LLMs Are Not (Yet) All You Need for Stochastic World Models

Anonymous ACL submission

Abstract

Large language models (LLMs) have demonstrated impressive capabilities in processing and generating human-like text, positioning them as potential candidates for "world models" that can simulate real-world processes. However, extending their reasoning to model complex probabilistic processes remains an open challenge. In this work, we investigate the ability of LLMs to model stochastic processes, ranging from simple Markov processes to more complex non-Markovian and Poisson processes. We find that LLMs consistently fail to model stochastic distributions when the parameters are specified through prompting alone, even in simple cases like biased coin flips or die rolls. While we show that techniques such as chain-of-thought (CoT) reasoning and in-context learning (ICL) lead to marginal performance improvements, they remain insufficient for accurately modeling complex stochastic phenomena. Towards addressing these challenges, we introduce a new benchmark, TOSS-v1, for evaluating LLMs' performance in modeling random processes, aiming to inspire future improvements in mechanisms for more accurate stochastic process modeling.

1 Introduction & Background

World models are computational frameworks that aim to simulate the complexities of real-world environments, enabling AI systems to represent, predict, and reason about complex, real-world environments (Allen and Koomen, 1983; Ha and Schmidhuber, 2018). These models are fundamental in fields like robotics (Polydoros and Nalpantidis, 2017) and cognitive science (McClelland, 2009) where they provide a way for agents to interpret their surroundings, anticipate outcomes, and plan actions based on simulated scenarios. Recent work has touted the capability of large language models (LLMs) to go beyond their impressive capabilities in processing and generating text and act as such

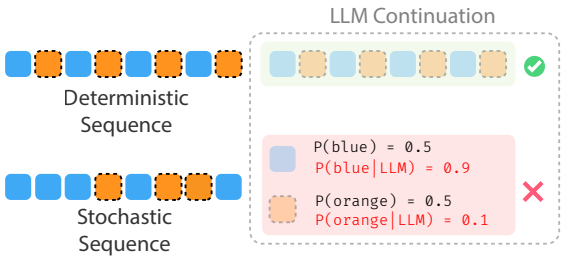


Figure 1: How well do LLMs act as world models in stochastic worlds? While recent work has shown that LLMs are capable pattern matchers for deterministic patterns, in this work, we find that LLMs struggle to model simple stochastic processes, even with techniques such as Chain of Thought (CoT) and In-Context Learning (ICL).

world models in domains including simulations in robotics (Dagan et al., 2023; Song et al., 2024), driving (Zhao et al., 2024), courtrooms (Chen et al., 2024a), human behavior (Aher et al., 2023; Park et al., 2023; Moon et al., 2024; Axtell and Farmer, 2022; Argyle et al., 2023; Loyall, 1997) and the scientific process (Shojaee et al., 2024) among others (Ge et al., 2024; Yang et al., 2024; Nottingham et al., 2023; Xie et al., 2024).

Despite modest improvements in performance, particularly in pattern matching (Mirchandani et al., 2023), LLMs have often failed comprehensively represent environments. For example, in robotics, LLMs can struggle in planning in complex and higher-dimensional situations (Kambhampati et al., 2024; Verma et al., 2024; Bohnet et al., 2024; Valmeekam et al., 2024; Guan et al., 2023). In general, LLM-based world models have struggled with over-fitting to data (Kapoor et al., 2024), long-tailed knowledge (Liu et al., 2024a) and adversarial attacks (Liu et al., 2024a). Even more important for "world-modeling," LLMs struggle to learn causal reasoning (Kıcıman et al., 2023; Kosoy et al., 2022) and stochasticity (Hopkins et al., 2023; Van Koevering and Kleinberg, 2024; Liu et al., 2024b).

These shortcomings in planning and reasoning point to a deeper, more fundamental challenge:

LLMs cannot manage dynamic, uncertain situations (Kambhampati et al., 2024). Uncertainty is a key element of many systems, whether in robotics, finance, or other AI applications. Engaging with unpredictable environments requires more than deterministic methods; it calls for a strong grasp of stochastic process modeling and knowing when to balance language modeling with world modeling (Chen et al., 2024b). Such an understanding of stochastic processes is essential for current LLM applications that involve time-series forecasting or direct stochastic sampling (Xu et al., 2024; Liu and Quan, 2024; Gruver et al., 2024; Jin et al., 2023).

Understanding how models understand and use stochastic phenomena is an under-explored area of natural language processing. Several works have taken a sampling-based approach to quantifying LLM performance in approximating simple random distributions. Hopkins et al. (2023) use a sampling process to determine how well models can sample from fixed distributions, focusing on uniform random number generation and probabilistic context-free grammars, demonstrating that LLMs often struggle to produce low-error, consistent distributions, and show that non-auto-regressive sampling can perform better than auto-regressive sampling when approximating stochastic phenomena. Interestingly, Van Koevering and Kleinberg (2024) show that LLMs, including GPT-4 and Llama-3, replicate and even amplify human biases when generating random sequences. Recently, Liu et al. (2024b) demonstrated that in certain situations LLMs can replicate Markov chains specified *implicitly* through in-context learning, however, they do not explore to what extent natural language inputs affect the generation process.

In this paper, we aim to (1) explore whether, and how, LLMs can model stochastic processes, and (2) examine whether existing methods for chain-of-thought reasoning and in-context learning techniques can mitigate inefficiencies in this modeling. Previous work (Hopkins et al., 2023; Van Koevering and Kleinberg, 2024) has primarily focused on sample-based estimates of probability for limited classes of stochastic phenomena. In contrast, we broaden the scope to cover a wider range of processes. These include simpler, discrete, and independent stochastic events—such as coin flips and die rolls, where each outcome has a fixed probability and is independent of prior events—as well as more complex processes, like Markov Chains, where states depend on prior states, and Poisson processes, which describe the occurrence of rare events over time.

Our primary contributions are as follows: First, we demonstrate that LLMs consistently fail to model simple stochastic distributions when the parameters are provided solely through prompting for basic multinomial distributions, such as biased coin flips and die rolls, and even with the assistance of chain-of-thought (CoT) prompting. We then show that example-based prompting techniques, such as in-context learning (ICL), provide partial improvements but are insufficient to fully address these challenges, particularly in more complex stochastic processes like Poisson distributions and Markov decision processes. Next, we assess the impact of long-context prompts on decision-making and demonstrate that the placement and depth of decision descriptions within the context can impact model performance. Finally, to facilitate progress in stochastic modeling with LLMs, we introduce the TOSS-v1 benchmark, which provides a systematic evaluation of LLM performance across stochastic modeling tasks. Using this benchmark, we show that instruction-tuned models are generally less capable of modeling stochastic processes than their non-instruction-tuned counterparts.

2 Measuring Stochastic LLM outcomes

We take a two-stage approach to answer the question of how well LLMs model stochastic processes. In the first stage, we prompt an LLM with the description of a stochastic process (e.g., the definition of a multinomial distribution), and in the second stage, we ask that the LLM draw a sample from that process. We then compare the LLMs model of the process to the (known) true random distribution.

2.1 Preliminaries

We focus on open-source language models, and extract stochastic representations directly from the underlying learned model distributions. For a sequence of tokens, $x = \{x_1, x_2, ... x_n\}$ in a vocabulary V (of size $|V|$), a large next-token prediction-based language model, $\mathcal{M}$, approximates a probability distribution over the next token: $P_{\mathcal{M}}(x_{i+1}|x_i, ..., x_1)$.

To evaluate stochastic processes, for each process we define a fixed set of possible "outcomes" that a sample from the process can take. Formally, each outcome $o \in \Omega = \{o_1 ... o_k\}$ is a sequence of

tokens corresponding to a string value (for example, when flipping an coin, the outcomes are "heads" and "tails", corresponding to token sequences `[_heads]` and `[_t,ails]`). For each outcome, we then aim to compute $P_{\mathcal{M}}(o|\text{prompt})$, where the prompt is a sequence of tokens that both (1) describes the process and (2) asks for a sample. While several works estimate this probability by sampling (Hopkins et al., 2023; Van Koevering and Kleinberg, 2024), we found that sampling was often unreliable, and thus, we extract this distribution directly from the language model as:

$$P_{\mathcal{M}}(o|\text{prompt}) = \prod_{i=1}^{k} P_{\mathcal{M}}(o_i|o_{i-1},\ldots,o_1,\text{prompt}) \tag{1}$$

Note here that for multi-token sequences, we compute the probability conditioned on picking the correct token, and we assume that there is only one unique generator for the sequence o. Because these outcomes are a subset of all of the potential token sequences generated by the LLM, we re-normalize the distribution over the support of the options. See subsection C.1 for more details about the re-normalization process.

In this paper, we primarily measure the total variation distance (TVD) between the true distribution $P^*(o)$ and the normalized model distribution $\hat{P}_{\mathcal{M}}(o)$ over the support Ω:

$$\delta(P^*,\hat{P}_{\mathcal{M}}) = \frac{1}{2}\sum_{\omega\in\Omega}\left|P^*(\omega) - \hat{P}_{\mathcal{M}}(\omega)\right| \tag{2}$$

The TVD is an intuitive distance measure, which arises as the optimal transport cost between the distributions given a unit cost function. When the TVD is high, the distributions are quite different, and when it is zero, the distributions are identical.

In this paper, we explore the performance of several models including Gemma-2 (Team et al., 2024), Phi-2/Phi-3.5 (mini) (Abdin et al., 2024), Llama-3.1 (8B) (Dubey et al., 2024), Mistral 7B (Jiang et al., 2023) and OLMoE (7B) (Muennighoff et al., 2024) along with their instruction tuned variants. For more details on the models, see Appendix D.

2.2 Multinomial Distributions

Multinomial distributions model counts from N independent trials over Categorical distributions $P(X=i)$. We evaluate LLM's capacity to model two common multinomial distributions, coin flips and die rolls. For each scenario, we ask the LLM to model an unbiased random process (e.g. fair coins) as well as biased variants (e.g. unfair coins) by measuring the output token probability distributions over a closed set of options for each process.

We define several prompts for modeling each of the multinomial cases: (1) **Coin Flip:** The LLM is tasked with completing the prompt (or a similar prompt to) "`I flipped a coin and it landed on`", over the set {heads, tails}, and (2) **Die Roll:** "`I rolled a die and it landed on`" over the choices of one through six. For biased variants, we provided explicit biasing statements within prompts to the model such as: "When I flip coins, they land on heads X% of the time," where X is a percentage between 0% and 100%, or "When I roll dice, they land on N X% of the time."

Results from the coin flip experiment are presented in Figure 2. Non-instruct LLMs ignored the bias in their input prompts, with their token probabilities generally showing a tendency to ascribe $\approx$ 60%-80% probability to heads, independent of context. Instruct models performed slightly better, though they still exhibited a bias toward heads. Additionally, instruct models showed improved performance at the extremes of bias values, with TVD values dropping for 0% and 100% heads biases.

Results from the die roll experiment are presented in Figure 3. For each bias percentage, we averaged results across the six die faces and 50 prompt variants, totaling 300 trials per bias percentage. Non-instruct models generally performed better than their instruct counterparts, and best around a 50%-60% bias, struggling more with higher biases. Instruct model performance was more varied, with some models showing little change in behavior and others improving as the bias value increased.

2.3 Markov Chains

We evaluate the ability of LLMs to model Markov chains (Chung, 1967) within their token distributions. A Markov chain is defined as a tuple (S,P), where S is the state space and $P(s,s')$ is a transition distribution from states s to s' in S. In each of our experiments, we define a transition matrix $\mathcal{M}$, where $\mathcal{M}[i,j]$ represents $P(s_i,s_j)$. N trajectories of length k are sampled from $\mathcal{M}$, where each state is represented by the token ID from an LLM's tokenizer (e.g. state '1' is represented as token ID 23574). We provide the LLM with one trajectory

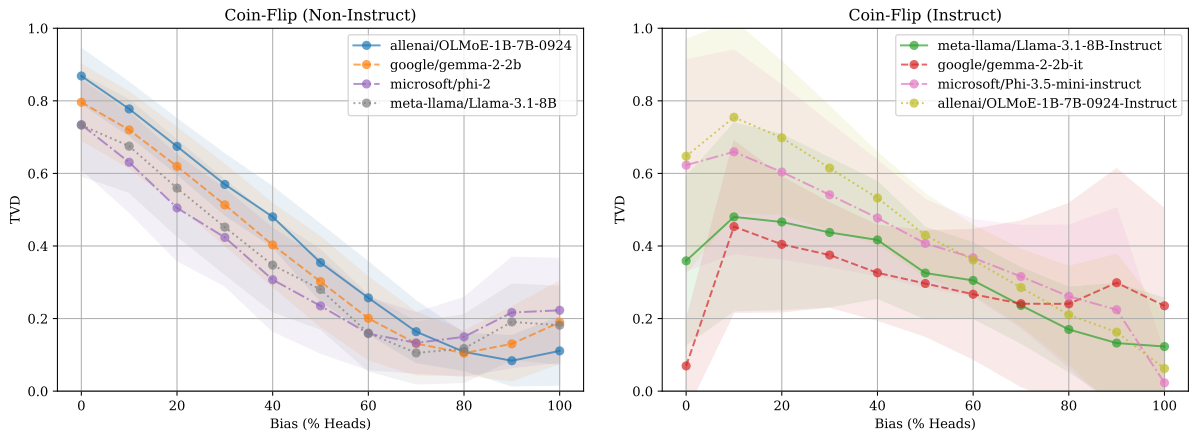


Figure 2: **Biased Coins:** Plots of mean total variation distance (TVD, ↓) against bias percentage for non-instruct (left) and instruct (right) models when aggregated across prompts (N=50) for the biased coin flip experiment. Shaded areas show one standard deviation. While non-instruct models both (1) ignore biasing instructions in the prompts and (2) almost always generate a biased distribution ($\approx 70\%$ heads), instruct-based models pay better attention to biasing information, and perform significantly better when modeling extreme bias (Always generating heads/tails).

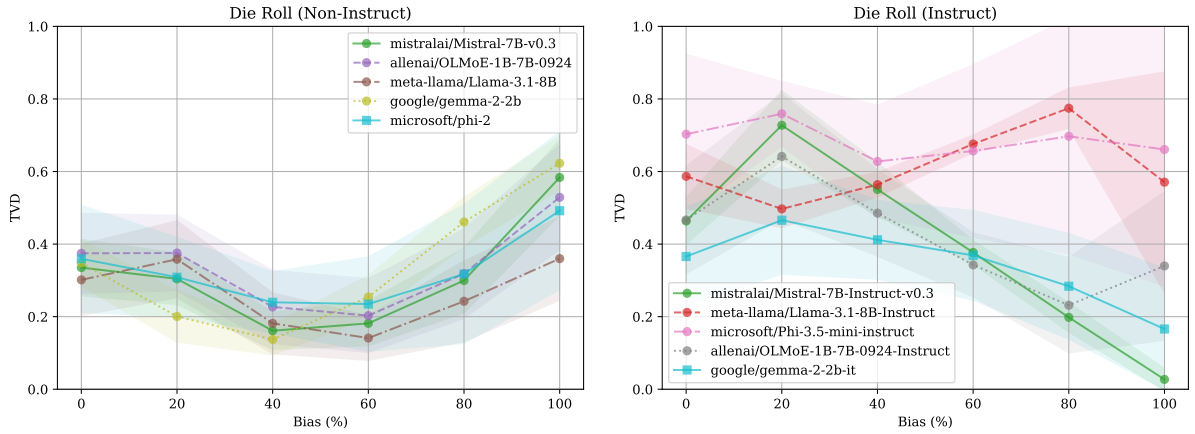


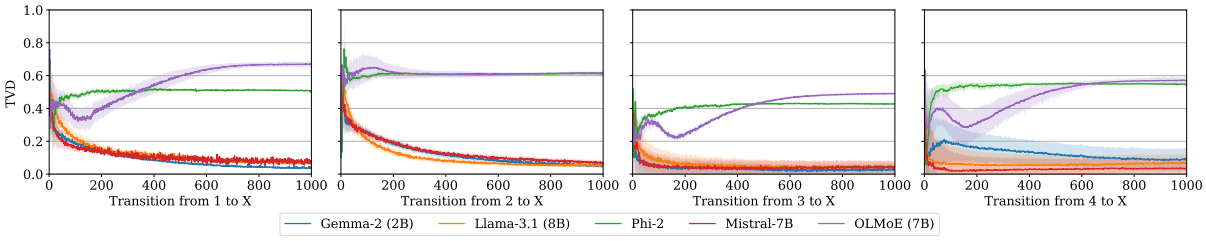
Figure 3: **Biased Die Rolls:** Plots of mean total variation distance (TVD, ↓) against bias percentage for non-instruct (left) and instruct (right) models when aggregated across prompts (N=50) for the biased die rolling experiment.

at a time over the entire dataset. For each state s_t in the sequence, we measure the normalized conditional probabilities $P(s'|s_t)$ from the LLM and compute its TVD to the corresponding row $\mathcal{M}[t]$ in the transition matrix. When a transition for a given state is observed at the same index across trials, they are averaged together.

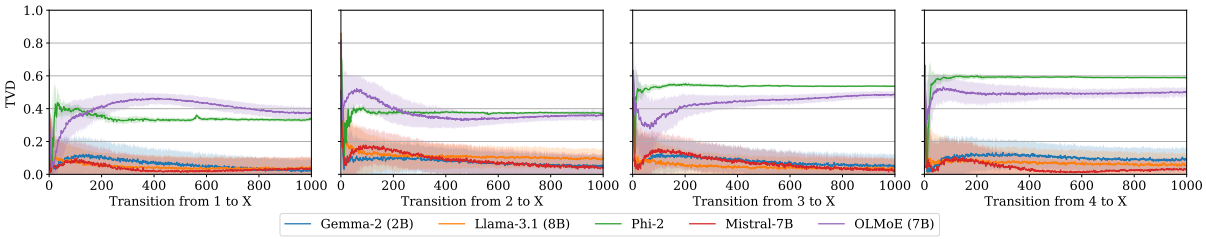
In our experiments, we utilize 4×4 transition matrices defining transition probabilities between states s_{1-4}. We vary the transition probabilities to probe various, interesting types of Markov chains. In each experiment, we assess the TVD of the sequence generated by the LLM as a function of sequence length. Conceptually, TVD should decrease as the sequence the LLM is exposed to gets longer. Most LLMs tested are able to recover the underlying transition matrix that trajectories are generated from as the sequence length they are conditioned on gets longer. We construct four experiments by specifying transition matrices $\mathcal{M}$ as follows (See Appendix F for exact definitions for each type of transition matrix.):

Simple chain We define a simplistic transition matrix that contains some missing connections between states, a low probability of self-transitions, and an "end" state that is hard to leave. As shown in Figure 4a, most LLMs are able to recover the underlying probabilities However, Phi-2 and OLMoE notably deviate from this trend.

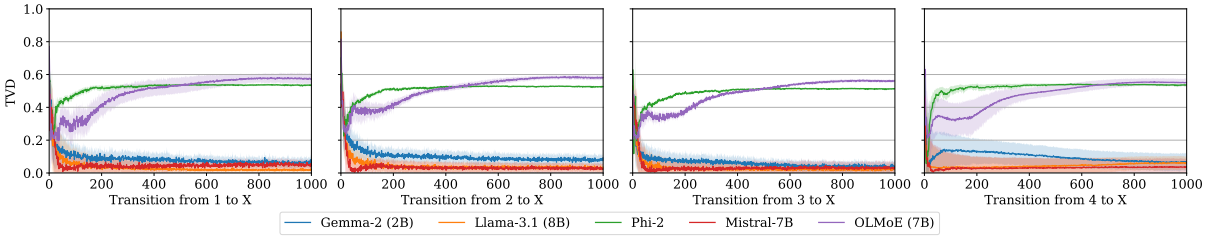
Quicksand chain Like walking in quicksand, this transition matrix places high probability on self-transitions and gets "stuck" in every transition. Trajectories should demonstrate long runs of the same state. Figure 4b demonstrates that almost all LLMs, besides Phi-2 and OLMoE, successfully



(a) Simple chain



(b) Quicksand chain



(c) Black hole chain

Figure 4: **Markov chains:** Plots of total variation distance (TVD, ↓) as a function of sequence length.

generate long, repeated subsequences of states as would be expected in this setup.

Black hole chain In this experiment, all states transition with high probability to the last state, which then transitions to itself with high probability. Ideal trajectories would have long runs of s_4. Similarly to the quicksand chain experiment, Figure 4c shows that almost all LLMs largely succeed at this task and appropriately generate long subsequences with s_4. Phi-2 and OLMoE both fail to perform well at this task.

Increasing "entropy" We construct a series of transition matrices that are modulated via the addition of uniform noise as follows:

$$(\mathcal{M}+uN)_+, N \sim U[-1,1] \tag{3}$$

In effect, u is a uniform transition matrix that becomes more "modal" as u approaches 1. We demonstrate results for Llama 3.1-8B in Figure 5 (with results for all models in Appendix F). The uniformity of the transition matrices impacts the ability of the LLM to recover the underlying transition probabilities demonstrated by the trajectories in the experiment. A surprising result is the inability of LLMs to achieve near-perfect performance on $u=0.0$, a result which mirrors findings in subsection 2.2.

2.4 Poisson Distributions

Poisson distributions, $X \sim \text{Pois}(\lambda)$, are discrete probability distributions that model how likely it is for an event to occur k times within a fixed time interval, where event occurrences are independent of each other. The parameter $\lambda = \text{E}(X)$ represents the average number of times the event occurs.

In these experiments, we deviated from extracting probabilities over a predefined set of choices and instead sampled sequence completions from LLMs. In each trial, the LLM was provided with an in-context learning (ICL) prompt containing a story with a special character inserted between words at a rate following a Poisson distribution $\text{Pois}(\lambda)$ (see Appendix E for more details) and was tasked with generating the next 100 tokens. We computed the average number of occurrences of the special character in the LLM's output and measured the L1 distance between this and the provided λ. Distributions were parameterized with $\lambda \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. We varied ICL context lengths and used a set of 10

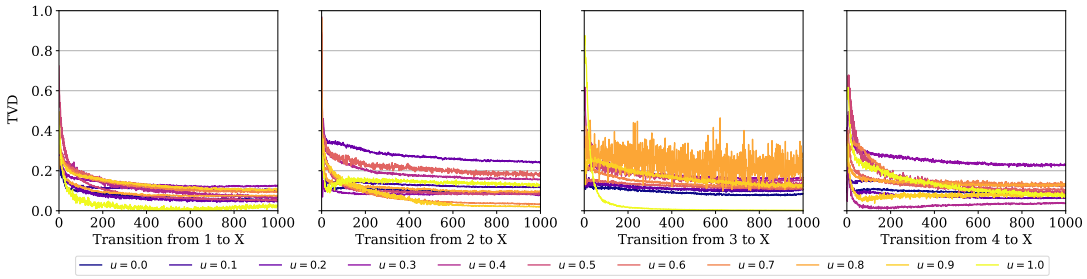


Figure 5: **Transition matrix distribution affects LLM performance.** Despite gradual changes in the uniformity of the underlying transition matrix that trajectories are sampled from, Llama-3.1 presents high sensitivity to the way transition matrices are constructed.

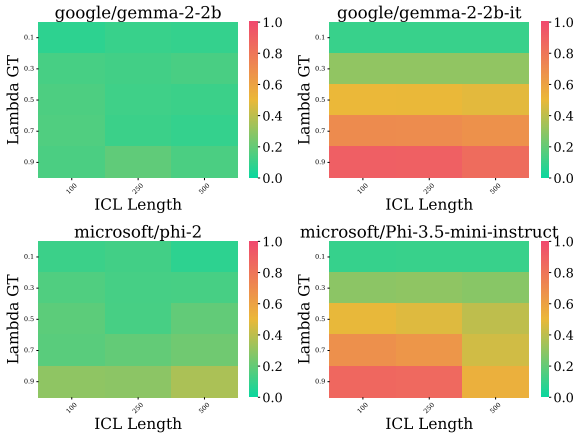


Figure 6: **Poisson In-Context Learning**: L1 error between the λ used to parameterize a Poisson distribution and the λ extracted from LLM's completion. We find that instruction fine-tuned LLMs struggle to continue generation under the appropriate Poisson distribution while their base variants perform well. For more models, see Appendix E.

special characters (e.g., '@').

We present the results in Figure 6. Non-instruct models were able to achieve very low insertion rate error, even with small ICL context windows. Interestingly, non-instruct variants generally struggled with this task, and performance degraded as λ increased, suggesting that they tended to insert fewer special characters into their completions.

3 Reasoning Techniques

In this section, we explore if leveraging reasoning techniques such as Chain-of-Thought (CoT) reasoning or In-Context Learning (ICL) can improve stochastic modeling performance.

Chain of Thought (CoT) Reasoning: In this experiment, we augment the distribution with a reasoning sentence that explains the phenomenon of the distribution. For example, on the coin flip experiment, we augment the original prompt "When I flip coins they land on heads 20% of the time." to 'When I flip coins they land on heads 20% of the time. Let's think about what this means for my distribution of coin flips, it suggests that if I flip the coin 10 times, I will see heads approximately 2 times." (similarly on other distributions). We fix the completion prompt to "`I flipped a coin and it landed on`" and task the LLM to complete this prompt based on the CoT-augmented input. We provide the results of coin flip and die roll experiments in Figure 10, where we calculate the TVD difference between CoT and non-CoT over the different biases. We observe that CoT doesn't yield a consistent improvement over the bias distribution. Additionally, we find that adding CoT changes the performance of the instruct models over different biases, while almost having no impact on std models. We provide more details in Appendix H.

In-Context Learning (ICL): In this experiment, we evaluate LLMs' ability to model multinomial distributions when provided with in-context examples. Specifically, rather than explicitly specifying the underlying distribution as in Section 2.2, we implicitly specify it by providing the LLM with a sequence of samples from the distribution in its prompt (e.g., "I flipped a coin and it landed on heads, then on tails, then on tails, then on tails, then on" for a coin biased toward tails). We then assess the probability of the outcomes under each model after presenting these ICL prompts.

Figure 8, shows results from the coin flip experiment on Llama-3.1-8B (see Appendix I for results from other models), where performance generally improves as more examples are added. Although the model's approximations become fairly accurate after 100 examples, it is unable to completely close the gap. Figure 9 shows a comparison between the instruct and non-instruct variants of Llama-3.1-8B (see Appendix J for other models). While both variants benefit from increasing numbers of

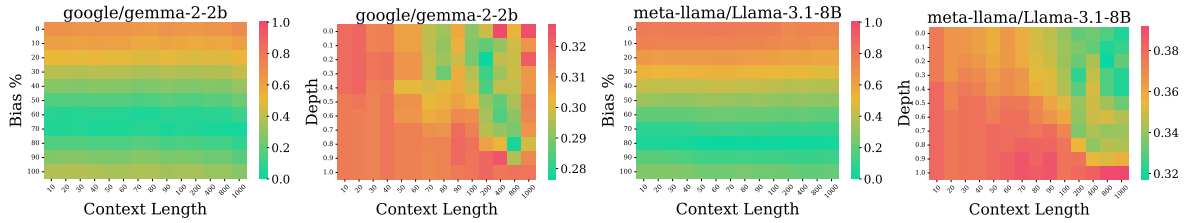


Figure 7: **Random Needle in a Haystack:** Total variation distance for Gemma-2 (Left) and Llama-3.1 (Right) on biased-coin experiments for both needle depth and bias percentage across several context lengths. In general, while context length and bias appear to be unrelated in terms of model performance, the depth of the needle displays interesting effects, with models preferring long context lengths, but biasing needles close to the beginning of the sentence.

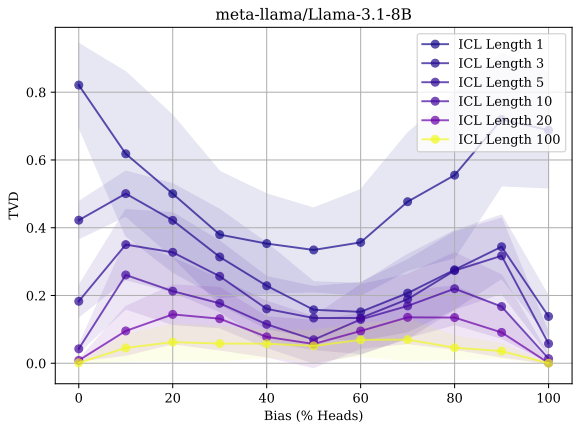


Figure 8: **Biased Coins + ICL:** Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths for the Llama-3.1-8B model. As the number of in-context samples increases, the performance of the model at modeling the stochastic process improves as well. Notably, adding as few as 3 in-context examples significantly improves performance, but even adding 100 in-context examples does not fully allow the model to capture the biased distribution. For other models, see Appendix I.

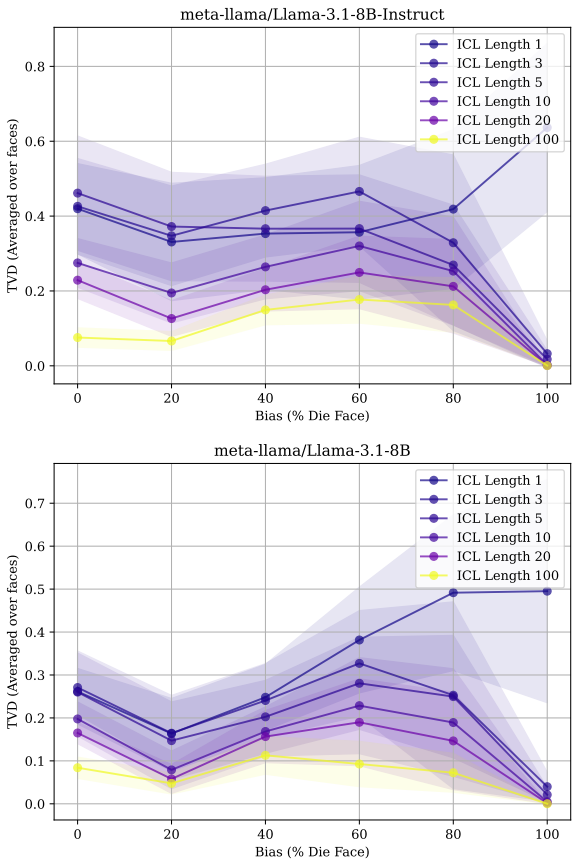


Figure 9: **Die Rolls + ICL:** Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths for the Llama-3.1-8B instruct (top) and non-instruct (bottom) variants on die rolls. Non-instruct variants generally benefit more from ICL than instruct variants. Interestingly, all models perform best when the die is completely biased towards a given head. For other models, see Appendix J.

examples, the non-instruct variant benefits more and generally exhibits better performance.

4 Long-Context Causal Conditioning

Sometimes the definitions of the stochastic process are separated in token space from the time that the stochastic process is used/sampled from. In this case, we would like LLM-based world models to be capable of discovering conditioning signals buried within a longer context, and use them to guide the generation of new text. To evaluate this capability, similar to Kamradt (2023), we introduce the biasing statement at varying context lengths and context depths, and measure the impact of these two variables on models' modeling of the biased random process.

The results are shown in Figure 7 for the multinomial biased coin-flip experiment. While we can see that context length is largely unrelated to the bias percentage, the depth at which the biasing needle is placed in the context can have a marked effect on the overall performance of the model. For Llama 3.1, the model is optimized for long contexts, with the biasing needle close to the sampling case, while for Gemma-2, there is a less distinct, but still evident sweet spot for depth and context length. In-

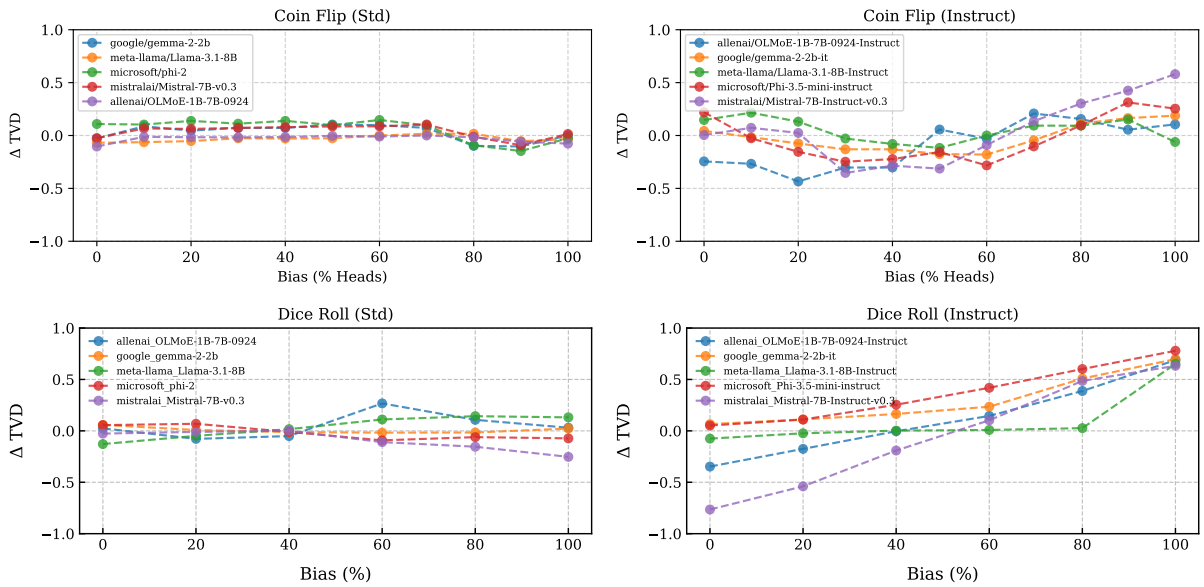


Figure 10: **Chain of Thought (CoT):** We show TVD difference between CoT and non-CoT input (Δ TVD, ↓). We observed that CoT has almost no impact on the performance of the non-instruct models. For the instruct models, CoT changes the model performance on different biases, but doesn't enhance the overall performance.

Model	**TOSS-v1 (↓)**
Mistral-7B-Instruct-v0.3	**0.8782**
Gemma-2-2b-it	0.8999
OLMoE-1B-7B-0924-Instruct	1.0682
Phi-3.5-mini-instruct	1.2980
Llama-3.1-8B-Instruct	1.3152
Llama-3.1-8B	**0.7917**
Mistral-7B-v0.3	0.8295
OLMoE-1B-7B-0924	0.8622
Phi-2	0.8783
Gemma-2-2b	0.8877

Table 1: Scores on the TOSS-v1 benchmark. In general, non-instruction-tuned models achieve the best scores, with instruction-tuned models showing degraded performance on the benchmark. Interestingly, Gemma degrades the least when converting to an instruction-tuned model, suggesting that their fine-tuning process preserves the most performance.

terestingly, for both models, longer prefixes tend to help the performance of the model, suggesting that additional "garbage" conditioning signal can help the model to approximate stochastic distributions.

5 TOSS: Measuring LLM Performance on Simple Stochastic Processes

Beyond simple analysis, we also aim to inspire further optimization of LLM behavior on simple stochastic processes. Towards such goals, we introduce the TOSS benchmark, which scores model performance on several stochastic processes across the suite of experiments in this paper. The **TOSS-v1** score consists of a weighted average of the normalized TVD-AUC for the multinomial and Poisson experiments in this paper (See Appendix K for details).

The performance of the models tested in this paper is presented in Table 1. Interestingly, we can see that while instruct-tuned models often perform worse than non-instruct-tuned models on the benchmark, Gemma degrades the least of the instruct-tuned models, suggesting that their fine-tuning process preserves the most stochastic ability from the base model. Notably, Llama-3.1 has the lowest TOSS-v1 score overall, suggesting that for most applications, it is the most capable open-source stochastic model.

6 Conclusion

In this work, we take some initial steps toward understanding and benchmarking LLM performance across various stochastic processes. While we find LLMs face significant challenges in modeling stochastic processes - especially when relying solely on prompting - our experiments with both in-context learning and Markov chains show that LLMs are capable pattern matchers (even when those patterns are stochastic). By both introducing a new family of benchmarks, through TOSS, and the experiments in this paper, we hope to provide a foundation for future research aimed at improving LLMs' capabilities as world models for probabilistic reasoning.

7 Limitations

While this paper provides an interesting look into the potential weaknesses of LLMs when performing stochastic modeling, the approach we have taken here has some weaknesses.

One of the key weaknesses of this approach is that this method for evaluating stochasticity only captures a limited slice of the potential probability distribution. Instead of saying `The coin came up "heads"`, for example, the model could have generated `The coin came up "on the side of heads"`. While the second completion is far less likely, this additional probability is not incorporated by our proposed approach and benchmark. If such options form a heavy tail of the probability distribution, then the proposed approach may incorrectly characterize the stochastic performance of a model.

Further, while the benchmarks we explore here provide a useful measure of LLM performance in stochastic modeling, they do not fully capture the complexity of real-world randomness. The limited set of probabilistic outcomes analyzed might miss out on certain subtle phenomena, such as rare or unlikely events (e.g., "the coin landed on the side").

Another weakness of our approach is the inability to effectively evaluate closed-source models and methods. Because our proposed approach relies on extracting the logits from the underlying probability distribution (artifacts that are not often made public to a downstream user of a closed-source model), we can't directly apply our methodology to models such as GPT-4 or Claude. Our approach could be approximated using sampling from the APIs, however, it would be fairly cost-intensive, and could easily be affected by API engineering decisions such as results caching or re-sampling.

Beyond the weaknesses of the proposed approach, we wanted to point out some weaknesses of using LLMs as stochastic world models as a whole. While it has already begun to occur, leveraging LLMs as world models induces a potential over-reliance on LLMs for probabilistic reasoning in complex stochastic processes. While LLMs show promise in simpler scenarios, their difficulty in handling biased random distributions and more intricate stochastic processes may lead to inaccurate predictions or simulations. This could result in flawed conclusions when applying these models to real-world contexts, especially in high-stakes fields like finance, scientific modeling, or decision-making systems. The benchmarks we introduce may offer insight into performance, but they do not guarantee that LLMs will generalize effectively across diverse and unpredictable real-world processes.

Additionally, there is a risk that the proposed benchmarks themselves may be misinterpreted. While they are designed to assess LLM performance in stochastic modeling, they may not fully capture the nuances of real-world randomness or bias, leading to overconfidence in model capabilities. LLMs can also exhibit bias amplification when working with biased random distributions, which may pose ethical concerns if not properly mitigated. Furthermore, models tested under specific benchmark conditions could over-fit those particular processes, limiting their robustness and applicability in broader or unforeseen contexts. These limitations emphasize the importance of careful oversight when deploying LLMs in stochastic modeling and real-world simulations.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*, pages 337–371. PMLR.

James F Allen and Johannes A Koomen. 1983. Planning using a temporal world model. In *Proceedings of the Eighth international joint conference on Artificial intelligence-Volume 2*, pages 741–747.

Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351.

Robert L Axtell and J Doyne Farmer. 2022. Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature*, pages 1–101.

Bernd Bohnet, Azade Nova, Aaron T Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. Exploring and benchmarking the planning capabilities of large language models. *ArXiv preprint*, abs/2406.13094.

Guhong Chen, Liyang Fan, Zihan Gong, Nan Xie, Zixuan Li, Ziqiang Liu, Chengming Li, Qiang Qu, Shiwen Ni, and Min Yang. 2024a. Agentcourt: Simulating court with adversarial evolvable lawyer agents. *ArXiv preprint*, abs/2408.08089.

Yongchao Chen, Harsh Jhamtani, Srinagesh Sharma, Chuchu Fan, and Chi Wang. 2024b. Steering large language models between code execution and textual reasoning. *ArXiv preprint*, abs/2410.03524.

Kai Lai Chung. 1967. Markov chains. *Springer-Verlag, New York*.

Gautier Dagan, Frank Keller, and Alex Lascarides. 2023. Dynamic planning with a llm. *ArXiv preprint*, abs/2308.06391.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zhiqi Ge, Hongzhe Huang, Mingze Zhou, Juncheng Li, Guoming Wang, Siliang Tang, and Yueting Zhuang. 2024. Worldgpt: Empowering llm as multimodal world model. *ArXiv preprint*, abs/2404.18202.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2024. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Advances in Neural Information Processing Systems*, 36:79081–79094.

David Ha and Jürgen Schmidhuber. 2018. World models. *ArXiv preprint*, abs/1803.10122.

Aspen K Hopkins, Alex Renda, and Michael Carbin. 2023. Can llms generate random numbers? evaluating llm sampling in controlled domains. In *ICML 2023 Workshop: Sampling and Optimization in Discrete Space*.

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. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. *ArXiv preprint*, abs/2310.01728.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

G. Kamradt. 2023. Needle in a haystack - pressure testing llms. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack`.

Sayash Kapoor, Benedikt Stroebl, Zachary S Siegel, Nitya Nadgir, and Arvind Narayanan. 2024. Ai agents that matter. *ArXiv preprint*, abs/2407.01502.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *ArXiv preprint*, abs/2305.00050.

Eliza Kosoy, Adrian Liu, Jasmine L Collins, David Chan, Jessica B Hamrick, Nan Rosemary Ke, Sandy Huang, Bryanna Kaufmann, John Canny, and Alison Gopnik. 2022. Learning causal overhypotheses through exploration in children and computational models. In *Conference on Causal Learning and Reasoning*, pages 390–406. PMLR.

Shuyuan Liu, Jiawei Chen, Shouwei Ruan, Hang Su, and Zhaoxia Yin. 2024a. Exploring the robustness of decision-level through adversarial attacks on llm-based embodied models. *ArXiv preprint*, abs/2405.19802.

Toni JB Liu, Nicolas Boullé, Raphaël Sarfati, and Christopher J Earls. 2024b. Llms learn governing principles of dynamical systems, revealing an in-context neural scaling law. *ArXiv preprint*, abs/2402.00795.

Zefang Liu and Yinzhu Quan. 2024. Tpp-llm: Modeling temporal point processes by efficiently fine-tuning large language models. *ArXiv preprint*, abs/2410.02062.

Aaron Bryan Loyall. 1997. Believable agents: building interactive personalities.

James L McClelland. 2009. The place of modeling in cognitive science. *Topics in Cognitive Science*, 1(1):11–38.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Suvir Mirchandani, Fei Xia, Pete Florence, Brian Ichter, Danny Driess, Montserrat Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large language models as general pattern machines. *arXiv preprint arXiv:2307.04721*.

Suhong Moon, Marwa Abdulhai, Minwoo Kang, Joseph Suh, Widyadewi Soedarmadji, Eran Kohen Behar, and David M Chan. 2024. Virtual personas for language models via an anthology of backstories. *ArXiv preprint*, abs/2407.06576.

Niklas Muennighoff, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Jacob Morrison, Sewon Min, Weijia Shi, Pete Walsh, Oyvind Tafjord, Nathan Lambert, et al. 2024. Olmoe: Open mixture-of-experts language models. *arXiv preprint arXiv:2409.02060*.

Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. 2023. Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling. In *International Conference on Machine Learning*, pages 26311–26325. PMLR.

OpenAI. 2022. Introducing chatgpt.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Athanasios S Polydoros and Lazaros Nalpantidis. 2017. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. 2024. Llm-sr: Scientific equation discovery via programming with large language models. *ArXiv preprint*, abs/2404.18400.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization for llm agents. *ArXiv preprint*, abs/2403.02502.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2024. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Katherine Van Koevering and Jon Kleinberg. 2024. How random is random? evaluating the randomness and humaness of llms' coin flips. *ArXiv preprint*, abs/2406.00092.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. On the brittle foundations of react prompting for agentic large language models. *ArXiv preprint*, abs/2405.13966.

Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. 2024. Large multimodal agents: A survey. *ArXiv preprint*, abs/2402.15116.

Yongjian Xu, Akash Nandi, and Evangelos Markopoulos. 2024. Application of large language models in stochastic sampling algorithms for predictive modeling of population behavior. *Artificial Intelligence and Social Computing*, 122:10–20.

Yijun Yang, Tianyi Zhou, Kanxue Li, Dapeng Tao, Lusong Li, Li Shen, Xiaodong He, Jing Jiang, and Yuhui Shi. 2024. Embodied multi-modal agent trained by an llm from a parallel textworld. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26275–26285.

Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Xinze Chen, Guan Huang, Xiaoyi Bao, and Xingang Wang. 2024. Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. *ArXiv preprint*, abs/2403.06845.

Appendix

The appendix consists of the following further discussion:

- Appendix A discusses the data used and created in this paper, and the licenses and usage.
- Appendix B discusses the use of artificial intelligence in the creation of this manuscript.
- Appendix C explains the methodologies including distribution normalization and comparisons with prior work.
- Appendix D details the models used in this study, their specifications, and training sources.
- Appendix E details how ICL prompts are generated for the Poisson distribution experiments.
- Appendix F elaborates upon the exact setups for the Markov chain experiments and provides additional results.
- Appendix G specifies details of how prompts are generated for the long context experiments.
- Appendix H covers the Chain-of-Thought (CoT) experiment on biased dice rolls.
- Appendix I presents the multinomial (coin flip) experiments with in-context learning.
- Appendix J presents the multinomial (die roll) experiments with in-context learning.
- Appendix K discusses the details of the TOSS-v1 benchmark.

A Data Usage

To perform NIAH experiments, we leverage WikiText (Merity et al., 2017) as filler content. The WikiText dataset is released under the CC BY-SA 4.0 license.

This paper further relies on several model artifacts including:

- Gemma-2 (Team et al., 2024) released under the Gemma license.
- Llama-3.1 (Dubey et al., 2024) released under the Llama 3 Community License Agreement.
- Phi-3.5 and Phi-3 (Abdin et al., 2024) released under the MIT license.
- Mistal 7B (Jiang et al., 2023) released under the Apache 2.0 license.
- Olmo 7B (Muennighoff et al., 2024) released under the Apache 2.0 license.

Our usage of the models is consistent with the above license terms. We release the TOSS benchmark under the MIT license.

B Use of Artificial Intelligence

This paper includes contributions generated with the assistance of AI tools. Specifically, AI assistants including Chat-GPT were used for sentence/paragraph-level editing of the content, the creation of LaTeX tables and figures from raw data sources, and as a coding assistant through GitHub Copilot. All intellectual and creative decisions, including the final content and conclusions, remain the responsibility of the authors. The use of AI in this process was supervised to ensure accuracy and alignment with the intended research outcomes.

C Methods

C.1 Distribution Normalization

Because the set of outcomes Ω is only a small part of the possible sequences that the LLM can generate, it is often necessary to re-normalize the probability distribution against the support Ω, instead of the full vocabulary space V. There are many options that could be picked for re-normalization. In our experiments, we choose to use a linear re-normalization:

$$\hat{P}_{\mathcal{M}}(o) = \frac{P_{\mathcal{M}}(o|\text{prompt})}{\sum_{\omega\in\Omega} P_{\mathcal{M}}(\omega|\text{prompt})} \tag{C.1}$$

This is in contrast to prior work (Liu et al., 2024b), who normalize using a softmax distribution:

$$\hat{P}_{\mathcal{M}}(o) = \frac{exp(P_{\mathcal{M}}(o|\text{prompt}))}{\sum_{\omega\in\Omega} exp(P_{\mathcal{M}}(\omega|\text{prompt}))} \tag{C.2}$$

Unfortunately, in the limit of small probabilities, for $p_i, 1 < i < |\Omega|$, as $p_i \to 0$:

$$\lim_{p_i\to 0, p_j\to 0} \frac{e^{p_i}}{\sum_j e^{p_j}} = \frac{1}{\sum_j e^{p_j}} \approx \frac{1}{|\Omega|} \tag{C.3}$$

This can significantly impact the computation of downstream measures. Normalizing linearly avoids this issue, but can sometimes cause numeric instability.

C.2 Instruct Models Chat Templates

In order to make instruction-tuned models compatible with our formulation for extracting token probabilities, we employ chat templates in the following manner. First, we construct the chat as follows:

```
chat = {'user': instruct_prompt
, 'assistant': input_prompt}
```

Here `instruct_prompt` explicitly provides a directive to the LLM such as `"Please toss a coin and tell me whether it landed on heads or tails."`. The `input_prompt` field contains the portion to be completed by the LLM spoken in the first person, such as `"I tossed a coin and it landed on"`. Next, in order to properly extract token probabilities pertaining to the support of the random process at the end of the input sequence, we modify models' chat templates to omit the EOT tokens.

D Models

google/gemma-2-2b : Gemma-2 (Team et al., 2024) is a 2-billion parameter model from Google DeepMind, built on a decoder-only Transformer architecture. It uses a model dimension of 2304, 26 layers, 8 heads with Grouped-Query Attention (GQA), and a feedforward dimension of 18,432. The model features 256,128 vocabulary entries using a SentencePiece tokenizer optimized for multiple languages. Trained on 2 trillion tokens from sources like web documents, code, and scientific articles, it incorporates local sliding window attention (4096 tokens) and global attention (8192 tokens). It also uses knowledge distillation from larger models, which enables it to achieve strong results with more efficient training.

google/gemma-2-2b-it : Gemma-2-IT (Team et al., 2024) is the instruction-tuned version of Gemma-2, featuring the same architecture as Gemma-2-2B. The model has been fine-tuned on supervised datasets and further improved using RLHF (Reinforcement Learning from Human Feedback) for better instruction-following capabilities. It uses the same 256,128-entry vocabulary and was trained on similar data sources. Gemma-2-IT includes additional tuning to enhance safety and reduce hallucinations.

meta-llama/llama-3.1-8B: Llama-3 (Dubey et al., 2024) is a foundation model developed by Meta, built with an 8 billion parameter dense Transformer architecture. The model has 32 layers, a model dimension of 4096, a feedforward dimension of 14,336, and 32 attention heads. It supports multilingual tasks, coding, and reasoning with a context window of 8K tokens. Llama-3 was pre-trained on a dataset of 15 trillion tokens, spanning a variety of sources such as web documents, code, and multilingual texts, with a vocabulary size of 128,000 tokens using a tokenizer optimized for multilingual use.

meta-llama/llama-3.1-8B-Instruct: Llama-3-Instruct (Dubey et al., 2024) is the instruction-tuned variant of Llama-3, also comprising 8 billion parameters, 32 layers, 4096 model dimensions, and a feedforward dimension of 14,336. This version is fine-tuned to follow human instructions better, leveraging supervised fine-tuning and Direct Preference Optimization (DPO). It is designed for tasks requiring precise instruction following, including coding, reasoning, and complex dialogue, while supporting tools like code generation and multilingual text processing. It also includes additional tuning to enhance safety and reduce hallucinations.

microsoft/phi-3.5-mini-instruct : Phi-3 (Abdin et al., 2024) is a 3.8-billion parameter Transformer model designed by Microsoft, optimized for both small-scale deployment and high-performance tasks. The model has 32 layers, 3072 hidden dimensions, 32 attention heads, and a default context length of 4K tokens, extendable to 128K using LongRope. It was trained on 3.3 trillion tokens, with a dataset comprising heavily filtered publicly available web data and synthetic data. Its instruction-following capability is enhanced through supervised fine-tuning and Reinforcement Learning from Human Feedback (RLHF)

microsoft/phi-2 : Phi-2 (Abdin et al., 2024) is a 2.7-billion parameter model, part of Microsoft's Phi series, designed for efficient performance in smaller-scale models. Like Phi-3, it uses a transformer-based decoder architecture with Grouped-Query Attention (GQA) and a vocabulary size of 320641 tokens and is trained on a mixture of filtered web data and LLM-generated synthetic data.

mistalai/Mistral-7B: Mistral-7B (Jiang et al., 2023) is a 7-billion parameter model developed by Mistral AI, built with a Transformer architecture optimized for efficiency and performance. The model has 32 layers, a model dimension of 4096, a

feedforward dimension of 14,336, and 32 attention heads. Mistral-7B uses Grouped-Query Attention (GQA) and Sliding Window Attention (SWA) to handle sequences up to 8192 tokens.

mistralai/Mistral-7B-Instruct: Mistral-7B-Instruct (Jiang et al., 2023) is the instruction-tuned variant of Mistral-7B, featuring the same architecture with 7 billion parameters, 32 layers, 4096 model dimensions, and a feedforward dimension of 14,336.

allenai/OLMoE-1B-7B : OLMo-7B (Muennighoff et al., 2024) is a 7-billion parameter language model developed by Allen AI, designed for open access and transparency. The model consists of 32 layers, a model dimension of 4096, a feedforward dimension of 11,008 (due to its SwiGLU activation), and 32 attention heads. The vocabulary size is 50,280 tokens, based on a modified BPE tokenizer that includes special tokens for anonymizing personal identifiable information (PII). OLMo-7B was trained on Dolma, which comprises 2.46 trillion tokens from diverse sources like Common Crawl, GitHub, Wikipedia, and scientific papers.

allenai/OLMoE-1B-7B-Instruct : OLMo-7B-Instruct (Muennighoff et al., 2024) is the instruction-tuned variant of OLMo-7B, fine-tuned for enhanced instruction-following capabilities. Like OLMo-7B, it features 32 layers, a model dimension of 4096, and 32 attention heads, with a feedforward dimension of 11,008. This variant was fine-tuned using a mixture of human-annotated and distilled instruction data, optimized further using Direct Preference Optimization (DPO) for better alignment with human preferences.

E Details on Poisson Distribution Experiments

E.1 Poisson Prompt Generation

We generate prompts for in-context learning of Poisson distributions as follows. We use GPT-4o (OpenAI, 2022) to generate a story of approximately 1K tokens. Given a desired context length of N words, we split the story based on whitespace and keep the first N words. The story subset is then modified to insert a given special character at a rate following the provided average number of occurrences λ from the Poisson distribution:

```
def modify_story_with_poisson
(special_char,
 lambda_param, story, n_words=1000):
"""
Insert a special
 character into a story string
 following a poisson distribution
 with a given lambda parameter.
"""
# Split the story into words
words = story.split()

# Cut the story to the first n_words
words = words[:n_words]

# Initialize the modified story list
modified_story = []

# Iterate through the words
for word in words:
    modified_story.append(word)

    # Draw a
sample from the Poisson distribution
    insert_after_words
 = np.random.poisson(lambda_param)

    # If the sample is greater
than 0, insert the special character
    if insert_after_words > 0:

 modified_story.append(special_char)

# Join the list back into a string
return ' '.join(modified_story)
```

For experiments, we run trials over the following set of special characters:

```
special_chars = ['*', '^', '#'
, '@', '$', '%', '&', '!', '~', '<']
```

F Details on Markov chain experiments

F.1 Transition matrices for Markov chain experiments

Each Markov chain experiments generates $N = 1000$ trajectories from transition matrices defined below.

F.1.1 Simple chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.00 & 0.50 & 0.00 & 0.50 \\ 0.20 & 0.00 & 0.80 & 0.00 \\ 0.10 & 1.20 & 0.10 & 0.60 \\ 0.10 & 0.10 & 0.10 & 0.70 \end{pmatrix} \end{array}$$

F.1.2 Quicksand chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.70 & 0.10 & 0.10 & 0.10 \\ 0.10 & 0.70 & 0.10 & 0.10 \\ 0.10 & 0.10 & 0.70 & 0.10 \\ 0.10 & 0.10 & 0.10 & 0.70 \end{pmatrix} \end{array}$$

F.1.3 Black hole chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 0 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.10 & 0.10 & 0.10 & 0.70 \\ 0.10 & 0.10 & 0.10 & 0.70 \\ 0.10 & 0.10 & 0.10 & 0.70 \\ 0.10 & 0.10 & 0.10 & 0.70 \end{pmatrix} \end{array}$$

F.1.4 $u = 0.0$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix} \end{array}$$

F.1.5 $u = 0.1$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.25 & 0.20 & 0.30 & 0.25 \\ 0.33 & 0.29 & 0.16 & 0.22 \\ 0.28 & 0.28 & 0.29 & 0.15 \\ 0.30 & 0.33 & 0.18 & 0.19 \end{pmatrix} \end{array}$$

F.1.6 $u = 0.2$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.24 & 0.13 & 0.36 & 0.27 \\ 0.09 & 0.45 & 0.08 & 0.38 \\ 0.22 & 0.11 & 0.35 & 0.32 \\ 0.18 & 0.27 & 0.28 & 0.27 \end{pmatrix} \end{array}$$

F.1.7 $u = 0.3$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.25 & 0.35 & 0.36 & 0.04 \\ 0.49 & 0.10 & 0.28 & 0.13 \\ 0.32 & 0.32 & 0.12 & 0.24 \\ 0.15 & 0.40 & 0.40 & 0.05 \end{pmatrix} \end{array}$$

F.1.8 $u = 0.4$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.10 & 0.36 & 0.15 & 0.39 \\ 0.09 & 0.30 & 0.07 & 0.54 \\ 0.03 & 0.43 & 0.54 & 0.00 \\ 0.23 & 0.27 & 0.29 & 0.21 \end{pmatrix} \end{array}$$

F.1.9 $u = 0.5$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.00 & 0.62 & 0.38 & 0.00 \\ 0.75 & 0.00 & 0.00 & 0.25 \\ 0.50 & 0.00 & 0.02 & 0.48 \\ 0.19 & 0.81 & 0.00 & 0.00 \end{pmatrix} \end{array}$$

F.1.10 $u = 0.6$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.37 & 0.00 & 0.13 & 0.50 \\ 0.42 & 0.04 & 0.00 & 0.54 \\ 0.44 & 0.27 & 0.15 & 0.14 \\ 0.24 & 0.03 & 0.55 & 0.18 \end{pmatrix} \end{array}$$

F.1.11 $u = 0.7$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.23 & 0.34 & 0.43 & 0.00 \\ 0.58 & 0.00 & 0.42 & 0.00 \\ 0.23 & 0.24 & 0.29 & 0.24 \\ 0.68 & 0.00 & 0.00 & 0.32 \end{pmatrix} \end{array}$$

F.1.12 $u = 0.8$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.43 & 0.28 & 0.01 & 0.28 \\ 0.46 & 0.29 & 0.00 & 0.25 \\ 0.39 & 0.30 & 0.00 & 0.31 \\ 0.45 & 0.39 & 0.00 & 0.16 \end{pmatrix} \end{array}$$

F.1.13 $u = 0.9$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.23 & 0.48 & 0.29 & 0.00 \\ 0.43 & 0.00 & 0.57 & 0.00 \\ 0.40 & 0.16 & 0.00 & 0.44 \\ 0.47 & 0.11 & 0.36 & 0.06 \end{pmatrix} \end{array}$$

F.1.14 $u = 1.0$ chain

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.55 & 0.00 & 0.45 & 0.00 \\ 0.40 & 0.00 & 0.20 & 0.40 \\ 0.00 & 0.00 & 0.00 & 1.00 \\ 0.22 & 0.78 & 0.00 & 0.00 \end{pmatrix} \end{array}$$

F.2 Entropy experiments

We ran the entropy set of experiments on all tested models. We present the results below. Of note is the poor performance of OLMoE 7B which is counter to the trends demonstrated by all other LLMs.

G Details on Long Context

G.1 Long Context Prompt Generation

For the long context experiments in Section 4, we are interested in testing the effects of inserting biasing statements for our distributions at varying distances away from the final prompt (e.g. `[context;"I flipped a coin and it landed on"]`), where `context` contains unrelated text of approximately N tokens with the biasing statement inserted somewhere within (i.e. at a certain document depth %). Specifically, we sample 10 unique contexts for each context length and aggregate performance over samples for each bias statement document depth insertion. In order to maintain fluency in the contexts, we use Spacy[1] to split text from the dataset into sentences and insert sentences until the desired context length threshold is reached.

H Details on CoT

H.1 Detailed Setup for Dice Roll

For the dice roll, we augment the distribution prompt 'When I roll dice they land on 60% of the time." to the CoT augmented prompt "When I roll dice they land on 1 60% of the time. Let's think about what this means for my distribution of dice rolls, it suggests that if I roll the dice 10 times, I will see a 1 approximately 6 times." (other distributions are augmented in the same way). We then ask the LLM to complete the prompt "I tossed a dice and it landed on".

H.2 Detailed Experiments

We show the details of the experiment result of coin flip in Figure H.1 and dice roll in Figure H.2. For std models, CoT doesn't change the performance over different biases. For instruct models, interestingly, CoT changes the performance over different biases, but doesn't significantly improve the overall performance.

[1] https://spacy.io/

I Multinomial (Coin Flip)

In this section, we present the full set of results for the multinomial coin flip experiment.

I.1 Coin-Flip (ICL)

In Figure I.1, Figure I.2, Figure I.3, Figure I.4, and Figure I.5, we present the Mean Total Variation Distance (TVD, ↓) against bias percentage for several ICL (In-Context Learning) example lengths across different models. These plots help analyze how well each model handles bias in a coin flip prediction task as the ICL context varies. The lower the TVD score, the better the model performs in generating unbiased predictions.

J Multinomial (Die Roll)

In Figure J.1, Figure J.2, Figure J.3, Figure J.4, and Figure J.5, we present ICL plots measuring TVD for a variety of model variants on the die rolling task.

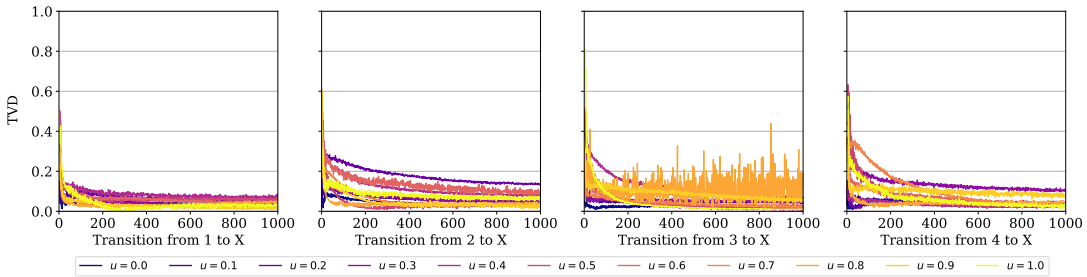
K TOSS-v1 Benchmark

The TOSS-v1 benchmark consists of an aggregate scoring across three experiments from the paper:

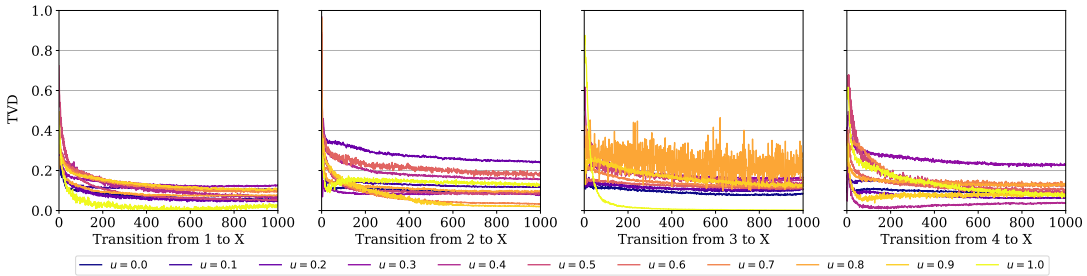
- Multinomial biased dice rolls
- Multinomial biased coin flips
- Poisson distribution matching

For each experiment, we compute the area under the TVD curve using the trapezoidal method. This area is then normalized by the mean of the experiments, which we freeze for comparison as 3.609 for the multinomial coin flip AUC, 4.104 for the multinomial die roll AUC, and 1.502 for the Poisson AUC. We then compute the weighted combination of these normalized means, using a weight of 0.4 for both multinomial experiments, and a weight of 0.1 for the Poisson process matching.

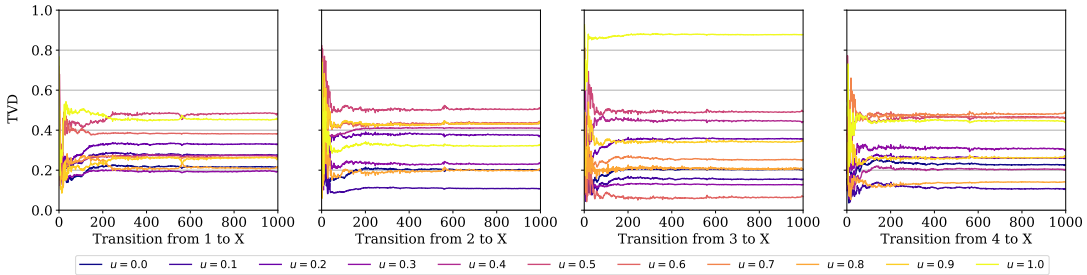
TOSS is made publicly available, and is compatible with any Huggingface Transformers model. TOSS is intended to be continuously updated as



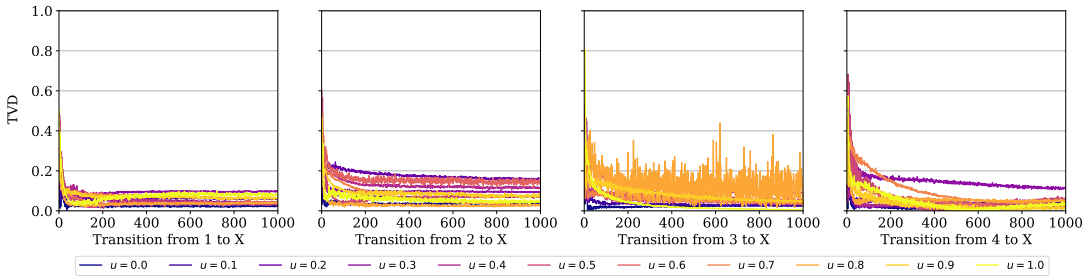
(a) Google Gemma-2 (2B)



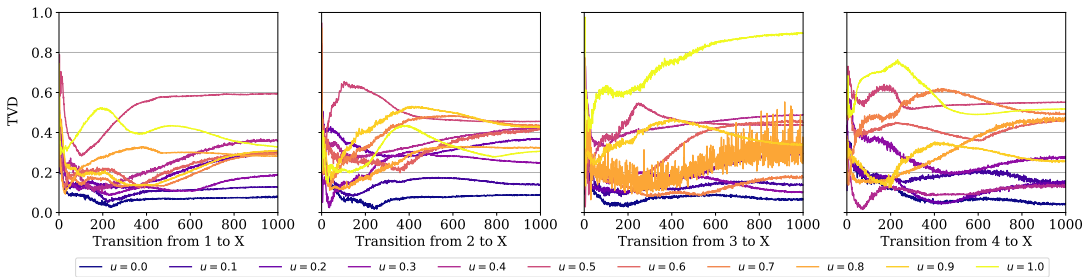
(b) Meta Llama-3.1 (8B)



(c) Microsoft Phi-2



(d) Mistral-7B



(e) AI2 OLMoE (7B)

Figure F.1: **Entropy experiments** for all models. AI2's OLMoE demonstrates inconsistent behavior versus all other tested LLMs.

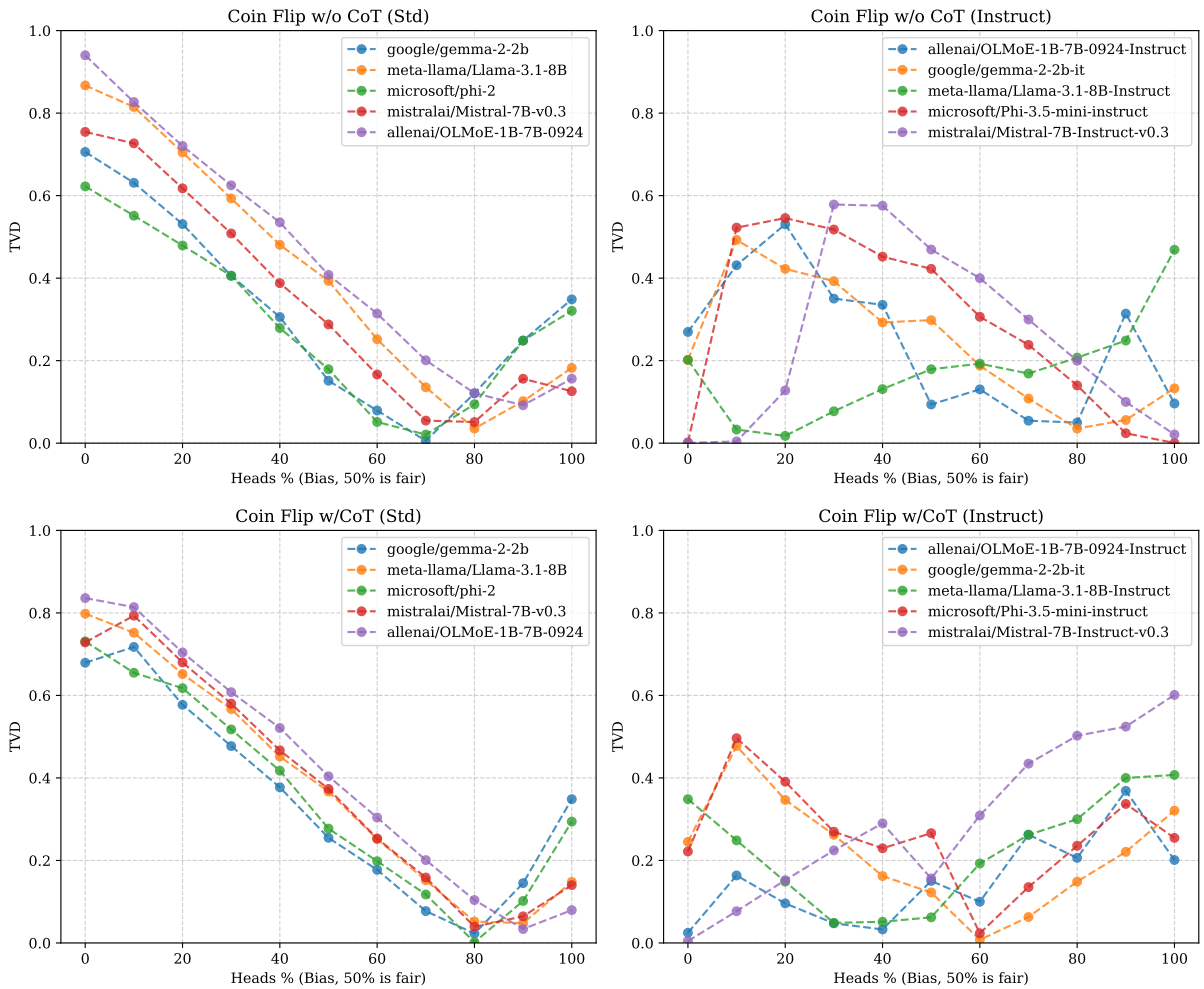


Figure H.1: **Biased Coins w/o CoT (Top) compared with Biased Coins w/ CoT (Bottom):** Plots of mean Total Variation Distance (TVD, ↓) against bias percentage for non-instruct (left) and instruct (right) models w/ (bottom) and w/o (top) CoT reasoning for the biased coin flip experiment. We use the fixed completion prompt "`I flipped a coin and it landed on`". We observed that on instruct models, CoT changes the trend over the biased distribution, but doesn't have a significant improvement over the performance. On non-instruct models, CoT doesn't change the trend much and doesn't improve the performance.

new models and architectures emerge, allowing for a standardized way to evaluate and compare the stochastic reasoning capabilities of LLMs. Future iterations of the TOSS benchmark (TOSS-v2 and beyond) will expand the suite of stochastic tasks to include more complex processes such as Brownian motion, autoregressive time series, and potentially even tasks from financial modeling or quantum processes. As the complexity of stochastic processes in AI applications continues to grow, we anticipate that TOSS could evolve into a key indicator of model robustness in real-world applications, where dealing with uncertainty and randomness is often crucial.

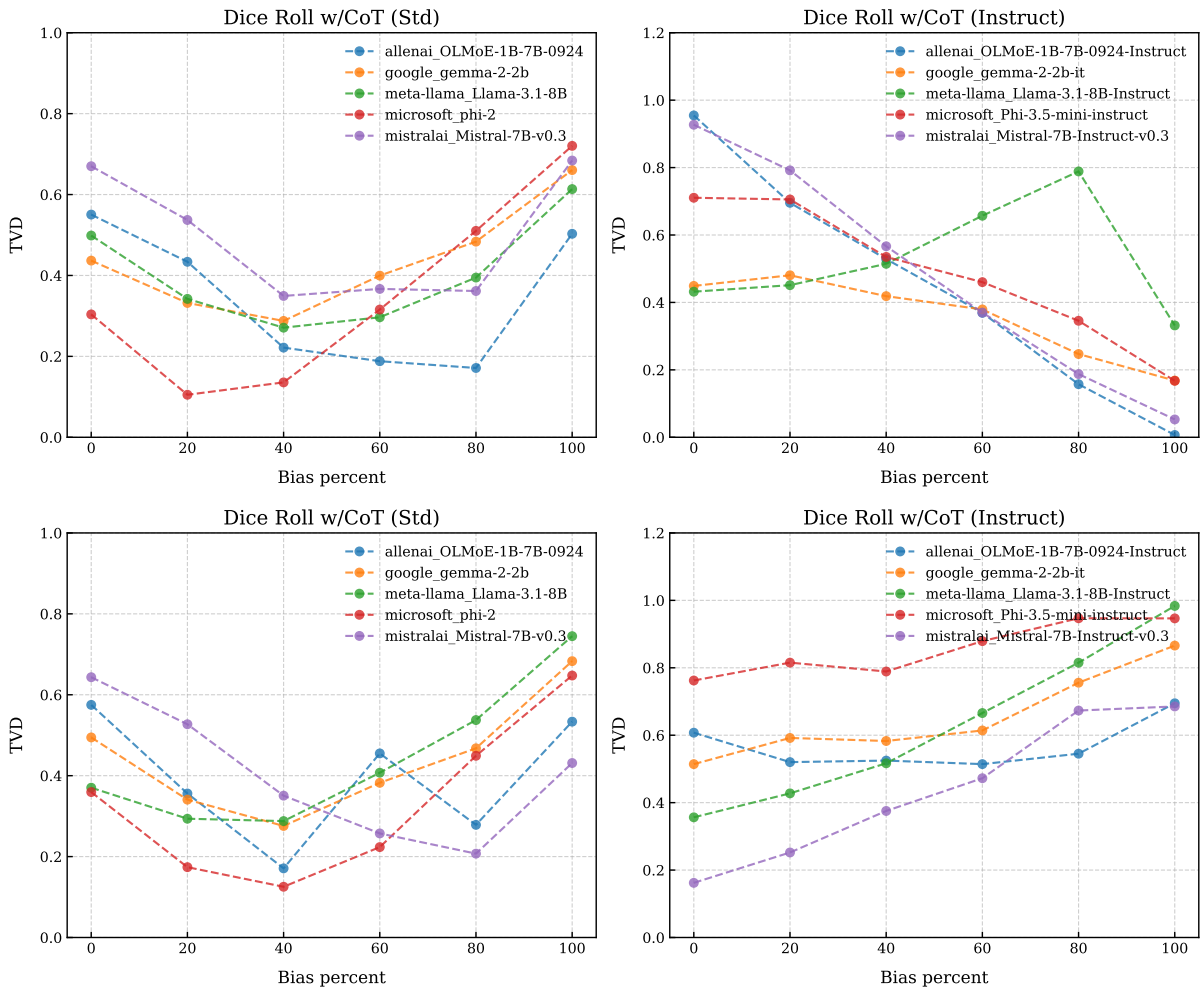


Figure H.2: **Biased Dice Roll w/o CoT (Top) compared with Biased Dice Roll w/ CoT (Bottom):** Plots of mean Total Variation Distance (TVD, ↓) against bias percentage for non-instruct (left) and instruct (right) models w/ (bottom) and w/o (top) CoT reasoning for the biased dice roll experiment. We use the fixed completion prompt "`I rolled a dice and it landed on`". Similar to what we observed in coin flip experiments, we observed that CoT doesn't consistently improve the overall performance. On instruct models, CoT changes the trend over the biased distribution. On non-instruct models, CoT doesn't change the trend.

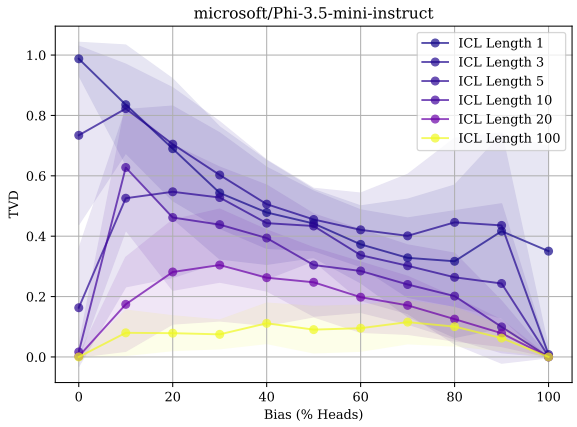


Figure I.1: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Phi-3.5-mini-instruct model.

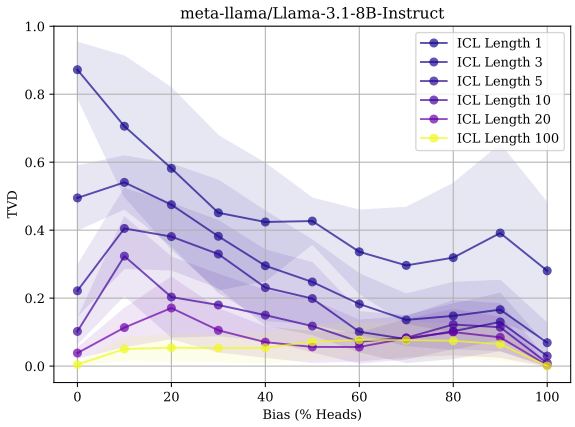


Figure I.2: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Llama-3.1-8B-Instruct model.

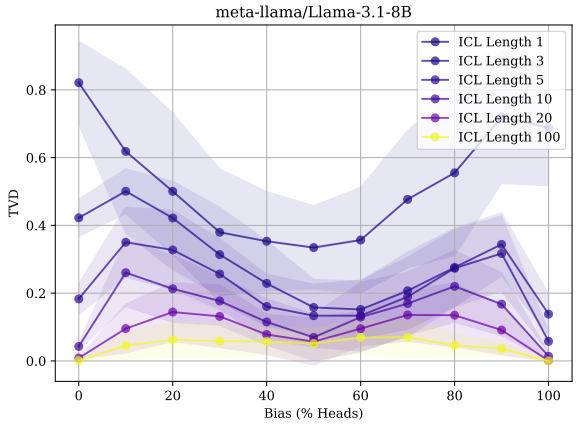


Figure I.3: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Llama-3.1-8B model.

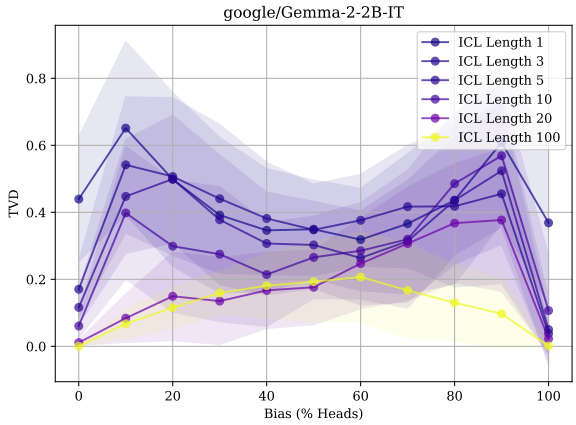


Figure I.4: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Gemma-2-2B-IT model.

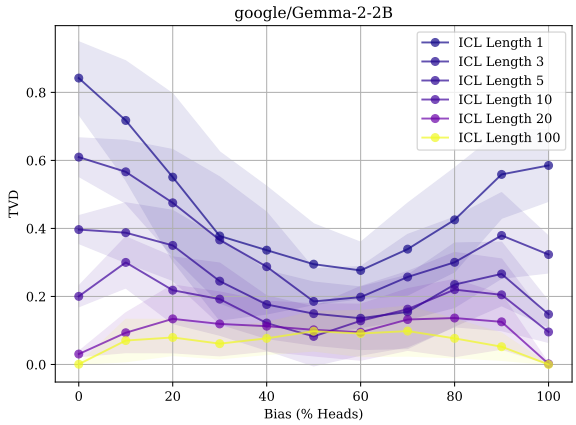


Figure I.5: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the coin flipping task for the Gemma-2-2B model.

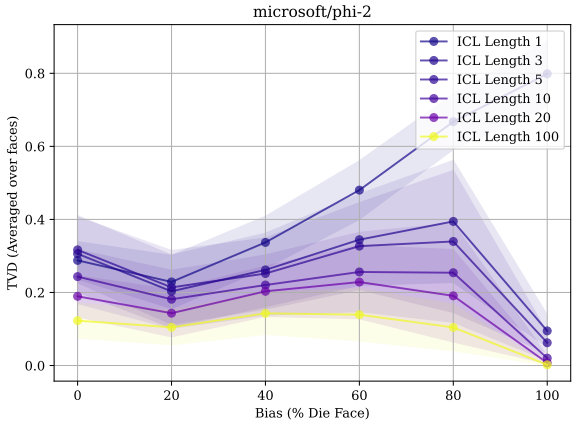


Figure J.1: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Microsoft Phi-2 model.

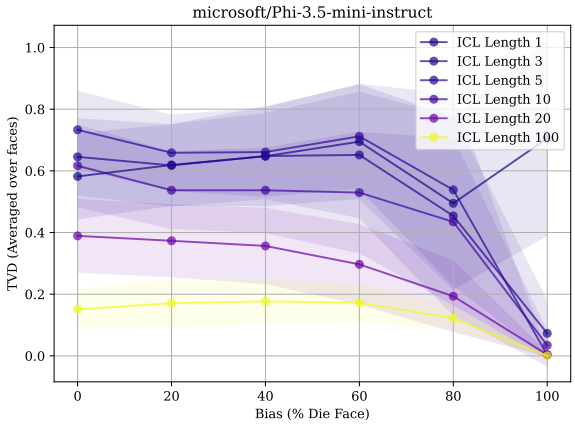


Figure J.2: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Microsoft Phi-3.5-mini-instruct model.

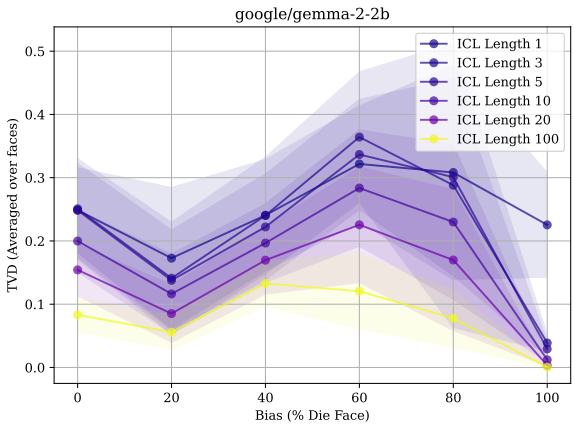


Figure J.3: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Google Gemma-2-2B model.

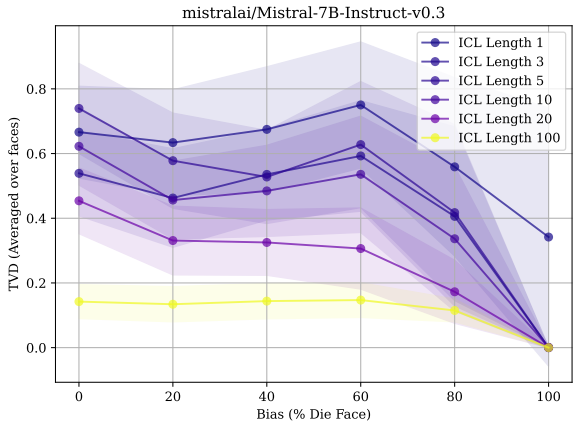


Figure J.4: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Mistral 7B-Instruct-v0.3 model.

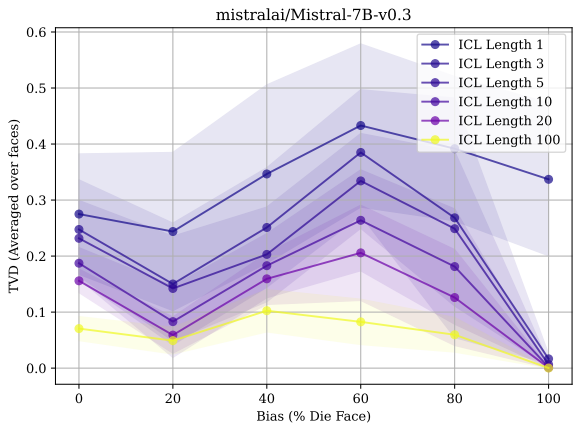


Figure J.5: Mean Total Variation Distance (TVD, ↓) vs. bias percentage for several ICL example lengths on the die rolling task for the Mistral 7B-v0.3 model.