBENCH reveal the limitations of current LLMs in adhering to format specifications and generating well-structured text. Furthermore, we design and implement format adaptation approaches, including format regulation, format tuning, and format refinement. The effectiveness of these approaches in improving format faithfulness is validated on FORMATBENCH.

Limitations

In this paper, we design and implement format tuning, as a means to improve the overall format faithfulness of LLMs by fine-tuning on format-related tasks. Experimental results confirm the efficacy of format tuning and demonstrate its ability to generalize to out-of-training tasks, particularly those with similar format requirements. However, it is also noted that format tuning suffers from significant catastrophic forgetting among tasks, which leads to unbalanced or even decreased performance in format faithfulness and general quality. Two limitations in format tuning implementation contribute to the catastrophic forgetting phenomenon. Firstly, the data used in fine-tuning lack variations, as only four specific tasks are utilized and no generation task is incorporated. Secondly, no regulation technique is employed in fine-tuning to mitigate catastrophic forgetting. A more diverse training set encompassing a broader range of categories within our proposed taxonomy combined with advanced regulation techniques could potentially alleviate the catastrophic forgetting problem.

In conclusion, our implemented format tuning may not fully realize its potential in improving format faithfulness due to the inadequacy in fine-tuning data and the absence of regulation techniques. We leave the investigation into more effective format tuning as a direction for future research.

Ethical Considerations

Dataset Collection We respect the terms of use of all the datasets we used with respect to academic purposes. In the event that it is required, our redistribution of the adapted datasets is under the same license as the original ones. Moreover, the detailed dataset collection process is shown in Appendix C.

Human Annotation Two authors of this paper serve as the annotators of our proposed FTime task. The annotators are informed to avoid and reject harmful annotations related to attacks or discrimination. After annotation, we randomly sample 3% data to conduct a cross-validation involving re-annotation by a different annotator. The consistency between the initial annotation and the re-annotation is found to be 98.01%, thereby confirming the trustworthiness of our annotated data.

Further Implications The lack of format faithfulness can pose a significant risk to the safety of LLM, particularly in the context of LLM-based agent grounding in the real world, as format errors may lead to unexpected behaviors. We believe this paper can inspire future research into the safety of LLMs.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Trischler Adam, Côté Marc-Alexandre, and Lima Pedro. 2019. First textworld problems, the competition: Using text-based games to advance capabilities of ai agents.

Rajat Agarwal and Katharina Kann. 2020. Acrostic poem generation. *arXiv preprint arXiv:2010.02239*.

Xuefeng Bai, Jialong Wu, Yulong Chen, Zhongqing Wang, and Yue Zhang. 2023. Constituency parsing using llms. *arXiv preprint arXiv:2310.19462*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Alina Karakanta, Matteo Negri, and Marco Turchi. 2020. MuST-cinema: a speech-to-subtitles corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3727–3734, Marseille, France. European Language Resources Association.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277.

Piji Li, Haisong Zhang, Xiaojiang Liu, and Shuming Shi. 2020. Rigid formats controlled text generation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 742–751.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

Bill Yuchen Lin, Ming Shen, Wangchunshu Zhou, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. Commongen: A constrained text generation challenge for generative commonsense reasoning. In *Automated Knowledge Base Construction*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank.

Grégoire Mialon, Roberto Dessì, Maria Lomeli, Christoforos Nalmpantis, Ram Pasunuru, Roberta Raileanu, Baptiste Rozière, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6834–6842.

OpenAI. 2021. gpt-3.5-turbo-instruct. https://openai.com.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. 2020. Exploring controllable text generation techniques. *arXiv preprint arXiv:2005.01822*.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. Cold decoding: Energy-based constrained text generation with langevin dynamics. *Advances in Neural Information Processing Systems*, 35:9538–9551.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Tim Schopf, Karim Arabi, and Florian Matthes. 2023. Exploring the landscape of natural language processing research. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 1034–1045.

Martin Seifrid, Robert Pollice, Andres Aguilar-Granda, Zamyla Morgan Chan, Kazuhiro Hotta, Cher Tian Ser, Jenya Vestfrid, Tony C Wu, and Alan Aspuru-Guzik. 2022. Autonomous chemical experiments: Challenges and perspectives on establishing a self-driving lab. *Accounts of Chemical Research*, 55(17):2454–2466.

Kirill Semenov, Vilém Zouhar, Tom Kocmi, Dongdong Zhang, Wangchunshu Zhou, and Yuchen Eleanor Jiang. 2023. Findings of the wmt 2023 shared task on machine translation with terminologies. In *Proceedings of the Eighth Conference on Machine Translation*, pages 663–671.

Marta Skreta, Naruki Yoshikawa, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. 2023. Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting. *arXiv preprint arXiv:2303.14100*.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Xiangru Tang, Yiming Zong, Yilun Zhao, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *arXiv preprint arXiv:2309.08963*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, and Shengyi Huang. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Raul Puri, Pascale Fung, Animashree Anandkumar, and Bryan Catanzaro. 2020. Megatron-cntrl: Controllable story generation with external knowledge using large-scale language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2831–2845.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023a. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Wang. 2023b. Syntax error-free and generalizable tool use for llms via finite-state decoding. *arXiv preprint arXiv:2310.07075*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

A Detailed Implementation

In this section, we outline the detailed implementations of (1) standard generation without format adaptation, (2) format regulation, (3) format tuning, and (4) format refinement. Notably, during the entire process of LLM generation, we employ the greedy decoding strategy, where beam search is not utilized and the token with the highest probability is selected.

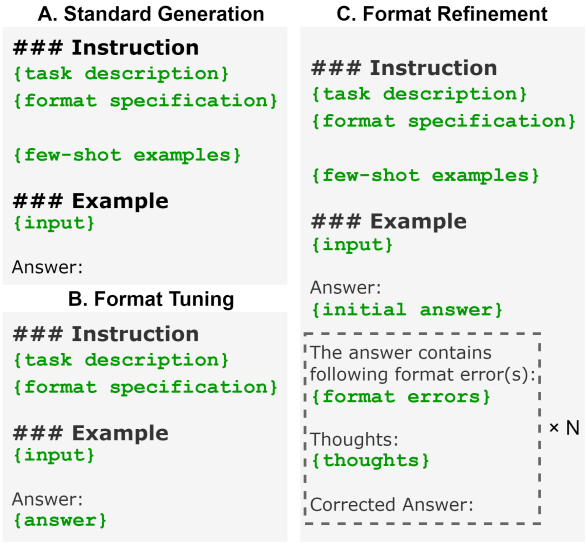

Figure 4: The prompt templates for standard generation (A), foramt tuning (B), and format refinement (C).

A.1 Standard Generation

The vanilla approach involves combining task instructions with few-shot examples to formulate a

prompt. Subsequently, LLMs generate the answer directly based on the provided prompt. The prompt design is illustrated in Figure 4 A.

A.2 Format Regulation

QC & Agent In classification tasks including QC and Agent, we aim at constraining the LLM generation to exclusively legal options. In doing so, we build a Trie consisting only token sequence representations of the legal options and mask all the tokens that do not fall within the Trie during the generation process.

AcroP In the case of AcroP, similar to classification tasks, we enforce the LLMs to produce the designated character in the given message. However, the constraints are only applied when the generated text ends with a line break. Moreover, we actively terminate the generation process once the acrostic message has been fully composed and a new line break is generated.

Parse In the post-generation edition of the Parse task, we determine the number of left and right brackets. Based on this calculation, we add left brackets to the beginning of the output if there is an excess of right brackets, or append right brackets if there is an excess of left brackets. Through this editing process, we guarantee that the bracket sequence achieves a balance.

A.3 Format Tuning

Data Processing We randomly sample 4,000 instances from each of the QC, QA, NER and CapGen tasks, resulting in a fine-tuning set containing 16,000 instances. We adapt each instance into a piece of text by applying the prompt template shown in Figure 4 B.

Tuning Configuration We conduct format tuning with `trl` (von Werra et al., 2020) library. We employ a completion only fine-tuning, where the only the tokens of generated answers contribute to gradients. Additionally, we use a parameter-efficient tuning approach, namely Low-Rank Adaptation (LoRA) (Hu et al., 2021).

Hyper-Parameters The hyper-parameters of format tuning are listed in Table 6.

Computational Usage We use one NVIDIA A800 80GB GPU to conduct format tuning.

learning rate	$2e-5$
optimizer	adamw
adam betas	$(0.9, 0.999)$
epoch	3
batch size	256
lr schedule	cosine
warm ratio	0.1
α	8
dropout	0
rank	16
target modules	attention layers

Table 6: Format tuning hyper-parameters. The bottom four lines are specifically related to LoRA.

A.4 Format Refinement Implementation

As is shown in Figure 3, format refinement includes compiler errors and LLM reflections to improve the format faithfulness. The employed prompt template is shown in Figure 4 C.

The stopping criterion of the refinement process includes (1) the format compiler detects no format errors, (2) the refinement step reaches the predefined limit (five in our implementation), (3) the prompt exceeds the maximum length supported by the model, and (4) the model repeats one of the previous answers.

B Detailed Metrics

B.1 Format Faithfulness

In this subsection, we list the format requirements for each task in FORMATBENCH. An answer is deemed to be faithfulness in format if it satisfies all specified requirements.

QC The answer should be among the legal class options.

Agent The answer should be among the admissible actions given by the game simulation engine.

QA The answer should be a segment of text, or span, from the corresponding reading passage.

NER (1) The sentence after the insertion of NER tags must be the same as the original sentence. (2) Every opening tag (`<PER>`, `<ORG>`, `<LOC>` or `<MISC>`) must be closed with a corresponding closing tag (`</PER>`, `</ORG>`, `</LOC>` or `</MISC>`).

CapGen (1) The sentence after the insertion of separator signs must be the same as the original sentence. (2) Each block must contain no more

than 2 lines. (3) Each line must contain no more than 42 characters.

MTT The terms in the source language should be translated as corresponding terms in the target language according to the terminology translation rules.

AcroP The first letter of each line should spell the given string.

FTime (1) For non-recurring times, the format should follow "YYYYMMDDTHHMMSS", where "T" is a separator. In cases where the "HHMMSS" part is unspecified in the instruction, the use of "?" with equal length is allowed for placeholder processing. (2) For recurring times, the number of recurrences should follow "Rn", where "n" is the number of cycles, "-1" for infinite recurrence. The triggering time adheres to the same format requirements as non-recurring times. For the recurring time section, the format strictly follows "PnYnMnDTnHnMnS". The three parts are spliced together to get the final result following the format "Rn/YYYYMMDDTHHMMSS/PnYnMnDTnHnMnS".

Parse (1) The final outcome should be a string with properly closed parentheses. (2) The sentence after the insertion of brackets and tags must be the same as the original sentence. (3) Labels should be selected from the given label set. (4) Word-level label must be assigned to a word in a leaf node. (5) Phrase-level label must be assigned to a text span in a non-leaf node. (6) Each subtree or word must be specific to a given label, and vice versa.

XDLGen Generated XDL codes should pass the compilation successfully without errors.

B.2 General Quality

This subsection outlines the metrics adopted in general quality evaluation in Table 5.

QC We calculate the accuracy, which is the percentage of correctly predicted instances out of the total instances in the test set, to evaluate classification performance in the QC task.

Agent We take the inner score of the game provided by the game engine, representing the extent to which the agent has reached the final goal. This score is then divided by the maximum score to assess the performance of the agent action trace.

QA We calculate the F1 score to measure the average overlap between the prediction and ground truth as previous works (Rajpurkar et al., 2016).

NER Similar to QA tasks, we use the F1 score by treating the prediction and the ground truth as bags of words (Sang and De Meulder, 2003).

CapGen As previous works (Karakanta et al., 2020), we calculate the F1 score by counting the correct_breaks and the total_breaks in prediction and ground truth respectively.

MTT We calculate the BLEU-4 score between the reference translation and the hypothesis for MTT task evaluation.

AcroP Following previous work (Agarwal and Kann, 2020), we utilize GPT-4-0125-preview to score generated acrostic poems based on four aspects: poetic essence, rhyme, content, and readability. Each aspect is scored from 0 to 5, with 0 being the lowest and 5 being the highest.

- Poetic essence: Does the poem embody the spirit of poetry and feel like a genuine piece of verse?
- Rhyme: Does the poem display a sense of rhyme?
- Content: How closely does the content of the poem relate to the provided acrostic words?
- Readability: How coherent are the vocabulary and grammar used in the poem?

Poems that fail to meet the requirements of the acrostic format often cannot form a poem. Therefore, we directly assign them zero scores in total.

FTime We evaluate the accuracy of the results. We give a full score if the result matches the reference exactly, and grant partial credit for partially correct answers. Specifically, for non-cyclic cases, if the result is misclassified as cyclic but the triggering time is correct, we assign $\frac{1}{3}$ score. For cyclic cases composed of three parts, each correct part earns $\frac{1}{3}$. If the answer is misclassified as non-cyclic but the triggering time matches, $\frac{1}{3}$ score is awarded.

Parse We calculate the F1 score as previous study[1]. Combining symbol, beginning, and ending into a constituent, we can get candidate brackets and gold standard brackets in prediction and ground truth respectively.

[1] https://www.cs.princeton.edu/courses/archive/fall19/cos484/lectures/lec10.pdf

C Dataset Details

AcroP We collect the acrostic poem dataset from Poem Hunter[2], focusing on the acrostic category to gather 927 acrostic poems via web scraping. We additionally combine Kaggle Poems Dataset[3] acrostic poems with these data. To ensure data quality and consistency, we eliminate redundant punctuation and standardize poem lines, retaining only the acrostic portion. Moreover, we filter out poems that do not meet acrostic requirements, such as initial letters failing to form coherent words or lacking relevance, to maintain data accuracy. The processed dataset undergoes manual inspection to verify quality and integrity, resulting in a dataset of 987 valid acrostic poems.

FTime Following ISO 8061 standard[4], we define the time format for FTime and categorize them into non-recurring time format and recurring time format. The non-recurring format is represented as `"YYYYMMDDTHHMMSS"`, and the recurring format is represented as `"Rn/YYYYMMDDTHHMMSS/PnYnMnDTnHnMnS"`.

In the definition of FTime task, we categorize the instructions into three classes. In the first category, the provided instruction contains an event interval (such as "remind me in 20 minutes"), and the final result is the reference time plus this time interval. In the second category, the instruction provided contains a specific time (such as "tomorrow morning at 8 o'clock"). The final result needs to be obtained based on the reference time and the time in the instruction. In the third category, the instruction provided includes a recurring event (such as "at 10 a.n. every Monday"). The final outcome requires determining the number of recurrences, the recurrence interval, and identifying the time of the first event trigger based on the reference time. The format in the first and second categories forms a non-recurring time, which that in the third category constitutes a recurring time. An LLM is to generate a time representation in accordance to the format based on the reference time, the given instruction, and the category to which the instruction pertains.

In the construction of FTime, three components need to be considered, namely reference time, natural language instruction, and the result. All the reference times and their corresponding category tags are automatically generated in advance. We compose the instruction part through two methods, including slot filling and manual composition. For slot filling method, we first design templates with placeholders for a time slot and an event slot. Then, for each of the three classifications mentioned above, we prepare a set of events to be filled in. Subsequently, several templates are randomly selected for each event, and the event slot is filled accordingly. In this way, we generated 4537 instruction instances. To diversify the data, we manually compose another 500 instruction instances. These manually composed data are not limited by the slot filling framework, thus obtaining better flexibility and higher difficulty. Combining the two parts together, we develop 5036 pieces of instruction in total. Given a reference time and a instruction, the corresponding result is annotated manually. After annotation, we randomly sample 3% data to conduct a cross-validation involving re-annotation by a different annotator. The consistency between the initial annotation and the re-annotation is found to be 98.01%, thereby confirming the trustworthiness of our annotated data.

[2] https://www.poemhunter.com/

[3] https://www.kaggle.com/datasets/michaelarman/poemsdataset

[4] https://en.wikipedia.org/wiki/ISO_8601

D Futility Analysis

An example of futility phenomenon in LLaMA2 is shown in Figure 5.

Figure 5: An instance of Agent task, where LLaMA2 repeatedly performs "go east" and "go west". The repetitive futile behavior results in an action trace that adheres faithfully to format, but lacks practical utility.