CS4: Measuring the Creativity of Large Language Models Automatically by Controlling the Number of Story-Writing Constraints

Anirudh Atmakuru[*]   **Jatin Nainani**[*]   **Rohith Siddhartha Reddy Bheemreddy**[*]
Anirudh Lakkaraju[*]   **Zonghai Yao**   **Hamed Zamani**   **Haw-Shiuan Chang**[*†]

University of Massachusetts, Amherst
140 Governors Dr., Amherst, MA, USA 01003
{aatmakuru, jnainani, rbheemreddy, alakkaraju, zonghaiyao}@umass.edu
{zamani,hschang}@cs.umass.edu

Abstract

Evaluating the creativity of large language models (LLMs) in story writing is difficult because LLM-generated stories could seemingly look creative but be very similar to some existing stories in their huge and proprietary training corpus. To overcome this challenge, we introduce a novel benchmark dataset with varying levels of prompt specificity: CS4 (**C**omparing the **S**kill of **C**reating **S**tories by **C**ontrolling the **S**ynthesized **C**onstraint **S**pecificity). By increasing the number of requirements/constraints in the prompt, we can increase the prompt specificity and hinder LLMs from retelling high-quality narratives in their training data. Consequently, CS4 empowers us to indirectly measure the LLMs' creativity without human annotations.

Our experiments on LLaMA, Gemma, and Mistral not only highlight the creativity challenges LLMs face when dealing with highly specific prompts but also reveal that different LLMs perform very differently under different numbers of constraints and achieve different balances between the model's instruction-following ability and narrative coherence. Additionally, our experiments on OLMo suggest that Learning from Human Feedback (LHF) can help LLMs select better stories from their training data but has limited influence in boosting LLMs' ability to produce creative stories that are unseen in the training corpora. The benchmark is released at https://github.com/anirudhlakkaraju/cs4_benchmark.

"Creativity is seeing what others see and thinking what no one else ever thought." — Albert Einstein

1 Introduction

Large language models (LLMs) can generate stories that surpass the quality of human-written ones, which intensifies debates among researchers and

---

[*] indicates equal contribution.
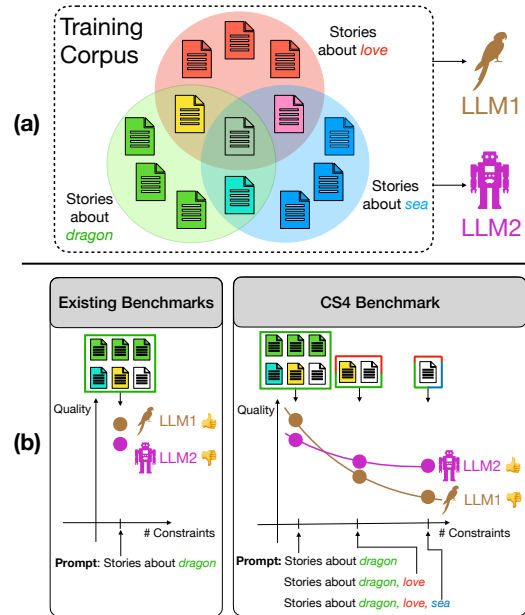[†] The work was mostly done at Amazon.

Figure 1: Comparison between CS4 and existing benchmarks. (a) Depiction of training corpora subsets for different narrative themes, illustrating the decreasing availability of training examples for LLMs as prompt specificity increases. (b) In response to general instructions, LLM1 tends to copy the relevant high-quality stories from its training corpus to achieve a good score in existing story-writing benchmarks. CS4 measures LLMs' creativity by comparing LLMs' performance drops for more specific instructions. Given more constraints, LLM2 could leverage very limited training data to output higher-quality stories than LLM1, so LLM2 is more creative.

professional writers about whether human writers may be replaced by LLMs anytime soon (Gómez-Rodríguez and Williams, 2023; Zhao et al., 2023b; Marco et al., 2024). One common argument on the human side is that LLMs still lack creativity (Boussioux et al., 2023; Chakrabarty et al., 2024; Gómez-Rodríguez and Williams, 2023), which is defined as the ability to produce original, unseen, and high-quality stories in this study.

Current evaluation methodologies are often in-

sufficient for measuring creativity in story writing. In existing creative-writing or instruction-following benchmarks such as FollowBench (Jiang et al., 2023b), IFEval (Zhou et al., 2023), Collie (Yao et al., 2023), CIF-Bench (Li et al., 2024b), InstructEval (Chia et al., 2024), CFBench (Zhang et al., 2024), and InfoBench (Qin et al., 2024), instructions are typically broad and impose limited constraints, which allow LLMs to perform well by slightly modifying (or even copying) relevant and high-quality narratives from their training datasets (Chang et al., 2023; Hartmann et al., 2023; Min et al., 2023a) without actually "understanding" the output story (West et al., 2023). However, it is challenging to directly check if the generated stories mostly come from the stories in the model's training corpora (Chen et al., 2024). This is because 1) LLMs could (or might often) output text that is semantically similar to but lexically different from a training text piece (Ippolito et al., 2023), 2) the training corpora are usually huge, and searching these for identifying LLM-generated text could be expensive, and 3) the training data used for training many popular LLMs are not publicly available.

To assess the creativity of LLMs in terms of generating original stories, we propose a novel evaluation benchmark, CS4, which provides instructions with up to 39 constraints. As we have more constraints in the prompt, the instruction becomes more specific and limits the LLM's ability to copy text from its training data. For example, as shown in Figure 1, given a prompt like *"Give me a story about a dragon"*, an LLM may reproduce the finest story from its training data that mentions the word *"dragon"*, which could naturally be superior to an average story crafted by a human. However, this might not be the case as the specificity of the prompt increases since the LLM would have fewer training examples to leverage, for example, *"Give me a story involving a dragon, love, and sea"*.

Adding more constraints to a prompt can not only push LLMs to be more creative but also help us assess how effectively they handle the intricate challenges human writers face (He et al., 2024a; Wen et al., 2024; Pham et al., 2024). This makes our CS4 benchmark both scientifically valuable and practically relevant for professional writers, particularly in the book and film industries. These writers are tasked with crafting intricate, high-quality narratives that meet various constraints from the plots from the existing chapters (Ippolito et al., 2022), from the rules of their imaginary world, and by

the different needs of editors, the marketing team, and the readers. Understanding the extent to which LLMs can replicate this process will not only help us deep dive into their behavior but could also be essential for determining their applicability in these creative domains.

Similar to Jiang et al. (2023b); Yao et al. (2023); Zhou et al. (2023), we synthesize our prompts by asking GPT-4 (Achiam et al., 2023) to generate up to 39 constraints for the story-writing instructions from users. The constraints in CS4 are synthesized in two ways: 1) manually writing constraints for some user instructions as few-shot examples, which help GPT-4 generate constraints for a new user instruction, and 2) using GPT-4 to extract the constraints from a human-written story to make sure the constraints are realistic and satisfiable.

CS4 measures common metrics such as instruction-following ratio, coherence, and diversity. However, unlike the existing benchmarks that provide a single score for each metric, CS4 uses multiple sets of prompts with different numbers of constraints to compare LLMs' performances across a range of prompt specificities (Figure 1). A more specific prompt (i.e., one with more constraints) is usually more difficult to satisfy, but a more creative LLM should have a smaller performance degradation.

In our experiments, we first found that all the explored LLMs output significantly worse stories as the prompt specificity increases. This indicates that more constraints indeed pose more creativity challenges on LLMs for generating novel responses (Chakrabarty et al., 2024; Lu et al., 2024). Second, given more constraints, we found that the stories from some LLMs deteriorate faster than those from others. For example, increasing the number of constraints from 7 to 39 makes the constraint satisfaction probability drop by approximately 35% for LLaMA-2 7B (Touvron et al., 2023) but by only approximately 20% for Gemma-7B (Team et al., 2024). Third, not all metrics degrade at the same speed. For example, when we increase the number of constraints from 7 to 23, LLaMA-2 outputs the stories that are similarly coherent but satisfy 25% less constraints. Finally, by comparing OLMo Instruct and OLMo SFT (Groeneveld et al., 2024), we found that the performance enhancement due to Direct Preference Optimization (DPO) (Rafailov et al., 2024) or more generally, Learning from Human Feedback (LHF) (Ouyang et al., 2022) is much smaller given more constraints. The

results support the hypothesis that LHF achieves better performance but worse diversity by encouraging the LLMs to select good stories from the training data (Kirk et al., 2024; Le Bronnec et al., 2024; Xiao et al., 2024; Lake et al., 2024).

2 Benchmark Construction

As illustrated in Figure 2, the methodology adopted in this research can be broken down into three steps – 1) generating constraints, 2) generating stories, and 3) evaluating the stories based on the constraints. In this section, we first explain the challenges encountered in constructing the benchmark and the solution we propose in each step to overcome these challenges. Then, we describe some important details of the first two steps.

2.1 Challenges and Solutions

When constructing the CS4 benchmark, we iteratively improve our evaluation process and prompting strategies. Our goal is to build a benchmark that allows us to observe an LLM's response given many constraints making it challenging for the LLM. In each iteration, we run small-scale experiments on ChatGPT (GPT-3.5-Turbo) and manually evaluate their responses. We summarize the challenges faced and our solutions below. The challenges also highlight the differences between CS4 and previous studies.

Generating lots of constraints: In the previous instruction-following studies, the number of constraints for each user instruction is usually fixed and small. For example, even the "hard" set of InfoBench (Qin et al., 2024) only contains 6.3 constraints on average and each instruction has only one set of constraints.

In CS4, we first synthesize many constraints for every user instruction using GPT-4 (Achiam et al., 2023) to progressively sample the intervals from these constraints later to control the number of constraints of each instruction. We call this method the instruction-based approach because GPT-4 generates several constraints based only on the user instruction.

Not generating very difficult constraints: When there are many synthesized constraints, the constraints might contradict each other, making some constraints unsatisfiable while writing a story. In the instruction-based approach, we prompt GPT-4 to make the constraints satisfiable. Besides, we propose a story-based approach, which asks GPT-4 to derive the constraints from the user instruction and a reference story that satisfies the instruction (Li et al., 2024a; Pham et al., 2024), to further address this issue.

In both approaches, our prompts encourage the GPT-4 to generate atomic constraints (Min et al., 2023b), which means each constraint cannot be broken down further into simpler parts. This alleviates the problem of generating very complex constraints that are hard to follow.

Not generating very easy constraints: An instruction following benchmark that is easy for LLMs may have low discriminative power for LLMs' creative capabilities (Jiang et al., 2023b; Zhang et al., 2024). For prompts generated using the instruction-based approach, we observe that ChatGPT tends to output fantasy stories. Since anything could happen in a fantasy world, LLMs could easily satisfy the arbitrary number of constraints. To solve this problem, we restrict the user instructions to queries on realistic fictions, which force LLMs to respect common sense and logical rules in the real world.

For the story-based approach, our first version of the prompt makes GPT-4 sequentially modify each sentence in the reference story as a constraint. This would allow a story writer to simply convert the constraints back to sentences, forming a story very similar to the reference story. To address this issue, we prompt GPT-4 to generate constraints that span multiple sentences of the story and encourage the production of constraints that do not follow the order of the story. In Figure 8, we compare the constraints before and after addressing this issue.

Reducing the instability of evaluation: It is difficult to find a metric that can objectively compare the quality of two stories from two different genres. For example, incoherence or logical flaws might be acceptable in an adventure story but disastrous in a detective story. Some human or LLM evaluators might also prefer one genre over another. To reduce the bias and variance in the evaluation, we first ask GPT-4 to generate a "base" story for the user instruction without seeing the constraints. Then, the 7B-LLMs that are being evaluated are prompted to modify the base story to satisfy the constraints. Because of the similarity of output stories from different testing LLMs, it is easier to compare their qualities.

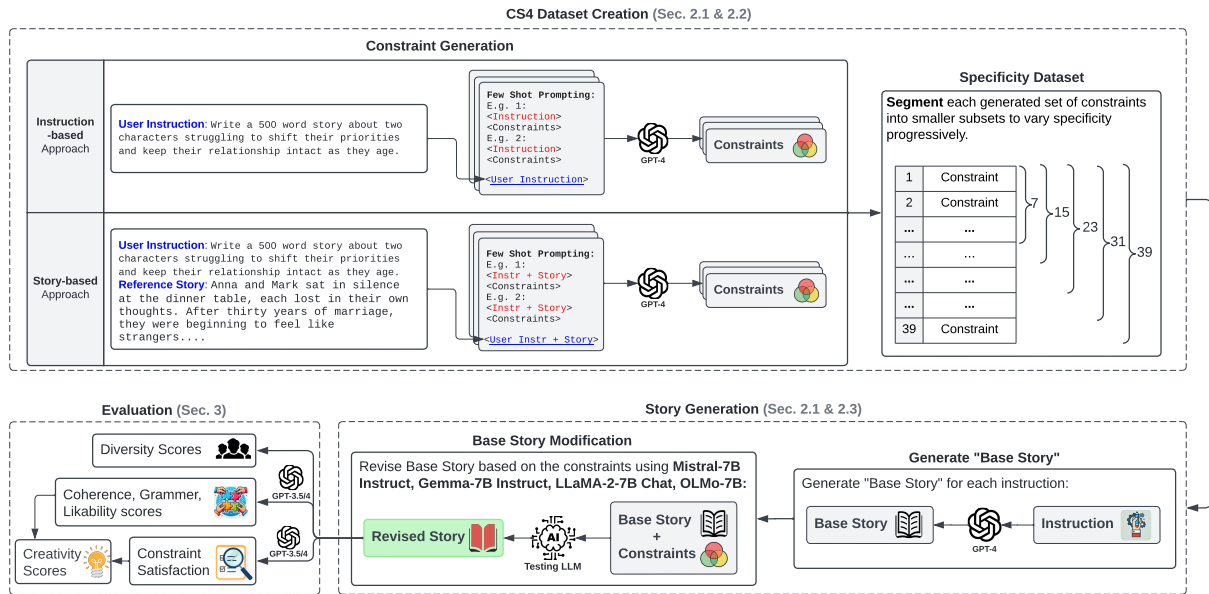The extra base story generation step also prevents the tested LLMs from "cheating". The LLMs

Figure 2: An overview of the evaluation process in CS4 benchmark. First, we use two different few-shot in-context learning approaches to synthesize 39 constraints from every user instruction and conduct sub-sampling to create the prompts with fewer constraints. Next, for each user instruction, the testing LLMs of interest revise a base story, which is generated without seeing the constraints, to satisfy the constraints. The revised stories are evaluated in terms of their constraint satisfaction ratio, quality, and diversity. Finally, we estimate LLMs' creativity by summarizing their coherence scores and instruction satisfaction ratios for different number of constraints.

might have seen and memorized the reference story used in the story-based approach and the base story prevents the LLMs from outputting the reference story. Besides, without the base story, LLMs might output a story that contains just the copied input constraints from the instruction-based prompt with some simple transitions between constraints.

2.2 CS4 Dataset Creation

As illustrated in Figure 2 and discussed in the previous section, we use two strategies to synthesize the constraints and enrich the diversity of the CS4 benchmark. For each constraint generation strategy, our benchmark dataset consists of 50 user instructions.

Instruction-based Approach: We collect 50 realistic fiction ideas from Kindlepreneur[1] as user instructions and manually create two few-shot examples of instructions with ten constraints each. These in-context examples guide GPT-4 in generating high-quality constraints, which include more style-related or open-ended elements compared to the story-based approach. Please see Figure 7 for an example of such constraints.

Story-based Approach: We collect 50 user instructions and their corresponding human-written reference stories from the Writing Prompts dataset (Fan et al., 2018) and our manually written few-shot examples encourage GPT-4 to extract more abstract and non-sequential constraints from the reference story. Compared to the instruction-based approach, this method synthesizes more plot-related constraints.

Constraint Segmentation: For each instruction from the above constraint generation strategies, we segment sets of 7, 15, 23, 31, and 39 constraints cumulatively. This segmentation results in 500 unique prompts that could be used for story generation (50 instructions $*$ 2 constraint generation approaches $*$ 5 sets of constraints).

2.3 Story Generation

We first prompt GPT-4 to write a "base" story with less than 500 words given the user instruction alone. Then, we provide the instruction, the base story, and the constraints to the chosen 7B-LLMs asking them to modify the base story such that the constraints are satisfied in the new story in about 500 words. This constraint on the story length increases the task difficulty and reduces the costs of the subsequent LLM evaluation.

3 Evaluation Metrics

In this work, we primarily use four metrics to evaluate the stories generated. These include constraint satisfaction, story coherence, output diversity, and LLM's creativity. In the interest of evaluation cost, we adopt LLM-as-the-Judge (Liu et al., 2023) and use GPT-3.5-Turbo to evaluate constraint satisfaction and story coherence by default. Besides the scores, we also ask GPT-3.5-Turbo to provide justifications to improve and support its evaluation judgement (Chiang and Lee, 2023).

3.1 Constraint Satisfaction

We use GPT-3.5-Turbo to judge whether the responses from the testing LLMs satisfy each constraint in the prompt (Jiang et al., 2023b; Qin et al., 2024; Chia et al., 2024; Zhang et al., 2024; Wen et al., 2024). Then, our constraint satisfaction metric is defined as the ratio of the number of constraints satisfied by the story to the total number of constraints in the prompt. You can find our prompt in Figure 11. Our preliminary evaluations show that the judgment of GPT-3.5-Turbo is highly correlated with human judgments in CS4.

3.2 Story Coherence

Adding more constraints tends to make the stories incoherent. Further, coherence is a more objective metric than likability, so we use coherence as our main metric to evaluate the generated story. In Appendix C, we also report the results of grammar and likeability. We evaluate coherence using GPT-3.5-Turbo, which has been shown to match the performance of human evaluators (Gilardi et al., 2023).

We found that GPT-3.5-Turbo tends to be too generous (i.e., often assigning the highest score) when evaluating the stories generated by LLMs in CS4 (Gmyrek et al., 2024). To solve the issue, we choose a generated story with the middle number of constraints (23 in our experiments) as a baseline for every user instruction, and compare all stories for that user instruction with the baseline. To avoid the positional bias, we randomly swap the order of each generated story and the baseline in the comparison (Zeng et al., 2023). This evaluation method reduces the times of running GPT-3.5-Turbo from the N^2 to $N - 1$ compared to the exhaustive pairwise comparison (Liusie et al., 2024), where N is the number of generated stories per instruction.

In each comparison, GPT-3.5-Turbo is prompted to provide a score out of 5 for both stories for the criteria of grammar, coherence, and likeability. To get a normalized coherence score for the LLM given the number of constraints, we first average the scores across the user instructions and then divide the scores by 5 (the maximum possible coherence score).

3.3 Output Diversity

We use two methods to measure the generation diversity of the testing LLMs: self perplexity and dist-n diversity (Li et al., 2016). Self perplexities are calculated using the same LLM that generates the story. A higher self perplexity means the testing LLM also assigns probabilities to other tokens, which implies high generation diversity (Hashimoto et al., 2019). Dist-n is the ratio of distinct n-grams to the total number of n-grams and we compute the products of dist-2, dist-3, and dist-4 (Li et al., 2022):
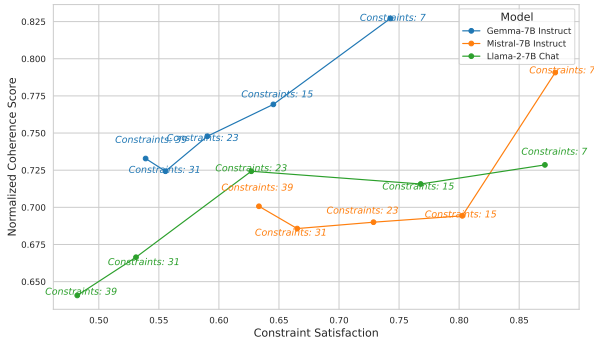
$$\text{Dist-n Diversity} = \prod_{n=2}^{4} \frac{|\text{unique n-grams}|}{|\text{total n-grams}|}. \quad (1)$$
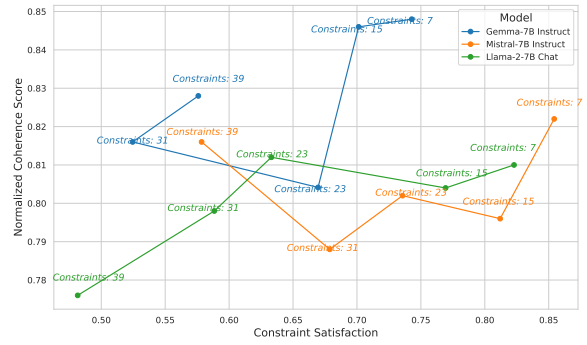
3.4 Creativity Measurements

By comparing the curves of LLMs in Figure 1 (b), we can understand that LLM2 is more creative than LLM1 since it performs better given more constraints and this performance decays slower as more constraints are introduced. To quantify such observations, we propose two new creativity metrics: Quality Under n Constraints (QUC$_n$), and Relative Creativity Score (RCS$_{m,n}$), which is the quality difference in two stories generated using m constraints and n constraints, respectively.

For a story generated using n constraints, QUC$_n$ is defined as the product of the normalized coherence score and the average percentage of constraints satisfied. When the prompt contains many constraints (i.e., large n), a good output story should still be both coherent and adhere to many constraints, leading to a high to a high QUC$_n$.
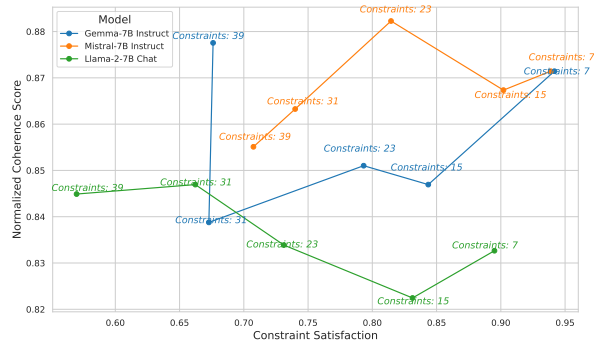
To measure the quality decay speed, we define the relative creativity score RCS$_{m,n}$ as the difference between the story quality from the smallest number of constraints m (QUC$_m$) and the story quality from the largest number of constraints n (QUC$_n$). A smaller RCS$_{m,n}$ of an LLM indicates its slower decay speed and thus better creativity.
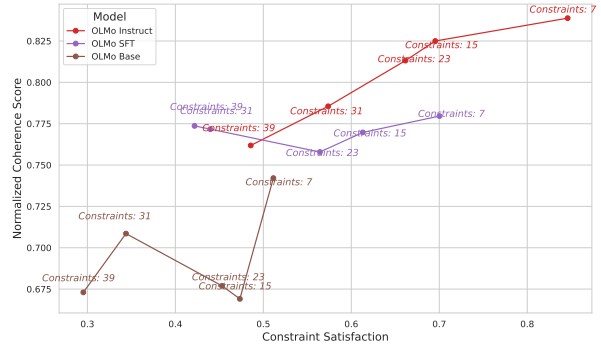
(a) Story-based constraints and GPT-4 evaluator

(b) Story-based constraints and GPT-3.5-Turbo evaluator

(c) Instruction-based constraints and GPT-3.5-Turbo evaluator

(d) Story-based constraints and GPT-3.5-Turbo evaluator

Figure 3: Analyzing the trade-off between coherence and constraint satisfaction.

4 Experiments

This section describes our experimental setup and analyzes the results obtained for each evaluation metric used in this study. We first highlight the trade-off between a model's instruction-following ability and the coherence of the generated stories. We then analyze the effect of LHF and SFT on text generation using OLMo, discuss our creativity measurements and the diversities of different LLMs, and identify which types of constraints are not satisfied.

4.1 Setup

In this work, we consider six models for story generation, namely LLaMA-2-7B Chat (Touvron et al., 2023), Gemma-7B Instruct (Team et al., 2024), Mistral-7B Instruct (Jiang et al., 2023a), and three versions of OLMo-7B (Groeneveld et al., 2024) - OLMo Base, OLMo SFT (the model obtained after SFT), and OLMo Instruct (the model obtained after LHF). Each model generates stories for all the 500 unique prompts described in Section 2.2, resulting in 3000 stories. The temperature of the generation is 0.8 and the threshold of top-p is 0.95. We compare the coherence and constraint satisfaction performances of different LLMs based on the

3000 stories.

4.2 Trade-off Between Coherence and Constraint Satisfaction

If an LLM outputs the best story in its memory regardless of the prompt, it could achieve the highest coherence while having the worst constraint satisfaction. In contrast, an LLM could simply copy all the plot-related constraints into the story without proper transitions, resulting in a high constraint satisfaction score but poor coherence. Figure 15 shows an example of such a trade-off. Hence it is important to evaluate both constraint satisfaction and coherence and study the interplay between them.

To compare this trade-off in different LLMs, we plot both metrics under different numbers of constraints in Figure 3. Points at the top-right corner of the figures correspond to high coherence and constraint satisfaction scores. In Figure 3a, we observe that coherence and constraint satisfaction do not decay at the same rate. Instead, different LLMs adjust the balance between coherence and satisfaction differently as the number of constraints increases. For example, **LLaMA-2** focuses on maintaining coherence by sacrificing the satisfaction probability

when the number of constraints ranges between 7 and 23.

We use GPT-3.5-Turbo to evaluate the coherence and constraint satisfaction in the rest of the experiments because running all the experiments using GPT-4 exceeds our budget. Replacing GPT-4 in Figure 3a with GPT-3.5-Turbo makes the curves in Figure 3b noisier, but the overall trends remain similar.

Finally, we find that different constraint-synthesizing approaches could lead to different results. For example, **Gemma-7B** outputs more coherent stories in response to story-based prompts compared to **Mistral-7B** in Figure 3b, but the trend is reversed for instruction-based prompts in Figure 3c.

4.3 Effects of LHF

A similar analysis was also conducted on **OLMo Base**, **OLMo SFT**, and **OLMo Instruct** in Figure 3d to understand the impact of different stages of LLM training. Their curves are smoother compared to Figure 3b, probably because it is easier for GPT-3.5-Turbo to evaluate the stories by comparison when the stories from LLMs share similar styles and have relatively worse qualities.

Figure 3d demonstrates that **OLMo Instruct** is much better than **OLMo SFT** given only 7 constraints, but they almost converge when they are presented with 31 or 39 constraints. We hypothesize that this is because LHF can help LLMs select better stories when the number of constraints is not large. For example, at Figure 1 (b), by learning from human preferences, LLMs could select the six most coherent stories that satisfy the constraints. However, if only one relevant training story satisfies all three constraints, human preference data will not help LLMs increase the story quality or compose new stories. These insights are hard to discover in the previous benchmarks whose number of constraints is seldom greater than 10.

4.4 Creativity Evaluation

Table 1 compares the overall creativity of different LLMs. High $QUC_{n=39}$ scores for **Gemma-7B** and **Mistral-7B** indicate that these models produce stories of good quality even in the presence of several constraints. **Gemma-7B** has also achieved the lowest $RCS_{m=7,n=39}$ score indicating that it is more creative because the quality of its stories degrades slower as more constraints are introduced.

Model	QUC_{39}	$RCS_{7,39}\downarrow$	Median Length	Self Perplexity
Gemma-7B Instruct	**0.4768**	**0.1531**	**488**	3.6620
Mistral-7B Chat	0.4720	0.2301	657	1.5834
LLaMA-2-7B Instruct	0.3736	0.2928	513	2.2870
OLMo-7B Base	0.1988	0.1807	531	**3.6740**
OLMo-7B SFT	0.3263	0.2195	572	3.6172
OLMo-7B Instruct	0.3700	0.3395	843	3.6007

Table 1: Overall comparison of different LLMs on story-based constraints. Smaller RCS indicates higher creativity. Models adhering closely to the 500-word limit in our instruction show better compliance and higher self perplexities mean more diverse outputs. Both metrics are computed across all the generated stories. The best values are highlighted.

An LLM could add more relevant plots or transitions to a longer story to satisfy more constraints or make it more coherent. Thus, although **OLMo Instruct** has a higher $QUC_{n=39}$ score than **OLMo SFT**, it is possible that the source of improvement comes from its tendency to output longer responses by ignoring the $500-$word constraint in our prompt (Singhal et al., 2023; Dubois et al., 2024). Finally, **Gemma-7B** outputs more concise and diverse stories compared to **Mistral-7B**. Having multiple ways to satisfy many constraints in shorter stories demonstrates the creativity of **Gemma-7B**.

4.5 Story Diversity Evaluation

In Figure 4b, **OLMo Instruct** has a slightly lower self perplexity, which is likely a consequence of LHF during model training (Kirk et al., 2024; Le Bronnec et al., 2024; Xiao et al., 2024). The vocabulary diversity of **LLaMA-2-7B Chat** also decays in Figure 4c. Except for these two exceptions, Figure 4 shows that the diversities of all LLMs do not have significant changes overall as we add more constraints into the prompt.

These results might be caused by the balance of two opposite trends: as the number of constraints increases, LLMs have fewer relevant training examples to leverage, potentially leading to decreased diversity. On the other hand, there are more possible combinations of constraints they can choose from to satisfy, which could increase diversity. As a result, we can focus on analyzing the trade-off between coherence and constraint satisfaction, without pondering about the implications on diversity.
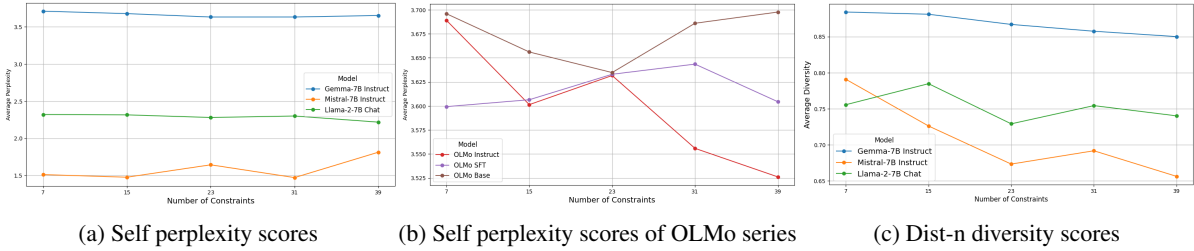
(a) Self perplexity scores   (b) Self perplexity scores of OLMo series   (c) Dist-n diversity scores

Figure 4: Generation diversity for stories written using story-based constraints.

4.6 Error Analysis on OLMo

To delve deeper into the impact of SFT and LHF in text generation, we perform an error analysis on the responses to prompts with 39 constraints. In the error analysis, we manually inspect around 600 constraint satisfactions, and with the help of GPT-4, we examine all individual constraints violated by **OLMo Base**, **OLMo SFT**, and **OLMo Instruct**.

We observed that when the constraints were satisfied by all three models, they were usually specific actions or events about the story, or were related to the style or tone of the story. The LLMs could simply copy such constraints to the story, thus satisfying most of them. When there was a slight depth added to the character or plot, **OLMo Base** was unable to satisfy them. **OLMo Base** was also unable to satisfy inference-based constraints, for example, events reflecting societal norms in the story. Furthermore, when there was any use of obscure knowledge or interactions between different characters or plot lines, all models failed. Finally, when complex narrative patterns (like plot twists), specific roles to characters, or futuristic/imaginative concepts were given, both **OLMo Base** and **OLMo SFT** fail but **OLMo Instruct** succeeds. One possible reason could be that **OLMo Instruct** tends to output a much longer story, which has more room to satisfy more complicated constraints.

5 Related Work

As LLMs become popular, more researchers are curious about whether LLMs could do well in tasks requiring creativity such as humor generation (Zhong et al., 2024), comedy creation (Mirowski et al., 2024), generating creative analogies (Kang et al., 2022; Ding et al., 2023; Liu et al., 2024), and creativity tests in psychology (Bellemare-Pepin et al., 2024). Previous findings suggest that LLMs could reduce the creativity of a group (i.e., homogenizing effect) (Moon et al.; Anderson et al., 2024) while improving the creativity of individual humans (Kang et al., 2022; Ding et al., 2023; Liu et al., 2024). Nevertheless, these studies do not try to measure the creativity of LLMs while preventing LLMs from memorizing something that looks creative in the training corpus. Our goal is to provide an automatic framework for evaluating the creativity of LLMs under up to 39 constraints, which differs from the focus of previous benchmarks. For example, Boussioux et al. (2023); Chakrabarty et al. (2024) recruit humans to judge if the outputs from LLMs are creative. CFBench (Zhang et al., 2024) analyzes the satisfaction ratios given mostly less than 8 constraints. CELLO (He et al., 2024b), ComplexBench (Wen et al., 2024), and Suri (Pham et al., 2024) provide more complex instructions without measuring LLMs' creativity using prompt specificities.

Finally, our methodology is related to the benchmarks that test LLMs' reasoning ability. For example, Wu et al. (2023); Nezhurina et al. (2024) propose reasoning tasks that are challenging for LLMs to test their limits. Another concurrent study (Lu et al., 2024) proposes to evaluate LLMs' coding creativity by prohibiting the usage of common functions such as a for-loop. Instead, our benchmark focuses on measuring story-writing creativity in practical settings.

6 Conclusion

In this work, we propose a novel way to measure LLMs' capability in creating original stories. The resulting CS4 benchmark has the following desired attributes, which makes it particularly useful for LLM developers. 1) **Inexpensive:** We do not need human annotators to evaluate LLMs' output. For the LLMs like OLMo, the GPT3.5-Turbo evaluator is sufficient. 2) **Practical:** The in-depth analysis of the benchmark reveals several actionable and undiscovered insights. For example, prioritization of coherence and instruction-following shifts for different LLMs, types of prompts, and prompt speci-

ficities; LHF's effectiveness diminishes severely in the presence of more constraints. 3) **Simple:** By simply synthesizing more constraints and designing proper domain-specific prompts, we can measure the creativity of LLMs. The simplicity makes our methodology applicable to many domains other than story generation.

7 Acknowledgement

8 Limitations

As prompts become more specific, tasks could also become difficult for humans (Tulving and Thomson, 1973). Since we haven't conducted human experiments, we are unable to compare LLMs' performances with human performances as in West et al. (2023); Tian et al. (2024). Next, professional writers have diverse ways to leverage LLMs (Ippolito et al., 2022), we do not know how representative our prompts in CS4 are. Furthermore, while LLM-as-the-Judge is supported and widely adopted by previous research (Chiang and Lee, 2023; Gilardi et al., 2023; Jiang et al., 2023b; Qin et al., 2024; Chia et al., 2024; Zhang et al., 2024; Wen et al., 2024), we would like to perform a human evaluation of the generated stories for their quality and compare the correlation between the evaluation judgment of LLMs and humans.

Although we can run the LLM evaluators automatically, the dataset construction still requires several iterations of prompt refinements. If we want to measure the creativities for more applications, automating the prompt design and prompt difficulty control is an interesting future project. To achieve the goal, it might be necessary to measure the specificity/difficulty of each constraint and automatically detect conflicts between constraints. Moreover, we haven't investigated if various agent-like or sophisticated prompting strategies such as Zhong et al. (2024); Zhao et al. (2023a); Mirowski et al. (2024); Wei et al. (2024) could boost the creativity in CS4.

Finally, there are many different ways to define and measure creativity. LLMs could do very well in psychological creativity tests (Bellemare-Pepin et al., 2024), but not so well in other measurements such as comedy creation (Bellemare-Pepin et al., 2024). We only adopt one creativity definition (i.e., outputting original contents) and provide one inexpensive yet indirect way to measure the creativity of LLMs. The creativity measurements are tangled with other abilities of LLMs. For example, an LLM could be good at outputting an original story but bad at following instructions. CS4 currently may not be able to assign a high creativity score on such an LLM. In this case, directly measuring the similarity between the output story and the stories in the training data might be expensive but necessary.

9 Impact Statement

Creativity is a signature of human intelligence, so measuring the creativity gap between humans and AI models could help us predict if or when AI models could achieve artificial general intelligence. By meticulously assessing the impact of prompt specificity, our research can advance our understanding of the gap between artificial creativity and human-like storytelling and potentially be extended to measure creativity in more domains.

Other researchers could extend our conclusions, potentially leading to negative impacts on our research. For example, they might use our observations as evidence to argue that an LLM is merely a parrot without reasoning abilities, or that LHF is not an effective approach.

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*.

Barrett R Anderson, Jash Hemant Shah, and Max Kreminski. 2024. Evaluating creativity support tools via homogenization analysis. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Antoine Bellemare-Pepin, François Lespinasse, Philipp Thölke, Yann Harel, Kory Mathewson, Jay A Olson, Yoshua Bengio, and Karim Jerbi. 2024. Divergent creativity in humans and large language models. *arXiv preprint arXiv:2405.13012*.

Leonard Boussioux, Jacqueline Ng Lane, Miaomiao Zhang, Vladimir Jacimovic, and Karim R Lakhani. 2023. The crowdless future? generative ai and creative problem solving. *Organization Science*, (ja).

Tuhin Chakrabarty, Philippe Laban, Divyansh Agarwal, Smaranda Muresan, and Chien-Sheng Wu. 2024. Art or artifice? large language models and the false promise of creativity. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–34.

Kent K Chang, Mackenzie Cramer, Sandeep Soni, and David Bamman. 2023. Speak, memory: An archaeology of books known to chatgpt/gpt-4. *arXiv preprint arXiv:2305.00118*.

Tong Chen, Akari Asai, Niloofar Mireshghallah, Sewon Min, James Grimmelmann, Yejin Choi, Hannaneh Hajishirzi, Luke Zettlemoyer, and Pang Wei Koh. 2024. Copybench: Measuring literal and non-literal reproduction of copyright-protected text in language model generation. *arXiv preprint arXiv:2407.07087*.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. 2024. Instructeval: Towards holistic evaluation of instruction-tuned large language models. In *Proceedings of the First edition of the Workshop on the Scaling Behavior of Large Language Models (SCALE-LLM 2024)*, pages 35–64.

Cheng-Han Chiang and Hung-yi Lee. 2023. A closer look into using large language models for automatic evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8928–8942.

Zijian Ding, Arvind Srinivasan, Stephen MacNeil, and Joel Chan. 2023. Fluid transformers and creative analogies: Exploring large language models' capacity for augmenting cross-domain analogical creativity. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 489–505.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120.

Paweł Gmyrek, Christoph Lutz, and Gemma Newlands. 2024. A technological construction of society: Comparing gpt-4 and human respondents for occupational evaluation in the uk. *British Journal of Industrial Relations*.

Carlos Gómez-Rodríguez and Paul Williams. 2023. A confederacy of models: A comprehensive evaluation of llms on creative writing. *arXiv preprint arXiv:2310.08433*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. 2024. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*.

Valentin Hartmann, Anshuman Suri, Vincent Bindschaedler, David Evans, Shruti Tople, and Robert West. 2023. Sok: Memorization in general-purpose large language models. *arXiv preprint arXiv:2310.18362*.

Tatsunori B Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1689–1701.

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.

Daphne Ippolito, Florian Tramer, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, and Nicholas Carlini. 2023. Preventing generation of verbatim memorization in language models gives a false sense of privacy. In *Proceedings of the 16th International Natural Language Generation Conference*, pages 28–53.

Daphne Ippolito, Ann Yuan, Andy Coenen, and Sehmon Burnam. 2022. Creative writing with an ai-powered writing assistant: Perspectives from professional writers. *arXiv preprint arXiv:2211.05030*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Hyeonsu B Kang, Xin Qian, Tom Hope, Dafna Shahaf, Joel Chan, and Aniket Kittur. 2022. Augmenting scientific creativity with an analogical search engine. *ACM Transactions on Computer-Human Interaction*, 29(6):1–36.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the effects of rlhf on llm generalisation and diversity. In *The Twelfth International Conference on Learning Representations*.

Thom Lake, Eunsol Choi, and Greg Durrett. 2024. From distributional to overton pluralism: Investigating large language model alignment. *arXiv preprint arXiv:2406.17692*.

Florian Le Bronnec, Alexandre Vérine, Benjamin Negrevergne, Yann Chevaleyre, and Alexandre Allauzen. 2024. Exploring precision and recall to assess the quality and diversity of llms. In *62nd Annual Meeting of the Association for Computational Linguistics*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119, San Diego, California. Association for Computational Linguistics.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024a. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. 2022. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*.

Yizhi Li, Ge Zhang, Xingwei Qu, Jiali Li, Zhaoqun Li, Zekun Wang, Hao Li, Ruibin Yuan, Yinghao Ma, Kai Zhang, et al. 2024b. Cif-bench: A chinese instruction-following benchmark for evaluating the generalizability of large language models. *arXiv preprint arXiv:2402.13109*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522.

Yiren Liu, Si Chen, Haocong Cheng, Mengxia Yu, Xiao Ran, Andrew Mo, Yiliu Tang, and Yun Huang. 2024. How ai processing delays foster creativity: Exploring research question co-creation with an llm-based agent. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–25.

Adian Liusie, Potsawee Manakul, and Mark Gales. 2024. Llm comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 139–151.

Yining Lu, Dixuan Wang, Tianjian Li, Dongwei Jiang, and Daniel Khashabi. 2024. Benchmarking language model creativity: A case study on code generation. *arXiv preprint arXiv:2407.09007*.

Guillermo Marco, Julio Gonzalo, Ramón del Castillo, and María Teresa Mateo Girona. 2024. Pron vs prompt: Can large language models already challenge a world-class fiction author at creative text writing? *arXiv preprint arXiv:2407.01119*.

Sewon Min, Suchin Gururangan, Eric Wallace, Hannaneh Hajishirzi, Noah A Smith, and Luke Zettlemoyer. 2023a. Silo language models: Isolating legal risk in a nonparametric datastore. *arXiv preprint arXiv:2308.04430*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023b. Factscore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*.

Piotr Mirowski, Juliette Love, Kory Mathewson, and Shakir Mohamed. 2024. A robot walks into a bar: Can language models serve as creativity supporttools for comedy? an evaluation of llms' humour alignment with comedians. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 1622–1636.

Kibum Moon, Adam Green, and Kostadin Kushlev. Homogenizing effect of large language model (llm) on creative diversity: An empirical comparison.

Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. 2024. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *arXiv preprint arXiv:2406.02061*.

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.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371*.

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*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language

model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Prasann Singhal, Tanya Goyal, Jiacheng Xu, and Greg Durrett. 2023. A long way to go: Investigating length correlations in rlhf. *arXiv preprint arXiv:2310.03716*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Yufei Tian, Tenghao Huang, Miri Liu, Derek Jiang, Alexander Spangher, Muhao Chen, Jonathan May, and Nanyun Peng. 2024. Are large language models capable of generating human-level narratives? *arXiv preprint arXiv:2407.13248*.

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*.

Endel Tulving and Donald M Thomson. 1973. Encoding specificity and retrieval processes in episodic memory. *Psychological review*, 80(5):352.

Kaiwen Wei, Jingyuan Zhang, Hongzhi Zhang, Fuzheng Zhang, Di Zhang, Li Jin, and Yue Yu. 2024. Chain-of-specificity: An iteratively refining method for eliciting knowledge from large language models. *arXiv preprint arXiv:2402.15526*.

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*.

Peter West, Ximing Lu, Nouha Dziri, Faeze Brahman, Linjie Li, Jena D Hwang, Liwei Jiang, Jillian Fisher, Abhilasha Ravichander, Khyathi Chandu, et al. 2023. The generative ai paradox:"what it can create, it may not understand. In *The Twelfth International Conference on Learning Representations*.

Zhaofeng Wu, Linlu Qiu, Alexis Ross, Ekin Akyürek, Boyuan Chen, Bailin Wang, Najoung Kim, Jacob Andreas, and Yoon Kim. 2023. Reasoning or reciting? exploring the capabilities and limitations of language models through counterfactual tasks. *arXiv preprint arXiv:2307.02477*.

Jiancong Xiao, Ziniu Li, Xingyu Xie, Emily Getzen, Cong Fang, Qi Long, and Weijie J Su. 2024. On the algorithmic bias of aligning large language models with rlhf: Preference collapse and matching regularization. *arXiv preprint arXiv:2405.16455*.

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*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating large language models at evaluating instruction following. *arXiv preprint arXiv:2310.07641*.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. 2024. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Jiachen Zhao, Zonghai Yao, Zhichao Yang, and Hong Yu. 2023a. Self-explain: Teaching large language models to reason complex questions by themselves. *arXiv preprint arXiv:2311.06985*.

Zoie Zhao, Sophie Song, Bridget Duah, Jamie Macbeth, Scott Carter, Monica P Van, Nayeli Suseth Bravo, Matthew Klenk, Kate Sick, and Alexandre LS Filipowicz. 2023b. More human than human: Llm-generated narratives outperform human-llm interleaved narratives. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 368–370.

Shanshan Zhong, Zhongzhan Huang, Shanghua Gao, Wushao Wen, Liang Lin, Marinka Zitnik, and Pan Zhou. 2024. Let's think outside the box: Exploring leap-of-thought in large language models with creative humor generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13246–13257.

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

A Appendix

In the supplementary material, we describe more details about our methods and prompting strategy, present some examples from our CS4 benchmark, and discuss more experimental results, focusing on those obtained from instruction-based constraints.

A.1 Prompting Strategy for Constraints Generation

Creating the CS4 benchmark and evaluating different stories generated using the benchmark depends largely on the prompts given to the LLMs. Here, we detail the various carefully crafted prompts that primarily use few-shot prompting, reasoning, and LLM-as-the-judge prompting strategies. Qin et al. (2024) noted that on average, prompts with over 6.29 constraints (or requirements, as they call it), are hard for LLMs to satisfy comprehensively. To challenge LLMs in creative story writing, we thus start by providing LLMs with prompts containing 7 constraints, and in our CS4 benchmark, we progressively increase the number of constraints by 8, obtaining prompts with 7, 15, 23, 31, and 39 constraints. As elaborated in Section 2.2, we use two strategies for generating constraints, namely instruction-based and story-based constraints generation. While generating constraints solely based on an input instruction given to GPT-4, controlling the diversity of the generated constraints is crucial. Given no restrictions, GPT-4 generates constraints corresponding to different story genres for the same instruction. Such constraints depreciate the quality of the benchmark since it makes story writing extremely difficult. To overcome this, we restricted the genre of constraints to Realistic Fiction. For the same purpose, we prompted the model to avoid generating constraints that contradict one another. We also prompted the model to avoid generating constraints that require specific domain knowledge or using poetry-related terminology. Having such constraints would highly curb a model's ability to write creative, original stories. Finally, we prompted the model to come up with atomic constraints (Min et al., 2023b) to 1) ensure that prompts with more constraints are more specific and complex than those with fewer constraints, and 2) make evaluating stories generated using CS4 easy. We provided few-shot examples illustrating each of these requirements. Figure 5 provides a general framework of the prompt used to develop instruction-based constraints.

While generating story-based constraints, some of the above issues become irrelevant since the model generates constraints based on a story that already exists. For example, it is not possible to generate contradicting constraints or constraints about different genres from the same story. However, it is more likely that the model incorporates different proper nouns from the story into the constraints, copies specific events from the plot into the constraints, and maintains the flow of the plot in the constraints. All this would allow any LLM or story writer that uses the CS4 benchmark to copy the constraints one after another to generate a meaningful, coherent story. This could be considered "cheating" and would not allow one to appropriately evaluate the model for its creativity and coherence. To handle these issues, we prompted GPT-4 to generate constraints avoiding proper nouns and encouraging it to generate constraints that are non-linear with respect to the plot of the story and satisfy multiple lines of the plot rather than a highly specific event. Finally, to generate 39 diverse constraints, it is crucial to have stories that are long enough. If not, the model would start repeating constraints or generate more constraints that are style-related, than plot-related. To overcome this, we sorted the stories from the Writing Prompts dataset in decreasing order of length and picked stories that did not violate any of GPT-4's human-alignment policies. We provided few-shot examples illustrating each of these requirements. Figure 6 provides a general framework of the prompt used to develop story-based constraints. Figures 7 and 8 show some examples of constraints generated using the instruction-based and story-based strategies, respectively.

A.2 Prompting Strategy for Story Generation

Compared to the prompts for generating constraints in Appendix A.1, the prompts for story generation are straightforward. First, we prompt GPT-4 to generate a "base" story providing just the instruction as the input (Figure 9). This is done by setting the parameters of temperature, top_p (sampling strategy), and max_tokens to 0.8, 0.95, and 4096, respectively. The six 7B models are next tasked with modifying these base stories to satisfy different constraints (and hence prompts of different specificities) in CS4 (Figure 10). All stories are generated using the NVIDIA superpod-a100 GPU consisting of 12 nodes, and VLLM for efficient resource utilization, faster inference, and scalability.

Figure 5: General framework of the system prompt used to generate instruction-based constraints.

Figure 6: General framework of the system prompt used to generate story-based constraints.

A.3 Prompting Strategy for Evaluating Constraint Satisfaction

Figure 11 shows the framework of the prompt used to evaluate the stories for their constraints satis-

Figure 7: Example of instruction-based constraints along with the corresponding instruction.

Figure 8: Example of story-based constraints before and after accounting for atomicity, linearity, specificity, and proper nouns.

Figure 9: General framework of the system prompt used to generate the base story using GPT-4.

faction. Qin et al. (2024) suggests breaking down prompts into simpler criteria that can be easily evaluated to understand a model's instruction following ability. Inspired by this framework, we use a reasoning-based approach to compute the number of constraints satisfied by each story. For each con-

Figure 10: General framework of the system prompt used to modify the GPT-4-generated base story to accommodate different constraints.

straint, we ask the evaluator model to reason if a constraint is not satisfied. On the other hand, if a constraint is satisfied, we ask the evaluator model to print the sentences of the stories that adhere to the constraint. Finally, we prompt the model to print the total number of constraints the story satisfies.

A.4 Prompting Strategy for Evaluating Coherence and Story Quality

To evaluate the stories for their quality, we develop a comparative framework that compares the quality of a story in terms of likeability, grammar, and coherence, with another story. We use LLM-as-the-judge, prompting GPT-4 to provide a score out of 5 for the two stories under consideration based on likeability, grammar, and coherence, along with a reason for the scores (Figure 12). We then ask the model to pick a winner between the two stories for each category, and an overall winner based on all three categories. The obtained coherence scores are then normalized and used to plot the trade-off curves in Section 4.2. An example of such a pairwise comparison is shown in Figure 13.

However, we do not compare every unique pair of stories. Instead, as described in Section 3.2 we pick a story written using 23 constraints as the middle story with which all other stories are compared. We don't compare the middle story with itself. The story set as the middle story is compared 14 more times than any other story. So, we instead divide its coherence score by 14 to provide a fair comparison and normalized coherence score.

B Evaluation Details

This section briefly discusses additional results obtained upon evaluating the stories generated using the CS4 benchmark, focusing primarily on those obtained from instruction-based constraints. First, as the number of constraints increases, the LLMs find it harder to satisfy all of them while coming up with the stories. This essentially means that the model's instruction-following ability decreases as the specificity of the prompt increases. Figure 14 illustrates this result by measuring the average percentage of constraints satisfied across all stories generated from the CS4 benchmark in this work. It shows that the percentage of constraints satisfied by the models decreases with an increase in the number of constraints, i.e., an increase in prompt specificity. In this paper, GPT-3.5-Turbo refers to *gpt-3.5-turbo-0125*.

Additionally, as discussed in Section 3, evaluating the stories for both instruction-following and quality is crucial. This is because models tend to compromise one for the other when prompt specificity increases. Figure 15 shows a toy example of such a trade-off. Figures 3c and 16 depict this trade-off quantitatively for stories generated using instruction-based constraints. While it can be observed that these stories satisfy more constraints compared to those generated from story-based constraints because of the inherent nature of the generation process, the trade-off can still be observed clearly. Figure 3d and 16 further illustrate that while LHF is helpful in the presence of fewer constraints, its impact in leveraging useful data in the presence of several constraints decreases. The convergence of OLMo SFT and OLMo Instruct curves as the constraints become 31 or 39 evidence this.

Next, to perform error analysis, we considered all stories generated by the three OLMo models for prompts with 39 constraints. These represent the most complex inputs in CS4. From the 2 constraint generation strategies, 50 instructions in each, and 3 OLMo models, we obtain a total of 300 stories for evaluation. We manually inspected all stories corresponding to 5 different instructions for their constraint satisfaction, resulting in a manual inspection of 585 constraints (5 instructions $*$ 39 constraints $*$ stories from 3 OLMo models). We then uploaded all 300 stories, their corresponding constraint satisfaction results, and our observations from manual inspection to the OpenAI website asking GPT-4 to refine our findings with some examples. After

```
System Prompt: You are an expert reader. I will give you a story followed by a set of
constraints.
Your task is to carefully read both of them and tell how many constraints are being
satisfied in the story.
As the output, print yes/no for each constraint based on whether it is being satisfied
or not, followed by a 1 line explanation of why it is being satisfied/violated.
In case a constraint is being satisfied, print the sentence/line from the story in
which it is being satisfied. If a constraint is not being satisfied, give an
explanation of how it is being violated. Be very strict in your evaluation.
Mark a constraint as satisfied ("yes") only if it is being completely satisfied in the
story. For no satisfaction/partial satisfaction, mark a "no".
Finally, print the number of constraints that are being satisfied.

Here are some examples: -
<Example 1>:
Story: - The crew of the Depth Reaver were charting a course through the...
Constraints: -
1. Write a story based on the following constraints in less than 377 words.
2. Start the story with the sentence: "Week 18 aboard the Depth Reaver, Circa 2023"
3. Include a revelation of an unexpected large-scale phenomenon observed in space."
Output -
1. Yes - The story is 302 words long, meeting the constraint of being less than 377
words.
2. No - The story starts with the sentence: " The crew of the Depth Reaver…" which does
satisfy the given constraint.
3. Yes - The revelation of the moon cracking open to reveal a colossal human face
qualifies as an unexpected large-scale phenomenon observed in space.
Number of constraints satisfied: 2
<Example 2>:
...

<User input: Story and constraints>
```

Figure 11: General framework of the system prompt used to evaluate stories for constraint satisfaction.

verifying the examples, we concluded our analysis in Section 4.6. Figure 19 illustrates these results. It depicts the number of constraints satisfied by different combinations of OLMo models, with the Instruct model satisfying the greatest number of constraints.

As the prompt specificity increases, two opposing forces could affect the perplexity and diversity of the stories. With more constraints, LLMs have fewer training examples to leverage, resulting in fewer options to generate the next word at any given instance, leading to a low perplexity and diversity score. On the other hand, with more constraints, LLMs are forced to generate more novel stories which would reflect as high perplexity and diversity scores. They could also generate stories with high perplexities by choosing to satisfy only a few of the several constraints in the prompt. To study which of these forces is stronger, it becomes important to compute perplexities and diversity scores for each story. Perplexities are calculated using the same model that generated the story. To analyze the dist-n diversity of the models, we generated three

sets of outputs for each input prompt and computed the diversity across these generations. This resulted in a generation of 4500 stories (500 prompts $* 3$ LLMs $* 3$ stories $= 4500$ stories. While Figures 4a, 4b, and 4c show perplexity and diversity results for story-based constraints, Figures 17 and 18 show the corresponding results for the instruction-based constraints.

Finally, Table 2 denotes the QUC_{39} and $RCS_{7,39}$ scores for stories generated using 7 and 39 instruction-based constraints. These essentially denote the stories generated using the simplest and most complex prompts from CS4. Here, the story length indicates the number of words in the story which is computed by splitting each story into words using white spaces. In Table 3, we average the results from Table 1 and 2 to present a more holistic view of how models perform irrespective of the constraint-generation strategy. The results corresponding to all stories developed from all levels of prompt specificities are shown in Figure 20. While OLMo Instruct can be seen to have the steepest decrease in terms of the QUC_{39} score,

```
System Prompt: You are an English writing expert and you can compare and evaluate story
essays on these metrics with the following definitions -
1. Grammar: Which story has better writing and grammar comparatively?
2. Coherence: Which story has a better logical flow and the writing fits together with
respect to the plot?
3. Likability: Which story do you find more enjoyable to read?
You will be given two Stories - Story A and Story B. Add a rating out of 5 for each
category, specify which story you prefer for each metric by providing a one line reason
for your preference.
For each category, provide a category winner story as the letter "A" or "B", based on
the ratings.
Finally, assign an overall winner story as the letter "A" or "B" based on the ratings
and category wins.

EXAMPLE OUTPUT 1:
Grammar Preference: A
A - 5/5: Story A has a few minor grammatical issues, but overall, it demonstrates
strong control of language.
B - 4/5: Story B is well-written but has slightly more noticeable issues in grammar and
sentence structure.
Coherence Preference: A
A - 4.5/5: Story B has a strong coherence, effectively conveying the emotional journey
and the progression of events.
B - 4/5: Story A maintains a consistent and engaging narrative flow, though some parts
are a bit abstract.
Likability Preference: A
A - 4/5: Story B's realistic and emotional narrative is likely to resonate more with a
wide range of readers.
B - 3.5/5: Story A is imaginative and intriguing, but its abstract nature might not
appeal to all readers.
Overall Winner: A
<EXAMPLE OUTPUT 2:>

<User Input: Story A, Story B>
```

Figure 12: General framework of the system prompt used to compare two stories in terms of coherence, grammar, and likability.

Figure 13: Example of a single pair-wise comparison for evaluating story quality.

Mistral and Gemma show relatively consistent performance across different prompt specificities in the CS4 benchmark.
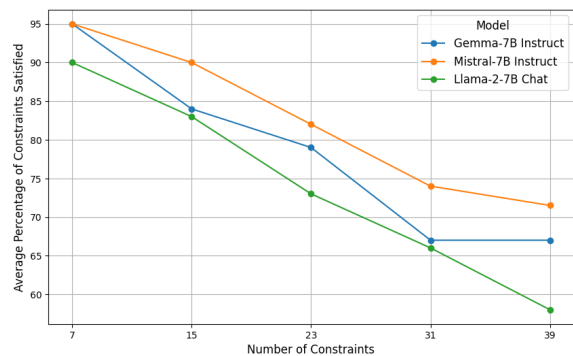
Figure 14: Percentage of constraints satisfied by storied generated by different LLMs.

C Grammar and Likability Scores

Finally, we briefly explain the results obtained while evaluating the stories for their grammar quality and likability. These metrics are computed with coherence using the same pipeline. Figure 21 shows the normalized grammar scores for stories written using different models with the CS4 bench-

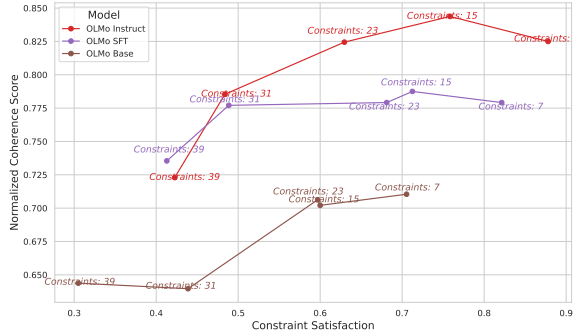Figure 15: An example of a trade-off between constraint satisfaction and coherence.

Figure 16: Trade-off results for OLMo Base, OLMo SFT, and OLMo Instruct models for instruction-based constraints.

Model	QUC$_{39}$	RCS$_{7,39}$ ↓	Median Length	Self Perplexity
Gemma-7B Instruct	0.5933	0.2273	**533**	3.7096
Mistral-7B Chat	**0.6049**	**0.2131**	818	1.4958
LLaMA-2-7B Instruct	0.4814	0.2637	585	2.3285
OLMo-7B Base	0.1963	0.3047	374	**3.9865**
OLMo-7B SFT	0.3036	0.3363	542	3.6753
OLMo-7B Instruct	0.3054	0.4188	826	3.6151

Table 2: Overall comparison of different LLMs on instruction-based constraints. Smaller RCS indicates higher creativity. Models adhering closely to the 500-word limit in our instruction show better compliance and higher self perplexities mean more diverse outputs. Both metrics are computed across all the generated stories. The best values are highlighted.

Model	QUC$_{39}$	RCS$_{7,39}$ ↓	Median Length	Self Perplexity
Gemma-7B Instruct	0.5350	**0.1902**	**510.5**	3.6858
Mistral-7B Chat	**0.5384**	0.2216	737.5	1.5396
LLaMA-2-7B Instruct	0.4275	0.2783	549	2.3077
OLMo-7B Base	0.1975	0.2427	452.5	**3.8302**
OLMo-7B SFT	0.3150	0.2779	557	3.6462
OLMo-7B Instruct	0.3377	0.3792	839.5	3.6079

Table 3: QUC$_{39}$, RCS$_{7,39}$, story length, and self-perplexity scores averaged across story-based and instruction-based constraints.

mark. Similarly, Figure 22 depicts the normalized likability scores. These figures generally indicate
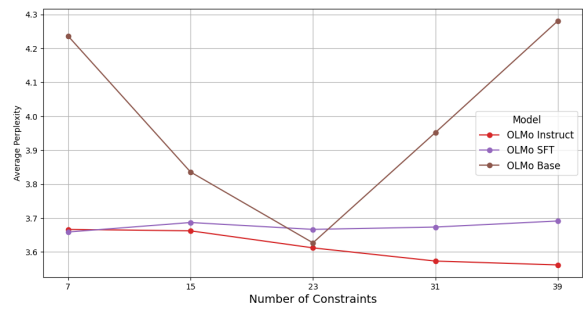
that these scores do not vary drastically as the number of constraints increases. While one can see a slight dip in these scores with increased prompt specificity in many cases, these dips are not as significant as the trade-off between constraint satisfaction and coherence. This suggests that models produce stories of similar grammar and likability properties irrespective of the prompt specificity, indicating that these metrics are associated with the model architecture rather than the prompt. For example, one can observe that Mistral, Gemma, and LLaMA-2 always produce competitive grammar and likability scores while OLMo SFT and OLMo Instruct outperform OLMo Base. This means that while LHF can not help produce highly coherent stories in the presence of several constraints, they can still alleviate the model's performance in terms of grammar and likability.

D Author Contributions

- **Anirudh Atmakuru**: A major contributor to constraint generation, LLM evaluation, error analysis, evaluating constraints satisfaction, and paper writing.

- **Jatin Nainani**: A major contributor to defining the project direction, designing evaluation methodologies, performing experiments, and developing the analysis code.

- **Rohith Siddhartha Reddy Bheemreddy**: A major contributor to paper writing, code development, and the creation of story generation and LLM evaluation systems.

- **Anirudh Lakkaraju**: A major contributor to constraint generation, paper writing, and code releasing.

- **Zonghai Yao**: An advisor who contributes to the brainstorming and paper writing.

- **Hamed Zamani**: The faculty advisor who contributes to the paper writing.

- **Haw-Shiuan Chang**: The lead supervisor of the work who proposes the ideas, manages the team, and writes the paper.

(a) Perplexity scores for Mistral, Gemma, and LLaMa-2 on instruction-based constraints.

(b) Perplexity scores for OLMo Base, SFT, and Instruct on instruction-based constraints.

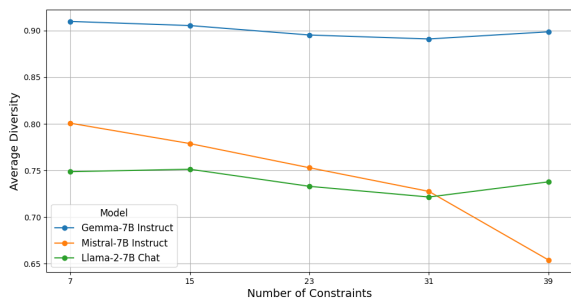Figure 17: Perplexity scores for stories developed using instruction-based constraints.

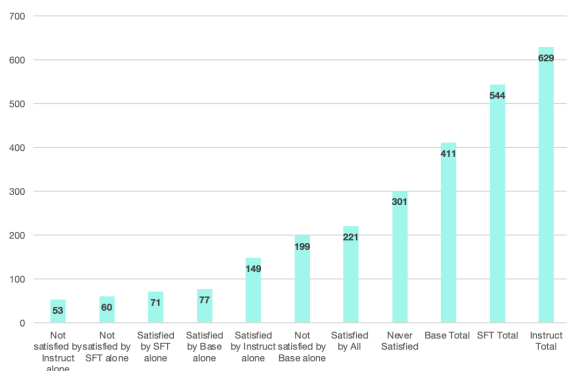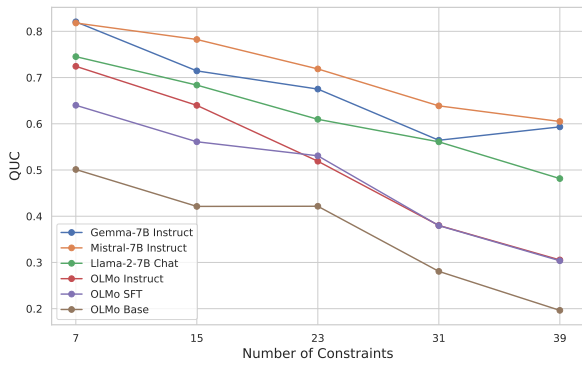Figure 18: Diversity scores for stories developed using instruction-based constraints.

Figure 19: Error analysis for stories developed using story-based prompts with 39 constraints.

(a) QUC scores for stories developed using instruction-based constraints.

(b) QUC scores for stories developed using story-based constraints.

Figure 20: QUC scores comparison for stories developed using instruction-based and story-based constraints in CS4 benchmark.
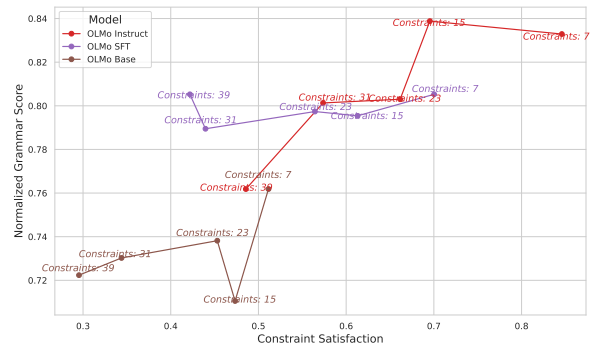

(a) Grammar scores for stories generated using Gemma, Mistral, and LLaMA-2 using instruction-based constraints.

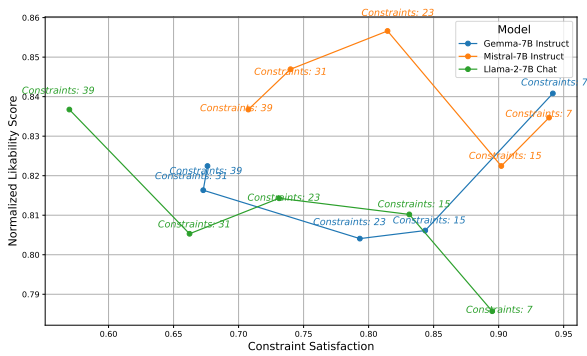(b) Grammar scores for stories generated Gemma, Mistral, and LLaMA-2 using story-based constraints.

(c) Grammar scores for stories generated using OLMo Base, SFT, and Instruct using instruction-based constraints.
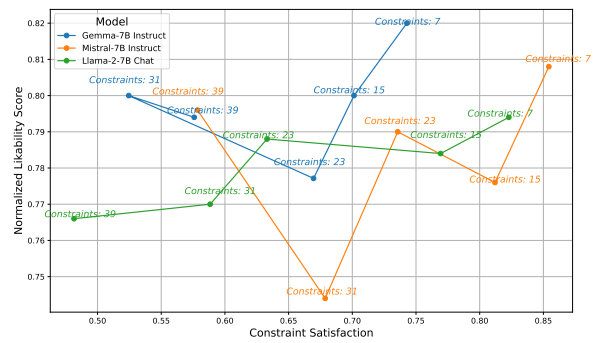
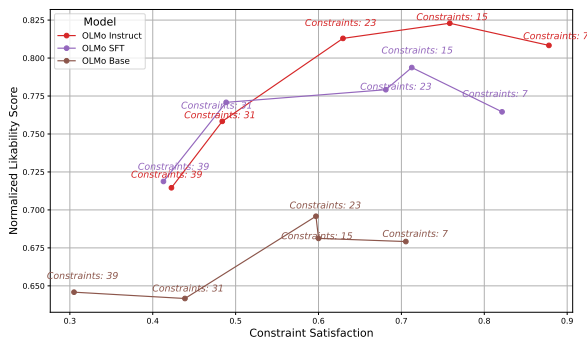(d) Grammar scores for stories generated using OLMo Base, SFT, and Instruct using story-based constraints.

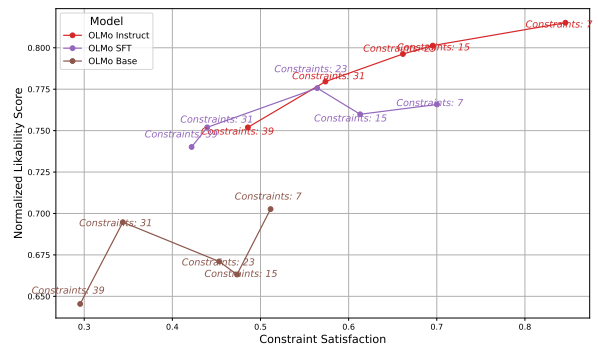Figure 21: Grammar scores for stories generated using CS4 benchmark.

(a) Likability scores for stories generated using Gemma, Mistral, and LLaMA-2 using instruction-based constraints.

(b) Likability scores for stories generated using Gemma, Mistral, and LLaMA-2 using story-based constraints.

(c) Likability scores for stories generated using OLMo Base, SFT, and Instruct using instruction-based constraints.

(d) Likability scores for stories generated OLMo Base, SFT, and Instruct using story-based constraints.

Figure 22: Likability scores for stories generated using CS4 benchmark.