Waste Not, Want Not[*]; Recycled Gumbel Noise Improves Consistency in Natural Language Generation

Damien de Mijolla, Hannan Saddiq, Kim Moore
Faculty Science Ltd
damien.de-mijolla@faculty.ai

Abstract

Consistency in the output of language models is critical for their reliability and practical utility. Due to their training objective, language models learn to model the full space of possible continuations, leading to outputs that can vary significantly in style and content, even for similar or repeated inputs. To address this, we propose a novel decoding algorithm that enhances response consistency across different prompts with no degradation in response quality. By incorporating a latent variable into the next-token sampling process based on the Gumbel reparametrisation trick, our method outperforms standard sampling by up to 10% across semantic and stylistic consistency benchmarks. Additionally, our approach integrates seamlessly with existing sampling methods with negligible computational overhead, providing a practical solution for improving the reliability of language model outputs.

1 Introduction

In recent years, state-of-the-art language models (LMs) have demonstrated remarkable performance across a wide range of benchmarks, often rivaling human capabilities in tasks such as translation, summarization, and question-answering (Brown et al., 2020; Dubey et al., 2024). However, these advancements have not always translated into practical usefulness for real-world applications, where reliability and consistency are crucial (Kaddour et al., 2023).

One of the primary challenges is the inconsistency of these models' responses, which can vary significantly in style, factual accuracy, and tone (Bommasani et al., 2021). This inconsistency, a byproduct of the probabilistic nature of language model training, can lead to a range of issues, including reduced trust in outputs, exposure to more diverse failure modes, and less reliable behaviour (Ye et al., 2023).

Although traditional methods (e.g the use of random seeds) can be applied to introduce determinism in natural language generation, ensuring identical responses for identical inputs, they do not help ensure similar responses when inputs are similar. In practice, due to the richness of language, input queries can often be reworded in many ways while retaining their meaning. To achieve greater consistency, it is desirable for the model to generate similar responses across all these variations (Ribeiro et al., 2018).

In this paper, we investigate whether next-token sampling procedures can be modified to enhance consistency across different prompts. Our main contributions include:

1. We propose a simple, computationally inexpensive sampling procedure that (i) can be applied to any model, (ii) does not require any additional training, and (iii) has negligible impact on inference costs. We also ensure that the probability of any individual response is unchanged and so does not compromise response quality.

2. We also leverage an auxiliary approach to further improve consistency between model responses using distributional ensembling, which can be applied in conjunction with our aforementioned sampling procedure.

3. We investigate the performance of our approach against standard sampling across a number of benchmarks covering semantic and stylistic similarity, across a number of different models.

In particular, we highlight that our combined sampler outperforms standard sampling across all benchmark suites and models tested, by up to 10% in some cases.

[*]*Definition*: If you use resources wisely and avoid waste, you'll never suffer from a shortage

2 Related works

Decoding approaches Language model decoding strategies can be broadly classified into two categories: optimization and sampling-based approaches (Ji et al., 2024). Optimization-based approaches, such as greedy decoding and beam search (Lowerre, 1976; Jurafsky and Martin, 2009), frame text generation as an optimization problem, searching for sequences that maximize a specific metric such as probability, whereas sampling-based approach incorporate stochasticity into the next-token selection process. Optimization-based approaches are typically perceived as yielding less engaging but more accurate responses and so are often favoured for closed-ended tasks expecting a fixed answer (Holtzman et al., 2020). However, recent work has put into question the greater accuracy of their responses (Renze and Guven, 2024).

In contrast, sampling-based approaches are usually preferred for open-ended tasks, as they typically yield more engaging answers (Basu et al., 2021; Ji et al., 2024). Our proposed method falls within this category. Many existing methods in the literature, such as nucleus sampling and mirostat (Holtzman et al., 2020; Basu et al., 2021; Fan et al., 2018), aim to improve text generation quality by directly modifying the probability distribution from which tokens are sampled. We consider these methods, which directly alter the next-token distribution, as complementary to our approach, which maintains the next-token distribution and instead modifies the joint distribution over responses.

Our approach is methodologically most closely related to methods (Vilnis et al., 2023; Kool et al., 2019) which also adjust the joint distribution of sampled responses. However, while these methods aim to maximize response diversity—an advantage when ensembling multiple responses as done in self-consistency voting (Wang et al., 2023)—our approach is distinct in its focus on minimizing response diversity to achieve more consistent outputs.

Self-Consistency Language models lack robustness to prompt variations (Huang et al., 2024; Elazar et al., 2021) and give contradictory responses in such cases, motivating the need for enhanced self-consistency. Self-consistency in language models has been studied from many different angles, but usually with a focus on factual rather than stylistic consistency. Prior work has proposed a number of fine-tuning approaches for increasing self-consistency, including fine-tuning approaches for increasing the ability of language models to respond consistently to paraphrases of questions (Elazar et al., 2021; Yan et al., 2024), and approaches for correcting model contradictions using a factor graph over beliefs (Mitchell et al., 2022).

Our approach is methodologically orthogonal to previous approaches for enhancing self-consistency. Previous work has relied on fine-tuning which not only is more cumbersome to implement but also modifies the raw next-token probabilities, potentially affecting responses in unforeseen ways or contributing to catastrophic forgetting.

Since our approach only modifies the joint distribution over responses without modifying the next-token probability distribution, it does not suffer from the same issues, and comes with principled guarantees around maintaining the model's original response style and quality. Additionally, it enhances *all* aspects of self-consistency, not just factual consistency of responses.

3 Problem statement

Let X be a language model prompt composed of a sequence of tokens drawn from a vocabulary of size N_v, and let π_θ be a language model trained on the task of next-token-prediction. For the remainder of the paper we denote a forward pass through the language model by $h_t = \pi_\theta(X, Y_{1:t-1})$ where $h_t \in \Delta^{N_v-1}$ represents a probability distribution over the token vocabulary, with Δ^{N_v-1} denoting the (N_v-1)-dimensional probability simplex. The sequence $Y_{1:T}$ represents the full response obtained by auto-regressively applying the language model with the next token at each step sampled from the categorical distribution parameterized by the model, $Y_t \sim \text{Cat}(h_t)$. In what follows, we use a subscript to represent position in a sequence, and a superscript to represent the token index. So for example, h_t^i represents the probability of sampling token i at position t.

Suppose that U is a different prompt that is semantically similar to X for which we generate a response $V = V_{1:M}$. Motivated by the inconsistency of LM responses, our goal is to modify the LM sampling procedure in a way that increases the similarity between responses Y and

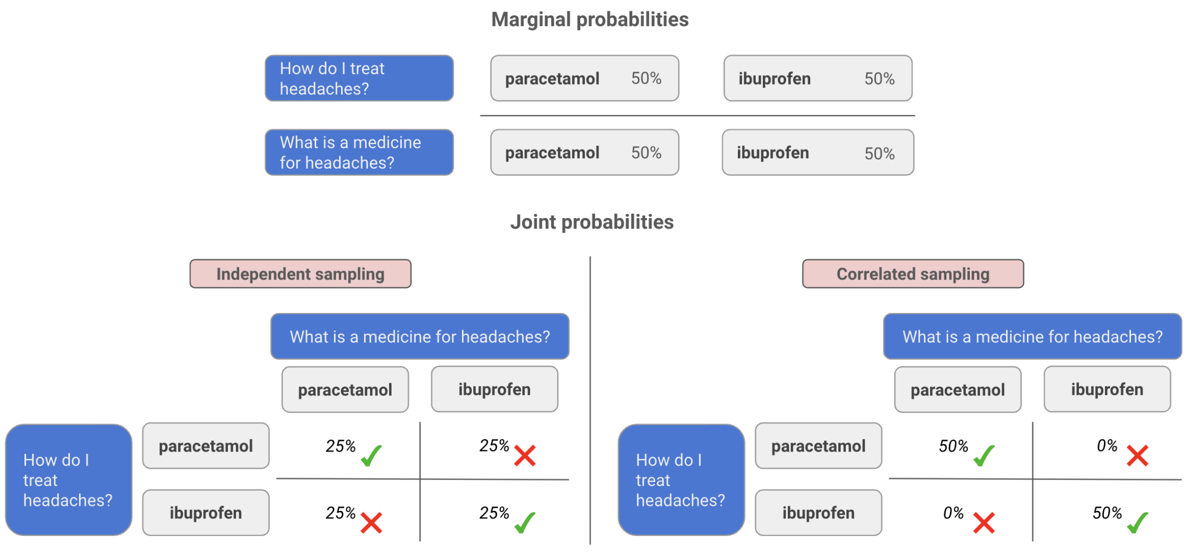

Figure 1: Motivating toy example highlighting the aim of our approach. Even when language models yield similar probability distributions over responses (top), responses sampled independently (bottom left) can be inconsistent or contradictory due to the inherent stochasticity of sampling. By generating responses in a correlated manner (bottom right) it is possible to alleviate inconsistencies across responses while still respecting the marginal probabilities of each response. In this paper we propose, Gumbel Consistent Sampling, an approach for increasing response consistency through drawing correlated responses, by conditioning all responses on a shared latent variable, that is robust to differences between probability distributions over responses.

V according to some yet-to-be-specified notion of similarity. Furthermore, we focus on sampling approaches that modify the joint probability of responses $p(Y, V)$ without affecting the marginal probability of individual responses, $p(Y)$ and $p(V)$, to guarantee that quality of the original responses is maintained.

4 Approach

Our proposed sampling approach, motivated in Figure 1, modifies the joint probability distribution over responses by introducing a latent variable g to the sampling process. Conditioning the generation of distinct responses on a common realisation of this latent variable introduces a statistical dependency between them. Generating responses with greater similarity can then be straightforwardly done by conditioning the generation of all responses on a common realisation of the latent variable, that is to say to sample $Y \sim p(Y|X, g)$ and $V \sim p(V|U, g)$.

To ensure the efficacy of the approach, we design the latent variable in such a way that conditioning responses on a common value of the latent variable makes responses as similar as possible. To ensure the preservation of the probability distribution parameterized by the language model, we sample the latent variable from a probability distribution $g \sim p(g)$ such that marginalising over the latent variable recovers the original distribution over responses, $\mathbb{E}_g[p(Y|g)] = p(Y)$.

To construct a latent variable with the above properties, we employ the reparametrization trick for categorical distributions. Introduced for normal distributions in (Kingma and Welling, 2014) and extended to categorical distributions in (Maddison et al., 2014, 2017; Jang et al., 2017), the reparametrization trick is a procedure that refactors the sampling from a distribution into a deterministic function of the parameters and a draw from some independent noise with a fixed distribution. For a categorical distribution with parameters $p^1, ..., p^{N_v}$, this can be cast as first drawing random noise $g = (g^1, ..., g^{N_v})$ where each $g^i \sim G(0, 1)$ is independently drawn from the Gumbel distribution (Gumbel, 1954) and selecting a category k according to $k = \arg\max_i(\log p^i + g^i)$.

Theorem 4.1. *Suppose we have two different categorical distributions parametrized by* $p^1, ..., p^{N_v}$ *and* $q^1, ..., q^{N_v}$*. Define a joint distribution over pairs of categories* (Y, V) *by defining*

$$Y = \arg\max_i(\log p^i + g^i),$$

$$V = \arg\max_i(\log q^i + g^i),$$

where $g^1, ..., g^{N_v} \sim G(0,1)$ are independent. We have that

$$P(Y=k, V=k) = \frac{p^k q^k}{p^k q^k + \sum_{i \neq k} \max\{p^i q^k, q^i p^k\}}.$$

Theorem 4.1 (proved in Appendix A) shows that interpreting the Gumbel noise as a latent variable and conditioning sampling events on the same realisation of this latent variable increases the probability of selecting the same category with both distributions compared to sampling from each categorical distribution independently, with identical sampling outcomes in the limit where p and q become identical.

Since generating a response using a LM consists of successive draws from categorical distributions, the above idea can be applied to language modelling in order to increase the token overlap across distinct responses. Indeed, we can generate ahead of time a sequence of independent Gumbel latent vectors, $g_{1:t}$, one for each position in the sequence up to the maximum sequence length, and sample each token using the Gumbel latent vector assigned to that position in the sequence when generating a response. That is to say, drawing $Y_t \sim p(Y_t|h_{y,t}, g_t)$ and $V_t \sim p(V_t|h_{v,t}, g_t)$, where here we denote by $h_{y,t}$ and $h_{v,t}$, the next-token probabilities obtained by running the language model on the context-up-to-now (i.e. $h_{y,t} = \pi_\theta(X, Y_{1:t-1})$, $h_{v,t} = \pi_\theta(U, V_{1:t-1})$). We refer to the above approach as **Gumbel Consistency Sampling, GCS**.

This sequential Gumbel sampling approach increases similarity of responses by increasing the rate at which identical tokens are generated at fixed positions in the sequence $p(Y_i = k, V_i = k)$ but has the limitation of not increasing the co-occurrence across sequence positions $p(Y_j = k, V_i = k)$. We expect that two similar responses are likely to contain some of the same tokens, but likely in different positions, so it would be advantageous for our final sampling approach to reflect this.

Introducing such an inter-position correlation in sampling outcomes across sequences is made challenging by the requirement of conditional independence between sampling steps. Indeed, to respect the LM's probability distribution, it is necessary for sequential sampling steps to be independent of each other, i.e. for $p(Y_{t+1}|X, Y_{1:t}) = p(Y_{t+1}|h_{t+1}) = \text{Cat}(Y_{t+1}; h_{t+1})$ which prevents the direct reuse of Gumbel samples across sequence positions.

Theorem 4.2. *Consider a sequence of tokens $Y_{1:T}$ generated auto-regressively according to the following update rule, where $k := \arg\max_j (g_t^j + \log h_t^j)$, $Q(\cdot)$ is the quantile function for the $G(0,1)$ distribution and $\pi_\theta(\cdot)$ is a language model:*

$$\begin{aligned} g_1 &\sim G(0,1) \\ h_{t+1} &= \pi_\theta(X, Y_{1:t}) \\ g_{t+1}^k \mid g_t, h_t &\sim G(0,1) \\ g_{t+1}^i \mid g_t, h_t &= Q\left(\frac{Q^{-1}(g_t^i)}{Q^{-1}(g_t^k + \log h_t^k - \log h_t^i)}\right), \end{aligned}$$

for $i \neq k$

$$Y_{t+1} = \arg\max_j \left(\log h_{t+1}^j + g_{t+1}^j\right)$$

With this update procedure, the probability distribution over a given token conditioned on preceding tokens is

$$p(Y_{t+1} \mid X, Y_{1:t}) = \mathit{Cat}(Y_{t+1}; h_{t+1})$$

In Theorem 4.2 we introduce a procedure for recycling a Gumbel vector after applying the Gumbel reparameterization trick. We prove in Appendix B.1 that this procedure is functionally equivalent to independently sampling each token from the true model probability distribution. This means that repeated application of the Gumbel reparametrization trick with this recycling procedure yields sequences that are indistinguishable from those obtained by independent sampling at each step from the model's categorical distribution.

This property enables the generation of highly correlated responses while preserving adherence to the model's probability distribution. By sampling a single Gumbel vector and reusing it across all generated responses, each response remains faithful to the model's predicted probabilities while also being inherently correlated due to the shared Gumbel noise. Moreover, because the Gumbel noise remains highly similar before and after recycling, responses exhibit strong inter-position correlations across different sequence positions.

We explicitly present the overall procedure in Algorithm 1 as well as an illustrative Python implementation in Appendix I. We refer to this generation approach as **Gumbel Consistency Sampling with Recycling, (GCSwR)**. At each sequence position, the algorithm resamples a new

Algorithm 1 Gumbel Consistency Sampling with Recycling (GCSwR)

```
Input: Context $X$, sequence length $T$, language model parameters $\theta$
Output: Generated token sequence $Y_{1:T}$
 1: Initialize $g \sim G(0,1) \in \mathbb{R}^{N_{\text{vocab}} \times T}$ and $c = [0, 0, \dots, 0] \in \mathbb{R}^{N_{\text{vocab}}}$
 2: for $t = 1$ to $T$ do
 3:     $h_t \leftarrow \pi_\theta(X, Y_{1:t-1})$
 4:     $k \leftarrow \arg\max_j \left( g^j_{c_j} + \log h^j_t \right)$
 5:     $Y_t \leftarrow k$
 6:     $c_k \leftarrow c_k + 1$
 7:     for each $i \neq k$ do
 8:         $g^i_{c_i} \leftarrow Q\left( \frac{Q^{-1}(g^i_{c_i})}{Q^{-1}\left(g^k_{c_k} + \log h^k_t - \log h^i_t\right)} \right)$
            {$Q(\cdot)$: Gumbel quantile function}
 9:     end for
10: end for
11: return $Y_{1:T}$
```

Gumbel noise value for the position corresponding to the chosen token while recycling a rescaled version of the existing Gumbel values for all other positions. To prevent divergences among responses due to resampling, we precompute and store Gumbel noise resamplings for each token in the vocabulary. This allows the same noise values to be reused across different responses. In the algorithm, this process is managed using the counter variable c.

The standard procedure for autoregressive token sampling, which is equivalent to independent sampling of a new Gumbel latent vector for every sequence position and every sequence, acts as a baseline for subsequent experiments, and is denoted as **Independent Sampling, (IS)**.

5 Ensembling semantically similar responses

A complementary approach to enhance consistency between responses given semantically similar prompts is to reduce the impact of semantically irrelevant prompt attributes on the next-token probability distributions, which can be achieved by increasing the similarity between the sampling distributions.

In our experiments, we explore sampling tokens from an ensembled probability distribution over semantically equivalent prompts as a means of minimising impact of semantically irrelevant prompt variations on responses. Specifically, we generate semantically equivalent variations of the user prompt by asking a separate LM (`gpt-4o mini`) to rephrase the prompt. We then run the target LM separately on all of the prompts, producing a set $\{P_i\}$ of next-token probability distributions. We then sample from an ensembled distribution, ensembled using the following formula:

$$Q^j = \frac{1}{Z} \prod_{i=1}^{n} (P^j_i)^{\frac{1}{n}} \qquad (1)$$

where Z is the normalisation constant that ensures Q defines a valid probability distribution function:

$$Z = \sum_j \prod_{i=1}^{n} (P^j_i)^{\frac{1}{n}}$$

This formula corresponds to selecting the categorical distribution that minimizes the average forward-KL divergence over all next-token probability distributions (see Appendix D). We found that direct averaging (which can equivalently be shown to minimize the reverse-KL distribution) tended to generate worse-quality responses due to at times sampling tokens that were only high-probability for a subset of question rewordings.

Note that, contrary to our proposed Gumbel sampling approach, ensembling comes at a cost of additional inference-time compute and also modifies the language model probability distributions. We highlight that ensembling can be applied in conjunction with any of the three samplers discussed in section 4, and we investigate the performance of each sampler with and without ensembling in our experiments.

6 Experiments

In our experiments, we empirically demonstrate the utility and limitations of GCS and GCSwR. We begin by quantifying the utility of the procedure for enhancing semantic similarity of responses, and highlight a number of stylistic dimensions of text along which Gumbel sampling improves consistency. Details for reproducing experiments are shown in Appendix E.

6.1 Semantic similarity

We start by quantifying the improvement in the semantic similarity between responses for semantically equivalent queries by using our Gumbel sampling variants (GCS and GCSwR). To measure semantic similarity, we use $\text{E5}_{\text{mistral-7b}}$, a specialised state-of-the-art model trained specifically on the task of semantic similarity (Wang et al., 2024).

We create semantically equivalent pairs of questions for evaluation by randomly sampling 300 questions from the Alpaca dataset (Taori et al., 2023) — a popular human-preference dataset - and rephrasing them using `gpt-4o mini`. We then generate responses to the original and rephrased version of each question using `Meta-Llama-3-8B-Instruct`, `Meta-Llama-3-8B`, `Mistral-7B-v0.1`, `Llama-2-7b-chat-hf` (AI@Meta, 2024; Touvron et al., 2023; Jiang et al., 2023). In all cases we sample from the raw unmodified next-token probabilities predicted by the language models (i.e. temperature of 1) and for Gumbel sampling, we resample the Gumbel latent vector for each pair of questions such that responses are correlated within but not between pairs.

The aggregated results, shown in Table 1, demonstrate that the most performant sampling scheme tested (GCSwR with ensembling) significantly increases response similarity to semantically equivalent questions across all models considered, by more than 10% when compared to the baseline in some cases. We note more pronounced enhancements from Gumbel sampling for unaligned models like Mistral and Llama3 Base, which we hypothesise is caused by their lower base semantic similarity compared to their instruction fine-tuned counterparts.

The above trends appear to be consistent across different choices of semantic similarity metric which we show in Appendix H, where we reproduce results using the Jaccard similarity, a simple token overlap metric, and using `all-mpnet-base-v2`, the semantic similarity model recommended by the popular `sentencetransformer` repository (Reimers and Gurevych, 2019). In both cases, we find relative performances between approaches to be consistent with those quoted in the main paper body.

6.2 Semantic similarity as a function of temperature

Next, we investigate how the effectiveness of GCSwR varies with sampling temperature. We compare the semantic similarity metric on the Alpaca dataset as a function of temperature in Figure 2 with IS as a baseline, without using ensembling in both cases. GCSwR improves the semantic consistency of responses across all temperatures, except temperature 0, where the model probabilities with and without GWSwR become identical due to the fully deterministic nature of model outputs at this temperature[1] Example responses for Llama3 models at temperature 0.8 can be found in Appendix G.

It is also interesting to note that although GCSwR improves self-consistency at all non-zero temperatures, the highest self-consistency achieved is with greedy decoding (i.e. temperature 0) which is where both approaches behave identically. However, we caution that this result does not imply that greedy decoding will always be preferable to higher-temperature Gumbel sampling. Using greedy decoding is widely considered to decrease the quality of responses across a number of important dimensions and so model providers typically use non-zero default temperatures (Basu et al., 2021; Ji et al., 2024; Zhang et al., 2021). Gumbel sampling offers a way of increasing the consistency of responses without the negative side-effects associated with excessively lowering the sampling temperature. We also note that using Gumbel sampling is much more effective at increasing self-consistency of responses than decreasing temperature, with temperatures needing to be roughly halved in order to match the benefits of using Gumbel consistency sampling.

6.3 Stylistic similarity

In this section, we study Gumbel consistency sampling's ability to enhance stylistic consistency across several distinct stylistic dimensions, evaluating GCSwR without ensembling using `Mistral-7B-v0.1` (Jiang et al., 2023).

We conduct our experiments on two datasets: Code-Alpaca and Aleatoric-List. The Code-Alpaca dataset (Chaudhary, 2023) consists of coding-related questions, from which we select a subset of 20 random questions that are agnostic to programming languages. For this dataset, we assess stylistic consistency based on several factors: whether the response contains a code snippet, whether the response starts directly with the code snippet or begins with freeform text, whether the code snippet includes comments, and the programming language used in the response (such as Python, JavaScript, or C++).

[1]We note that responses can still differ under greedy decoding if several tokens are tied for maximum probability. In experiments this occurred a non-negligible amount of times due to the limited numerical precision of bfloat16.

Model	Sampler	Without Ensembling	With Ensembling
Llama2 Chat	IS	86.34±0.07	87.56±0.29
	GCS	88.28±0.10	90.26±0.27
	GCSwR	**88.61±0.15**	**90.38±0.25**
Mistral	IS	72.00±0.27	72.34±0.93
	GCS	78.55±0.22	81.17±0.77
	GCSwR	**80.94±1.05**	**82.74±0.81**
Llama3 Instruct	IS	85.61±0.18	86.90±0.16
	GCS	86.81±0.46	89.01±0.35
	GCSwR	**87.37±0.27**	**89.68±0.08**
Llama3 Base	IS	71.23±0.41	71.46±0.70
	GCS	76.68±0.80	78.71±0.82
	GCSwR	**80.10±0.80**	**82.04±0.81**

Table 1: Average semantic similarity results by sampler type with and without our ensembling approach as measured by E5$_{\text{mistral-7b}}$. Scores shown as mean±std.err with std.err obtained from 3 independent runs. Bold indicates highest scores for each model in both ensembling categories.

Dataset	Stylistic Dimension	Sampler	
		IS	GCSwR
Code-Alpaca	Is Python	0.67	**0.73**
	Is JavaScript	0.78	**0.84**
	Is C++	0.92	**0.94**
	Contains Code Snippet	0.71	**0.81**
	Answers Directly	0.50	**0.73**
	Contains Comments	0.71	**0.80**
Aleatoric-List	Does Not Use Bullets	0.75	**0.82**
	Uses Numerical Bullets	0.82	**0.87**
	Terseness	0.50	**0.64**

Table 2: Comparison of Gumbel consistency sampling with recycling (GCSwR) vs. independent sampling (IS) on Stylistic Consistency

The second dataset, Aleatoric-List, is a synthetic dataset we created containing 20 questions that ask for five different items fitting a specific category. An illustrative example question is "Give me the names of five capital cities in Europe." For this dataset, we evaluate stylistic consistency based on whether the answer is terse, whether it contains bullet points, and whether these bullet points are numerical.

To evaluate stylistic consistency along each dimension, we begin by generating 100 Gumbel latent vectors. Then, for each Gumbel vector, we generate a response to all questions in the dataset which we classify along each of the stylistic dimensions through prompting `gpt-4o mini` (with prompts shown in Appendix F). For each factor, we then define the stylistic consistency as the probability that responses to two randomly selected questions share the same label, denoted as p_{repeat}. We then compare this probability with the equivalent probability when the responses are generated with our independent sampling baseline (IS).

Let Z be a Bernoulli random variable that denotes whether a randomly sampled response is labelled with a given stylistic dimension, $p(Z = 1) = p$. For IS, $p_{repeat} = p^2 + (1-p)^2$. However, for GCS and GCSwR, $p_{repeat} = \mathbb{E}_g[p_g^2 + (1-p_g)^2]$ where p_g denotes the probability of a randomly sampled response generated using Gumbel latent

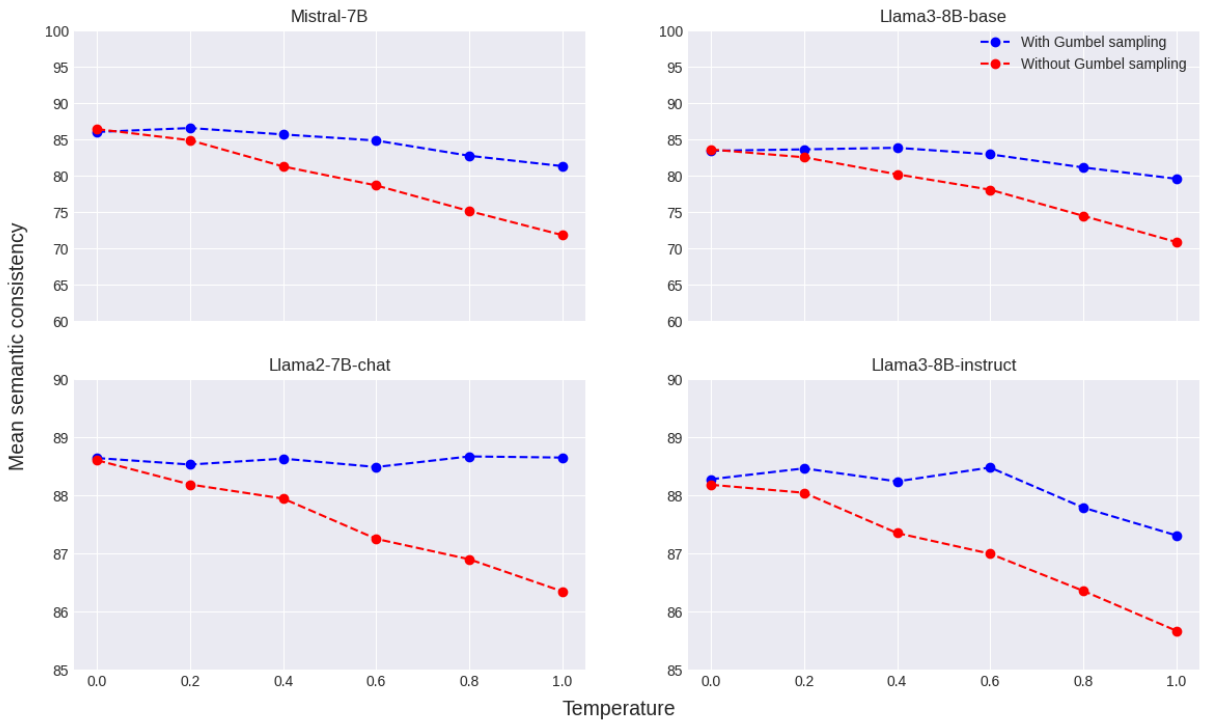

Figure 2: Mean semantic consistency between responses to paraphrased questions as a function of temperature, comparing independent sampling (IS) against Gumbel Consistency Sampling with Recycling (GCSwR).

vector g taking value $Z = 1$. These expressions follow directly from the conditional independence of responses generated with a common initial Gumbel latent vector g and generated independently, and additionally from marginalisation over initial Gumbel latent vectors g.

Although the estimator $\hat{p} = \frac{1}{n}\sum_{i=1}^{n} Z_i$ is an unbiased estimator of p, $\hat{p}^2 + (1-\hat{p})^2$ yields a biased estimator of $p^2 + (1-p)^2$. To correct for this bias, we use the following estimator $\frac{N}{N-1}(\hat{p}^2 + (1-\hat{p})^2) - \frac{1}{N-1}$ which we show in Appendix C to be unbiased.

We show, in Table 2, the results of this experiment, using Mistral-7B to generate responses. Across all stylistic dimensions considered, using GCSwR increases the frequency with which generated responses follow a common style. For many factors, the increase is significant (>10%), showing that Gumbel consistency sampling can have an appreciable impact on style consistency.

7 Conclusion

We have introduced Gumbel consistency sampling, a straightforward and computationally inexpensive sampling approach for increasing consistency amongst model responses. The method requires no additional fine-tuning, additional language model calls or apriori knowledge of what prompts will be used, and guarantees responses indistinguishable to those obtained using standard sampling at the level of individual responses. The approach enhances consistency by sampling responses in a correlated manner through the introduction of a latent variable, in a way that increases the token overlap across responses. In our experiments, we find that this approach is not only able to enhance semantic similarity between responses but also stylistic similarity. These results showcase how Gumbel consistency sampling offers a principled quick and easy way of enhancing language model consistency.

Future work could extend the Gumbel consistency sampling to imposing local rather than global correlation to responses. Currently, all responses are globally coupled due to dependence on the same global latent variable, which makes localised adjustments to model behaviour impossible. However, the framework could easily enable for latent variables to be varied locally depending on question specifics, which would enable finer-grain control of model behaviour and could increase the overall response diversity. Another, promising direction for extending the work could be to treat the Gumbel noise as a learnable task-specific parameter. Such an approach may be especially useful for building stronger model safeguards while preserving general utility.

8 Limitations

Increasing the consistency amongst a set of responses necessarily decreases their diversity making our proposed sampling approach unsuitable for use cases requiring high response diversity. In particular, using the proposed sampling approach, leads to fully deterministic sampling where responses will always be identical for identical input prompts.

More generally, use of the sampling approach is likely to lead to responses favoring specific topics and figures of speech over others. This arises due to the specific Gumbel noise value utilised during text generation encoding relative preferences between tokens and is not inherently a weakness of the approach. Indeed, analogously every members of the human population also exhibit their own individual preferences and mannerisms.

Finally, it is important to emphasize that while Gumbel consistency sampling enhances consistency amongst responses it does not guarantee it. Responses, generated using the approach, may still lack self-consistency making the approach on its own inadequate for use cases requiring perfect consistency.

References

AI@Meta. 2024. Llama 3 model card.

Sourya Basu, Govardana Sachithanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. 2021. Mirostat: a neural text decoding algorithm that directly controls perplexity. In *International Conference on Learning Representations*.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models. *ArXiv*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1223–1231, Red Hook, NY, USA. Curran Associates Inc.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron,

Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Cantón Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab A. AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriele Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guanglong Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Laurens Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Ju-Qing Jia, Kalyan Vasuden Alwala, K. Upasani, Kate Plawiak, Keqian Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline C. Muzzi, Mahesh Babu Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melissa Hall Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri S. Chatterji, Olivier Duchenne, Onur Celebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Chandra Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yiqian Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zhengxu Yan, Zhengxing Chen, Zoe Papakipos, Aaditya K. Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adi Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Ben Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Shang-Wen Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzm'an, Frank J. Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory G. Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Han Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kaixing(Kai) Wu, U KamHou, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, A Lavender, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar

Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollár, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sung-Bae Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Andrei Poenaru, Vlad T. Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xia Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Emil Julius Gumbel. 1954. *Statistical theory of extreme values and some practical applications: a series of lectures*. 33. US Govt. Print. Office.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yue Huang, Lichao Sun, Haoran Wang, Siyuan Wu, Qihui Zhang, Yuan Li, Chujie Gao, Yixin Huang, Wenhan Lyu, Yixuan Zhang, Xiner Li, Hanchi Sun, Zhengliang Liu, Yixin Liu, Yijue Wang, Zhikun Zhang, Bertie Vidgen, Bhavya Kailkhura, Caiming Xiong, Chaowei Xiao, Chunyuan Li, Eric P. Xing, Furong Huang, Hao Liu, Heng Ji, Hongyi Wang, Huan Zhang, Huaxiu Yao, Manolis Kellis, Marinka Zitnik, Meng Jiang, Mohit Bansal, James Zou, Jian Pei, Jian Liu, Jianfeng Gao, Jiawei Han, Jieyu Zhao, Jiliang Tang, Jindong Wang, Joaquin Vanschoren, John Mitchell, Kai Shu, Kaidi Xu, Kai-Wei Chang, Lifang He, Lifu Huang, Michael Backes, Neil Zhenqiang Gong, Philip S. Yu, Pin-Yu Chen, Quanquan Gu, Ran Xu, Rex Ying, Shuiwang Ji, Suman Jana, Tianlong Chen, Tianming Liu, Tianyi Zhou, William Yang Wang, Xiang Li, Xiangliang Zhang, Xiao Wang, Xing Xie, Xun Chen, Xuyu Wang, Yan Liu, Yanfang Ye, Yinzhi Cao, Yong Chen, and Yue Zhao. 2024. Trustllm: Trustworthiness in large language models. In *Forty-first International Conference on Machine Learning*.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Haozhe Ji, Pei Ke, Hongning Wang, and Minlie Huang. 2024. Language model decoding as direct metrics optimization. In *The Twelfth International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Dan Jurafsky and James H. Martin. 2009. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, Upper Saddle River, N.J.

Jean Kaddour, J. Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *ArXiv*, abs/2307.10169.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3499–3508. PMLR.

B. T. Lowerre. 1976. *The Harpy speech recognition system*. Ph.D. thesis, Carnegie Mellon University, Pennsylvania.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* Sampling. In *Advances in Neural Information Processing Systems 27*.

Eric Mitchell, Joseph Noh, Siyan Li, Will Armstrong, Ananth Agarwal, Patrick Liu, Chelsea Finn, and Christopher Manning. 2022. Enhancing self-consistency and performance of pre-trained language models through natural language inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1754–1768, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Matthew Renze and Erhan Guven. 2024. The effect of sampling temperature on problem solving in large language models. *CoRR*, abs/2402.05201.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging nlp models. In *Annual Meeting of the Association for Computational Linguistics*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

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

Luke Vilnis, Yury Zemlyanskiy, Patrick Murray, Alexandre Passos, and Sumit Sanghai. 2023. Arithmetic sampling: parallel diverse decoding for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Tianyi Yan, Fei Wang, James Y. Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive instruction tuning. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10288–10302, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Wentao Ye, Mingfeng Ou, Tianyi Li, Yipeng Chen, Xuetao Ma, Yifan Yanggong, Sai Wu, Jie Fu, Gang Chen, Haobo Wang, and Junbo Zhao. 2023. Assessing hidden risks of llms: An empirical study on robustness, consistency, and credibility. *CoRR*, abs/2305.10235.

Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.

A Proof for Theorem 4.1

Theorem. *Suppose we have two different categorical distributions parametrised by* $p^1, ..., p^{N_v}$ *and* $q^1, ..., q^{N_v}$*. Define a joint distribution over pairs of categories* (Y, V) *by defining*

$$Y = \arg\max_i(\log p^i + g^i),\ V = \arg\max_i(\log q^i + g^i), \tag{2}$$

where $g^1, ..., g^{N_v} \sim G(0,1)$ *are independent. We have that*

$$P(Y = k, V = k) = \frac{p^k q^k}{p^k q^k + \sum_{i \neq k} \max\{p^i q^k, q^i p^k\}}.$$

Proof. If

$$k = \arg\max_i\{\log p^i + g^i\} = \arg\max_i\{\log q^i + g^i\},$$

then we must have that for all $i \neq k$,

$$g^i < \log p^k + g^k - \log p^i \text{ and } g^i < \log q^k + g^k - \log q^i,$$

i.e.,

$$g^i < g^k + \min\{\log(p^k/p^i), \log(q^k/q^i)\} \quad \forall i \neq k.$$

Denoting by F the CDF of the Gumbel distribution, we can write

$$P(Y = k, V = k | g^k) = \prod_{i \neq k} F(g^k + \min\{\log p^k/p^i, \log q^k/q^i\}).$$

Denoting the PDF of the Gumbel distribution by f and marginalising we deduce that

$$P(Y = k, V = k) = \int_{-\infty}^{\infty} \prod_{i \neq k} F(g + \min\{\log p^k/p^i, \log q^k/q^i\}) f(g) dg.$$

Expanding, we can write this as

$$\begin{aligned}
P(Y = k, V = k) &= \int \prod_{i \neq k} \exp\{-\exp\{-g - \min\{\log p^k/p^i, \log q^k/q^i\}\}\} \exp\{-g - \exp(-g)\} dg \\
&= \int \exp\{-g - \exp(-g) - \sum_{i \neq k} \exp\{-g - \min\{\log p^k/p^i, \log q^k/q^i\}\}\} dg \\
&= \int \exp\{-g - \exp(-g)(1 + \sum_{i \neq k} \exp\{-\min\{\log p^k/p^i, \log q^k/q^i\}\}\} dg.
\end{aligned}$$

Recall that

$$\frac{d}{dx} e^{e^x} = e^x e^{e^x} = e^{x + e^x},$$

and so for any A independent of x we have that

$$\frac{d}{dx} e^{-Ae^{-x}} = Ae^{-x} e^{-Ae^{-x}} = Ae^{-x - Ae^{-x}}.$$

Therefore, we may deduce that

$$\begin{aligned}
P(Y = k, V = k) &= \left[\frac{1}{1 + \sum_{i \neq k} \exp\{-\min\{\log p^k/p^i, \log q^k/q^i\}\}} e^{-Ae^{-g}}\right]_{-\infty}^{\infty} \\
&= \frac{1}{1 + \sum_{i \neq k} \exp\{-\min\{\log p^k/p^i, \log q^k/q^i\}\}}.
\end{aligned}$$

Since

$$\begin{aligned}\exp\{-\min\{\log p^k/p^i, \log q^k/q^i\}\} &= \max\{\exp\{-\log p^k/p^i\}, \exp\{-\log q^k/q^i\}\} \\ &= \max\{p^i/p^k, q^i/q^k\},\end{aligned}$$

we deduce

$$P(Y = k, V = k) = \frac{1}{1 + \sum_{i \neq k} \max\{p^i/p^k, q^i/q^k\}}.$$

as claimed. □

B Proofs relating to Gumbel recycling procedure

Note that in the following proofs, we denote for notational simplicity that for a random vector x, where each element of x is independently sampled according to a Gumbel distribution, $x^k \sim G(0,1)$, $p(x) = \prod_k G(x^k; 0, 1) = G(x; 0, 1)$.

B.1 Proof for Theorem 4.2

Theorem. *Consider a sequence of tokens $Y_{1:T}$ generated auto-regressively according to the following update rule, where $k := \arg\max_j (g_t^j + \log h_t^j)$, $Q(\cdot)$ is the quantile function for the $G(0,1)$ distribution and $\pi_\theta(\cdot)$ is a language model:*

$$\begin{aligned} g_1 &\sim G(0,1) \\ h_{t+1} &= \pi_\theta(X, Y_{1:t}) \\ g_{t+1}^k \mid g_t, h_t &\sim G(0,1) \\ g_{t+1}^i \mid g_t, h_t &= Q\left(\frac{Q^{-1}(g_t^i)}{Q^{-1}(g_t^k + \log h_t^k - \log h_t^i)}\right), \quad \textit{for } i \neq k \\ Y_{t+1} &= \arg\max_j \left(\log h_{t+1}^j + g_{t+1}^j\right) \end{aligned}$$

With this update procedure, the probability distribution over a given token conditioned on preceding tokens is

$$p(Y_{t+1} \mid X, Y_{1:t}) = \mathit{Cat}(Y_{t+1}; h_{t+1})$$

Proof. We proceed through proof by induction. We make two assumptions that following expressions hold for t, then prove that the expressions hold for $t+1$ under those assumptions (and that they hold for the base case). The assumptions are that:

Assumption 1.

$$p(Y_t \mid X, Y_{1:t-1}) = \mathit{Cat}(Y_t; h_{t+1})$$

Assumption 2.

$$p(g_t \mid X, Y_{1:t-1}) = G(g_t; 0, 1)$$

N.B that in the base cases ($p(Y_1 \mid X)$ and $p(g_1 \mid X)$), the expressions are trivially valid by the Gumbel reparameterization trick and by construction of g_1 respectively.

Now, let's prove that the expressions hold for $t+1$. We will first prove the following:

$$p(g_{t+1} \mid X, Y_{1:t}) = G(g_{t+1}; 0, 1)$$

by first considering the probability $p(g^i_{t+1} < a \mid X, Y_{1:t})$:

$$\begin{aligned}
p(g^i_{t+1} < a \mid X, Y_{1:t}) &= \int p(g^i_{t+1} < a, g_t \mid X, Y_{1:t})\, dg_t \\
&= \int p(g^i_{t+1} < a \mid X, Y_{1:t}, g_t) p(g_t \mid X, Y_{1:t-1}, Y_t)\, dg_t \\
&= \int p(g^i_{t+1} < a \mid X, Y_{1:t}, g_t) \frac{p(Y_t \mid X, Y_{1:t-1}, g_t) p(g_t \mid X, Y_{1:t-1})}{p(Y_t \mid X, Y_{1:t-1})}\, dg_t \\
&= \frac{1}{p(Y_t \mid X, Y_{1:t-1})} \int p(g^i_{t+1} < a \mid h_t, Y_t, g_t) p(Y_t \mid h_t, g_t) G(g_t; 0, 1)\, dg_t
\end{aligned}$$

Firstly, consider the case where $Y_t = i$. In this case, we know that g^i_{t+1} is newly sampled from $G(0, 1)$. Therefore, using the Gumbel reparameterization trick for the last step, we have that:

$$\begin{aligned}
p(g^i_{t+1} < a \mid X, Y_{1:t}) &= \frac{1}{h^i_t} \int p(g^i_{t+1} < a \mid h_t, Y_t, g_t) p(Y_t \mid h_t, g_t) G(g_t; 0, 1)\, dg_t \\
&= \frac{1}{h^i_t} \int Q^{-1}(a) p(Y_t \mid h_t, g_t) G(g_t; 0, 1)\, dg_t \\
&= \frac{h^i_t}{h^i_t} Q^{-1}(a) = Q^{-1}(a)
\end{aligned}$$

Turning our attention to the case where $Y_t = j \neq i$

$$\begin{aligned}
p(g^i_{t+1} < a \mid X, Y_{1:t}) &= \frac{1}{p(Y_t = j \mid X, Y_{1:t-1})} \int p(g^i_{t+1} < a \mid h_t, Y_t = j, g_t) p(Y_t = j \mid h_t, g_t) G(g_t; 0, 1)\, dg_t \\
&= \frac{1}{h^j_t} \int p(g^i_{t+1} < a \mid h_t, Y_t = j, g_t) p(Y_t = j \mid h_t, g_t) G(g_t; 0, 1)\, dg_t
\end{aligned}$$

We simplify notation by denoting the following events:

$$\begin{aligned}
E' &= \left\{ Q\left(\frac{Q^{-1}(g^i_t)}{Q^{-1}(g^j_t + \log h^j_t - \log h^i_t)} \right) < a \right\} \\
E_p &= \left\{ g^p_t + \log h^p_t < g^j_t + \log h^j_t \right\}
\end{aligned}$$

Now, we can rewrite the following probabilities using these definitions:

$$\begin{aligned}
p(g^i_{t+1} < a \mid h_t, Y_t = j, g_t) &= \mathbf{1}_{E'(g_t)} \\
p(Y_t = j \mid h_t, g_t) &= \left(\prod_{p \neq j} \mathbf{1}_{E_p(g_t)} \right) \\
p(g^i_{t+1} < a \mid X, Y_{1:t}) &= \frac{1}{h^j_t} \int \mathbf{1}_{E'(g_t)} \left(\prod_{p \neq j} \mathbf{1}_{E_p(g_t)} \right) G(g_t; 0, 1)\, dg_t
\end{aligned}$$

Since $Q^{-1}(x)$ is a monotonic function, E_i is equivalently defined as:

$$E_i = \left\{ Q^{-1}\left(g^i_t\right) < Q^{-1}\left(g^j_t + \log h^j_t - \log h^i_t\right) \right\}$$

Additionally, E' can be rewritten as

$$E' = \left\{ Q^{-1}\left(g^i_t\right) < Q^{-1}(a) Q^{-1}\left(g^j_t + \log h^j_t - \log h^i_t\right) \right\}$$

Since $Q^{-1}(a) \in [0,1]$, the occurrence of E' is a sufficient condition for the occurrence of E_i. Therefore, we can simplify the integral to:

$$p(g^i_{t+1} < a \mid X, Y_{1:t}) = \frac{1}{h^j_t} \int \mathbf{1}_{E'(g_t)} \left(\prod_{p \neq i,j} \mathbf{1}_{E_p(g_t)} \right) G(g_t; 0, 1)\, dg_t$$

The CDF of the Gumbel distribution can be written $Q^{-1}(x) = e^{-e^{-x}}$, so $Q^{-1}(x+c) = \left(Q^{-1}(x)\right)^{e^{-c}}$. With this fact and application of the monotonic transformation $Q(\cdot)$, we can rewrite the events :

$$E' = \left\{ Q^{-1}\left(g^i_t\right) Q^{-1}\left(g^j_t\right)^{-\frac{h^i_t}{h^j_t}} < Q^{-1}(a) \right\}$$

$$E_p = \left\{ Q^{-1}\left(g^p_t\right) < Q^{-1}\left(g^j_t\right)^{\frac{h^p_t}{h^j_t}} \right\}$$

We now use the fact that $Q^{-1}(g^i_t) \coloneqq U^i_t \sim \mathcal{U}[0,1]\ \forall i$ to rewrite the events like so:

$$E' = \left\{ U^i_t \left(U^j_t\right)^{-\frac{h^i_t}{h^j_t}} < Q^{-1}(a) \right\}$$

$$E_p = \left\{ U^p_t < \left(U^j_t\right)^{\frac{h^p_t}{h^j_t}} \right\}$$

In conjunction with lemma B.1, this gives us the desired cumulative density function:

$$p(g^i_{t+1} < a \mid X, Y_{1:t}) = \frac{1}{h^j_t}(h^j_t) Q^{-1}(a) = Q^{-1}(a)$$

Since the cumulative density function in both cases ($Y_t = i$ and $Y_t \neq i$) is $Q^{-1}(a)$, we have that, under our initial assumptions, $p(g_{t+1} \mid X, Y_{1:t}) = G(g_{t+1}; 0, 1)$.

Finally, we then introduce and marginalise over the Gumbel noise vector at the previous timestep for the distribution over Y_{t+1}, where the final step follows from the Gumbel reparameterization trick:

$$\begin{aligned}
p(Y_{t+1} \mid X, Y_{1:t}) &= \int p(Y_{t+1}, g_{t+1} \mid X, Y_{1:t})\, dg_{t+1} \\
&= \int p(Y_{t+1} \mid X, Y_{1:t}, g_{t+1}) p(g_{t+1} \mid X, Y_{1:t})\, dg_{t+1} \\
&= \int p(Y_{t+1} \mid h_{t+1}, g_{t+1}) G(g_{t+1}; 0, 1)\, dg_{t+1} \\
&= \mathrm{Cat}(Y_{t+1}; h_{t+1})
\end{aligned}$$

Therefore, since the expressions are valid for the base case of $t = 1$, and we have shown them to be valid for $t + 1$ if assumptions 1 and 2 hold, they must be true for all t, by induction. □

B.2 Statement and Proof of lemma B.1

Lemma B.1. *X, Y and $Z_{1:N}$ are random variables each independently drawn from $\mathcal{U}[0,1]$. A, B, $C_{1:N}$ and D are positive constants between 0 and 1, and $A + B + \sum_n C_n = 1$. Defining the events $E^* = \left\{XY^{-\frac{A}{B}} < D\right\}$ and $E_n = \left\{Z_n < Y^{\frac{C_n}{B}}\right\}$, the probability of the intersection of events is given by:*

$$P\left(E^* \cap \bigcap_{n=1}^{N} E_n\right) = BD$$

Proof. We can write down the following probabilities that are conditional on Y:

$$P(E^*|Y) = P\left(X \leq DY^{\frac{A}{B}}\right) = DY^{\frac{A}{B}}$$
$$P(E_n|Y) = P\left(Z_n \leq Y^{\frac{C_n}{B}}\right) = Y^{\frac{C_n}{B}}$$

Therefore, the probability of the complement is given by integrating the product of these quantities over $p(y)$:

$$\begin{aligned} P\left(E^* \cap \bigcap_{n=1}^{N} E_n\right) &= \int_0^1 P(E^*|Y) \prod_{n=1}^{N} P(E_n|Y) dY \\ &= \int_0^1 \left(DY^{\frac{A}{B}}\right) \prod_{n=1}^{N} \left(Y^{\frac{C_n}{B}}\right) dY \\ &= \int_0^1 \left(DY^{\frac{A+\sum_n C_n}{B}}\right) dY \\ &= D\frac{1}{\frac{A+\sum_n C_n}{B}+1} \\ &= D\frac{1}{\left(\frac{A+\sum_n C_n + B}{B}\right)} = BD \end{aligned}$$

□

C Proof of unbiased estimator for p_{repeat}

Claim. *Let p denote the probability of some Bernoulli event. an unbiased estimator of p given by a finite set N of samples $Z_{1:N}$ from the distribution is given by:*

$$\hat{p} = \frac{1}{N}\sum_{i=1}^{N} Z_i$$

An unbiased estimator of $p_{repeat} = p^2 + (1-p)^2$ is:

$$\frac{N}{N-1}(\hat{p}^2 + (1-\hat{p})^2) - \frac{1}{N-1}$$

Proof. Calculate the expectation of $\hat{p}^2$:

$$\mathbb{E}(\hat{p}^2) = E\left(\left(\frac{1}{N}\sum_{i=1}^{N} Z_i\right)^2\right)$$

Expand the square inside the expectation:

$$\mathbb{E}(\hat{p}^2) = \frac{1}{N^2}\mathbb{E}\left(\sum_{i=1}^{N} Z_i^2 + \sum_{i\neq j} Z_i Z_j\right)$$

Since $Z_i^2 = Z_i$, and by linearity of expectation:

$$\mathbb{E}(\hat{p}^2) = \frac{1}{N^2}\left(Np + N(N-1)p^2\right)$$

Simplify the expression:

$$\mathbb{E}(\hat{p}^2) = \frac{Np + N^2p^2 - Np^2}{N^2} = \frac{p + (N-1)p^2}{N}$$

Using this result, we have the following:

$$\begin{aligned}
\mathbb{E}(\hat{p}^2 + (1-\hat{p})^2) &= E\left(2\hat{p}^2 - 2\hat{p} + 1\right) \\
&= 2E(\hat{p}^2) - 2E(\hat{p}) + 1 \\
&= 2\left(\frac{1}{N}p + \frac{N-1}{N}p^2\right) - 2p + 1 \\
&= \frac{1}{N}\left((N-1)(2p^2 - 2p + 1) + 1\right) \\
&= \frac{N-1}{N}\left((2p^2 - 2p + 1) + \frac{1}{N-1}\right) \\
&= \frac{N-1}{N}p_{repeat} + \frac{1}{N}
\end{aligned}$$

Therefore, we can debias the naive estimator using the following expression:

$$\frac{N}{N-1}(\hat{p}^2 + (1-\hat{p})^2) - \frac{1}{N-1}$$

□

D Justification for ensembling procedure

Theorem. *Suppose we have a set of categorical distributions $\{P_i\}_{i=1}^n$, define Q^* as the distribution minimizing the average forward Kullback-Leibler divergence to each $\{P_i\}_{i=1}^n$:*

$$Q^* = \arg\min_Q \frac{1}{n}\sum_{i=1}^n D_{KL}(Q\|P_i) \tag{3}$$

then $Q^*(x)$ can be expressed as

$$Q^*(x) = \frac{1}{Z}\prod_{i=1}^n P_i(x)^{\frac{1}{n}} \tag{4}$$

where Z is the normalisation constant to ensure Q^* defines a valid probability distribution function

$$Z = \sum_x \prod_{i=1}^n P_i(x)^{\frac{1}{n}}$$

Proof. Expanding the KL divergence

$$\frac{1}{n}\sum_{i=1}^n D_{\text{KL}}(Q\|P_i) = \frac{1}{n}\sum_{i=1}^n\sum_x Q(x)\log\frac{Q(x)}{P_i(x)}$$

Changing the order of sums, this can be re-expressed as

$$\frac{1}{n}\sum_{i=1}^n D_{\text{KL}}(Q\|P_i) = \frac{1}{n}\sum_x Q(x)\log\frac{Q(x)^n}{\prod_{i=1}^n P_i(x)} = \sum_x Q(x)\log\frac{Q(x)}{\prod_{i=1}^n P_i(x)^{\frac{1}{n}}}$$

Introducing the normalisation constant Z

$$\frac{1}{n}\sum_{i=1}^n D_{\text{KL}}(Q\|P_i) = \sum_x Q(x)\log\frac{\frac{1}{Z}Q(x)}{\frac{1}{Z}\prod_{i=1}^n P_i(x)^{\frac{1}{n}}} = \frac{1}{n}\sum_{i=1}^n D_{\text{KL}}(Q\|P_i)$$

separating the Z in the numerator

$$\frac{1}{n}\sum_{i=1}^{n} D_{\mathrm{KL}}(Q\|P_i) = \sum_x Q(x)\log\frac{Q(x)}{\frac{1}{Z}\prod_{i=1}^{n} P_i(x)^{\frac{1}{n}}} + \log\frac{1}{Z}$$

thus

$$\frac{1}{n}\sum_{i=1}^{n} D_{\mathrm{KL}}(Q\|P_i) = D_{\mathrm{KL}}(Q\|\frac{1}{Z}\prod_{i=1}^{n} P_i^{\frac{1}{n}}) + \log\frac{1}{Z}$$

this will be minimised when the right-hand side KL is equal to zero which occurs at $Q^*(x) = \frac{1}{Z}\prod_{i=1}^{n} P_i(x)^{\frac{1}{n}}$ □

E Experimental details

For all experiments, answer generation is done using language models quantised to bfloat16 (Dean et al., 2012). Chat and instruction-tuned models are prompted using default templates whereas base models (mistral, llama3-base) are prompted with a template consisting of a single in-context example to help steer away from off-topic answers. The addition of this in-context example was found to not materially impact the efficacity of Gumbel sampling but impact the quality of responses.

To avoid excessive experiment run-times, we restrict generated responses to a maximum length after which we interrupt text generation. This limit was set to 50 new tokens for semantic similarity experiments and to 200 new tokens for stylistic similarity experiments. We ran small-scale experiments with larger maximum response length and did not find material evidence of experimental findings being impacted by this response truncation.

When measuring semantic similarity between responses we measure consistency between responses rather than response-question pairs with any follow-up questions or answers hallucinated by the language model removed programmatically.

F Stylistic consistency prompts

F.1 Aleatoric-list

Listing 1: Zero-shot classification prompt for whether model-generated response contains bulletpoints. Placeholders for question-specific content are shown in red.

```
Does the following response separate items in the answer using bullet points (*/-),
    letters(a/b/c...), numerics (1,2,3) or if items are not separated respond with `
    doesn't separate`?

Response:```{response}```.
To make your answer easy to extract respond with only one of the following options `
    uses bullets`/`uses letters`/`uses numerics`/`doesn't separate`
```

Listing 2: Zero-shot classification prompt for whether model-generated response is terse. Placeholders for question-specific content are shown in red.

```
Is the response terse or not?
Response:```{response}```.
To make your answer easy to extract respond with only one of the following options `
    terse`/`not terse`.
```

F.2 Code-Alpaca

Listing 3: Zero-shot classification prompt for determining programming language of model-generated response. Placeholders for question-specific content are shown in red.

```
What is the programming language used in the provided response. If no programming
    language is used return None
Response:```{response}```.
Your response should only contain the answer and nothing else.
```

Listing 4: Zero-shot classification prompt for determining if model-generated response contains comments. Placeholders for question-specific content are shown in red.

```
Does the code snippet in the following response contain comments.
Response:```{response}```.
Answer either with yes/no. Your response should only contain the answer and nothing
    else.
```

Listing 5: Zero-shot classification prompt for determining if model-generated response contains code snippet. Placeholders for question-specific content are shown in red.

```
Does the following response contain a code snippet.
Response:```{response}```.
Answer either with yes/no. Your response should only contain the answer and nothing
    else.
```

G Example responses and semantic similarity

G.1 Llama3-base-7B without gumbel consistency sampling at T=0.8

Table 3: Examples of semantic similarity between responses for various questions

Question	**Response 1**	**Response 2**	**Similarity (%)**
Break down the application of word embeddings in Natural Language Processing / Illuminate the use of word embeddings in Natural Language Processing	Natural Language Processing(NLP) is a branch of Artificial Intelligence (AI) which deals with the interaction between computers and human languages. Word2vec is the most popular class of word embeddin...	It is used to represent words and their context with numerical values.	68.0
Expound on the function of a computer motherboard. / Illustrate the function of a computer motherboard.	The motherboard is the framework of the computer. It houses the CPU, RAM, and other components.	The computer's motherboard serves a number of functions. It directs information from the CPU to the memory, makes sure the proper voltage levels are reached, and directs the data flow in and out of th...	80.5
Decode the abbreviation DNA. / Share the meaning behind DNA.	DNA is an abbreviation for deoxyribonucleic acid.	DNA is also known as Deoxyribonucleic Acid. It is found in every single cell in the body and each cell contains the complete set of genetic information. It is the genetic blueprint that determines the...	78.0

G.2 Llama3-base-7B with gumbel consistency sampling (recycling) at T=0.8

Table 4: Examples of semantic similarity between responses for various questions

Question	Response 1	Response 2	Similarity (%)
Break down the application of word embeddings in Natural Language Processing / Illuminate the use of word embeddings in Natural Language Processing	Embedding is the encoding of words in the context with a real-valued vector of a certain dimension. The low dimension helps to reduce the computational complexity of the models. Embedding can be learn...	Embedding is the encoding of words in the context of a sentence with a real-valued vector. The use of word embeddings helps to solve the problem of 'Word sense disambiguation'.	85.0
Expound on the function of a computer motherboard. / Illustrate the function of a computer motherboard.	The motherboard of a computer is responsible for providing a workspace where all the hardware components can be connected. The motherboard also controls the communication between the various component...	The motherboard of a computer is responsible for providing communication between all the various components. The motherboard also controls basic functions such as the timing of the hardware, start-up ...	90.0
Decode the abbreviation DNA. / Share the meaning behind DNA.	DNA stands for deoxyribonucleic acid	DNA means Deoxyribonucleic acid	98.0

G.3 Llama3-instruct-7B without gumbel consistency sampling at T=0.8

Table 5: Examples of semantic similarity between responses for various questions

Question	Response 1	Response 2	Similarity (%)
Break down the application of word embeddings in Natural Language Processing / Illuminate the use of word embeddings in Natural Language Processing	Word embeddings are a fundamental component in Natural Language Processing (NLP) that allow machines to understand the nuances of language by mapping words to vectors in a high-dimen...	The wonders of word embeddings! Word embeddings are a fundamental concept in Natural Language Processing (NLP) that revolutionized the way we represent words as vect...	91.0
Expound on the function of a computer motherboard. / Illustrate the function of a computer motherboard.	The motherboard is the main circuit board of a computer, connecting and supporting all the hardware components of the system. It serves as a communication highway, allowing different...	The computer motherboard is the main circuit board of a computer that connects all the hardware components together. It is responsible for facilitating communication and data transfe...	91.0
Decode the abbreviation DNA. / Share the meaning behind DNA.	A classic one! The abbreviation DNA stands for: D - Deoxy N - Nucleic A - Acid. In other words, DNA is Deoxyribose N...	What an fascinating topic! DNA (Deoxyribonucleic acid) is a molecule that contains the genetic instructions used in the development and function of all living organisms. It's often r...	78.0

G.4 Llama3-instruct-7B with gumbel consistency sampling (recycling) at T=0.8

Table 6: Examples of semantic similarity between responses for various questions

Question	Response 1	Response 2	Similarity (%)
Break down the application of word embeddings in Natural Language Processing / Illuminate the use of word embeddings in Natural Language Processing	Embedding is the encoding of words in the context with a real-valued vector of a certain dimension. The low dimension helps to reduce the computational complexity of the models. Embedding can be learn...	Embedding is the encoding of words in the context of a sentence with a real-valued vector. The use of word embeddings helps to solve the problem of 'Word sense disambiguation'.	85.0
Expound on the function of a computer motherboard. / Illustrate the function of a computer motherboard.	The motherboard of a computer is responsible for providing a workspace where all the hardware components can be connected. The motherboard also controls the communication between the various component...	The motherboard of a computer is responsible for providing communication between all the various components. The motherboard also controls basic functions such as the timing of the hardware, start-up ...	90.0
Decode the abbreviation DNA. / Share the meaning behind DNA.	DNA stands for deoxyribonucleic acid	DNA means Deoxyribonucleic acid	98.0

H Evaluation of semantic similarity using different metric choices

H.1 Jaccard similarity

In Table 7, we reproduce mean semantic similarity results quoted in Section 6.1 using Jaccard similarity where we measure Jaccard similarity on the set of tokens produced by each model's own associated tokenizer.

Model	Sampler	Without Ensembling	With Ensembling
Llama2 Chat	IS	0.351±0.002	0.385±0.011
	GCS	0.420±0.006	0.495±0.005
	GCSwR	**0.444±0.005**	**0.521±0.007**
Mistral	IS	0.115±0.004	0.119±0.005
	GCS	0.266±0.010	0.340±0.020
	GCSwR	**0.335±0.013**	**0.393±0.008**
Llama3 Instruct	IS	0.320±0.011	0.346±0.000
	GCS	0.365±0.017	0.442±0.004
	GCSwR	**0.410±0.003**	**0.479±0.008**
Llama3 Base	IS	0.086±0.001	0.096±0.009
	GCS	0.224±0.015	0.281±0.009
	GCSwR	**0.314±0.012**	**0.371±0.010**

Table 7: Model results by sampler type, for the Jaccard similarity, with and without our ensembling approach. Scores shown as mean±std.err with std.err obtained from 3 independent runs. Bold indicates highest scores for each model in both ensembling categories.

H.2 Sentencebert

In Table 8, we reproduce mean semantic similarity results quoted in Section 6.1 using the `all-mpnet-base-v2` model recommended by the popular popular `sentencetransformer` repository (Reimers and Gurevych, 2019).

Model	Sampler	Without Ensembling	With Ensembling
Llama2 Chat	IS	0.758±0.006	0.772±0.002
	GCS	0.799±0.001	0.827±0.003
	GCSwR	**0.801±0.005**	**0.834±0.005**
Mistral	IS	0.463±0.006	0.461±0.025
	GCS	0.600±0.006	0.647±0.022
	GCSwR	**0.630±0.017**	**0.664±0.016**
Llama3 Instruct	IS	0.778±0.002	0.794±0.001
	GCS	0.795±0.008	0.827±0.005
	GCSwR	**0.800±0.004**	**0.832±0.003**
Llama3 Base	IS	0.429±0.019	0.443±0.014
	GCS	0.558±0.013	0.593±0.012
	GCSwR	**0.597±0.009**	**0.641±0.009**

Table 8: Model results by sampler type, for the `all-mpnet-base-v2` model, with and without our ensembling approach. Scores shown as mean±std.err with std.err obtained from 3 independent runs. Bold indicates highest scores for each model in both ensembling categories.

I Python implementation

We provide below a self-contained reference Python implementation of our GCSwR algorithm.

```python
from typing import Optional, Union

import numpy as np
import numpy.typing as npt
import torch


def uniform_to_gumble_fn(z, mu=0, beta=1):
    samples = mu - beta * torch.log(-torch.log(z))
    return samples


def gumbel_to_uniform_fn(z, mu=0, beta=1):
    uniform_samples = torch.exp(-torch.exp(-(z - mu) / beta))
    return uniform_samples


class GumbelSampler:
    def __init__(
        self,
        rng_seed: int,
        memory_size: int = 100,
        recycle_strategy: Union[str, int] = "always",
    ) -> None:
        self.rng_seed = rng_seed
        self.is_initialised = False
        self.memory_size = memory_size
        self.recycle_strategy = recycle_strategy

    def sample(self, logprobs: torch.Tensor) -> int:
        gumbel_noise = self.get_current_gumbel(num_cats=len(logprobs))
        sampled_idx = torch.argmax(gumbel_noise.squeeze() + logprobs.squeeze()).item
            ()
        self.recycle_gumbel(gumbel_noise, logprobs, sampled_idx)
        return sampled_idx

    def get_current_gumbel(self, num_cats: int = None) -> torch.Tensor:
        """Get the current gumbel noise vector and initialize gumbel noise if it has
            not yet been initialized."""
        if not self.is_initialised:
            self.rng = np.random.default_rng(self.rng_seed)
            self.gumbel_mem = self.make_gumbel_noise(
                num_cats=num_cats, num_samples=self.memory_size, rng=self.rng
            )
            self.mem_loc = torch.zeros(num_cats, dtype=torch.int64)
            self.is_initialised = True
        gumbel_noise = torch.gather(self.gumbel_mem, 1, self.mem_loc[:, None])
        return gumbel_noise

    def set_current_gumbel(self, gumbel_noise: torch.Tensor) -> None:
        self.gumbel_mem.scatter_(1, self.mem_loc[:, None], gumbel_noise[:, None])

    def recycle_gumbel(
        self,
        gumbel_noise: torch.Tensor,
        logprobs: torch.Tensor,
        sampled_idx: int,
    ) -> None:
        if self.recycle_strategy == "never":
            self.mem_loc += 1
        else:
            uniform_noise = gumbel_to_uniform_fn(gumbel_noise)
            scaler = gumbel_to_uniform_fn(
                logprobs[sampled_idx] + gumbel_noise[sampled_idx] - logprobs
            )
            updated_gumbel = uniform_to_gumble_fn(uniform_noise.squeeze() / scaler)
            self.set_current_gumbel(updated_gumbel)
            self.mem_loc[sampled_idx] += 1

    @staticmethod
    def make_gumbel_noise(
        num_cats: int,
        num_samples: int,
        rng: Optional[np.random._generator.Generator] = None,
    ) -> np.array:
        if rng:
            return torch.Tensor(rng.gumbel(0, 1, size=(num_cats, num_samples)))
        else:
            return torch.Tensor(np.random.gumbel(0, 1, size=(num_cats, num_samples))
                )

    def reset(self):
        self.is_initialised = False


def sample_n_new(
    n: int,
    N_vocab: int,
    rng_seed: Optional[int] = 0,
    logprobs: Optional[npt.NDArray] = None,
) -> list[int]:
    if logprobs is None:
        norm_probs = np.random.dirichlet(np.ones(N_vocab))
        logprobs = np.log(norm_probs)
    sampler = GumbelSampler(rng_seed=rng_seed)
    sequence = []
    for _ in range(n):
        sequence.append(sampler.sample(torch.Tensor(logprobs)))
    return sequence


if __name__ == "__main__":
    n = 2
    N_vocab = 2
    num_samples = 10000
    logprobs = np.log(np.array([0.4, 0.6]))

    results = []
    for seed in range(num_samples):
        results.append(
            sample_n_new(
                n,
                N_vocab,
                seed,
                logprobs,
            )
        )

    print(f"IS: Probability of sampling 1 in both positions: {np.exp(logprobs)[1]}")
    print(
        f"GCSwR: Empirical probability of sampling 1 in positions 1 & 2: {np.array(
            results).mean(axis=0)}"
    )
```