CFBench: A Comprehensive Constraints-Following Benchmark for LLMs

Tao Zhang[1*♣] **Chenglin Zhu**[2*] **Yanjun Shen**[1*] **Wenjing Luo**[1] **Yan Zhang**[1]
Hao Liang[2,3] **Tao Zhang**[1◇] **Fan Yang**[1] **Mingan Lin**[1] **Yujing Qiao**[1]
Weipeng Chen[1] **Bin Cui**[2] **Wentao Zhang**[2,3†] **Zenan Zhou**[1†]
[1]Baichuan Inc. [2]Peking University [3]Zhongguancun Academy
{zhangtao2, shenyanjun, zhouzenan}@baichuan-inc.com
zhuchenglin@stu.pku.edu.cn, wentao.zhang@pku.edu.cn

Abstract

The adeptness of Large Language Models (LLMs) in comprehending and following natural language instructions is critical for their deployment in sophisticated real-world applications. Existing evaluations mainly focus on fragmented constraints or narrow scenarios, but they overlook the comprehensiveness and authenticity of constraints from the user's perspective. To bridge this gap, we propose CFBench, a large-scale Chinese Comprehensive Constraints Following Benchmark for LLMs, featuring 1,000 curated samples that cover more than 200 real-life scenarios and over 50 NLP tasks. CFBench meticulously compiles constraints from real-world instructions and constructs an innovative systematic framework for constraint types, which includes 10 primary categories and over 25 subcategories, and ensures each constraint is seamlessly integrated within the instructions. To make certain that the evaluation of LLM outputs aligns with user perceptions, we propose an advanced methodology that integrates multi-dimensional assessment criteria with requirement prioritization, covering various perspectives of constraints, instructions, and requirement fulfillment. Evaluating current leading LLMs on CFBench reveals substantial room for improvement in constraints following, and we further investigate influencing factors and enhancement strategies. The data and code will be made available.

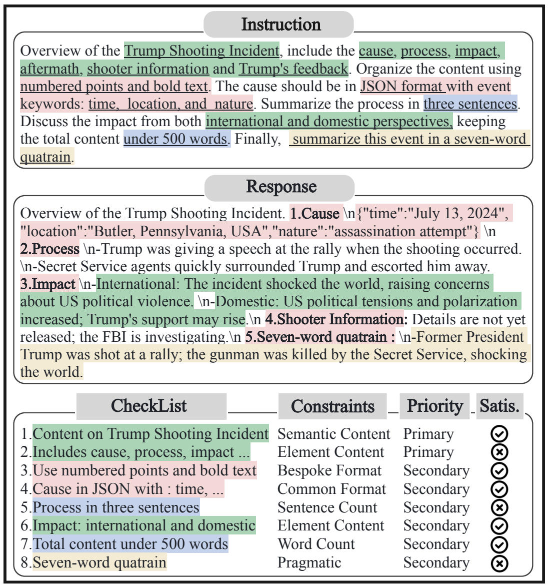


Figure 1: Sample data from CFBench. A checklist, constraint type, requirement priority, and satisfaction constitute our evaluation criteria.

1 Introduction

Large Language Models (LLMs) have become the cornerstone of numerous cutting-edge research tasks and are widely utilized in real-world scenarios (Brown et al., 2020; Chowdhery et al., 2023; Achiam et al., 2023; Touvron et al., 2023). In real-world scenarios, human instructions are inherently complex and accompanied by explicit constraints, requiring models to both understand intricate requirements and strictly comply with these constraints (Yang et al., 2023; Zhong et al., 2021; Mishra et al., 2022; Wei et al., 2021; Sanh et al., 2022). The proficiency of LLMs in comprehending requirements and adhering to natural language constraints is essential, as it ensures tasks are executed precisely and resolved perfectly according to user instructions.

The prevailing method for evaluating a model's instruction-following ability involves using quantitative programs, human evaluators, or advanced LLMs to assess response quality across single constraints, complex problems, and finite constraints (Zhou et al., 2023a; Wang et al., 2023; Li et al., 2023; Zheng et al., 2024; Xu et al., 2023). Laskar et al. (2024) underscores the importance of evaluating data quality, highlighting the necessity for real and extensive data distribution, along with its applicability to real-world scenarios. Sun et al. (2024b)

* Equal contribution.
† Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 32926–32944
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

also stresses that realistic evaluation metrics reflect model capabilities and guide iteration. Constraints-following evaluation faces analogous challenges, particularly within complex real-world scenarios, where data sources and contexts are diverse, and where evaluation is both subjective and arduous. Fig. 1, which addresses the aforementioned challenges, presents a sample from CFBench illustrating the Trump assassination event with different colors representing various constraint types. The instruction include multiple constraints, and the evaluation method uses a checklist to break down complex requirements into independent checkpoints, annotating constraint types and priorities. LLMs are then used to assess each checkpoint. For the English-Chinese comparison example, see Appendix Fig. 11. Consequently, we introduce two more profound challenges in constraints-following assessment.

Q1: How to construct high-quality evaluation data? Many studies focus on evaluating single constraint (Chen et al., 2022; Tang et al., 2023), lacking comprehensive analysis across diverse constraints. He et al. (2024b) examines LLM performance on complex real-world instructions but neglect constraint diversity and scenario coverage. Jiang et al. (2023) incrementally incorporate fine-grained constraints to craft multi-level instructions. However, with only 75 instances of mixed type, which risks variability due to limited data, and equating difficulty with constraint quantity oversimplifies the task. Recent work focuses on evaluating constraints combinability (Wen et al., 2024). To ensure data quality, we systematically categorize constraints by mining real-world online data and using classification, synthesis, and expert design, covering 10 primary categories and over 25 subcategories. We also cross-match these constraints with various domains and scenarios, ensuring balanced representation and expert-validated reasonableness.

Q2: How to evaluate accurately and meticulously? Evaluating LLMs' adherence to constraints is challenging and typically involves manual, automated, and programmatic assessments using various metrics. Representative work computes outcomes for verifiable instructions using code (Zhou et al., 2023a; He et al., 2024b). Jiang et al. (2023) uses scripts and constraint-evolution paths to handle diverse challenging instructions, introducing three metrics tailored to the data's characteristics. The DRFR method decomposes complex constraints into binary judgments, with GPT evaluating each criterion (Qin et al., 2024). Indeed, previous work has ensured the feasibility and objectivity of evaluations through various methods, but they have overlooked assessments from the user's multiple perspectives. We deconstruct complex instructions from the user's perspective into multiple sub-needs, categorizing them by priority and constraint type, with LLMs evaluating each checkpoint. Furthermore, a multi-dimensional evaluation criteria is proposed using three metrics from the perspectives of constraints, instructions, and requirements priority.

We introduce CFBench, a comprehensive Chinese benchmark designed to thoroughly evaluate the constraint comprehension and following capabilities of LLMs. CFBench comprises 10 primary categories and over 25 secondary subcategories organized through taxonomic and statistical methodologies. CFBench features 1,000 meticulously curated samples spanning more than 200 real-life scenarios across 20 domains and over 50 NLP tasks, enhancing the breadth and generality of the evaluation data. Additionally, we have seamlessly integrated original instructions and constraint types within each sample, paying particular attention to nuanced combinations, ensuring each constraint is credibly and coherently embedded. Our advanced evaluation methodology incorporates multi-dimensional assessment criteria, which prioritizing requirements to align LLM outputs with user needs, enhance interpretability, and facilitate iterative development. Finally, extensive experiments and exploratory discussions provide strong support for evaluation and optimization.

Overall, our contributions are mainly four-fold:

- To the best of our knowledge, we are the pioneers in systematically defining an instruction constraint framework utilizing both taxonomic and statistical methodologies.
- We introduce CFBench, a meticulously annotated, large-scale, high-quality Chinese benchmark that encompasses a broad spectrum of real-world scenarios and NLP tasks.
- We propose a multi-dimensional evaluation framework to comprehensively assess model capabilities while prioritizing user-centric needs.
- We exhaustively evaluated prominent LLMs,

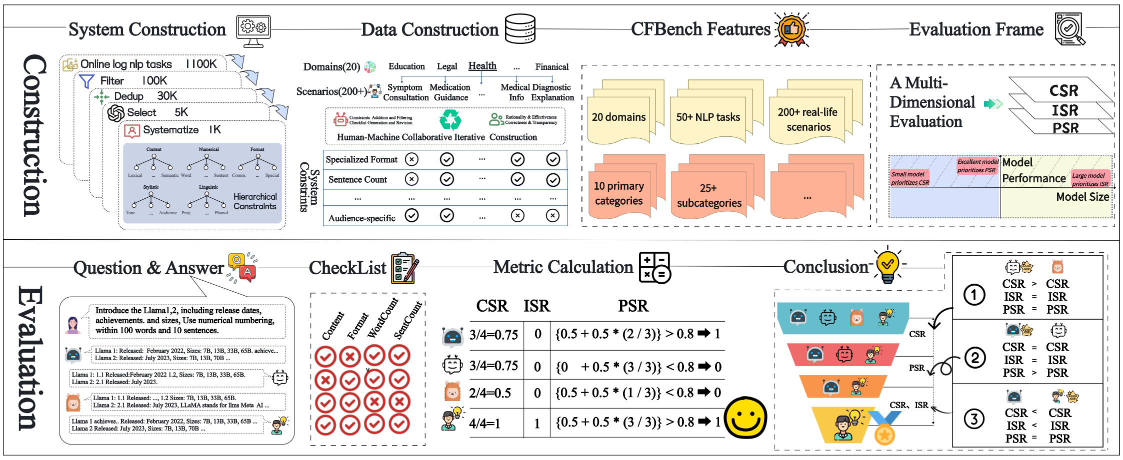


Figure 2: The construction pipeline and evaluation sample of CFBench. Initially, it entails the construction of the constraint system, followed by the assembly of the dataset, and culminating in the proposal of a multi-perspective user view evaluation.

uncovering significant deficiencies in constraints following and exploring performance factors and optimization strategies.

2 Related Work

2.1 Instruction Following

Fine-tuning LLMs with annotated instructional data enhances their ability to follow general language instructions (Weller et al., 2020; Sanh et al., 2022). Studies show that more complex or constrained instructions further improve this ability. For instance, six methods to create intricate instructions from a small set of handwritten seed data are proposed (Xu et al., 2023), while Mukherjee et al. (2023) elevate training data complexity by having GPT-4(Achiam et al., 2023) generate reasoning steps for simple instructions. The latest work (Sun et al., 2024a; He et al., 2024a; Dong et al., 2024) suggests that increasing the number and variety of constraints can enhance the complexity of instructions, thereby further improving the model's ability to follow constraint-based instructions.

2.2 Evaluation of Constraints Following

Constraints such as word count, position, topics, and content have garnered significant attention in the field of Controlled Text Generation (Yao et al., 2023; Zhou et al., 2023b). Zhou et al. (2023a) centers on assessing 25 verifiable instructions. Numerous studies have explored the adherence of LLMs to format constraints, such as complex tabular data (Tang et al., 2023) and customized scenario formats (Xia et al., 2024). Qin et al. (2024) decomposing a single instruction into multiple constraints. He et al. (2024b) gathered constraints from real-world scenarios and developed a sophisticated benchmark using detailed task descriptions and inputs. Jiang et al. (2023) progressively integrates fine-grained constraints to develop multi-level instructions, thereby enhancing complexity across six distinct types. Concurrent work (Wen et al., 2024), constructs a novel benchmark by synthesizing and refining data from the aforementioned benchmarks, with an emphasis on the combinatorial types of constraints. However, previous studies suffered from fragmented constraints, limited scenarios, and misaligned evaluation methods with user perspectives.

3 CFBench

As depicted in Fig. 2, the CFBench construction pipeline includes several key components. First, we collect and systematize constraint expressions from real-world scenarios and various NLP tasks. Using this system, we create high-quality evaluation data by combining instructions from these scenarios with advanced LLMs and manual curation. We then introduce innovative multi-perspective evaluation method. Additionally, we conduct a thorough statistical analysis and validate the quality from various angles to highlight reliability and applicability.

3.1 Constraints System

3.1.1 Constraints Collection

We amass a diverse corpus of instructions from real-world scenarios and various NLP tasks (Xia et al., 2024; Li et al., 2024) to ensure a comprehen-

Split Set	Basic Info					Constraints Count									
	Num.	Len.	Prim.	Cons.	Type.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
Easy Set	500	413	1.69	3.59	2.83	613	214	180	170	134	92	82	95	90	79
Hard Set	500	605	1.98	4.89	3.58	772	345	233	241	168	122	115	145	137	81
Full Set	1000	509	1.84	4.24	3.20	1385	559	413	411	302	214	197	240	227	160

Table 1: CFBench Statistic. The abbreviations of 'Num.', 'Len.', 'Prim.', 'Cons.', 'Type.' denote the sample number, average instruction length, average primary requirements number, average constraint number, average constraint type number, respectively. The designations 'C1'-'C10' denote the Primary Constraint types of content, numerical, style, format, linguistic, situation, example, inverse, contradictory, and rule constraint, respectively.

sive system. Initially, we aggregate several million instructions from online logs and NLP tasks, refining these through length filtering and clustering to distill 30,000 high-quality instructions. Utilizing advanced LLM techniques, we extract and expand atomic constraints through evolutionary methods. Using LLMs, we carefully select meaningful atomic constraints, resulting in over 5000 unique constraints. Domain experts first filter out unreasonable or meaningless atomic constraints and then synthesize these into a structured framework with 10 primary categories and 25 subcategories, guided by principles of statistics, taxonomy, and linguistics.

3.1.2 Constraints System

Content constraints control the scope and depth of output content by specifying certain conditions (Zhang et al., 2023), and can be divided into lexical constraints, element constraints, and semantic constraints based on their granularity. **Numerical constraints**, which ensure that output content meets length and quantity requirements (Yao et al., 2023), can be classified into word-level, sentence-level, paragraph-level, document-level based on the objects involved in the planning. **Stylistic constraints** impart a unique flavor and color to the output, revealing the author's traits and chosen social objectives (Tsai et al., 2021), can be subdivided into tonal, formal, audience, and authorial style constraints based on the perspective of application. **Format constraints** (Tang et al., 2023) standardize expression to guide the generation of complex content and can be categorized into fundamental, bespoke, and specialized scenario constraints based on their usage scenarios. **Linguistic constraints** (Zhou et al., 2023b) adapt to various scenarios by controlling internal features and logic, grouped into Pragmatic, Syntactic, Morphological, Phonological, and other constraints. **Situation constraints** (Liu et al., 2023) guide response appropriateness through background or situational parameters, can be classified into role-based, task-specific, and complex contextual constraints. **Example constraints** regulate new responses by leveraging the intrinsic patterns established by a limited set of samples, with an emphasis on assessing the model's proficiency in contextual constraint learning. **Inverse constraints** narrow the response space through the mechanism of indirect exclusion. **Contradictory constraints** denote conditions that are mutually exclusive, rendering it impossible for the response to fulfill all requirements concurrently, which are prevalent in online logs and are often easily overlooked. **Rule constraints** define logic flows or actions and meticulously crafted to standardize the road of responses. Details are in Appendix Tab. 9.

3.2 Dataset Construction

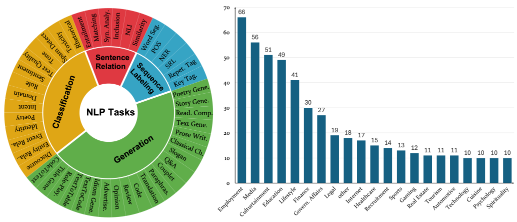
To guarantee data quality in terms of authority and thorough coverage, we utilize a collaborative iterative methodology that synergizes expertise with the capabilities of LLMs.

3.2.1 Data Source and Selection

Real-world scenarios and NLP tasks form the foundation for CFBench's initial instructions. By harnessing advanced LLMs, we rigorously assess each instruction for constraint types and quantities within a predefined system, filtering out those with unreasonable or ineffective constraints. Subsequently, we balance the scenarios and constraint types, resulting in a refined set of 2,000 instructions covering all scenarios and NLP tasks. Prompts and checklist generation are in Appendix.

3.2.2 Iterative Refinement

Professional annotators carefully review and refine the data, ensuring the rationality of constraints and gold answer. If modifications are needed, instructions are revised, and LLMs generate responses



(a) Classification of NLP tasks (b) Scenario-specific domain distributions

Figure 3: The distribution of NLP tasks and domains

with refined evaluation criteria, repeating this process until satisfactory results are achieved. Ultimately, comprehensive support is formulated for each sample, detailing high-quality instructions, the ideal answer, specific assessment criteria, constraint types, and priority levels.

3.3 Dataset Statistics

3.3.1 Overall Statistics

Table 1 provides a statistical overview of CFBench, highlighting substantial differences between the two sets. The Hard Set has more detailed instructions, a greater variety, and a higher number of constraints, indicating higher complexity compared to the Easy Set. The table also shows the diversity and balanced distribution of primary constraint types within CFBench, outperforming other benchmarks. See the Appendix for division details.

3.3.2 Tasks and Domains Distribution

CFBench covers 20 major real-life domains and includes over 200 common scenarios and 50+ NLP tasks. Fig. 3(a) illustrates the classification of NLP tasks, including four major types: classification, generation, sequence labeling, and sentence relation, along with their corresponding specific tasks. Fig. 3(b) shows the real-life scenario-specific domain distribution, where Employment is the most prevalent category, and the other domains are relatively balanced. Our objective is to balance the real distribution with an equitable distribution. Overall, Fig. 3 illustrates that CFBench has evolved into a comprehensive and well-balanced benchmark.

3.3.3 Comparison with Other Benchmarks

As shown in Tab. 3, we thoroughly compare our benchmark with various relevant ones. In terms of size, our benchmark contains approximately twice the number of samples as others. FollowBench (Jiang et al., 2023) increases difficulty by adding the number of constraints, but focuses on the incremental increase of a single constraint type. ComplexBench (Wen et al., 2024) places more emphasis on the combination relationships between different constraint types, but only designs four types. IFEval (Zhou et al., 2023a) focuses on constraints that can be verified, but lacks generalization. Compared to others, CFBench provides comprehensive scenario coverage, diverse systematic constraints, numerous high-quality samples, and multidimensional evaluation. For details, see Appendix Tab. 11, including Case and Features.

3.4 Evaluation Protocol

3.4.1 Evaluation Criteria

We breaking down instructions into multiple simple, independent checkpoints to ensure evaluation accuracy, inspiration was drawn from DRFR (Qin et al., 2024). Unlike DRFR, our method emphasizes defining ideal response characteristics and critical evaluation points. The previous sections detailed the checklist generation process, a key part of our evaluation criteria. Furthermore, we employ GPT-4o, as the evaluation model. By repeatedly feeding it the instruction, test model response, and checklist with a carefully tuned prompt, we ensure that the judged response fully meets the judgement format check. This iterative process aims to maximize confidence in our evaluation. The specific evaluation prompt is in the Appendix.

3.4.2 Evaluation Metrics

Aligned with different perspectives, we define the Constraint Satisfaction Rate (CSR), Instruction Satisfaction Rate (ISR) as follows:

$$\mathrm{CSR} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{n_i}\sum_{j=1}^{n_i} s_i^j\right) \quad (1)$$

$$\mathrm{ISR} = \frac{1}{m}\sum_{i=1}^{m} s_i \quad (2)$$

where $s_i^j = 1$ if the j-th constraint of i-th instruction is satisfied and $s_i^j = 0$ otherwise. $s_i = 1$ indicates that all constraints in the i-th instruction are satisfied and $s_i = 0$ otherwise. The requirements Priority Satisfaction Rate (PSR) is defined as follows:

$$\mathrm{PSR} = \frac{1}{m}\sum_{i=1}^{m}(PSR_i) \quad (3)$$

Let the average score for secondary requirements be A. When all primary requirements are met, $score = 0.5 + 0.5 \times A$. If $score > 0.8$, then

Models	Easy Set			Hard Set			Full Set		
	CSR	ISR	PSR	CSR	ISR	PSR	CSR	ISR	PSR
o1-preview†	0.926	0.806	0.844	0.814	0.462	0.592	0.870	0.634	0.718
DeepSeek-V3†	0.948	0.836	0.864	0.831	0.460	0.616	0.890	0.648	0.740
DeepSeek-R1†	**0.960**	**0.874**	**0.894**	**0.856**	**0.524**	**0.672**	**0.908**	**0.699**	**0.783**
GPT-4o†	0.956	0.868	0.888	0.816	0.438	0.582	0.886	0.653	0.735
GPT-4-Turbo-20240409†	0.924	0.792	0.826	0.783	0.370	0.518	0.853	0.581	0.672
GPT-4-0125-Preview†	0.923	0.790	0.826	0.763	0.310	0.468	0.843	0.550	0.647
Claude-3.5-Sonnet†	0.943	0.844	0.882	0.799	0.408	0.564	0.871	0.626	0.723
GLM-4-0520†	0.939	0.820	0.852	0.785	0.372	0.536	0.862	0.596	0.694
Yi-Large†	0.900	0.730	0.786	0.744	0.292	0.460	0.822	0.511	0.623
MoonShot-V1-8k†	0.919	0.764	0.812	0.758	0.308	0.464	0.838	0.536	0.638
Llama-3-8B-Instruct*	0.656	0.300	0.356	0.562	0.122	0.238	0.609	0.211	0.297
Llama-3-70B-Instruct*	0.750	0.422	0.498	0.642	0.178	0.330	0.696	0.300	0.414
Vicuna-33B-V13*	0.621	0.270	0.352	0.527	0.110	0.196	0.574	0.190	0.274
DeepSeek-V2-Lite-Chat	0.733	0.382	0.448	0.597	0.148	0.262	0.665	0.265	0.355
Qwen1.5-110B-Chat	0.905	0.724	0.792	0.730	0.276	0.438	0.818	0.500	0.615
Qwen2-72B-Instruct	0.944	0.836	0.880	0.791	0.342	0.530	0.867	0.589	0.705
BaiChuan2-13B-Chat	0.669	0.348	0.418	0.547	0.134	0.226	0.608	0.241	0.322

Table 2: The evaluation results of LLMs on CFBench and its splits. Notably, * stands for the model supporting mainstream languages excluding Chinese, and † represents calling through the API. The **bold**, underlined, and tilde denote the first, second, and third rankings, respectively.

Benchmarks	Data Quality			Evaluation	
	Num.	Type.	Syst.	Prio.	Meth.
IFEval	541	4*	✗	✗	
CELLO	523	4	✗	✗	
FollowBench	820	5	✗	✗	
InFoBench	500	5	✗	✗	
FoFoBench	494	1	✗	✗	
ComplexBench	1150	4	✓	✗	
CFBench	1000	10-25	✓	✓	

Table 3: Detailed Comparison of Relevant Benchmarks. * represents our constraint system. 'Num.', 'Type.', 'Syst.', 'Prio.', and 'Meth.' denote the number of samples, primary constraint types, presence of a constraint system, requirement prioritization, and evaluation method.

$PSR_i = 1$; otherwise, $PSR_i = 0$, especially when any primary requirement is not met. The threshold of 0.8 is based on user feedback, reflecting tolerance for LLMs adhering to constraints. Overall, CSR, ISR, and PSR reflect different levels of user perception from multiple perspectives, including constraints, instructions, and requirement priorities.

3.5 Data Quality

3.5.1 Quality Evolution

To enhance the quality of CFBench, we invested considerable effort and financial resources. First, in the instruction generation phase, we utilized multiple advanced LLMs, such as GPT-4 and Claude, to generate diverse instructions and responses for annotator candidates. Second, we implemented a stringent manual annotation process, including annotator training, cross-validation, batch validation, expert team involvement, and iterative refinement of instruction constraints and response quality. We also ensured the objectivity, evaluability, and prioritization of checkpoints. Additionally, we balanced the data for constraint types, scenarios, and NLP task distribution. Detailed information can be found in the Appendix.

3.5.2 Quality Evaluation

To investigate CFBench's quality, we randomly selected 100 samples for assessment. Three professional data inspectors evaluated them, resulting in high-quality rates of 94% for instructions, 94% for gold answers, and 93% for checklists (see Appendix Tab. 6). Additionally, three experts rated Qwen2-7B-Instruct outputs on a 0-1 scale. The kappa coefficient between GPT-4o PSR and expert evaluations was 0.77, highlighting the effectiveness of the PSR evaluation method and metrics, even for smaller models. Details are in Appendix Tab. 7.

4 Experiment

4.1 Evaluation Settings

We evaluated 50+ top-performing models from previous benchmarks (Hendrycks et al., 2020;

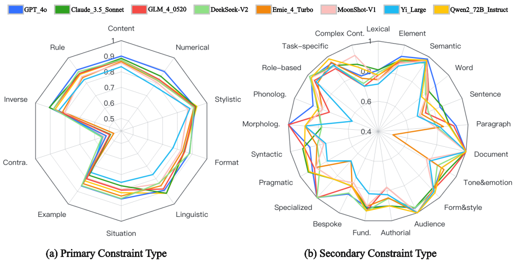


Figure 4: Different mainstream models' results under primary and secondary constraint categories.

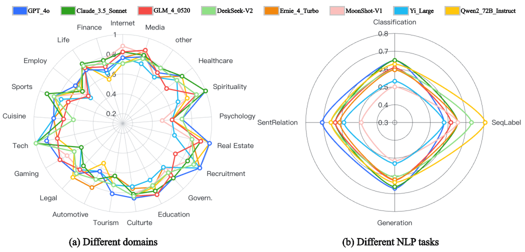


Figure 5: Different mainstream models' PSR results in real-world domains and NLP task types.

Cobbe et al., 2021), considering factors like model size, Chinese language support, access via API or weights, and fine-tuning with instruction data. During inference, we set the maximum generation length to 2048 and used default values for other parameters. For evaluation, we used GPT-4o as the judge model with a temperature of 0 for deterministic outputs.

4.2 Overall Result

Tab. 2 presents CFBench evaluation results for leading models. DeepSeek-R1 (DeepSeek-AI et al., 2024) leads overall, followed by GPT-4o and DeepSeek-V3 (DeepSeek-AI et al., 2025) in third. Claude-3.5-Sonnet (Anthropic, 2024) and Qwen2-72B-Instruct (Yang et al., 2024) performed well, though DeepSeek-V3 and Qwen2-72B-Instruct showed slight drops in the Full Set. The highest PSR in the Hard Set was 0.582, indicating room for improvement.

While CSR favors weaker models, ISR and PSR highlight differences in stronger models. API-accessed models like GPT-4o outperformed most open-source models, though DeepSeek-V3 and Qwen2-72B-Instruct performed well among open-source models.

Model	MMLU	GSM8K	CFBench
GPT-4o	88.7	90.5	0.698
Claude-3.5-Sonnet	88.7	96.4	0.691
Qwen2-72B-Instruct	82.3	91.1	0.672
DeepSeek-V2	78.5	79.2	0.665
Qwen1.5-110B-Chat	80.4	-	0.584
Qwen1.5-72B-Chat	77.5	79.5	0.577

Table 4: Performance Comparison on Benchmarks

4.3 Constraints-categorized Performance

To assess performance across different constraint types, we calculated satisfaction scores for the top 8 LLMs (see Fig. 4). Many models struggled with contradictory constraints, highlighting their limitations. GPT-4o excelled across various constraints, while other models alternated in leading different types. For secondary constraints, all models performed poorly in lexical, word, and sentence count constraints but did better in document count and audience style constraints. No single model consistently led across most constraint types. In summary, even the most advanced LLMs have significant room for improvement, with each model showing specific weaknesses, providing valuable insights for future iterations.

4.4 Domain and Task-categorized Performance

As depicted in Fig. 5, we evaluate performance across 21 domains and 4 major NLP task types, each with 500 examples from the two main sources of CFBench. For domain performance, employment and psychology require significant attention, while technology and recruitment are strengths for most models. For NLP tasks, GPT-4o excels in sentence relationship tasks, while Qwen2-72B-Instruct is strong in sequence labeling, likely due to its optimization for Chinese. In general, models exhibit different rankings across domains and tasks, indicating no clear absolute leader. Comprehensive improvements are needed for better constraint follow across multiple domains and tasks.

5 Discussions

5.1 Comparisons between Capabilities

Tab. 4 presents a comprehensive comparison of CFBench's PSR with two prominent LLM evaluation benchmarks. MMLU (Hendrycks et al., 2020) focuses on knowledge proficiency, while GSM8K (Cobbe et al., 2021) emphasizes mathematical ability. GPT-4o ranks first on CFBench

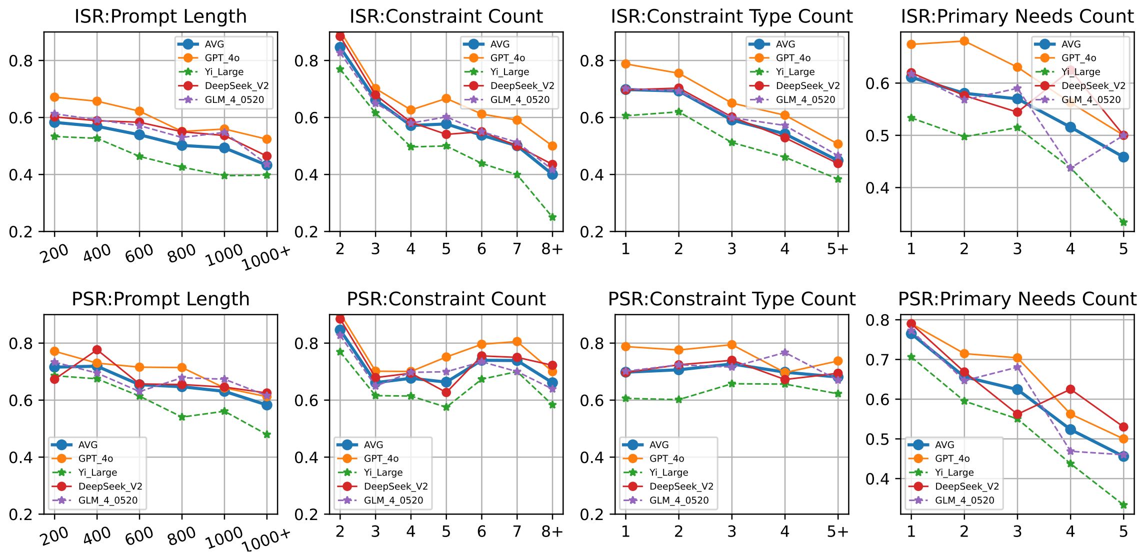


Figure 6: Factors Influencing Constraints-Following Performance

but significantly lags behind, ranking third on GSM8K. Qwen2-72B-Instruct performs worse than DeepSeek-V2 on CFBench but outperforms it on MMLU. Notably, the rankings of LLMs on CF-Bench do not entirely correspond with those on the other two benchmarks, indicating that CFBench provides a novel perspective for LLMs evaluation.

5.2 Factors influencing constraints-following

Previously, we identified a significant gap in LLM constraints following performance, prompting us to further explore the influencing factors. We analyzed the impact of prompt length, number of constraints, constraint types, and primary requirements on evaluation results across five top-performing models and their average values. As shown in Fig. 6, all four factors are positively correlated with the ISR metric, with the number of constraints having the most significant effect. For PSR, the number of constraints and constraint types do not show a completely positive correlation, while the number of primary requirements has a greater influence. Users are more affected by unmet constraints when there are fewer, but become more tolerant of unmet non-primary constraints when there are many.

5.3 How to improve constraint-following ability?

In Appendix Tab. 8, we investigated methods to potentially enhance constraint following. Firstly, Supervised Fine-Tuning (SFT) significantly improves performance, with nearly all models that undergo instruction fine-tuning exhibiting substantial improvements in effectiveness, as demonstrated by the Qwen series. Secondly, model size is also an important factor, as evidenced by Qwen2-72B-Instruct showing a 40% relative PSR improvement over Qwen2-7B-Instruct. Additionally, replicating Conifer's models (Sun et al., 2024a) reveals that fine-tuning with complex constraint instructions further enhances performance, and recent work has also been directed towards this approach (He et al., 2024a). Further exploration is intended to be pursued in future work.

6 Conclusion

This study examines the constraints-following capabilities of LLMs. CFBench, was introduced with 1000 manually annotated samples covering more than 200 real-world scenarios and over 50 NLP tasks, encompassing a wide range of systematically defined constraint types. Each sample in CFBench includes detailed evaluation criteria, providing metrics that accurately reflect model performance and real user needs across multiple dimensions. Extensive experiments on CFBench revealed significant limitations and challenges that advanced LLMs face in following constraint instructions. Key factors and potential strategies to improve constraint following were also analyzed, and numerous insightful findings can provide valuable guidance for the optimization of LLMs' performance. In conclusion, CFBench offers a novel perspective for evaluating LLM capabilities, providing new directions for performance assessment and improvement.

7 Limitations

7.1 Experimental Setup

This study primarily focuses on models with strong Chinese language capabilities, lacking a comprehensive survey of a broader range of English models. Additionally, while we conducted preliminary analyses on the differences in instruction-following abilities between Chinese and English, a more in-depth comparative study is absent.

7.2 Limited Exploration of Reasoning Models

Currently, deep reasoning models like R1 continue to achieve commendable results. However, there is a lack of in-depth research into these models, particularly concerning the factors that enhance their instruction-following abilities.

7.3 Evaluation Model Bias

The evaluation of models predominantly relies on GPT-4o as the judge model. Future research could explore the impact of different evaluation models on assessing the performance of other models.

8 Ethics Statement

This research adheres to the ethical guidelines set forth by the Association for Computational Linguistics (ACL). We have ensured that all data collection and experimental designs comply with privacy protection and informed consent principles, fully respecting and safeguarding the rights of all participants. Furthermore, we have evaluated the potential societal impacts of our research findings, ensuring that their application does not result in adverse effects on society.

9 Acknowledgements

This work is supported by the National Key R&D Program of China (2024YFA1014003), National Natural Science Foundation of China (92470121, 62402016), and High-performance Computing Platform of Peking University.

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*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

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.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. 2022. Controllable text generation with language constraints. *arXiv preprint arXiv:2212.10466*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun,

T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. 2024. Deepseek-v3 technical report. *Preprint*, arXiv:2412.19437.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024a. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *arXiv preprint arXiv:2404.15846*.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024b. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Md Tahmid Rahman Laskar, Sawsan Alqahtani, M Saiful Bari, Mizanur Rahman, Mohammad Abdullah Matin Khan, Haidar Khan, Israt Jahan, Amran Bhuiyan, Chee Wei Tan, Md Rizwan Parvez, et al. 2024. A systematic survey and critical review on evaluating large language models: Challenges, limitations, and recommendations. *arXiv preprint arXiv:2407.04069*.

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

Yizhi Li, Ge Zhang, Xingwei Qu, Jiali Li, Zhaoqun Li, Zekun Wang, Hao Li, Ruibin Yuan, Yinghao Ma, Kai

Zhang, et al. 2024. Cif-bench: A chinese instruction-following benchmark for evaluating the generalizability of large language models. *arXiv preprint arXiv:2402.13109*.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*.

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)*, pages 3470–3487.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

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

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024a. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Kun Sun, Rong Wang, Haitao Liu, and Anders Søgaard. 2024b. Comprehensive reassessment of large-scale evaluation outcomes in llms: A multifaceted statistical approach. *arXiv preprint arXiv:2403.15250*.

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

Alicia Tsai, Shereen Oraby, Vittorio Perera, Jiun-Yu Kao, Yuheng Du, Anjali Narayan-Chen, Tagyoung Chung, and Dilek Hakkani-Tur. 2021. Style control for schema-guided natural language generation. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 228–242.

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)*, pages 13484–13508.

Jason Wei, Maarten Bosma, Vincent Y 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. *arXiv preprint arXiv:2109.01652*.

Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E Peters. 2020. Learning from task descriptions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1361–1375.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, et al. 2024. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978*.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*.

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. *arXiv preprint arXiv:2304.12244*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. 2023. Foundation models for decision making: Problems, methods, and opportunities. *arXiv preprint arXiv:2303.04129*.

Shunyu Yao, Howard Chen, Austin W Hanjie, Runzhe Yang, and Karthik Narasimhan. 2023. Collie: Systematic construction of constrained text generation tasks. *arXiv preprint arXiv:2307.08689*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2856–2878.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023b. Controlled text generation with natural language instructions. In *International Conference on Machine Learning*, pages 42602–42613. PMLR.

A Appendix

A.1 Constraint System Construction

The construction of the constraint system commenced with the aggregation of data from diverse real-world scenarios and NLP tasks. This encompassed 800,000 query logs from LLM websites over the preceding six months, alongside over 300,000 data points from various NLP tasks. Instructions that were excessively long or short were filtered out, and a vector clustering deduplication algorithm was employed. This meticulous process culminated in a refined dataset comprising approximately 30,000 instructions. Subsequently, GPT was utilized to extract constraint atoms from these instructions, thereby ensuring the comprehensiveness of the constraint system. The prompt employed for GPT-4 extraction, as illustrated in Fig. 7, resulted in the identification of approximately 5,000 unique atomic constraint instructions. Three seasoned experts meticulously refined these into 1,000 significant atomic constraints. By integrating statistical analysis, classification, and linguistic principles, a hierarchical constraint system was developed using Top-Down Organization and Bottom-Up Synthesis methodologies. This system comprises 10 primary categories and 25 secondary categories. The system comprehensively categorizes all types of constraints, ensuring that nearly all specific constraints can be systematically classified within its framework. Detailed information regarding the constraint system is presented in Tab. 9.

A.2 Dataset Construction

We adopted an innovative Human-Machine Collaborative Iterative Construction approach to ensure the highest quality of data. This method involved leveraging advanced LLMs to augment original instructions with additional constraints and generate corresponding responses. These responses were meticulously reviewed for constraint validity, followed by the creation of detailed checklists for each example. Multiple experts participated in this iterative process, continuously refining the outputs by addressing issues encountered by the LLMs and regenerating or manually correcting any substandard samples. The prompts used for GPT to enhance constraints and generate checklists are illustrated in Fig. 8 and Fig. 10. Due to the limited attention given to real-life scenarios, we have meticulously organized and covered 20 domains and over 200 scenarios in our CFBench system, as detailed in Tab. 10. In the end, we gathered 1,000 high-quality data points: 500 from real-world scenarios and 500 from different NLP tasks. Specifically, we implemented the following steps to enhance data quality for manual annotations. In Tab. 5, we list the proportions of the four major NLP task categories.

Task Type	Prop.	Samples' num
Generation	46%	230
Classification	30%	150
Sentence Relation	12%	60
Sequence Labeling	12%	60

Table 5: Distribution of four major NLP task categories.

A.2.1 Annotator Training

We sourced annotation contractors from the public and selected 21 candidates for training by seasoned data scientists. After a one-week training program, the annotators engaged in multiple rounds of trial annotations, which were then assessed by data experts. From these assessments, the 9 annotators demonstrating the highest accuracy were selected for this dataset.

A.2.2 Cross-Validation

To reduce the likelihood of missed and incorrect annotations, we implemented an inter-annotator validation process. Three annotators independently reviewed the labeled instructions, responses, and evaluation criteria, achieving a notable agreement rate of 94%. Any discrepancies that emerged were resolved through expert adjudication, ensuring both consistency and accuracy.

Prompt Template for Atomic Constraint Extraction

You are a professional atomic constraint extractor. Your task is to extract as many atomic constraint expressions as possible from the given {**Instruction**}.
Definition of atomic constraint expression: The smallest unit of description or constraint for the required task within the instruction.
Common types of constraints include: content constraints, numerical constraints, style constraints, format constraints.

[**Example**]
[**Instruction**]: {instruction}
[**Atomic constraint**]: {atomic constraint 1, atomic constraint 2, ...}

[**Input**]
[**Instruction**]: {instruction}
[**Your Answer**]:

Figure 7: Prompt Template Atomic Constraint Extract

Prompt Template for Adding Atomic Constraints

[Task Description]
1. I currently have a seed question, but the seed questions are relatively simple. To make the instructions for the seed question more complex, I want you to add more constraints to this question.
2. I will provide [Seed Question] and [Constraint References], and you can use these constraint references to increase the difficulty of the seed question.
3. [**Constraint References**] are just suggestions for constraints. When adding constraints, you can add one or more, freely combine from the constraint references, or add other constraints you deem appropriate.
4. Do not delete any information from the seed question. Your task is to rewrite the seed question and add constraints without omitting any key information from the seed question, such as reference texts.
5. Directly return the modified question (the question with added constraints), without any analysis.

[**Constraint References**]
1. Lexical content constraint : {Definition} {Example}
2. Word Count : {Definition} {Example}
...
25.Rule Constraint : {Definition} {Example}

[**Seed Question**] : {raw_question}

[**Modified Question**] :

Figure 8: Prompt Template for Adding Atomic constraint

Prompt Template for GPT-4o judge Evaluation

I want you to act as a quality evaluator.
Evaluate the [**Model Answer**] based on the [**User Instruction**], [**Reference Answer**], and [**CheckList**], scoring it as either 0 or 1. Both the [**Reference Answer**] and the [**Model Answer**] respond to the [**User Instruction**]. The [**CheckList**] defines the criteria for evaluation. Score each point in the [**CheckList**] as 1 if the [**Model Answer**] meets it, otherwise score it as 0.
Note: If the [**Reference Answer**] is empty, ignore it when evaluating the [**CheckList**] points.

[**Output Requirement**]:
1. Follow the order of the [CheckList] points, output one per line, separated by '\n\n'.
2. For each line, first output the [CheckList] content, then a '\t', and finally the [Evaluation Score] (0 or 1).
3. Please directly output your evaluation without any additional content.

[**Examples**]
[**Example1**] : {Example1}
[**Example2**] : {Example2}

[**User Instruction**] : {user instruction}
[**Reference Answer**] : {reference answer}
[**Model Answer**] : {model answer}
[**CheckList**] : {checklist}
[**Your Evaluation**]:

Figure 9: Prompt Template for Evaluation

A.2.3 Batch Validation

Due to the substantial size of the dataset, it was systematically divided for processing. Following a phased improvement approach, the initial batch sizes were set at 50, 100, and 200, gradually increasing to 400 for later batches. After the annotation process, 50% of the dataset was randomly selected for contractor review, while 20% of the dataset was examined by experts.

A.2.4 Data Split

We used a voting mechanism involving experts and ten models, including GPT-4o and Claude-3.5-Sonnet, to partition 10,000 CFBench entries into 'easy' and 'hard' categories. The 'hard' category includes entries where multiple models struggle with PSR performance and are also challenging for humans, as verified by experts.

A.3 Evaluation Method and Metric

The state-of-the-art GPT-4o model was employed as the judge to perform binary scoring (0 or 1) for each checkpoint in the checklist. The specific evaluation prompt is illustrated in Fig. 9. The Requirement Priority-Satisfaction Ratio (PSR) was proposed as an evaluation metric that simultaneously considers the prioritization of user requirements and satisfaction levels. PSR is calculated by first ensuring that all primary requirements are met. Subsequently, the satisfaction score is determined by averaging the fulfillment of the remaining constraints to obtain A. The final satisfaction score is then calculated using the formula $0.5 + 0.5 * A$. If the final score exceeds 0.8, PSR is set to 1. The threshold of 0.8 was established based on the average satisfaction levels derived from multiple users' feedback on the responses to the instructions.

A.4 Quality Assessment

We employed multiple methods to validate the quality of the benchmark on a randomly selected set of 100 samples. First, we engaged three experts to independently evaluate the quality of each sample's instruction, response, and criteria. The average quality rate determined by the three experts was consistently above 90%, as detailed in Tab. 6. To further validate the effectiveness of our proposed evaluation metric, PSR, we had the same three experts score the responses of Qwen2-7B-Instruct on these 100 cases using a 0-1 scale. Simultaneously, we utilized GPT-4o to directly score the responses, referred to as GPT-4o PSR. By calculating the kappa coefficient, we found a strong agreement between our proposed PSR evaluation metric and the human experts' assessments. The detailed results are presented in Tab. 7. Kappa coefficient scores are interpreted as follows: below 0.2 indicates slight agreement, 0.21 to 0.40 indicates fair agreement, 0.41 to 0.60 indicates moderate agreement, 0.61 to 0.80 indicates substantial agreement, and 0.81 to 1.00 indicates almost perfect agreement.

Set	Instruction	Gold Ans	CheckList
Easy Set	0.96	0.94	0.93
Hard Set	0.92	0.95	0.93
All Set	0.94	0.94	0.93

Table 6: The High-Quality Rate of 100 selected Samples

Set	Easy Set	Hard Set	Full Set
Avg.Expert	1	1	1
GPT-4o DS	0.58	0.61	0.60
GPT-4o PSR	0.76	0.77	0.77
Qwen2-72B-Inst. PSR	0.70	0.73	0.72

Table 7: The kappa coefficient between expert evaluations and various assessment methods

A.5 Experimental Setup and Results

A.5.1 Experiment Setting

We evaluated the most popular Large Language Models (LLMs), with the majority of these models being developed by companies based in China, primarily to accommodate our CFBench's focus on the Chinese language. Among the 50 evaluated models, they can be categorized into two groups based on their access method: API-based and open-source weight-based models. It is worth noting that the Llama series models do not primarily support the Chinese language, which results in noticeably lower performance. Both conifer-base and conifer-test are based on the Mistral-7B foundational model. Llama-3-8B-Instruct-CN and Llama-3-70B-Instruct-CN respectively represent Llama-3-8B-Instruct-Chinese and Llama-3-70B-Instruct-Chinese, both of which have undergone Chinese SFT (Supervised Fine-Tuning). For the base models, we used a 3-shot approach to ensure a fair evaluation. The complete list of evaluated models can be found in Tab. 8.

A.5.2 Explanation of Results

GPT-4o and Claude3.5-Sonnet have demonstrated near-absolute leadership, achieving outstanding performance across various metrics and categories. Similarly, models such as GLM-4-0510, ERNIE-4-Bot-0613, ERNIE-4-Turbo-0628, DeepSeek-V2-0628, and Qwen2-72B-Instruct have also exhibited strong capabilities. Many models that support less mainstream Chinese languages performed significantly worse, which is unfair to them and only serves to illustrate their relative rankings. This also confirms that performance are highly correlated with language, especially within the scope of language constraints. From the perspective of open-source versus closed-source models, open-source models have generally achieved comprehensive success. However, Qwen2-72B-Instruct, as an open-source model, also demonstrated notable constraint-following capabilities. Regarding model size, within the Qwen series, performance metrics clearly improve with increasing model size. Additionally, models that have undergone Supervised Fine-Tuning (SFT) show significantly enhanced instruction-following capabilities. The complete evaluation results and rankings can be found in Tab. 8.

Models	Easy Set			Hard Set			Full Set		
	CSR	**ISR**	**PSR**	**CSR**	**ISR**	**PSR**	**CSR**	**ISR**	**PSR**
GPT-4o†	**0.956**	**0.868**	**0.888**	**0.816**	**0.438**	**0.582**	**0.886**	**0.653**	**0.735**
GPT-4-Turbo-20240409†	0.924	0.792	0.826	0.783	0.370	0.518	0.853	0.581	0.672
GPT-4-0125-Preview†	0.923	0.790	0.826	0.763	0.310	0.468	0.843	0.550	0.647
GPT-3.5-Turbo-1106†	0.797	0.520	0.602	0.631	0.176	0.326	0.714	0.348	0.464
Claude-3.5-Sonnet†	0.943	0.844	0.882	0.799	0.408	0.564	0.871	0.626	0.723
GLM-4-0520†	0.939	0.820	0.852	0.785	0.372	0.536	0.862	0.596	0.694
DeepSeek-V2-0628†	0.946	0.830	0.868	0.786	0.350	0.524	0.866	0.590	0.696
ERNIE-4-Turbo-0628†	0.930	0.790	0.848	0.772	0.332	0.532	0.851	0.561	0.690
ERNIE-4-Bot-0613†	0.929	0.792	0.832	0.779	0.338	0.518	0.854	0.565	0.675
ERNIE-3.5-0613†	0.901	0.720	0.772	0.758	0.302	0.482	0.830	0.511	0.627
Yi-Large†	0.900	0.730	0.786	0.744	0.292	0.460	0.822	0.511	0.623
abab6.5-chat†	0.894	0.696	0.766	0.736	0.260	0.452	0.815	0.478	0.609
MoonShot-V1-8k†	0.919	0.764	0.812	0.758	0.308	0.464	0.838	0.536	0.638
Vicuna-7B-V13*	0.563	0.206	0.262	0.468	0.100	0.168	0.516	0.153	0.215
Vicuna-33B-V13*	0.621	0.270	0.352	0.527	0.110	0.196	0.574	0.190	0.274
Vicuna-13B-V13*	0.605	0.248	0.302	0.503	0.100	0.178	0.554	0.174	0.240
Llama-2-7B-Chat*	0.5268	0.198	0.250	0.448	0.096	0.152	0.487	0.147	0.201
Llama-2-13B-Chat*	0.574	0.242	0.280	0.488	0.094	0.178	0.531	0.168	0.229
Llama-3-8B-Instruct*	0.656	0.300	0.356	0.562	0.122	0.238	0.609	0.211	0.297
Llama-3-70B-Instruct*	0.750	0.422	0.498	0.642	0.178	0.330	0.696	0.300	0.414
Mistral-7B-Instruct-V03*	0.227	0.072	0.086	0.148	0.008	0.022	0.188	0.040	0.054
Conifer-Base*	0.510	0.184	0.232	0.300	0.018	0.048	0.405	0.101	0.140
Conifer-Test*	0.559	0.215	0.255	0.328	0.102	0.156	0.443	0.159	0.206
BaiChuan-13B-Chat	0.630	0.306	0.366	0.521	0.114	0.196	0.575	0.210	0.281
BaiChuan2-13B-Chat	0.669	0.348	0.418	0.547	0.134	0.226	0.608	0.241	0.322
Llama-3-8B-Instruct-CN	0.743	0.458	0.510	0.627	0.162	0.314	0.685	0.310	0.412
Llama-3-70B-Instruct-CN	0.756	0.482	0.536	0.636	0.190	0.322	0.696	0.336	0.429
DeepSeek-7B-Chat	0.695	0.378	0.442	0.580	0.150	0.270	0.638	0.264	0.356
DeepSeek-V2-Lite-Chat	0.733	0.382	0.448	0.597	0.148	0.262	0.665	0.265	0.355
DeepSeek-67B-Chat	0.802	0.516	0.578	0.662	0.180	0.350	0.732	0.348	0.464
InternLM2-Chat-7B	0.767	0.452	0.538	0.625	0.172	0.320	0.696	0.312	0.429
GLM-4-9B-Chat	0.885	0.678	0.742	0.742	0.288	0.450	0.813	0.483	0.596
YI-1.5-34B-Chat	0.881	0.672	0.740	0.745	0.302	0.474	0.813	0.487	0.607
Qwen1.5-4B	0.454	0.170	0.198	0.376	0.074	0.116	0.415	0.122	0.157
Qwen1.5-4B-Chat	0.652	0.310	0.362	0.536	0.104	0.198	0.594	0.207	0.280
Qwen1.5-7B	0.473	0.176	0.212	0.400	0.090	0.142	0.437	0.133	0.177
Qwen1.5-7B-Chat	0.799	0.534	0.592	0.654	0.194	0.338	0.726	0.364	0.465
Qwen1.5-14B	0.498	0.228	0.280	0.430	0.110	0.176	0.464	0.169	0.228
Qwen1.5-14B-Chat	0.822	0.558	0.626	0.671	0.202	0.370	0.746	0.380	0.498
Qwen1.5-32B	0.647	0.336	0.408	0.528	0.132	0.224	0.587	0.234	0.316
Qwen1.5-32B-Chat	0.883	0.678	0.744	0.704	0.228	0.412	0.793	0.453	0.578
Qwen1.5-72B	0.627	0.324	0.380	0.556	0.148	0.248	0.591	0.236	0.314
Qwen1.5-72B-Chat	0.896	0.710	0.776	0.730	0.254	0.436	0.813	0.482	0.606
Qwen1.5-110B-Chat	0.905	0.724	0.792	0.730	0.276	0.438	0.818	0.500	0.615
Qwen2-0.5B-Instruct	0.446	0.150	0.172	0.393	0.070	0.110	0.419	0.110	0.141
Qwen2-1.5B-Instruct	0.607	0.250	0.316	0.496	0.104	0.168	0.551	0.177	0.242
Qwen2-7B	0.576	0.260	0.316	0.478	0.120	0.192	0.527	0.190	0.254
Qwen2-7B-Instruct	0.835	0.584	0.642	0.682	0.198	0.362	0.758	0.391	0.502
Qwen2-72B	0.711	0.424	0.484	0.568	0.170	0.274	0.640	0.297	0.379
Qwen2-72B-Instruct	0.944	0.836	0.880	0.791	0.342	0.530	0.867	0.589	0.705

Table 8: The complete evaluation results and rankings of CFBench and its respective subsets. Notably, * stands for the model supporting mainstream languages excluding Chinese, and † represents calling through the API. The **bold**, underlined, and tilde denote the first, second, and third rankings, respectively. Llama-3-8B-Instruct-CN and Llama-3-70B-Instruct-CN respectively represent Llama-3-8B-Instruct-Chinese and Llama-3-70B-Instruct-Chinese, both of which have undergone Chinese SFT (Supervised Fine-Tuning). Both conifer_base and conifer_test are based on the Mistral-7B foundational model. For the base model, we used a 3-shot approach for generation.

Primary	Secondary	Definition	Example
Content Constraint	Lexical	Mandatory use of specific terms or symbols, including their inclusion and precise placement.	...must include the word "beautiful."
	Element	Mandates for including specific elements or concepts in responses, reflecting a scenario or object.	...highlights the Great Wall.
	Semantic	Directives on thematic content, perspective, or tone, emphasizing response significance.	Write a poem about London.
Numerical Constraint	Word Count	Limit the number of words or tokens.	A 50-word poem.
	Sentence Count	Limit the number of sentences.	... three sentences.
	Paragraph Count	Limit the number of paragraphs.	divided into 3 sections.
	Document Count	Limit the number of documents.	... list 3 articles.
Stylistic Constraint	Tone and emotion	The emotional tone must adhere to standards such as seriousness, anger, joy, humor, and politeness.	Write a letter in an angry and sarcastic tone.
	Form and style	Text expression standards ensure alignment with specific stylistic criteria in both presentation and perception.	Write a passage in an encyclopedic style.
	Audience-specific	Text should be tailored to specific audiences, ensuring clarity and relevance for children, students, or specialized groups.	Write a pome for a 6-year-old.
	Authorial style	Texts should emulate the styles of authors like Shakespeare to achieve artistic effects or depth.	Write a passage the style of Shakespeare.
Format Constraint	Fundamental	Widely accepted and utilized standard formats, including JSON, XML, LaTeX, HTML, Table, and Markdown.	Extract keywords and output in JSON format.
	Bespoke	Protocols for information expression tailored to specific needs,including paragraphing, headings, text emphasis, examples, and bullet points.	Summarize the main idea and output in unordered list format.
	Specialized	Formatting standards tailored for specialized applications or domains.	Conform to electronic medical record format.
Linguistic Constraint	Pragmatic	Contextual language study, encompassing speech acts, implicature, discourse, dialects, sociolects, and language policy.	Output in English, in classical Chinese style.
	Syntactic	Sentence structure, including phrases, constituents, subordinate clauses, ba-constructions, and imperatives.	Use imperatives with nouns and verb phrases.
	Morphological	The internal structure and formation rules of words, including roots, affixes, and morphological changes.	Output all content in lowercase English.
	Phonological	Study on phonological structures:phonemes, allophones, pitch, duration, and intensity.	Single-rhyme tongue twisters.
Situation Constraint	Role-based	Simulating characters based on context, emulating their traits, language, and behaviors.	You are Confucius, how do you decide?
	Task-specific	Offer tailored solutions based on a nuanced understanding of situational demands.	Must work from home, how to report?
	Complex context	Reasoning and problem-solving within intricate and multifaceted contexts.	4 on the left, 10 total, which from right?
Example Constraint	-	Regulate new responses by leveraging intrinsic patterns from a limited set of samples.	Example:input:xxx, output:{...}; input:xx, output?
Inverse Constraint	-	Narrow the response space through inverse constraints and indirect exclusion.	Prohibited from answer political topics.
Contradictory Constraint	-	Mutually exclusive constraints prevent fulfilling all requirements concurrently.	Write a five-character quatrain, 1000 words.
Rule Constraint	-	Standardize the road of responses through meticulously crafted logic flows or actions.	Each answer adds 1, 1+1=3, then 2+3=?

Table 9: Constraint System of CFBench

Domain	Scenarios List			
	Symptom Consultation	Diagnostic Explanation	Medication Guidance	Procedures
Healthcare	Wellness	Medical Info	Guidelines Inquiry	Public Health
	Medical Education			
	Teaching Methods	Resource Access	Curriculum Design	Communication
Education	Academic Counseling	Mental Health Support	Tutoring	Subject Q&A
	Reports	Interests		
	Market Research	Stock Analysis	Investment Analysis	Personal Finance
Finance	Corporate Tax	Insurance Management	Corporate Financing	Compliance & Risk
	Financial Education	Product Development	Customer Service	Financial Reports
	Regulatory Analysis			
	Legal Education	Legal Consultation	Document Review	Case Analysis
Legal	Statute Explanation	Regulation Analysis	IP Management	Legal Training
	Case Management	Compliance & Risk		
Media	Content Creation	Information Analysis	Marketing & Promotion	News Reporting
Tourism	Travel Consultation	Itinerary Planning	Route Introduction	
	JD Writing & Analysis	Resume Creation	Resume Screening	Interview Preparation
Recruitment	Interview Evaluation	Career Planning	Offer Comparison	Communication Skills
	Performance Review			
	Policy Research	Public Education	Service Guide	Public Services
Gov Affairs	Document Writing	Content Review	Business Procedures	Civil Servant Training
	Emergency Management			
	Purchase Planning	Market Trends	Property Policies	Development
Real Estate	Leasing	Property Valuation	Amenities	Financial Services
	Property Description	Content Creation	Sales & Marketing	Qualifications
	Renovation			
	Marketing & Sales	Driving & Safety	Customer Experience	Model Consultation
Automotive	Model Comparison	Loan Calculation	Insurance Evaluation	Claims Assessment
	Car Reviews	Maintenance & Repair		
	Romance	Family	Friendship	Workplace
Psychology	Self & Health	Social	Sexuality & Gender	Life Stages
	Organizational	Client Relations	Crisis Intervention	Public Psychology
	Business Analysis	Product Design	User Research	Coding & Debugging
Internet	Product Testing	Data Management	Cybersecurity	Computer Q&A
	Internet News	Marketing	Operations	UI/UX Design
Spirituality	Beliefs & Rituals	Divination	Feng Shui	Astrology
	Metaphysics	Spirituality	Healing	Content Review
	Training	Goal Setting	Nutrition	Workout Plans
Sports	Equipment & Tech	Performance	Injury Care	Specialized Training
	Mental Motivation	Data Tracking		
	Life Tips	Shopping Decisions	Instant Queries	Skincare
Lifestyle	Fashion & Styling	Naming	Recommendations	Planning
	Socializing	Life Creations	Q&A	
	Podcasts & Radio	TV & Film	Music	Literature
Culturtainment	Theater & Dance	Art	Cultural Events	Short Videos & Live
	Gossip	Content Creation		
Employment	Project Management	Translation	Office Efficiency	Marketing
	Administration	Customer Service	Team Collaboration	Collaboration
	Food & Restaurant Recs	Reviews & Feedback	Marketing & Promo	Food Content
Cuisine	Culinary Culture	Recipes & Menus	Cooking Techniques	Ingredient Prep
	Nutrition & Health	Food Safety	Culinary Training	
	Guides	Reviews	Hardware & Peripherals	News
Gaming	Software & Services	Development & Design	Operations	Mini Games
	Search	Marketing & Promotion	Esports & Tournaments	Culture & Education
Technology	Reviews	Launches	Buying Guides	Tips & Tricks
	Content Creation	Product Design	Marketing Copy	After-sales & Repairs

Table 10: Domain and Scenarios List

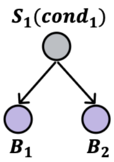
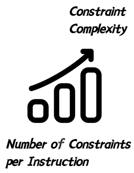
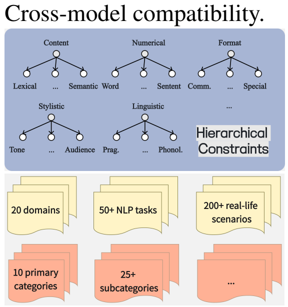
Benchmark	Case	Features	CFBench's Advantages
Complex Bench	**Instruction:** Please introduce the following painting. - If the work contains any animal, the description should be in English. - Otherwise, the description should be in Chinese. **Response:** Painting: "Mona Lisa"	Focus on the **combination** of instructions, including And/Chain/Selection/Nested. 	CFBench supports **simplicity in instruction compliance**, **broad applicability** across use cases, and **scalability** for various tasks.
Follow Bench	**Instruction:** Using following words to generate a story. "needed, money, computer, bought, happy". **Response:** In the heart of the city, there was a research facility that desperately needed funding for a crucial project. To gather the required money, ···	Focus on **gradual constraint escalation**. The more constraints per instruction, the greater the complexity. 	CFBench is **simple** to use, **resource-efficient**, and **scalable** for large models.
IFEval	**Instruction:** Given the sentence "Two young boys with toy guns and horns." can you ask a question? Please ensure that your response is in English, and in all lowercase letters. No capital letters are allowed. **Response:** what are the two young boys holding?	**Verifiable Instructions**. 	CFBench is **easy to implement**, **general-purpose across models**, and **scalable for large datasets**.
CFBench	**Instruction:** Overview of the Trump Shooting Incident, include the cause, process, aftermath, shooter information and Trump's feedback. Organize the content using numbered points and bold text. The cause should be in JSON format with event keywords: time, location, and nature. Summarize the process in three sentences. Discuss the impact from both international and domestic perspectives, keeping the total content under 500 words. Finally, summarize this event in a seven-word quatrain. **Response:** 1. Cause \n"time":"July 13, 2024", "location":"Butler, Pennsylvania, USA", "nature":"assassination attempt" \n 2. Process \n-Trump was giving a speech at the rally when the shooting occurred. \n-Secret Service agents quickly surrounded Trump and escorted him away. \n 3. ···	**Comprehensive Coverage**: Multi-dimensional constraints; Cross-task evaluation; Real-world complexity. **Standardized Framework**: Unified metrics; Cross-model compatibility. 	**Diverse instruction types**, **standardized evaluation**, **hierarchical constraints**.

Table 11: Benchmark Comparison for Different Instruction Types and Task Complexity

Prompt Template for CheckList Generation

Task Description

You are a professional instruction and response review expert. You need to extract all possible checklist items that could affect the evaluation of the result based on the [Instruction] and [Response] within the [Constraint System]. Your output should be as detailed as possible. Use numerical numbering and "\n" to list all points in the checklist.

Constraint System

1. Lexical content constraint: {Definition} {Example}
2. Word Count： {Definition} {Example}

...

25.Rule Constraint: {Definition} {Example}

Given Example

Instruction: Overview of the Trump Shooting Incident, include the cause, process, impact, aftermath, shooter information and Trump's feedback. Organize the content using numbered points and bold text. The cause should be in JSON format with event keywords: time, location, and nature. Summarize the process in three sentences. Discuss the impact from both international and domestic perspectives, keeping the total content under 500 words. Finally, summarize this event in a seven-word quatrain.

Response: Overview of the Trump Shooting Incident. **1.Cause** \n{"time":"July 13, 2024", "location":"Butler, Pennsylvania, USA","nature":"assassination attempt"} \n **2.Process** \n-Trump was giving a speech at the rally when the shooting occurred... **3.Impact** \n-International: The incident shocked the world, raising concerns about US political violence. \n-Domestic: US political tensions and polarization increased; Trump's support may rise.\n **4.Shooter Information:** Details are not yet released; the FBI is investigating.\n **5.Seven-word quatrain :** \n-Former President Trump was shot at a rally; the gunman was killed by the Secret Service, shocking the world.

CheckList

1.Content on Trump Shooting Incident 2.Includes cause, process, impact ... 3.Use numbered points and bold text 4.Cause in JSON 5.Process in three sentences 6.Impact: international and domestic 7.Total content under 500 words 8.Seven-word quatrain

Given Input

[**Instruction**]:{instruction} [**Respone**]:{response}

Your Answer about Checklist

Figure 10: Prompt Template for CheckList Generation

Figure 11: CFBench Example: English-Chinese Comparison, Data Itself in Chinese.