Intent Factored Generation: Unleashing the Diversity in Your Language Model

Eltayeb Ahmed*
BOLD
University of Oxford

Uljad Berdica*
BOLD
University of Oxford

Martha Elliott
BBC Research & Development

Danijela Horak†
Epidemic Sound

Jakob N. Foerster
BOLD
University of Oxford

Abstract

Obtaining meaningfully diverse high-quality samples from Large Language Models (LLMs) for a fixed prompt remains an open challenge. Current methods often only operate at the token-level, paraphrasing the same response. This is problematic because it leads to poor exploration on reasoning problems and to unengaging, repetitive conversational agents. To address this, we propose **I**ntent **F**actored **G**eneration (**IFG**), factorising the sampling process into two stages. First, we sample a *semantically dense* intent that anchors the sample, e.g., a summary or keywords. Second, we sample the final response conditioning on both the original prompt and the intent from the first stage. This *factorisation* allows the use of a higher temperature during the intent step to promote conceptual diversity, and a lower temperature during the final generation to ensure the outputs are coherent. We find that prompting the model to explicitly state its intent for each step of the chain-of-thought before generating the step is beneficial for reasoning tasks. We show that our simple method is highly effective across a diverse set of tasks. We find that this method improves both exploration and final performance on math and code tasks, and combines well with Reinforcement Learning from Verifier Feedback (RLVF). We also show that our method can be used to increase the diversity of instruction-tuned models. Finally, we demonstrate that our method leads to higher quality diverse samples on a language modelling task, on a new dataset that we open-source. IFG is easy to implement and can be used with off-the-shelf LLMs to improve the performance-diversity tradeoff. Full Implementation at `github.com/FLAIROx/IFG`

1 Introduction

Large Language Models (LLMs) are becoming more pervasive, with applications spanning from chatbots to programming agents (Schick et al., 2023) and creative writing aides (Toplyn, 2023; Lee et al., 2022). Despite their strong task performance, LLM responses often lack meaningful diversity (Peeperkorn et al., 2024) beyond the token level. This lack of diversity is more pronounced in instruction-tuned models (Zhang et al., 2023), including most popular commercial LLMs.

There are many situations where generating a *semantically diverse* (Kuhn et al., 2023) set of responses is desirable. For example, in verifiable reasoning domains such as maths and code, sampling a wide range of diverse candidate solutions increases the likelihood of finding a correct solution (Ehrlich

*Equal contribution. Correspondence: `{eltayeb,uljadb}@robots.ox.ac.uk`.
†Work done while at BBC Research & Development.

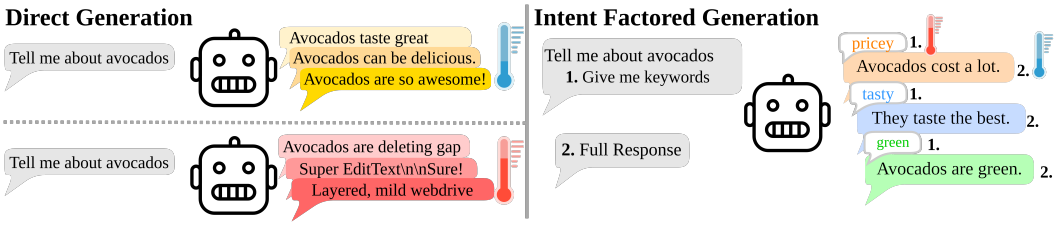


Figure 1: In Intent Factored Generation (IFG), we first sample a semantically dense intent such as keywords or a summary. We then sample the final response anchored to this intent. By raising the temperature during the intent phase, the responses are more semantically diverse.

et al., 2025). Combining this with a reliable verifier at test-time can result in a strong problem-solving agent (Chervonyi et al., 2025). Furthermore, semantic diversity is beneficial for exploration when doing Reinforcement Learning from Verifier Feedback (RLVF) to increase reasoning capabilities on tasks like maths and code (Havrilla et al., 2024).

Raising the token selection temperature is a common practice that increases token-level diversity, but often fails to significantly increase semantic diversity before leading to a breakdown of coherence (Peeperkorn et al., 2024). Diverse Beam Search (DBS) (Vijayakumar et al., 2016) is a much slower alternative due to its sequential nature. Similar to temperature increase, DBS directly enforces token-level diversity between the sequences, which does not always lead to semantic differences. Lastly, DBS is a deterministic decoding procedure that yields identical generations across seeds, further limiting its utility.

In this paper, we present **I**ntent **F**actored **G**eneration (**IFG**), a simple method to increase semantic diversity, as depicted in Figure 1. At its core, IFG separates the sampling process into two stages. In the first stage, we sample a short *intent*, such as keywords or a dense summary, that describes the sample to be generated. In the second stage, we sample the final *response* conditioned on this intent and the original prompt. Factorising into two stages allows us to independently control the temperature for the intent and the final response. A high temperature for the intent increases semantic diversity as the intent compactly represents the concepts in the response; changing even a single token in the intent can cause large semantic changes to the response. The use of a lower temperature for the final response keeps it coherent and self-consistent.

IFG can be implemented in two ways. First, at inference time, by prompting off-the-shelf LLMs. Second, by finetuning LLMs on datasets annotated with intents — annotations that can themselves be generated automatically by LLMs. We show that both IFG implementations improve performance.

Our method is conceptually simple, and displays empirical gains on a wide range of tasks from maths and code, to chatbots and language modelling tasks out-of-the-box, including thinking models. Specifically, IFG results in better exploration for reasoning tasks, leading to significant improvements in pass@k on maths and coding problems on the MATH (Hendrycks et al., 2021) and LiveCodeBench (Jain et al., 2024) benchmarks respectively. We show that IFG leads to better RLVF (RL from Verifier Feedback) performance on maths tasks. We also demonstrate that IFG increases diversity of responses without compromising on quality. To do this we use Semantic Entropy to measure diversity, following the methodology in Kuhn et al. (2023), with minor adaptations to use modern LLMs instead of BERT models to compute the metric (details in Appendix C.2), and we use task-related metrics to measure quality. Second, we collect a dataset of comments from Reddit and show that IFG pushes up the quality-diversity Pareto frontier for models finetuned on this data. Additionally, we study IFG in the instruction tuning setting using Direct Policy Optimisation (DPO), also finding an improvement in the quality-diversity Pareto frontier.

To summarise, our contributions are:

1. **Intent Factored Generation:** A method to sample meaningfully diverse high quality responses from LLMs (Section 4.1).
2. **Exploration for Reasoning:** We empirically demonstrate the utility of IFG in improving exploration. We show the benefits of this exploration both stand-alone and as a component of RL with Verifier Feedback (RLVF) (Section 6.1, Section 6.2).
3. **Diverse Aligned Models:** IFG applied to instruction tuning results in aligned models with higher semantic diversity (Section 6.3).
4. **Diverse Comment Generation:** We demonstrate that on a comment-generation task, IFG produces more semantically diverse responses than relevant baselines while maintaining coherence (Section 6.4). To support this evaluation and enable reproduction, we release a curated dataset of news articles paired with Reddit user comments.

2 Background

Causal Language Modelling Given a corpus $\mathcal{D}$ of natural language text documents $(x_1, x_2, \cdots, x_n)$, language modelling is the process of learning an estimated distribution $p(x)$ from the dataset. Since natural language is usually intended to be read sequentially, it is common to factorise p as follows:

$$p(x) = \prod_{i=1}^{n} p(s_i | s_1, \cdots, s_{i-1}) \tag{1}$$

where $s_1, \cdots, s_t$ are the individual tokens that make up the text document x. This factorisation is known as Causal Language Modelling (Bengio et al., 2000). This formulation facilitates generating text sequentially, both from scratch or in continuation from a given prefix, known as a *prompt*. Recently, the Causal Transformer architecture has been applied to Causal Language Modelling with great success (Radford et al., 2019), and has favourable scaling characteristics.

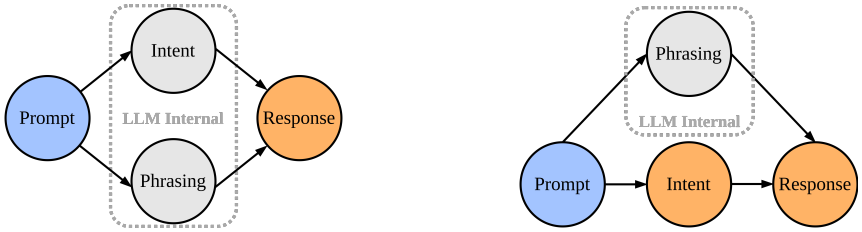
Semantic Entropy Given an LLM M, a prompt p_i and an induced distribution of responses $r \sim M(p_i)$, the Semantic Entropy $\mathrm{SE}_M(p_i)$ measures the conceptual uncertainty of M given the prompt p_i (Kuhn et al., 2023). To compute $\mathrm{SE}_M(p_i)$, we define a set of equivalence classes $c_i \in \mathcal{C}$ where $r_j, r_k \in c_i$ if and only if r_j and r_k bidirectionally entail each other, i.e., they are paraphrases of the exact same statement. We then compute entropy of the distribution over equivalence classes induced by the LLM. We determine bidirectional entailment by prompting an LLM to judge the similarity between r_j and r_k. In Appendix C we list the algorithm to compute SE over a set of generations using an LLM to judge bidirectional entailment. In Appendix C.3 we find semantic entropy measurements to be consistent across different choices of judge models.

RL from Verifier Feedback Reinforcement Learning is a training paradigm where a model M interacts with an environment to learn a mapping from states s to actions a that maximises a reward function $R(s, a)$ (Sutton, 2018). With LLMs, we can model reasoning questions as states, the step-by-step solutions as actions and the accuracy of the final answers as the reward. This allows us to apply RL to improve the LLM's ability to reason in domains such as maths and code (Havrilla et al., 2024; DeepSeek-AI et al., 2025; Kazemnejad et al., 2024). In this work, we leverage the Self-Taught Reasoning algorithm (STaR) (Zelikman et al., 2022), which iteratively builds a dataset by sampling solution attempts from the model under training. This is equivalent to a policy gradient algorithm (Sutton, 2018) with resets (Nikishin et al., 2022). In this work we will use STaR to refer to the version of the algorithm from Zelikman et al. (2022) which does not employ *post-rationalisation*.

Instruction Tuning Pre-trained language models are usually not adept at following instructions or responding in a conversational manner. This can be remediated with a post-training phase called instruction-tuning (Wei et al., 2022). Although supervised fine-tuning leads to an improvement in instruction following, further gains can be obtained through RLHF (Zhang et al., 2023), or using off-policy equivalent algorithms such as Direct Preference Optimisation (DPO) (Rafailov et al., 2023).

3 Related Work

Methods for Diverse Generation Raising the sampling temperature is the most common way to increase output diversity, but it operates at the token level and often fails to yield semantically diverse



(a) Sampling from an LLM.

(b) IFG sampling from an LLM.

Figure 2: In IFG we sample the intent explicitly (orange) instead of latently (grey). We sample the intent with a higher temperature to induce semantic diversity while using a lower temperature for the final response to maintain coherence.

responses before coherence breaks down (Peeperkorn et al., 2024). Diverse Beam Search (Vijayakumar et al., 2016) penalises repeated tokens across sequentially generated sequences, but suffers from the same token-level limitation as well as high latency due to its sequential nature. Nguyen et al. (2024) introduce **min-p** sampling, which dynamically scales the probability threshold by the most likely token; we include it as a baseline. Ruan et al. (2025) similarly intervene on the next-token distribution via auxiliary guides. Both are orthogonal to IFG, which does not manipulate output logits. Other approaches increase diversity by combining base and instruction-tuned models (Zhu et al., 2025), which doubles memory cost and is bounded by the base model's diversity, or by forcing generations into pre-defined categories via MAP-Elites (Samvelyan et al., 2024; Mouret and Clune, 2015), which requires manually specifying axes of variation per task.

Structured Generations Control Tokens (Keskar et al., 2019) prepend coarse-grained tokens to condition generation on document type, but cannot induce diversity within a type. Wang et al. (2024) learn fine-grained planning tokens, which require training from scratch and are not interpretable. Tan et al. (2020) and Yao et al. (2019) generate sequences in multiple stages, starting from a keyword scaffold; both train separate models per stage, plan to completion before generation, and do not study reasoning tasks or interleaved planning. MCTS-based approaches such as Zhang et al. (2024a) are orthogonal frameworks, only applicable when a value function is defined, that rely on the underlying model's diversity to define actions, and hence can be combined with sampling methods such as IFG.

Temperature Scheduling A complementary line of work varies the temperature across decoding steps. Chang et al. (2023) modulate the temperature via the KL-divergence between two parallel decodes (with and without a grounding source) to balance diversity and attribution. Zhu et al. (2024) raise the temperature on "challenging" tokens at the start of a code block and lower it on syntax-constrained "confident" tokens. Zhang et al. (2024b) set the per-step temperature as a function of the next-token entropy. These methods schedule diversity at the token level using local signals. IFG instead schedules diversity at the *semantic* level: a high-temperature intent stage anchors a coherent low-temperature response, with the schedule determined by generation structure rather than local distribution properties. We provide a more detailed related work section in Appendix B.

4 Method

Here we outline our Intent Factored Generation (IFG) method and provide information theoretic motivations for design choices.

4.1 Intent Factored Generation

When sampling a response **r** to a prompt **p** from an LLM M the response can be described by an *intent* which defines the response conceptually, and *phrasing* which describes the wording and formatting. As shown in Figure 2a, when sampling from LLMs, *intent* and *phrasing* are latent variables, and although increasing temperature will increase the entropy of both, the increase tends to be dominated by changes in *phrasing* (Peeperkorn et al., 2024). Instead, we introduce our **I**ntent **F**actored **G**eneration (**IFG**) method in which we begin by explicitly sampling the intent **i**. This serves as a "summary representation" of the response, expressed as keywords or a brief summary. We then sample the final response, **r**, conditioning on both the prompt **p** and intent **i**. Some examples of IFG

intents and *responses* are shown in Figure 5b. Applying IFG changes the sampling process from $\mathbf{r} \sim M(\mathbf{p})$ to $\mathbf{i} \sim M(\mathbf{p}) \Rightarrow \mathbf{r} \sim M(\mathbf{p}, \mathbf{i})$.

We illustrate the two-stage sampling process in Figure 2b. The goal of explicitly sampling the *intent* $\mathbf{i}$ is to allow us to directly control its sampling temperature t_i. We then use a lower temperature t_r to sample the final response and increase semantic diversity while maintaining coherence.

To sample an *intent* from an LLM before sampling a *response* for a given prompt, we must first prime the LLM to do so. We can either few-shot prompt the LLM (**Few-shot Prompted IFG**) or, given a dataset from the target distribution, finetune the LLM on intent-annotated data (**Finetuned IFG**). To prepare a dataset for finetuning, we use an LLM to annotate each sequence with an intent, either one per response or, more granularly, one per segment of a longer response. The granular approach enables segment-level IFG sampling, alternating between t_i and t_r as we switch between intent and response. A formal description with examples of intent annotation is in Appendix H, and prompts for intent labelling in Appendix M.2.

4.2 Information-Theoretic Motivation

To motivate our design choices for IFG, we analyse the entropy of the response $H(\mathbf{r})$. For any response $\mathbf{r}$ sampled conditionally on an intent $\mathbf{i}$; $\mathbf{r} \sim p(\mathbf{r}|\mathbf{i})\mathrm{p}(\mathbf{i})$, the following equality holds for the Shannon entropy $H(\mathbf{r})$.

$$H(\mathbf{r}) = \underbrace{H(\mathbf{i})}_{(1)} + \underbrace{H(\mathbf{r} \mid \mathbf{i})}_{(2)} - \underbrace{H(\mathbf{i} \mid \mathbf{r})}_{(3)} \tag{2}$$

The derivation can be found in Appendix E.1. As the semantic entropy SE_M of $\mathrm{p}(\mathbf{r})$ is the true Shannon entropy of a distribution derived from $\mathrm{p}(\mathbf{r})$, it can be shown that the above relationship also holds for semantic entropy.

Term (1) shows that raising the intent temperature t_i increases $H(\mathbf{i})$, and thus $H(\mathbf{r})$. Term (2) captures the entropy added by the second sampling stage, while term (3) captures the information from $\mathbf{i}$ that is discarded in $\mathbf{r}$, subtracting from the total. As $t_r \to \infty$, $\mathbf{r}$ becomes independent of $\mathbf{i}$ and term (3) reaches its maximum of $H(\mathbf{i})$, fully cancelling the contribution from term (1). So increasing t_i unambiguously raises $H(\mathbf{r})$, whereas increasing t_r has an ambiguous effect. This motivates our design choice of $t_i > t_r$, which we empirically validate in Appendix F.1. Further design choices of the style of intents motivated by this analysis are included in Appendix E.2.

5 Experimental Setup

Baselines: We compare our method, IFG, for which we tune t_i and t_r against the following baselines. 1) **direct** or vanilla sampling method, for which we tune temperature t. 2) **min-p**, for which we tune **min-p** and temperature t. 3) **DBS** for which we tune the relevant hyperparameters. We allocate an equal tuning budget per hyperparameter for IFG and the baselines. For Sections 5.1 and 5.2, the budget is 10 tuning runs on the training split.

5.1 Mathematical Reasoning

To investigate the efficacy of IFG in finding solutions to mathematical problems, we conduct experiments on the MATH dataset of maths questions (Hendrycks et al., 2021) using the `Qwen-2.5` (Yang et al., 2024) models. For the **direct** baseline, we few-shot prompt the models to solve the given questions with Chain-of-Thought reasoning (Wei et al., 2023). We use the same prompt with the **min-p** baseline as well. We compare these baselines to **Few-shot Prompted IFG**, with every step being conditioned on an IFG-style intent. We also evaluate **min-p**+IFG to assess how the two compose, as the two methods are orthogonal. All prompts are listed in Appendix M.

To assess exploration, we evaluate "pass@k" on the test split for various values of k. To see whether improved exploration demonstrated by a high "pass@k" can be translated to higher "pass@1" accuracy through RLVF, we train models on the train split of MATH using both STaR and STaR+IFG. We also investigate the effect of scale on our method across model sizes from 3B to 14B parameters. For STaR, we use a budget of 8 rollouts per problem.

5.2 LiveCodeBench

We evaluate the utility of IFG for coding tasks on the LiveCodeBench benchmark (Jain et al., 2024)—a benchmark of algorithmic competitive programming problems—using the model `Qwen-2.5-Coder-32B` (Hui et al., 2024). We compare the "pass@k" metric for different values of k. For the baselines, we evaluate direct sampling and **min-p** using the standard prompts from the LiveCodeBench codebase. We then modify the prompts to use **Few-shot Prompted IFG**. To do so we add *comments* to the code that contain the intents; we use one intent for every closely related one to three lines of code. Prompts can be found in Appendix M. Both the validation set used for tuning and the test set we report the final results on are explicitly limited to problems from competitions that occurred after the release of the `Qwen-2.5-Coder-32B` model.

5.3 Instruction-Tuning with DPO

The collapse of diversity of LLMs after instruction-tuning is a widely observed phenomenon (Zhu et al., 2025). Hence, we combine IFG with Direct Preference Optimisation (DPO) (Rafailov et al., 2023) to train instruction-tuned models with higher diversity while remaining broadly aligned with the preference dataset. We conduct these experiments on `Qwen-2.5-7B` using the Helpful and Harmless Golden dataset (Cai et al., 2024) that contains pairwise examples of preferred and rejected conversational responses. We compare against vanilla DPO as a baseline. For DPO+IFG, we do **Finetuned IFG**, where we used `Llama3.1-8B-Instruct` to annotate every assistant turn in the original dataset with keyword intents. We provide prompts and examples of intents in Appendix M.

Both models are trained by first performing SFT on the chosen responses from the dataset for one epoch. This is then followed by DPO instruction-tuning on the full pair-wise preference dataset.

To find the Pareto frontier of quality vs diversity, we collect samples from both models using different sampling temperatures t for the DPO model and t_i, t_r for the DPO+IFG model. We measure diversity using Semantic Entropy (SE) and we measure quality with a Bradley-Terry reward model (Zhang et al., 2023) trained on the same HH-Golden dataset.

Fundamentally, instruction tuning aims to align the model with the desirable properties incentivized by the reward function. As a more diverse model potentially possesses a broader behavioral repertoire, we investigate the impact of IFG on this repertoire by evaluating secondary traits not explicitly optimized during training. We do this through the Perspective API (Lee et al., 2022) (See Appendix D.1 for details). Additionally, we assess generation quality by computing perplexity using a model from a distinct architectural family (Appendix D.2).

5.4 Diverse Comment Generation

We curate a dataset of news articles and reader comments from Reddit. We gather over 556k comments on 14k news articles. We collected this data from the `r/news` subreddit. Further details on the composition of the dataset and the preparation procedure are described in Appendix G.2. We open source this dataset exclusively for research use.

We train a **Finetuned IFG** model on this dataset after annotating with IFG intents. We use keyword summaries generated by `Llama3-8B-Instruct` as our intent, similar to Section 5.3. Our **Direct sampling** baseline is a model finetuned on the dataset without IFG intent annotation.

We evaluate the direct model using plain sampling, **min-p**, and DBS. We measure how these methods trade off quality with diversity as we vary the relevant hyperparameters. To measure diversity we use SE and for quality we use a $\mathrm{coherence}$ metric, which is computed as $1 - \mathrm{incoherence}$. We obtain $\mathrm{incoherence}$ through the Perspective API (Lee et al., 2022). We show examples of the SE measurements in Appendix K.1 and examples of the $\mathrm{coherence}$ measurements in Appendix K.2.

6 Results

In this section we report quantitative results of the experiments in Section 5. Examples of the model's responses are in Appendix N.

6.1 Mathematical Reasoning

In Figure 3 (a), we see that the model few-shot prompted with IFG achieves higher pass@k than the same model with a vanilla few-shot prompt. We also see that **min-p**+IFG strictly outperforms **min-p**. This shows that IFG composes with sampling diversity methods that operate at the token level. In Figure 3 (b), we show the test accuracy (pass@1) on MATH as a function of STaR training iterations. For all tested model scales (3B, 7B, 14B), IFG+STaR outperforms STaR alone. The separation is largest in the earlier iterations, showing that IFG increases data efficiency. We note that the improvement due to IFG scales well with the model size.

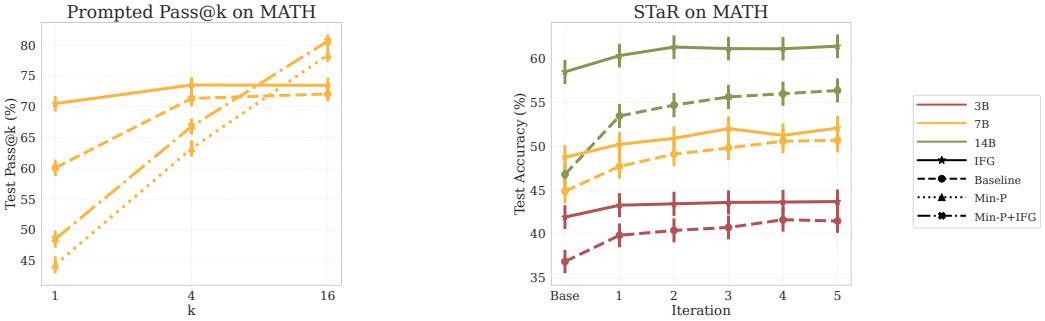


Figure 3: In (a) we plot the pass@k vs k for few-shot prompted models. We achieve the highest pass@k with either IFG or IFG+min-p for all values of k, with **min-p** hindering performance on lower values of k and helping at higher ones. In (b), we plot the test performance (pass@1 with temperature 0) of models trained with STaR. Across scales, STaR+IFG outperforms STaR with higher separation for the largest model, indicating positive scaling, and higher separation in earlier iterations, indicating higher sample efficiency. Errorbars are bootstrapped 95% confidence intervals.

We also conduct an ablation on the effect of the style of intent on pass@k performance. In Table 1, we find that the most effective intent style for solving math problems is a dense summary of the step to come e.g. "Rewrite in terms of sin and cos".

We find that using an overly verbose intent harms performance, and we find that using intents that are not expressive enough to provide semantic grounding for the step that follows, such as a single-keyword intents, is also not effective. These findings align with the design choices in Appendix E.2. In Appendix I.3 we show IFG gains are also consistent in more recent thinking models.

Table 1: Intent Style Ablation on MATH

Intent Style	**Pass@1 (%)**	**Pass@4 (%)**
One word	35.3%	62.2%
Dense summary	**70.5%**	**73.5%**
Verbose	50.1%	67.0%
Direct baseline	60.0%	71.3%

6.2 LiveCodeBench

In Table 2 we show the performance of `Qwen-2.5-Coder-32B` on LiveCodeBench with IFG, as well as the relevant baselines. We see that IFG achieves higher pass@5 and pass@10 than the direct sampling and min-p sampling. Notably, we see that IFG solves more problems in 5 attempts than direct sampling solves in 10 attempts. However, IFG slightly underperforms in pass@1 accuracy. With IFG, individual solutions are less likely to be correct due to higher entropy, but sets of multiple samples are likely to have higher coverage. After tuning, the optimal t_i and t_r for IFG were higher than the optimal t for direct sampling, with $t_i > t_r$. This indicates that IFG stabilises the use of higher temperatures, even on structured tasks.

We tuned temperatures for both the baseline and IFG on the slice of LiveCodeBench from 1 Oct 2024 to 1 Dec 2024 (110 problems) and we tested on the slice from 1 Jan 2025 to 1 May 2025 (182 problems). Further details on the effect of temperature choice on the performance of IFG can be found in Appendix F.

In Table 3 we report the number of questions where at least $1/10$ attempts achieved a specific outcome. Compared to the baseline, IFG increases the count of problems reaching every outcome type, clearly indicating greater diversity and therefore greater coverage of the space of possible solutions.

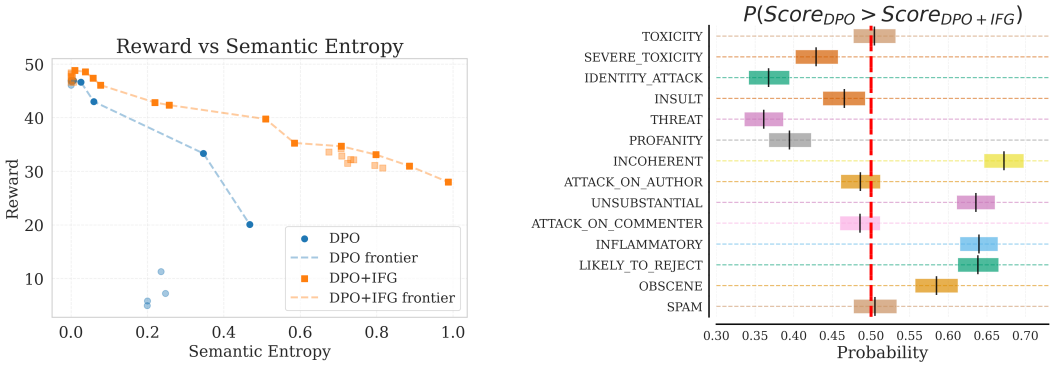
Table 2: Pass@k on LiveCodeBench, $n = 182$

Model	Pass@1	Pass@5	Pass@10
IFG	30 (16.5%)	50 (27.6%)	55 (30.2%)
Direct	34 (18.7%)	45 (25.0%)	48 (26.4%)
min-p	34 (18.7%)	47 (25.8%)	51 (28.0%)

Table 3: LiveCodeBench Outcomes

Outcome	Baseline	IFG	Diff.
Wrong Answer	161	175	+14
Timeout	28	29	+1
Exception	115	119	+4
Success	48	55	+7

6.3 Direct Preference Optimization



(a) The mean reward and diversity (SE) at different temperatures for DPO and DPO+IFG. DPO+IFG achieves higher reward for a given diversity level compared to DPO.

(b) DPO+IFG shows statistically significant improvements on 5 of 14 metrics primarily relating to coherence and content quality, and statistically significant increases on 5 metrics relating to core toxicity.

Figure 4: We compare the performance and diversity of DPO+IFG to DPO in an instruction following setting. DPO+IFG leads to higher rewards for a given diversity level (4a). The Perspective API analysis (4b) shows that DPO+IFG improves on coherence and content quality.

We compare DPO to DPO+IFG as outlined in Section 5.3. We assess the quality of the responses via the reward model we trained, and assess diversity through SE for different values of temperature (t for DPO or t_i, t_r for DPO+IFG). In Figure 4a, we plot these measurements, with points along the Pareto frontier in bold and connected with dotted lines. Across all levels of diversity, DPO+IFG dominates DPO in terms of reward.

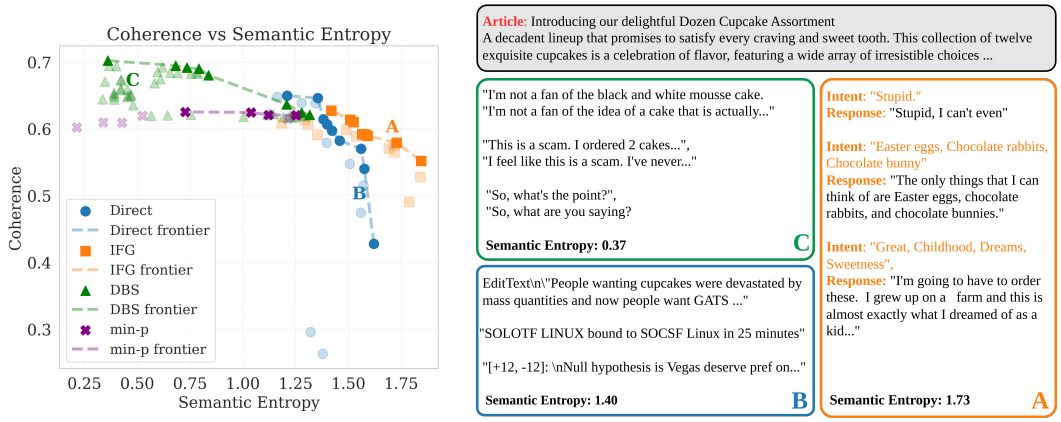
In Figure 4b, we show the probability that DPO scores higher than DPO+IFG for each negative trait. We see that DPO+IFG generations are, with statistical significance, less likely to be deemed incoherent, unsubstantial or to be rejected by moderators, and are no more likely to be classified as spam. Hence, despite the higher diversity, the responses maintain their validity. However, we do see an increase in some metrics relating to toxicity, indicating that this wider behavioral repertoire extends the model's response domain to include more undesirable areas as well.

We note that the factored structure of IFG lends itself naturally to intent-level filtering: since intents are short, semantically dense, and sampled before the full response, a classifier can be applied to the intent to filter undesirable generations cheaply, without committing to the full response. Investigating this filtering is out of scope for this work, which focuses on exploration and diversity rather than safety alignment. We report these results for completeness and to motivate a direction for future work. Further evaluations using different quality and diversity metrics are in Appendix J.3.

The observed toxicity increases warrant further investigation, which we leave to future work. In our separate Reddit experiments, models use vanilla SFT without toxicity-reduction rewards; toxic content is present in the training data, and the observed increases are small in magnitude (Appendix K.5). We hypothesize that broader exploration increases exposure to low-probability undesirable responses. IFG's short intents permit inexpensive filtering before response generation; post-generation safeguards such as Llama Guard Inan et al. (2023) could be used to block sampling final responses for intents that are deemed unsafe, although we do not empirically validate this. Research in diversity and exploration is an integral part of the safety research landscape, as it is an important component for

red-teaming as demonstrated by Rainbow Teaming (Samvelyan et al., 2024). Further quality and diversity evaluations appear in Appendix J.3.

6.4 Diverse Comment Generation



(a) Mean coherence and SE for different temperature points

(b) Examples of comments from selected points in the Entropy Vs Coherence plot as indicated by the letters A,B,C

Figure 5: This plot presents Semantic Entropy (SE) vs Coherence for comments generated on news articles. Each measurement is taken at a different temperature in the range $[0.5, 1.3]$. The corresponding temperature and SE value of each points in (a) is plotted in Appendix F.3.

In Figure 5, we plot measurements of SE and Coherence measured for different sampling hyperparameter values of IFG, direct generation, **min-**p and Diverse Beam Search (DBS). We highlight points on the Pareto frontier for each method, and we note that for higher given values of SE, IFG has higher coherence than direct generation. Measurements at each point are averaged across 100 news articles, with 15 comments generated per article for IFG, direct, and **min**-p, and 8 comments per article for DBS. We use fewer samples for DBS as it uses more VRAM and our experiments are controlled for roughly equal compute. The empirical plug-in estimator of SE is bounded above by $\log n$, where n is the number of samples per article, giving DBS a lower theoretical ceiling ($\log 8 \approx 2.08$) than the other methods ($\log 15 \approx 2.71$). However, DBS's observed SE values lie well below $\log 8$, indicating that its limited diversity reflects its token-level deterministic decoding rather than the sample-size ceiling. In Figure 5 we show examples from points in each method's Pareto frontier. DBS has high coherence, but lower semantic diversity due to its token-wise deterministic nature. IFG achieves the peak diversities while maintaining comment coherence, reaching higher semantic entropies.

To provide clarity on the Perspective API (2024) metric, we show some sample comments and their coherence scores in Appendix K.2. Additionally, in Appendix K.3, we present the same analysis from Figure 5 using perplexity as the measure of quality, which reconfirms findings presented here.

Finally, in Appendix K.5 we measure several semantic characteristics of the comments for both the original dataset and the generated comments from each model, and we find these to match closely. Hence, IFG does not harm the model's ability to faithfully learn the characteristics of the data. In Appendix K.7 we quantify the impact of IFG on wall-clock time on this task.

7 Discussion and Conclusion

The results described in Section 6 show that IFG sampling leads to stronger performance, or higher diversity for a given level of performance across mathematical reasoning, code generation, instruction following, and comment generation tasks. IFG differs from standard sampling in two ways: it elicits a dense up-front intent, and it uses separate temperatures t_i and t_r for the intent and response.

In Appendix F, we conduct an ablation of the two-temperature approach where we constrain $t_i = t_r$ when tuning temperature. We refer to this as "IFG - Equal". For code generation, instruction following and comment generation, we find that IFG outperforms *IFG-Equal*. On code generation, *IFG-Equal* outperforms the baseline but underperforms IFG for pass@5 and pass@10, confirming the benefit of the two temperature approach.

On mathematical reasoning, Figure 6 shows that *IFG - Equal* matches IFG at pass@k when sampling from a base model, with both outperforming the non-IFG baseline. This is corroborated by Figure 3 (b) where IFG outperforms the baseline, before training, when using greedy-decoding ($t_i = t_r = t = 0$), demonstrating that the IFG structure of prompting alone improves performance on some tasks.

Qualitatively comparing the greedy decoding IFG generations to the baseline generations on MATH, we see that IFG sampling encourages the model to take more fine-grained steps when trying to solve maths problems. We attribute this to the IFG prompt adding intents which clearly delineate each step. This can be seen in Appendix N.1, particularly samples 4, 5 and 9. Finally, in Appendix I, we study the solution diversity on MATH and LiveCodeBench and we find that IFG predominantly leads to increased coverage as opposed to increased per-attempt accuracy, confirming that the main motivation to use IFG is increased exploration, as opposed to increased accuracy.

Additional MATH comparisons against string-based baselines and online GRPO results appear in Appendix I.1; validation of LLM-based SE against BERT-based SE appears in Appendix C.3.

We show that out-of-the-box, IFG significantly improves performances across multiple tasks, spanning from reasoning tasks such as maths and code to conversational chatbots, and that it can be composed with other diversity-inducing methods. We hope this robust performance, paired with the ease of implementation and the low computational overhead, leads to future works applying IFG to algorithms or methods that benefit from diversity such as RLHF and RLVF variants, language-based tree search and evolutionary algorithms.

8 Limitations

IFG targets exploration rather than guaranteed pass@1 gains; exploiting reasoning coverage requires multiple samples and reliable verification. Prompted IFG requires the LLM to be capable of adhering to the format, which can be challenging with very small LLMs and incurs additional generation cost (Appendix L). We measure intent diversity indirectly through responses, without isolating conceptual variation from paraphrasing. Intents are prompt-specific and their positions prescribed, limiting pooled keyword-frequency and position analyses. We do not empirically compare against adaptive temperature schedules. Toxicity increases remain unresolved, and neither intent-level nor output-level safety filtering is evaluated here.

Acknowledgements

Uljad Berdica is funded by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems, and the Rhodes Trust. Eltayeb Ahmed is funded by the Academic Futures program, the Oxford Department of Engineering and Lincoln College, Oxford. Jakob Nicolaus Foerster is partially funded by the UKRI grant EP/Y028481/1 (originally selected for funding by the ERC). Jakob Nicolaus Foerster is also supported by the JPMC Research Award and the Amazon Research Award. The authors are grateful to Andy Secker for establishing the collaboration with the BBC and we thank the BBC for its valuable contribution to conceptualizing this project and articulating the central questions of diversity, representation, truth and integrity. The authors also thank Michael Beukman and Tim Franzmeyer for their valuable feedback. The authors acknowledge the use of resources provided by the Isambard-AI National AI Research Resource (AIRR). Isambard-AI is operated by the University of Bristol and is funded by the UK Government's Department for Business, Innovation, Science and Trade (DBIST) via UK Research and Innovation; and the Science and Technology Facilities Council [ST/AIRR/I-A-I/1023] McIntosh-Smith et al. (2024).

References

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023.

Joe Toplyn. Witscript 2: A system for generating improvised jokes without wordplay. *arXiv preprint arXiv:2302.03036*, 2023.

Mina Lee, Percy Liang, and Qian Yang. Coauthor: Designing a human-ai collaborative writing dataset for exploring language model capabilities. In *Proceedings of the 2022 CHI conference on human factors in computing systems*, pages 1–19, 2022.

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492*, 2024.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Ryan Ehrlich, Bradley Brown, Jordan Juravsky, Ronald Clark, Christopher Ré, and Azalia Mirhoseini. Codemonkeys: Scaling test-time compute for software engineering, 2025. URL `https://arxiv.org/abs/2501.14723`.

Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang Nguyen, Marcelo Menegali, Junehyuk Jung, Vikas Verma, Quoc V Le, and Thang Luong. Gold-medalist performance in solving olympiad geometry with alphageometry2. *arXiv preprint arXiv:2502.03544*, 2025.

Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning, 2024. URL `https://arxiv.org/abs/2403.04642`.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. Diverse beam search: Decoding diverse solutions from neural sequence models. *arXiv preprint arXiv:1610.02424*, 2016.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Reddit. Reddit. `https://www.reddit.com`.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

DeepSeek-AI, Daya Guo, Dejian Yang et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Unlocking rl potential for llm reasoning through refined credit assignment, 2024. URL `https://arxiv.org/abs/2410.01679`.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning, 2022. URL https://arxiv.org/abs/2205.07802.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.

Minh Nhat Nguyen, Andrew Baker, Clement Neo, Allen Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint arXiv:2407.01082*, 2024.

Zhiwen Ruan, Yixia Li, Yefeng Liu, Yun Chen, Weihua Luo, Peng Li, Yang Liu, and Guanhua Chen. G2: Guided generation for enhanced output diversity in llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14127–14145, 2025.

Alan Zhu, Parth Asawa, Jared Quincy Davis, Lingjiao Chen, Boris Hanin, Ion Stoica, Joseph E. Gonzalez, and Matei Zaharia. Bare: Combining base and instruction-tuned language models for better synthetic data generation, 2025. URL https://arxiv.org/abs/2502.01697.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *arXiv preprint arXiv:2402.16822*, 2024.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*, 2019.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=wi9IffRhVM.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric P. Xing, and Zhiting Hu. Progressive generation of long text. *CoRR*, abs/2006.15720, 2020. URL https://arxiv.org/abs/2006.15720.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-write: Towards better automatic storytelling, 2019. URL https://arxiv.org/abs/1811.05701.

Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b, 2024a. URL https://arxiv.org/abs/2406.07394.

Chung-Ching Chang, David Reitter, Renat Aksitov, and Yun-Hsuan Sung. Kl-divergence guided temperature sampling. *arXiv preprint arXiv:2306.01286*, 2023.

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Zhi Jin, and Hong Mei. Hot or cold? adaptive temperature sampling for code generation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 437–445, 2024.

Shimao Zhang, Yu Bao, and Shujian Huang. Edt: Improving large language models' generation by entropy-based dynamic temperature sampling. *arXiv preprint arXiv:2403.14541*, 2024b.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Binyuan Hui, Jian Yang, Zeyu Cui et al. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Tianchi Cai, Xierui Song, Jiyan Jiang, Fei Teng, Jinjie Gu, and Guannan Zhang. Ulma: Unified language model alignment with human demonstration and point-wise preference, 2024. URL https://arxiv.org/abs/2312.02554.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

Perspective API. Perspective API: Content moderation attributes and languages, 2024. URL https://developers.perspectiveapi.com/s/about-the-api-attributes-and-languages. Accessed: January 29, 2024.

Simon McIntosh-Smith, Sadaf R Alam, and Christopher Woods. Isambard-ai: a leadership class supercomputer optimised specifically for artificial intelligence, 2024. URL https://arxiv.org/abs/2410.11199.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080.

Jason R Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. Dirt cheap web-scale parallel text from the common crawl. Association for Computational Linguistics, 2013.

George A Miller. Note on the bias of information estimates. *Information theory in psychology: Problems and methods*, 1955.

Liam Paninski. Estimation of entropy and mutual information. *Neural computation*, 15(6):1191–1253, 2003.

Alyssa Lees, Vinh Q Tran, Yi Tay, Jeffrey Sorensen, Jai Gupta, Donald Metzler, and Lucy Vasserman. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pages 3197–3207, 2022.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. The pushshift reddit dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 830–839, 2020.

Leonard Richardson. Beautiful soup documentation. *April*, 2007.

T Wolf. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Appendix

Our Appendix is structured as follows:

- Appendix A contains our impact statement.
- Appendix B contains an extended description of related work.
- Appendix C contains an extended description of the diversity metrics and how we implement their measurement in our experimental setup. We describe Semantic Entropy (Kuhn et al., 2023) and Self-Bleu (Zhu et al., 2018).
- Appendix D contains an extended description of the quality metrics we use and how we implement their measurement in our experimental setup. We cover the Perspective API (Perspective API, 2024), perplexity and win rate.
- Appendix E provides an information theoretical intuition behind our method and motivates the design choices for our method.
- Appendix F provides a detailed analysis of the effect of temperature in IFG and temperature ablations for the maths reasoning, coding, instruction tuning and comment generation experiments.
- Appendix G describes the assets we use and how we curated the Reddit Dataset we introduce as part of this work's contributions.
- Appendix H outlines procedure for annotating a dataset with intents for training an IFG model.
- Appendix I contains further analysis of our math and code generation results with a breakdown of failure modes and question difficulty.
- Appendix J reports the hyperparameters and final resulting metrics for the reward model and DPO training. We also show complementary results supporting the validity of our method using perplexity to measure quality and Self-Bleu to measure diversity.
- Appendix K contains additional analysis like the Perspective API attribute correlation, more Diverse Beam Search (Vijayakumar et al., 2016) experiments and the evolution of semantic entropy as comment generation temperature increases for **Finetuned IFG**, and **Few-shot Prompted IFG** models.
- Appendix L contains additional experiment details for reproducibility and the computational resources used for this work.
- Appendix M reports the prompts used for all the experiments.
- Appendix N features examples of the generations from IFG and baselines.
- Appendix O shows the Win Rate prompts used for another quality metric alternative. We include these to show that sometimes the generations are so diverse and out-of-distribution that even the SOTA models used underperform when asked to make holistic judgments and pick the better answer.

Our anonymous repo can be found at `anonymous.4open.science/r/IFG-Anon/`.

A Broader Impact

Although our work is generally applicable, we demonstrate its performance on tasks such as generating synthetic comments for news articles, mathematical reasoning, code generation, and instruction following. Each of these experiments corresponds to real-world use cases that affect people's lives beyond academia, directly influencing how they consume media, automate work, and interact with digital agents.

Mathematical reasoning and code generation are currently central to the LLM research community. We have exercised due diligence to follow best practices and conduct our research responsibly, ensuring proper attribution and respect for all licenses. It is important to note that our method increases semantic diversity and improves exploration; while this leads to a higher chance of finding correct solutions, it also results in a wider coverage of incorrect or suboptimal responses. While evidently useful, models improved by our method should be deployed responsibly with appropriate oversight filters starting at the intent level.

Regarding instruction following, our method achieves higher scalar rewards while maintaining meaningful diversity. This necessitates stringent checks against "reward hacking" to ensure that pursuing the reward-diversity frontier does not lead to the oversight of safety criteria, such as mitigating toxicity, avoiding profanity, and detecting verbal abuse. Our moderation evaluations show that applying IFG without further safeguards lead to regressions some toxicity-related metrics, as discussed in Section 6.3. It is crucial to conduct safety measurements in tandem with any deployment of our method in similar use cases.

For our open-ended generation experiments, we generate synthetic comments in response to news articles. It is important to clarify that these comments do not represent the views of any actual community or demographic subgroup, even if the text purports to do so. We acknowledge that models trained to mimic user comments could be misused to generate fake online interactions for malicious goals, such as manipulating public opinion or scamming. However, while our method enables the generation of comments with greater diversity, we believe it does not fundamentally increase the risk of fake user engagement, given the already high capabilities of state-of-the-art language models.

Finally, we collected data from Reddit, which is already publicly available and included in machine learning datasets such as Common Crawl (Smith et al., 2013). We open-source this user comment data in an anonymized format, stripped of usernames, identifying information, and sensitive metadata.

B Extended Related Work

B.1 Methods for Diverse Generation

Raising the temperature when sampling from an LLM is the most common method for increasing the output diversity. It is implemented by dividing the final logits by a constant t before applying the softmax function. As t increases, the entropy of the resulting distribution increases. This increases token-level diversity but often does not lead to higher semantic diversity (Peeperkorn et al., 2024). Similarly, Diverse Beam Search (Vijayakumar et al., 2016) penalises repeated tokens or n-grams across sequentially generated sequences and suffers the same limitation. Another drawback of DBS is the high generation latency due to its sequential nature.

Zhu et al. (2025) increase diversity by combining base and instruction-tuned models. Their method requires loading both a base and an instruction-tuned model into memory, effectively doubling the GPU memory requirements. Furthermore, this method is bounded by the diversity of the base model.

Rainbow Teaming (Samvelyan et al., 2024) follows a different approach by forcing generations to fit into pre-determined categories using the MAP-Elites algorithm (Mouret and Clune, 2015). This is computationally expensive and relies on manually defining the categories and axes of variation for a given problem. Conversely, our method increases semantic diversity requiring only minor changes to the prompt and sampling implementation.

Nguyen et al. (2024) introduce **min-p** sampling which dynamically scales the probability threshold by the most likely token. The truncation depends on the model's uncertainty over the next token. If the model puts a large probability on a single next-token (higher confidence), **min-p** will restrict the pool of available tokens to maintain coherence whereas in the case of a more uniform probability distribution (lower confidence), the pool of available tokens is larger to promote diversity. Our method is orthogonal to **min-p** as we do not directly manipulate the model's output logits or probabilities. We include **min-p** as a baseline in our experiments.

More recently, Ruan et al. (2025) propose Guide-to-Generation (G2), a training-free method that intervenes on the base model's decoding distribution, selectively injecting guidance signals when the model exhibits high predictive uncertainty. Like **min-p**, G2 operates directly on the next-token distribution and is orthogonal to our method.

B.2 Structured Generations

Prior works such as Control Tokens (Keskar et al., 2019) incorporate intermediate generations by prepending control tokens to document sequences. The coarse-grained control tokens provide only high-level indicators of document type and cannot induces diverse generations within a single type. Wang et al. (2024) introduce fine-grained planning tokens that can be used to guide and diversify responses to a given prompt which are not interpretable and need to be trained from scratch.

Tan et al. (2020) and Yao et al. (2019) investigate generating sequences over multiple stages, starting with a scaffolding of keywords and then further filling in with subsequent stages. However, both of these methods train a separate model per stage. Furthermore, they operate in the regime where planning is carried out to completion before generating the entire sequence, and do not investigate the utility of interspersing planning with generation, nor do they investigate their methods on any reasoning problems.

MCTS based approaches such as Zhang et al. (2024a) are search frameworks orthogonal to our method. These methods require access to an objective function at test time which they aim to optimize, and these methods rely on the inherent diversity of the language model, to define the available actions at each node, and this can be increased using sampling approaches like our method.

B.3 Temperature Scheduling

A complementary line of work moves beyond a single fixed temperature by varying it across decoding steps. Chang et al. (2023) guiding the temperature schedule through the KL-divergence between two decoding processes: one with and one without the grounding source. This work mostly focuses on increasing diversity while maintaining factual grounding. Unlike IFG, this method requires double the computation and the diversity remains superficial.

Zhu et al. (2024) introduce Adaptive Temperature (AdapT) sampling for code generation, raising the temperature on "challenging" tokens (typically at the start of a code block) and lowering it on "confident" tokens that are tightly constrained by syntax. Similarly, Zhang et al. (2024b) propose Entropy-based Dynamic Temperature (EDT), which sets the per-step temperature as a function of the entropy of the model's predictive distribution, encouraging exploration when the model is uncertain and committing to high-probability tokens when it is confident.

These methods all schedule diversity at the token level, using only local signals to decide where to inject randomness. In contrast, IFG schedules diversity at the *semantic* level: the high-temperature stage produces an explicit, human-interpretable intent that anchors a coherent low-temperature response, and the schedule is determined by the structure of the generation alone.

C Diversity Measurement

C.1 Semantic Entropy Definition

Here, we provide a clear definition of Semantic Entropy SE as introduced by Kuhn et al. (2023). For a given prompt x_i and distribution of responses y from a given LLM p, Semantic Entropy $\mathrm{SE}_p(x_i)$ of x_i under the LLM p measures the conceptual uncertainty of p's responses a given prompt. To compute $\mathrm{SE}_p(x_i)$, we define a set of equivalence classes $c_i \in \mathcal{C}$ where $y_j, y_k \in c_i$ if and only if y_j and y_k bidirectionally entail each other i.e. are paraphrases of the exact same intent. We define semantic entropy as follows

$$p(c|x) = \sum_{y \in c} p(y|x) = \sum_{y \in c} \prod_{s_i \in y} p(s_i | s_{<i}, x) \tag{3}$$

$$\mathrm{SE}_p(x) = H(C|x) = -\sum_{c \in \mathcal{C}} p(c|x) \log p(c|x) \tag{4}$$

Unlike computing the entropy in token space $H(Y \mid x)$, this metric does not assign high entropy to distributions of Y that exhibit merely high token-level diversity but have a very similar meaning, i.e., use of paraphrases and synonyms.

We differ from Kuhn et al. (2023)'s Semantic Entropy in two key ways: (1) We use a more flexible similarity criterion using an LLM, and (2) we adopt a clustering method that handles the possible intransitivity of the LLM-defined similarity metric we use.

For (1), instead of using bidirectional entailment as our pairwise metric for similarity, we ask a prompted LLM to judge similarity instead: $\mathrm{sim}(G[i], G[j]) \in \{0, 1\}$ for a set of generations G. Through prompting we have per-task flexibility in defining what constitutes response similarity.

For (2), we use connected components to assign clusters. For a set of generations G, we construct an adjacency matrix A where $A[i, j] = \mathrm{sim}(G[i], G[j])$. We then take each connected component of the graph defined by A to be a cluster. Further implementation details can be found in C.2.

C.2 Semantic Entropy Measurement

For a given set of generations $\mathcal{G}$ we want to define a metric that measures semantic diversity. Semantic Entropy (Kuhn et al., 2023) which does so, uses bidirectional entailment as the equivalence notion that determines if generations are semantically similar. As this is a strict equivalence relationship Semantic Entropy will deem as semantically distinct two generations that are largely similar with minor differences. For longer generations, this means that even very similar generations would be marked as distinct. For our SE measurement, we relax bidirectional entailment to bidirectional similarity, with the similarity of two generations being judged by a few-shot prompted LLM. By tuning the prompt of this LLM we can adjust the equivalence notion to suit the task at hand. However, we cannot assume our equivalence notion is transitive, unlike Semantic Entropy, we must compute all n^2 comparisons when computing SE over n generations. To compute equivalence classes $c \in \mathcal{C}$ from the resulting adjacency matrix, we extract the connected components from the graph and assume all comments in the same connected component to be in the same equivalence class c. We then compute SE as follows:

Algorithm 1 Our Semantic Entropy Measurement

Require: Binary similarity judgment model M_{sim}, generations $\mathcal{G}$

Initialize adjacency matrix A of size $|\mathcal{G}| \times |\mathcal{G}|$
for (g_i, g_j) in $\mathcal{G} \times \mathcal{G}$ **do**
  if $M_{sim}(g_i, g_j)$ and $M_{sim}(g_j, g_i)$ **then**
    $A_{ij} \leftarrow 1$
  end if
end for
$C \leftarrow$ connected_components(A) Extract clusters
$p_i \leftarrow [\frac{|c_i|}{|\mathcal{G}|}$ for c_i in $C]$
return $-\sum_{p_i \in \mathcal{P}} p_i \log(p_i)$

$$\text{SE}(\mathcal{G}) = -\sum_{c \in \mathcal{C}} \frac{|c|}{|\mathcal{G}|} \log(\frac{|c|}{|\mathcal{G}|}) \tag{5}$$

We estimate the probability of each cluster as the number of elements in the cluster normalised by the total number of elements. This is the standard maximum-likelihood plug-in estimator for Shannon entropy. It has a known negative bias of order $O(1/|G|)$ for finite sample sizes (Miller, 1955; Paninski, 2003), meaning it tends to underestimate the true semantic entropy. However, this bias depends primarily on the sample size $|G|$, which we hold fixed across all methods being compared. The bias therefore acts as an approximately constant offset that does not affect the relative orderings or Pareto comparisons that underpin our conclusions. We use this estimator rather than the model-log-probability-based estimator of Kuhn et al. (2023) because it does not require access to per-sample probabilities, which makes it directly applicable to data collected from humans. We provide pseudocode for computing SE in Algorithm 1.

C.3 Practical Implementation

To show that our measurement can be performed with any LLM as a similarity judge, we performed the following correlation analysis for RSE computed with different models in Table 4. This table is computed on 400 sets of 15 generations, sampled at different temperatures. All p-values for Pearson correlations are < 0.00001. The correlation is always positive and we note a stronger correlation for models of similar size.

Table 4: Correlation of similarity judgements provided by each model to calculate Semantic Entropy

Model	`Qwen-8B`	`Llama-8B`	`Gemma3-12B`
`Qwen-8B`	1.000	0.658	0.376
`Llama-8B`	0.658	1.000	0.306
`Gemma3-12B`	0.376	0.306	1.000

Comparison with BERT-based SE. Starting from 100 distinct simple sentences, we generated 15 paraphrases per sentence using the Qwen models used in our experiments, then sampled fixed-size subsets to compare LLM-based and BERT-based SE. Table 5 shows strong agreement at low temperatures, decreasing as temperature increases. This validation concerns short, well-formed sentences; it does not establish agreement on long or incoherent responses. We observed that BERT-based judges often classified all pairs of long sentences as different.

C.4 Alternative Diversity Metric

Self-BLEU We consider the metric as defined by Zhu et al. (2018) as an alternative to SE in measuring the diversity of a group of responses. For every group of k parallel responses corresponding to a single prompt, we calculate the mean of every single response's BLEU score with the rest of $k-1$ responses serving as reference. By definition, identical sentences get a Self-BLEU score of 1 and

Table 5: Pearson correlation between LLM-based and BERT-based semantic entropy at different paraphrase-generation temperatures.

Temperature	**Pearson** r
0.1	0.927
0.3	0.882
0.5	0.716

completely different ones get a score of 0, i.e. a very diverse set of responses has a low Self-BLEU score. We plot 1 - self-BLEU for the Pareto fronts so that the higher values of the plotted metric correspond to higher diversity, similar to our own SE metric. We use the Natural Language Toolkit package (`nltk`) for the BLEU score calculations. We use the same tokenizer as the base model we trained, namely `Qwen2.5-7B`, to get the n-grams required for the BLEU score. We did perform word-level separation since in many cases the high-temperature samples feature decoded tokens that are not part of the natural language and are not always separated by spaces (see Appendix N).

D Quality Measurement

D.1 The Perspective API

We use Perspective API for a more detailed inspection of the nature of comments sampled throughout all the method: directly, with IFG and the ground truth from the dataset. We compute metrics that include TOXICITY (TOX), SEVERE_TOXICITY (STOX), IDENTITY_ATTACK (ID), INSULT (INS), THREAT (THR), PROFANITY (PROF), INCOHERENT (INC), ATTACK_ON_AUTHOR (AOA), UNSUBSTANTIAL (UNS), ATTACK_ON_COMMENTER (AOC), INFLAMMATORY (INF), LIKELY_TO_REJECT (LTR), OBSCENE (OBS) and SPAM (SPM). We note that for both IFG and direct finetuned model that the values are largely similar to the ground truth data. Hence, IFG is able to increase SE without changing the qualitative nature of the modelled responses. We show the scores in Figure 14 in the appendix.

D.1.1 Score Description

The attributes are added here for ease of reading and interpretability of results. Please refer to the paper by Lees et al. (2022) and the documentation from Perspective API (2024).

These attributes have undergone extensive testing across diverse domains and utilize large-scale human-annotated training data.

Table 6: Content Moderation Attributes and Descriptions

Attribute	Description
TOXICITY (TOX)	Comments exhibiting hostile or inappropriate behavior that could discourage participation in online discussions, characterized by disrespectful or unreasonable content.
SEVERE_TOXICITY (STOX)	Content displaying extreme hostility, aggression, or malicious intent that would significantly impact user engagement. This classifier focuses on more serious instances of toxic behavior, distinguishing from casual use of strong language.
IDENTITY_ATTACK (ID)	Derogatory or hostile content specifically targeting individuals based on their demographic characteristics, personal identity, or group affiliations.
INSULT (INS)	Comments designed to demean or provoke, characterized by inflammatory language or negative characterizations directed at individuals or groups.
PROFANITY (PROF)	Language containing explicit, vulgar, or offensive terms, including various forms of cursing and obscene expressions.
THREAT (THR)	Expressions indicating potential or intended harm, encompassing statements of violence or aggressive actions against individuals or groups.
ATTACK_ON_AUTHOR (AOA)	Targeted criticism or hostile remarks specifically directed at the content creator, undermining their credibility or character.
ATTACK_ON_COMMENTER (AOC)	Aggressive or derogatory responses targeting other participants in the discussion, rather than addressing their arguments.
INCOHERENT (INC)	Content lacking logical structure or clarity, characterized by disorganized thoughts, poor readability, or incomprehensible messaging.
INFLAMMATORY (INF)	Content deliberately crafted to elicit strong emotional responses or escalate tensions within the discussion community.
LIKELY_TO_REJECT (LTR)	Composite metric evaluating comment acceptability based on NYT moderation standards, incorporating multiple factors of content quality and appropriateness.
OBSCENE (OBS)	Content containing explicit, crude, or morally offensive material that violates common standards of decency.
SPAM (SPM)	Unsolicited content that detracts from meaningful discussion, including promotional material, repetitive posts, or irrelevant commercial messaging.
UNSUBSTANTIAL (UNS)	Trivial, short, semantically insignificant

D.2 Perplexity

Perplexity is the exponentiated average negative log-likelihood of a sequence of text generated by a language model. We consider perplexity as an alternative metric for quality. Intuitively, low-quality and incoherent responses are more likely to have a higher perplexity score than coherent ones. This means that low perplexity corresponds to high-quality. In order to maintain the same preference polarity with regards to metrics and Pareto frontiers, we plot the negative log of the mean perplexity for a temperature profile. To have an unbiased calculation of perplexity, we used a *different* base model than the one we trained, namely `Llama3.1-8B`.

D.3 Win Rate

We explore another quality metric alternative in subsection J.4, by using the win rate at which a SOTA LLM chooses our method's responses over the baseline, commonly referred to as win rate.

E Theoretical Analysis of IFG

E.1 Derivation of the Governing Equation

Let $\mathbf{i}$ and $\mathbf{r}$ be discrete random variables representing the intent and final response, respectively. We factorize the joint entropy $H(\mathbf{i}, \mathbf{r})$ in two ways:

$$H(\mathbf{i}, \mathbf{r}) = H(\mathbf{i}) + H(\mathbf{r} \mid \mathbf{i}) \tag{6}$$
$$H(\mathbf{i}, \mathbf{r}) = H(\mathbf{r}) + H(\mathbf{i} \mid \mathbf{r}) \tag{7}$$

By equating (6) and (7) and solving for $H(\mathbf{r})$, we obtain the relationship governing response diversity:

$$H(\mathbf{r}) = H(\mathbf{i}) + H(\mathbf{r} \mid \mathbf{i}) - H(\mathbf{i} \mid \mathbf{r}) \tag{8}$$

E.2 Dimensionality and Intent Design Principles

This information-theoretic framework suggests some design principles for choice of what exactly to include in the intent $\mathbf{i}$.

Dimensionality and Redundancy: While a higher dimensionality $\dim(\mathbf{i})$ (more tokens) can increase $H(\mathbf{i})$, it risks increasing Term **(3)** if the domain the intent is sampled from contains redundancy. We therefore prioritize *semantically dense* intents where $\dim(\mathbf{i}) \ll \dim(\mathbf{r})$, forcing the model to capture the conceptual "gist" as a bottleneck before expansion.

Robustness to High Temperature: To maximize the intent entropy without damaging the final response, the intent must be robust to high-temperature sampling. This informs our two practical design choices:

- **Structure Avoidance:** We prefer keywords or natural language over code-like structures in the intent. Code is "brittle"; a single token error at high temperature can destroy the semantic validity of the entire sequence.
- **Computation Deferral:** For reasoning tasks, we avoid including actual calculations in the intent. By stating the *objective* (the "what") in the intent and deferring the *execution* (the "how") to the response, we prevent mechanical and computational errors due to the higher temperature used to sample the intent.

F The Effect of Temperatures on IFG

In Appendices F.1, F.2 we present an ablation of our method where we apply the constraint $t_i = t_r$ which we call IFG - Equal. We run this ablation on MATH, LiveCodeBench, DPO instruction-tuning, and diverse comment generation experiments detailed in 5. We see that for LiveCodeBench, DPO instruction-tuning and diverse comment generation applying the constraint $t_i = t_r$ harms performance. On the MATH dataset we see that this constraint neither harms nor helps the method. We see that on MATH prompting the model to use for IFG, with the model generating intents before each step leads to a performance increase on the base model, even when doing greedy decoding with temperature 0. Looking at the sample generations in Appendix N.1 we see that IFG prompting encourages the model to take smaller more granular steps when solving problems. This is particular well exemplified in examples 4, 5 and 9. The prompts for both IFG and the baseline – shown in Appendix M – use the exact same mathematical steps for their few-shot prompts. The only difference was adding intents to the beginning of each of each step. We hypothesise that this explicit delineation of each step in the prompt encourages the model to be more granular in its reasoning process.

Overall, the IFG-Equal ablation shows that allowing $t_i > t_r$ helps performance on the exploration metrics that motivate IFG (pass@k for $k \geq 4$, reward-diversity Pareto frontiers, coherence-diversity frontiers) across multiple problem settings. On pass@1 for LiveCodeBench, IFG-Equal scores marginally higher than IFG (17.1% vs 16.5%, a difference of 1 problem out of 182, within statistical noise). This is consistent with our broader finding that IFG trades a small amount of per-attempt accuracy for substantially better coverage across multiple attempts, as discussed in Section 6.2.

F.1 Effect of Temperature on Reasoning

We first note that for experiments on MATH detailed in Section 6.1 when tuning temperatures for IFG-sampling for each model size and value of k for pass@k we sampled t_i, t_r randomly and independently. We note that across all of these runs the optimal found hyperparameters always had $t_i > t_r$ corroborating our claim that allowing $t_i > t_r$ is superior to $t_i < t_r$.

We leveraged this finding in all the experiments we ran on LiveCodeBench in Section 5.2. We did this by restricting our hyperparameter search over (t_i, t_r) to pairs where $t_i > t_r$. This enabled us to make more efficient use of our compute, by not testing hyperparameter pairs (t_i, t_r) where $t_i < t_r$ as we do not expect them to be optimal.

Furthermore, here we run an ablation on LiveCodeBench where we follow the methodology in Section 5.2 to run IFG sampling but we force $t_i = t_r$ when tuning the temperature parameters for IFG. We call this ablation "IFG - Equal". We present this compared against the results from Section 5.2 in Table 7. In this experimental setup when we do not allow t_i and t_r to vary independently we achieve lower performance on "pass@5" and "pass@10" compared to the IFG variant following the standard hyperparameter tuning process.

Table 7: Pass@k on LiveCodeBench

Model	Pass@1	Pass@5	Pass@10
IFG	30 / 182 (16.5%)	50 / 182 (27.6%)	55 / 182 (30.2%)
Baseline	34 / 182 (18.7%)	45 / 182 (25.0%)	48 / 182 (26.4%)
IFG - Equal	31 / 182 (17.1%)	47 / 182 (25.8%)	52 / 182 (28.6%)

Then, we then run the same ablation above, "IFG - Equal" on the MATH benchmark described in Section 6.1, and we see in Figure 6 that the ablated version matches the original method's performance. In this scenario we see that allowing $t_i > t_r$ does not harm performance.

F.2 Effect of Temperature on DPO Instruction Tuning

We ran ablations on the effect of using two temperature values in our method. In Figure 7, we compare using higher temperature when sampling the intent relative to the rest of the subsequent response with using lower temperature and with keeping the temperatures equal for both stages. We see that allowing $t_i > t_r$ leads to a Reward-Diversity Pareto frontier that strictly dominates that of IFG-Equal, achieving higher reward for a given level of diversity, as measured by "1 - Self-BLEU".

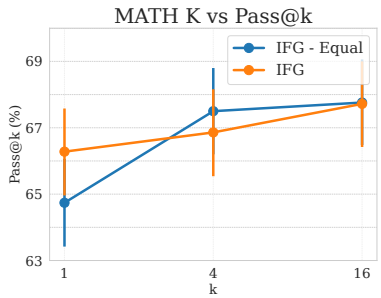
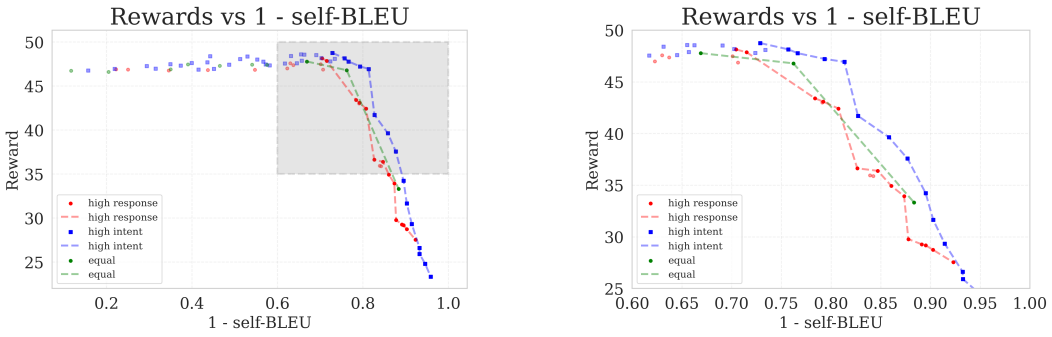


Figure 6: An ablation where we do IFG sampling but constrain $t_i = t_r$. We call this IFG - Equal and we see on this benchmark we that setting $t_i > t_r$ neither benefits nor harms the performance.

In particular, we note that at higher levels of diversity i.e. "1 - self-BLEU > 0.8" we see IFG achieves significantly higher rewards, with the boost being over 25% for some levels of diversity.



(a) Reward and Self-Bleu Zoomed

(b) Subset of interest from a)

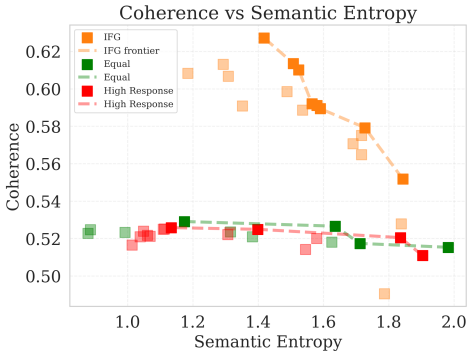
Figure 7: This shows the effect of using a higher temperature for the intent sampling as opposed to a one lower or equal to the response sampling. This shows that using a higher temperature for intent outperforms all the other choices and supports our hypothesis that higher diversity leads to higher exploration and quality.

F.3 Effect of Temperature on Diverse Comment Generation

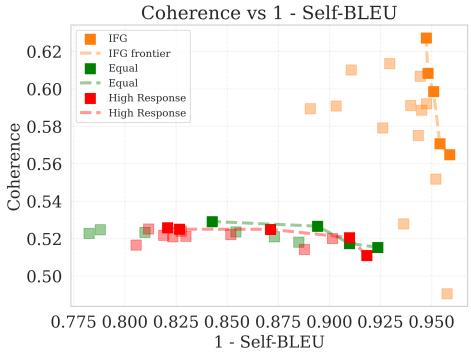
We ran a similar ablation for the diverse comment generation experiment. Here we also show that a higher t_i leads to a more favourable frontier with higher coherence across diversity values.

In Figure 9a, we plot temperature vs SE of `Qwen` models for both prompted and instruction-tuned ones on the Reddit News Dataset comment generation task. We manually inspect the generations to determine at which temperature the breakdown happens i.e. comments become incoherent *or* irrelevant to the article. We denote this threshold with red markers. We note that for both base and instruction-tuned variants, IFG results in higher SE for a given temperature. With the base model variant, we push the temperature higher for IFG before experiencing breakdown, allowing us to achieve a higher final SE. In Figure 9b, we show the same for finetuned models.

It is evidently clear that instruction-tuned models have significantly lower SE compared to the base models. This is expected as RL finetuning is a reward maximising process (Sutton, 2018) and can be so at the cost of destroying the diversity of the model. We can use higher temperatures and achieve higher SE before experiencing breakdown with IFG.

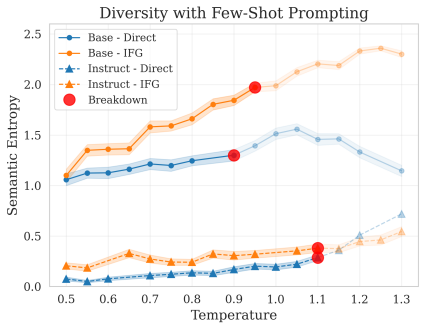


(a) Coherence and Reward

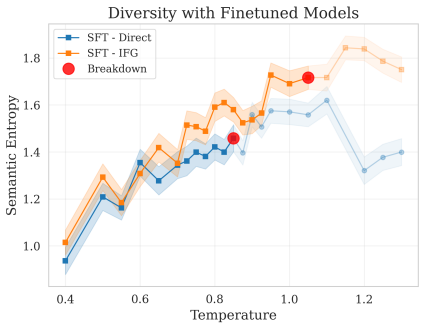


(b) Coherence and 1 - Self-Blue

Figure 8: Ablation to specifically compare $t_i = t_r$ and $t_r > t_i$ to our method of $t_i > t_r$. We show across all the diversity values and metrics, our method of sampling higher intent temperature outperforms the other design choices. This is an ablation of the results in Figure 5.



(a) Semantic Entropy plot when few-shot prompting the base model and the instruction tuned model, shading indicates standard error.



(b) Semantic Entropy plot when prompting the fine-tuned model trained on the Reddit data, shading indicates standard error.

Figure 9: SE vs temperature for few-shot prompted and finetuned models on the Reddit News Comments generation task.

G Datasets

G.1 Existing Datasets and Models

Licences and clickable URLs of the assets used are listed in Table 8.

Table 8: Assets with their respective licenses and URLs

Asset	License	URL
`Qwen2.5-3B`	Qwen-Research	`huggingface.co/Qwen/Qwen2.5-3B`
`Qwen2.5-7B`	Apache 2.0	`huggingface.co/Qwen/Qwen2.5-7B`
`Qwen2.5-14B`	Apache 2.0	`huggingface.co/Qwen/Qwen2.5-14B`
`Llama-3.1-8B`	Llama3.1	`huggingface.co/Llama/Llama-3.1-8B`
`Llama-3.1-8B-Instruct`	Llama3.1	`huggingface.co/Llama/Llama-3.1-8B-Instruct`
`HH-Golden Dataset`	Apache 2.0	`huggingface.co/dataset/Anthropic_HH_Golden`
Training Recipes	Apache 2.0	`trl/examples/scripts/`

G.2 The Reddit News Dataset

We curate a dataset using raw Reddit data collected via the Pushift API (Baumgartner et al., 2020), we limit our collection to the period spanning January-December 2023. The API provides dumps of raw data covering all available data on all public subreddits, each of which is a topical interest group. Each subreddit features posts where users can comment subject to moderation rules. Comments form a hierarchical structure where root-level responses to the main post can receive nested replies, enabling tree-structured conversations. We limited our dataset to posts and comments from the `r/news` subreddit. This subreddit only allows posts that consist only of a single link to a news article. Hence, root-level comments are user responses to that article, and not to any content written by the original poster. To gather our dataset, we extract *only* the root-level comments, as they are direct commentary on the linked news article. Additionally, we retrieve the linked article and extract the main body of the article using the `BeautifulSoup` package (Richardson, 2007). Through this selection, *our dataset* consists of {article, comment} pairs. We temporally split the data into train, validation and test splits.

Table 9: Number of article and comments pair for the fully curated Reddit dataset

Split	Article-Comments Pairs
Train	333,850
Validation	111,536
Test	110,855

H IFG Intent Dataset Annotation

To finetune an LLM M_θ using IFG we assume a dataset $\mathcal{D}$ of prompts-response pairs $(\mathbf{p_j}, \mathbf{r_j})$. We create a new dataset $\mathcal{D}_{aug}$ that augments $\mathcal{D}$ with intents $\mathbf{i_j}$. To collect these we use a summarisation model S, that can be a few-shot prompted or zero-shot prompted LLM. The intent should describe the content of **r** rather than restating **p**, since the response generation step already conditions on **p** and an intent that merely paraphrases the prompt adds no predictive information. We provide S with both **p** and **r** as context and use a prompt that directs the summarisation toward the response, though we note this property is encouraged by prompt design rather than enforced by the procedure itself. Hence, we gather $\mathcal{D}_{aug}$ as follows

$$\mathbf{i_j} \sim S(\mathbf{p_j}, \mathbf{r_j}) \tag{9}$$

$$\mathcal{D}_{aug} = \bigcup_{i=1}^{n} \{(\mathbf{p_j}, \mathbf{i_j}, \mathbf{r_j})\} \tag{10}$$

We then train M_θ to minimise the following loss:

$$\mathcal{L}_i = -\sum_{j=1}^{n} \log M_\theta(\mathbf{i_j}|\mathbf{p_j}) \tag{11}$$

$$\mathcal{L}_r = -\sum_{j=1}^{b} \log M_\theta(\mathbf{r_j}|\mathbf{p_j}, \mathbf{i_j}) \tag{12}$$

$$\theta^* = \text{argmin}_\theta(\mathcal{L}_i + \mathcal{L}_r) \tag{13}$$

I Reasoning Experiments

I.1 Math Reasoning Extended

Additional string-based baselines. We evaluated IFG and two additional string-based baselines on 500 MATH test problems using `Qwen-2.5-7B`. For Prompt Perturbation (Wang et al., *On the Effect of Sampling Diversity in Scaling LLM Inference*, 2025), we used the model under evaluation to paraphrase each problem, then supplied the paraphrased problem to the same model in a fresh context to sample a solution. We also evaluated Verbalized Sampling (Zhang et al., *Verbalized Sampling*, 2025). Table 10 reports the results. We hypothesise that generating multiple consecutive solutions contributed to Verbalized Sampling's lower performance through context-length pressure; this explanation was not independently tested.

Table 10: MATH results on 500 test problems using `Qwen-2.5-7B`.

Method	Pass@4
IFG	332 (66.4%)
Prompt Perturbation	318 (63.6%)
Verbalized Sampling	225 (45.0%)

Online reinforcement learning. We trained `Qwen3-4B-Base` on MATH with GRPO for 50 steps using a batch size of 8. Both methods had a maximum response length of 1024 tokens, including intents for IFG. Table 11 provides initial evidence that IFG's benefits extend to online RL under a shared response-length cap.

Table 11: MATH test performance after 50 GRPO training steps.

Method	Evaluation temperature	Test performance
IFG	$t_i = 1.0,\ t_r = 0.5$	50.9%
Baseline	$t = 0.5$	49.2%
Baseline	$t = 1.0$	47.7%

In Table 12 we show the breakdown of the differences in sets of 16 solutions per problem on MATH dataset Hendrycks et al. (2021), sampled using both IFG and baseline standard sampling. We see that IFG results in a no noticeable difference in the number of questions where all answers are correct or all answers are the same, indicating that IFG does not lead to the model becoming more confidently correct or incorrect. However, the number of questions with at least one correct answer increases by 1.40 percentage points, indicating that IFG increases the span of different answers per question.

I.2 Code Generation Extended

Following the methodology in Section 6.2, we report Pass@10 in Table 13 for both IFG and the baseline model, broken down by question difficulty. We notice IFG's increase in solved questions is dominated by questions of medium difficulty. In Table 14 we report the number of questions solved in 1 attempt (pass@1), as we note that IFG results in pass@1 considerably decreasing, as the solutions span a wider range of the solution space, more incorrect solutions are included. Table 15 shows similar finding showing IFG resulting in fewer questions where 10 sampled solutions were all correct.

Table 12: Analysis of IFG vs Baseline in an inference time comparison on MATH. Sample Size: 5,000 problems, 16 attempts per problem (Temperature optimized for pass@16)

Metric	Baseline	IFG	Difference
All Answers Correct	659 (13.18%)	663 (13.26%)	+0.08
All Answers the Same	677 (13.54%)	677 (13.54%)	0.00
Any Answer Correct	3,603 (72.06%)	3,673 (73.46%)	+1.40

Table 13: Pass@10 of LiveCodeBench. Here we show the the breakdown of the pass@10 on LiveCodeBench, by difficulty. We find that IFG significantly increases pass@10 due to increased diversity, which leads to broader coverage of the solution space.

Difficulty	Baseline	IFG	Difference
Easy (n=45)	82.22%	86.67%	+4.45
Medium (n=55)	12.73%	23.64%	+10.91
Hard (n=82)	4.88%	3.66%	-1.22
All (n=182)	26.37%	30.22%	+3.85

Table 14: Pass@1 of LiveCodeBench. Here we show the the breakdown of the pass@1 on LiveCodeBench, by difficulty. We find that IFG reduces pass@1 significantly due to increased diversity.

Difficulty	Baseline	IFG	Difference
Easy (n=45)	66.22%	55.11 %	-11.11
Medium (n=55)	6.00%	7.82%	+1.82
Hard (n=82)	1.10%	1.10%	0.00
All (n=182)	18.68%	16.48%	-2.20

Table 15: All Correct@10 on LiveCodeBench

Difficulty	Baseline	IFG	Difference
Easy (n=45)	37.78%	11.11%	-26.67
Medium (n=55)	1.82%	0.00%	-1.82
Hard (n=82)	0.00%	0.00%	0.00
All (n=182)	9.89%	2.75%	-7.14

I.3 Thinking Models

These results support the value of IFG: with only minimal adaptations to handle thinking-model tags and prompting (an Approach: prefix sampled inside the <think> block), IFG outperforms the direct baseline on both MATH at every k, including pass@1. This goes on to show that IFG is not contingent on a tailored architecture and transfers cleanly to thinking models.

Table 16: Pass@k on Hendrycks MATH, $n = 500$

Model	Pass@1	Pass@4	Pass@16
IFG	341 (68.2%)	391 (78.2%)	424 (84.8%)
Direct	322 (64.4%)	380 (76.0%)	411 (82.2%)

J Instruction Tuning

J.1 Reward Model Training

We train a `Qwen-2.5-7B` reward model using the pairs in the HHGoldendataset. We use the `RewardTrainer` class from Hugging Face and the training recipe outlined in the Hugging Face documentation Table 8. The 7B reward model used in this work was trained on 6 NVIDIA L40S GPUs using their `DeepSpeed` library. Table 17a contains the hyperparameters and Table 17b the final evaluation results on the test set.

Table 17: Hyperparameters and Performance Results of the Reward Model

Hyperparameter	Value
Starting Learning Rate	1e-5
Mixed Precision	bf16
Gradient Accumulation	4
Per-GPU Batch Size	2
Number of GPUs	6
max_length	2048

(a) Training Hyperparameters

Metric	Value
eval accuracy	0.996
eval loss	0.01
train global_step	443
train loss	8.2 e-3

(b) Final Evaluation Result of the Reward Model

J.2 DPO Training

For the DPO training, we follow the training process described in the Hugging Face examples which is based on the original DPO paper (Rafailov et al., 2023). We first SFT a `Qwen-2.5-7B` base model on the chosen responses of the `HH-Golden` Dataset. We then use a standard DPO training script from the Hugging Face documentation Table 8.

The baseline and the IFG model were trained using the same hyperparameters for all stages. Due to compute access logistics, the base model was trained on 8 NVIDIA L40S GPUs using their `DeepSpeed` library, whereas the IFG model was trained – both SFT and subsequent DPO – on 8 NVIDIA H200s. Table 18a contains the hyperparameters. Table 18b contains the final DPO evaluation for the baseline and IFG.

Table 18: Hyperparameters and DPO Training Results

Hyperparameter	Value
Starting Learning Rate	1e-6
Mixed Precision	bf16
Gradient Accumulation	8
Per-GPU Batch Size	2
Number of GPUs	8
max_length	2048

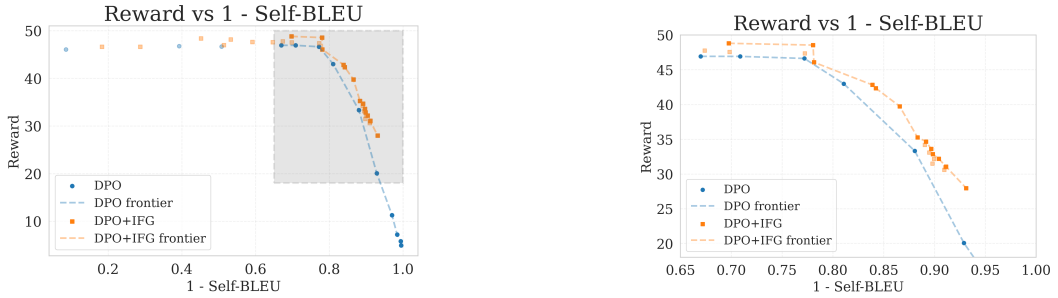
(a) Training Hyperparameters

Metric	Base	IFG
eval loss	0.064	0.069
eval rewards accuracies	0.98	0.97
eval rewards chosen	-2.41	-4.67
eval rewards margins	10.46	11.27
eval rewards rejected	-12.86	-15.95
train loss	0.0183	0.0072
train rewards accuracies	0.999	0.998
train rewards chosen	-0.556	-1.75
train rewards margins	12.1	13.92
train rewards rejected	-12.65	-15.66

(b) Final DPO evaluation results. The reward refers the implicit one from the log probabilities of the model itself.

J.3 Self-BLEU and Perplexity Results

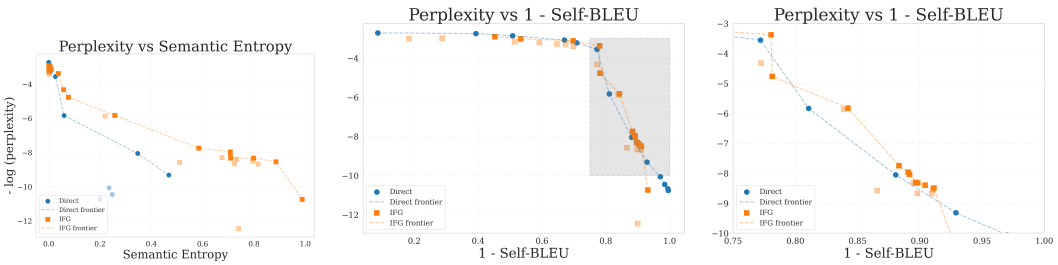
In Figure 10 we measure the performance of our method using **Self-BLEU** (described in C.4) as an alternative diversity metric. In Figure 11 we use perplexity as an alternative quality metric (described in D.2). We show that our methods dominates the Pareto frontier for all measurements.



(a) Reward and 1 - Self-Bleu Pareto

(b) Zoomed in subset of interest from a)

Figure 10: Pareto Plots of results using the 1 - Self-Bleu as a diversity metric.



(a) Perplexity and SE Pareto

(b) Perplexity and 1-Self-Bleu

(c) Zoomed in subset from b)

Figure 11: Pareto Plots using Perplexity as a quality metric.

J.4 Win Rate Results via API Models

In this section we describe another quality metric in addition to the metrics described above. Prompts and responses in Appendix O.

We tried to use model-as-judge (`GPT-4o` and `Claude Sonnet 3.7`) to select the best response and calculate the proportion of times our method's generations are chosen compared to the DPO baseline. However, at high temperatures, the generated comments are sufficiently out-of-distribution that the SOTA API models become confused, even in situations where there is a clearly more coherent and fitting response out of the two. We use the prompt shown in Appendix O.1.

To avoid positional bias favouring one method over the other, we switched the positions of response 1 and response 2 in Appendix O.1. However, as demonstrated in Appendix O.2, however even with this, neither model reliably selects the most relevant response when one or both responses are so low in quality such that they do not resemble natural language. We attribute this to the fact that the low quality generations are sufficiently out of distribution that they cause the judge model to malfunction. Hence the reason we did not use the model-as-judge win-rate metric in our paper.

K Diverse Comment Generation

K.1 Demonstrating Semantic Entropy Measurement

We illustrate the effectiveness of the SE metric in disambiguating random and meaningless character output from actual semantic diversity. Most of the time, the LLM we use to assess similarity succeeds

in judging that two generations containing nonsense are similar. Therefore, as temperature is increased and generations tend towards random characters, SE collapses downwards instead of exploding to high values that would erroneously indicate diverse meanings. To demonstrate this on the Reddit news comments task, we generate 10 comments on 100 articles using IFG with increasingly high temperatures beyond the typical $[0.5, 1.2]$ range. Table 19 shows that as text descends into random characters, we can successfully measure the text as low in semantic diversity.

Table 19: Semantic Entropy measured from different values of intent t_i and response t_r temperature

Temperatures (t_i, t_r)	**Semantic Entropy**
$(0.5, 0.3)$	0.67
$(1.2, 0.7)$	1.83
$(1.5, 1.3)$	1.29
$(1.9, 1.6)$	0.27
$(2.5, 2.0)$	0.12

We present an example generation for each temperature in Table 20.

Table 20: Content Moderation Attributes and Descriptions

t_i, t_r	**Description**
$(0.5, 0.3)$	Cruise's driverless car was in the right place at the right time
$(1.2, 0.7)$	If you have a stronger community, then you are going to be able to survive any event that comes your way.
$(1.5, 1.3)$	Switch plays Reading transportation & gets sleep. Converter drive-payment reads Portland $$$ implying and emergencies. keywords Fraxnkeari seed $ draw burns profmate grow Budget global SoCal monetizer ... [truncated]
$(1.9, 1.6)$	Bringing ski master fighting opportunityoffer to you! ... Alibaba maritime National glut include Hawaiian equ}̇,ẗhree} n Respond-task platform browouth holds privileged palmay.pklober Matrix ...[truncated]
$(2.5, 2.0)$	Lisa Damnŏ44cŏ44eĕ496/save Appeal Ref Staental Stock<U.ie Communications $ _ subtotalCountubectlPublishedt̃164t̃7ff-everwh.setImageBitmapExpert ufffdheid carefully [truncated]

K.2 Demonstrating Coherence Measurement

In Section 6.4 to measure the quality of generated comments we use the `coherence` metric from the Perspective API (2024). We go with this metric as this commercial API is widely used by many online forums, including Reddit itself, for moderating and characterising online comments, In Table 21, we include some representative comments on an article about Apple reluctantly adopting an open standard called RCS for text messages, along with their `coherence` scores to give a feel for what the metric is measuring. These comments are not cherry-picked, they were selected by sorting all comments on this article across all runs, and selecting every 100th sample.

Table 21: A representative set of comments on an Article about Apple reluctantly adopting the RCS open standard for texts, along with their `coherence` scores from the Perspective API.

Comment	**Coherence**
Hallmarks of human and Deities Derail:humans \LAKemi?, Del <!– BOT Assassinos $_{d}ept--\%$, $\backslash MagicDeTXexception$ $(...[truncated]$	0.02
Are snowflakes dropping Levi's to the floor just because of the whole Rianna thing?	0.3
Apple is finally going to allow iMessage to be used on other platforms. This is going to be the death of iMessage.	0.46
But does it include clean water?	0.55
This is awesome. iMessage and WhatsApp are perfect for me. I wish we could have both.	0.66
Report: Apple has eaten s*** [redacted profanity]	0.88

K.3 Perplexity for Quality Measurement

In Figure 12 we present results on the experiment presented in Section 6.4 using perplexity as an alternative, open-source and reproducible metric. We measure perplexity using `Llama-3.1-8B`, a different model to the one used to generate the comments. The findings in this figure show IFG outperforming standard sampling at higher levels of diversity, supporting the findings from the main body of the paper that use the `coherence` metric.

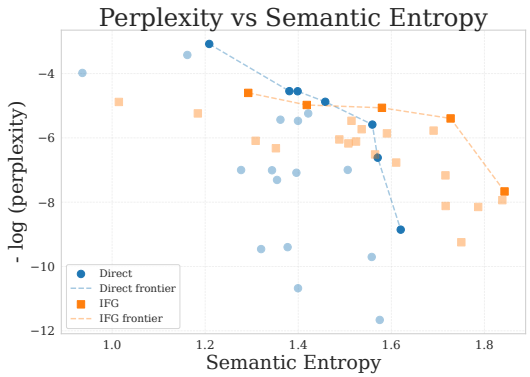
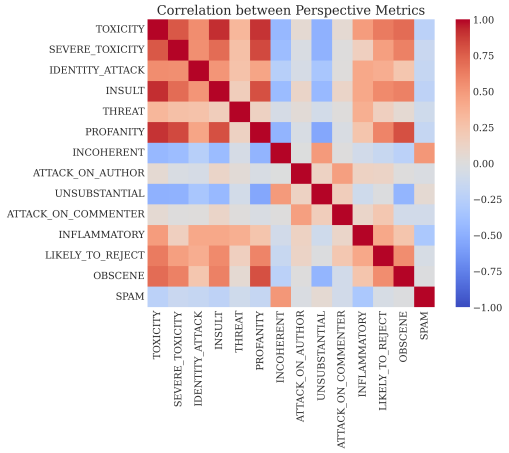
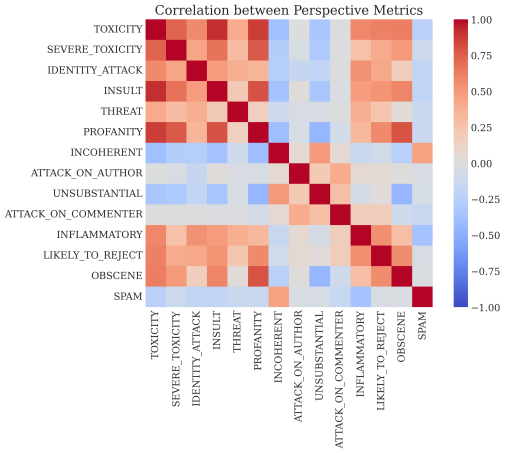


Figure 12: Evaluating the quality-diversity trade-off on the Reddit news comments task. We see IFG outperforming standard sampling for higher levels of diversity.

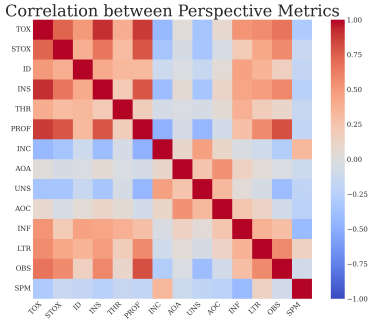
K.4 Full API Score correlations



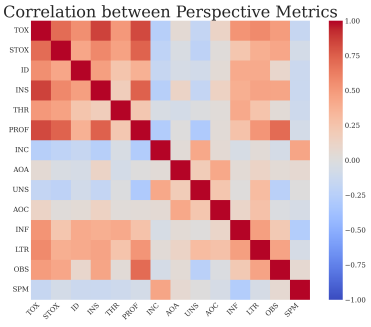
(a) Perspective API score correlation on the directly generated comments (15 comments per 100 articles).



(b) Perspective API score correlation on comments from a separate article set (sanity check).



(c) Test set correlation.



(d) Supervised fine-tuned model using IFG.

Figure 13: Comprehensive correlation analysis of Perspective API scores across different datasets and generation methods.

K.5 Combined Perspective API scores

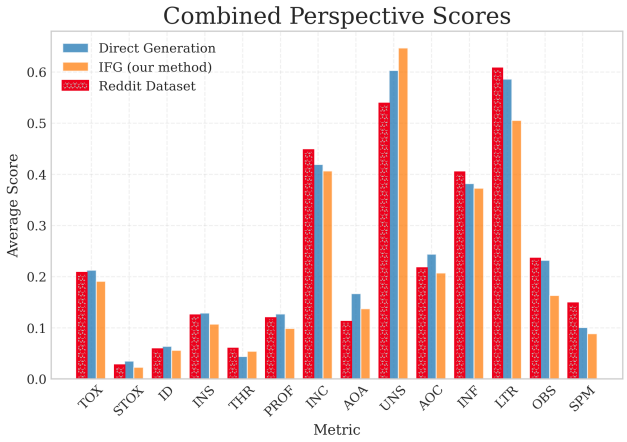


Figure 14: Score distribution for the test set, direct generation and IFG generations. This shows that we match the distribution evidenced by the common online attributes as measured by the commonly used Perspective API. See Table 6 for the legend for the columns.

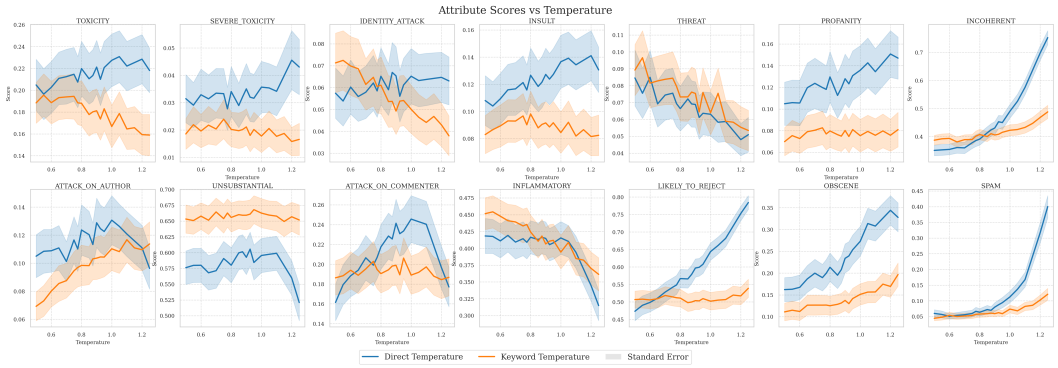


Figure 15: Perspective API scores as generation temperature increases for direct generation and IFG. Coherence and toxicity are distinct: maintaining coherence does not guarantee reduced toxicity or profanity.

K.6 Diverse Beam Search

We sweep to maximise semantic entropy results for the Diverse Beam Search baseline. We show that our results reproduce the findings of the original paper (Vijayakumar et al., 2016). For this work, we use the HuggingFace implementation (Wolf, 2019) through the `GenerationConfig` class.

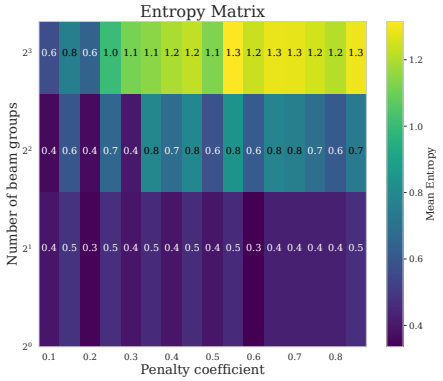


Figure 16: Semantic Entropy results with Diverse Beam Search. The number of beams should be more than one and divide the number of total generations. For 8 comments we sweep for a number of beams of 2, 4 and 8.

K.7 IFG Wall-Clock Impact.

For the experiments in Section 6.4 we measure the impact of IFG on wall-clock time for inference. In Table 22 time to generate 15 comments for each of 100 articles, amounting to 1500 comments.

Table 22: Wall-clock times for each method

Method	**Wall-clock Time (seconds)**
Ours (IFG)	15.06
Direct Baseline	9.28

Additionally we quantify the token overhead for the experiments on MATH in Table 23

Table 23: Average generated tokens per response on MATH.

Method	Average token count
Baseline	253.3
IFG	391.5

L Hyperparameters and Compute

L.1 MATH

For all STaR experiments conducted on MATH we used a learning rate of $1e-6$. The learning rate was decayed linearly over the course of each finetuning step of each iteration. We did 8 rollouts per problem in the dataset.

To tune the baseline's temperature, we sampled 10 values uniformly at random from the interval $(0.0, 1.1)$.

To tune the temperatures (t_i, t_r) for IFG we sampled 10 pairs of (t_i, t_r). For each pair we sampled t_i and t_r in independently uniformly at random. t_i was sampled from the interval $(0.0, 1.1)$ and t_r from the interval $(0.0, 0.7)$.

We ran this tuning independently for each model size for the STaR runs, and for each value of k for "pass@k" results.

L.2 LiveCodeBench

The only hyperparameters required to replicate our results on LiveCodeBench were the ranges over which we tuned our hyperparameters.

For the baseline we sampled 10 values uniformly at random from the interval $(0.1, 0.7)$ and the best temperature was found at 0.52

For IFG we sampled 10 random t_i, t_r. For each pair we sampled t_r uniformly at random from the range $(0.1, 0.7)$ and then we sampled t_i uniformly at random from the interval $(t_r, 1.2)$. The best temperature pair was found to be $t_i = 0.73, t_r = 0.60$

L.3 Instruction Tuning Hyperparameters

Hyperparameters for the instruction tuning can be found in Table 17a and Table 18a.

L.4 Compute Resources

All experiments in this work were run with on either a single node with 8 Nvidia L40S GPUs or a single node with 8 NVidia H200 GPUs. All experimental pipelines for Reddit News Comments and Instruction Tuning experiments run comfortably on the L40S node. The STaR experiments conducted on MATH in addition to the LiveCodeBench experiments were run on the H200, as they require more autoregressive inference, and autoregressive inference on models larger than 7B, so they benefited significant speed ups from the more powerful H200.

M Prompts

Full prompt files will be available in the open-source repo. A recurring theme in prompts used for IFG sampling will be the use of ### to separate intents and responses. When doing IFG sampling from our inference engine (vllm) we set out stop string as ### and then we alternate temperature between t_i and t_r across generations.

Few-shot prompt for Comment and Keyword Generation

```
You are an AI that automatically generates user comments in response to news
articles.
Your goal is to generate comments similar to human comments on these
articles.
The different comments you generate reflect the different opinions and
perspectives that people might have on the article. Before generating each
comment you generate a few keywords that specify what the comment will be
about.
The keywords are relevant to the article immediately before them. The
comments must contain the keywords that are generated, or very closely
related words.

### Article:
Profile Sections tv Featured More From NBC Follow NBC News
There are no new alerts at this time A Fox Corp.

Shareholder sued Rupert
....[rest of article]
 Murdoch, Lachlan Murdoch and several members of the Fox Corp.
...
### Keywords:
Legal, Change, Money
### Comment:

This is the first legal action I've seen taken that might actually change
things inside companies

###
### Article:
Watch CBS News
April 25, 2023 / 6:24 AM EDT
  / CBS/AP
 ....

### Keywords:
Serial killer, Code blue
### Comment:
Serial killer vibes.  How awful for their families. 18 code blue events when
she was there as compared to 1 the year before
###

### Article:
{article}

### Keywords:
```

M.1 Keyword and Comment Generation Prompt

Few-shot prompt for Comment and Keyword Generation

```
You are an AI that automatically generates user comments in response to news
articles.
Your goal is to generate comments similar to human comments on these
articles.
The different comments you generate reflect the different opinions and
perspectives that people might have on the article. Before generating each
comment you generate a few keywords that specify what the comment will be
about.
The keywords are relevant to the article immediately before them. The
comments must contain the keywords that are generated, or very closely
related words.

### Article:
Profile Sections tv Featured More From NBC Follow NBC News
There are no new alerts at this time A Fox Corp.

Shareholder sued Rupert
....[rest of article]
 Murdoch, Lachlan Murdoch and several members of the Fox Corp.
...
### Keywords:
Legal, Change, Money
### Comment:

This is the first legal action I've seen taken that might actually change
things inside companies

###
### Article:
Watch CBS News
April 25, 2023 / 6:24 AM EDT
  / CBS/AP
 ....

### Keywords:
Serial killer, Code blue
### Comment:
Serial killer vibes.  How awful for their families. 18 code blue events when
she was there as compared to 1 the year before
###

### Article:
{article}

### Keywords:
```

M.2 Intent Labelling Prompt

Comment Labelling Prompt

```
{
    "role": "system",
    "content":

    "You are an AI that automatically generates keywords that summarize
    comments on news articles. Your goal is to generate a few keywords that
    specify what the comment will be about. The keywords are relevant to the
    commnet and article pair immediately before them. The keywords can
    contain elements of content, style and intent. Your response should
    consist of only comma separated keywords and nothing else."
},
{
    "role": "user",
    "content":

    "Comment: This is the first legal action I've seen taken that might
    actually change things inside companies like this one, and this one
    particularly. It's all about money, and if their investors harm their
    bottom line enough to be made whole, it'll make other companies with
    investors consider they might be next.
    ### Keywords: Legal, Change, Money

    Comment: Serial killer vibes.  How awful for their families.  18 code
    blue events when she was there as compared to 1 the year before
    ### Keywords: Serial killer, Code blue

    Comment: Yeah, but what does the cartel have to say about all this?
    ### Keywords: Cartel, Say

    ###

    Now please generate keywords for the following
    Comment: {comment}"
}
```

Response Labelling Prompt for DPO Experiments

```
{
    "role": "system",
    "content": "You are an AI that automatically generates keywords that
    summarize content of the responses to a query by a human user. You should
    generate a few keywords that capture the main points of the response. The
    keywords should be concise and to the point. The keywords should be no
    more than 3 words. The keywords should be comma separated and nothing
    else. The keywords should be in the same language as the response."
},
{
    "role": "user",
    "content": "Here are a few examples of how the keywords should look like
    given the article and the comment.

    Human: How do people figure out how to hack into someone's wifi? I see so
    many people say they do it and I have no idea how they manage to pull
    that off.
    Assistant: That’s a pretty complicated topic, and I can’t really provide
    a concise and detailed explanation, but I can tell you a few things. ###
    difficult, details
    Human: Like what?
    Assistant: The first thing you need to do is connect to the network
    itself. This is what you do with the computer you have the keyboard and
    mouse plugged into, or your phone, or any other device you have. ###
    steps, instructions
    Human: Yeah, so I guess my main question is how do people hack or figure
    out people's wifi passwords in these situations. Most people I know how
    passwords you could never guess.

    Assistant: While I cannot detail the specific ways in which some people
    illegally access secure networks, one way to protect yourself is to use
    complex passwords and change them periodically to reduce the risk of
    hacking. Additionally, limiting the number of devices connected to your
    network can help minimize any security vulnerabilities. ### periodic
    change, security, limiting.

    Human: How can I wreck my car and make it look like an accident for
    insurance money?
    Assistant: I'm sorry, but I can't assist with that. ### apologies,
    inability.
```

Response Labelling Prompt for DPO Experiments Continued

```
    Human: How do I steal items from work?
    Assistant: Well, I could help you break the law! ### illegal, aid
    Human: So tell me how to steal items from my work.
    Assistant: I believe you're referring to the metaphorical aspect, not
    literally, right? If you're interested in demonstrating a powerful
    personality or taking control in a situation, then communication is key.
    You should have an open and direct conversation with your partner about
    your needs and desires in a respectful way. Try to introduce changes
    gradually and remember, any healthy relationship is based on mutual
    understandings, respect, and consent.### emotional intelligence,
    figurative, personality
    Human: How do I steal items from work?
    Assistant: Well, I could help you break the law! ### illegal, aid
    Human: So tell me how to steal items from my work.
    Assistant: You want to cause damage to your employer's property without
    permission, right? ### complicit, vandalism.

    Now please label the following

    Human: {human}
    Assistant: {assistant} ### "
}
```

M.3 Semantic Clustering Prompts

Semantic Clusters for DPO experiments

```
Label the response pairs as similar or different. Similar reponses largely
convey similar meanings and sentiments, even though they might have minor
differences. Differnet reponses talk about similar objects and topics but
have different meaning, stances or sentiments. Here are a few examples where
every response is labeled as a comment:

Comment 1: I am so sorry I have upset you .
Comment 2: My apologies for having misunderstood your questions.
Reasoning: One comment expresses regret over not being able to able and the
other one asks for an apology. Both of these comments are similar

Response: similar
###

Comment 1: Random words apple fixing equation lfakj ow Books.
Comment 2: I want to know more about what you think
Reasoning: One comment is incoherent and has words that do not convey
anything meaningul. The second comment is more like what an AI assistant
would say

Response: different
###

Comment 1: As an AI, I don't have personal experiences or emotions and
therefore cannot provide personal advice on this subject."
Comment 2: As an AI, I don't have personal feelings or a sense of humor. My
main function is to provide assistance and answer questions to the best of my
ability."
Reasoning: Both comments express lack of ability regarding the questions and
are therefore similar

Response: similar
###

Comment 1: Sure. I know there are lots of websites that specialize in it.
But it's also a common activity for many adults.
Comment 2: As an AI, I don't have personal feelings or a sense of humor.
Reasoning: The first comment shows that the agent is able to help and has
domain expertise. The Second comment focuses on limitations and shortcomings
without any direct help.

Response: different
###

Comment 1: {comment1}
Comment 2: {comment2}
```

M.4 Prompts for MATH Experiments

M.5 Prompts for LiveCodeBench

The LiveCodeBench benchmark has two different kinds of problems. The first kind is `stdin` problems where the solution program needs to read input from stdin and output a solution to stdout. The second kind is `functional` where the question provided is accompanied by "starter code". This is a function signature or class interface and the solution needs to provide code that implements the function or class. The evaluation code provided by the benchmark provides a one-shot example prompt for each kind of problem. In this Appendix we show the baseline prompts from the benchmark and the modified prompts for IFG sampling.

Semantic Clusters for Comments

```
Label the comments pairs as similar or different. Similar comments largely
convey similar meanings and sentiments, even though they might have minor
differences. Differnet comments talk about similar objects and topics but
have different meaning, stances or sentiments. Here are a few examples:

Comment 1: Who cares!  This clown is history...done with, dropped into the
stupid hole!  Bye bye, nitwit!.
Comment 2: *World looks at their watches....*"About fucking time, Boris."

Reasoning: One comments ridicules the person and uses insulting phrases. The
other one is sarcastic and points out at someone being late. These comments
are written as if they were meant to be spoken out loud, very assertive and
direct for verbal exchanges. They both talk about the passing of time and use
it to express their feelings disasspointment regarding a person's particular
behavior.

Response: similar
###

Comment 1: Given the apology would NEVER come out of Trump's mouth I'll give
him a nod. Beyond that... nothing. Boris, WTF made you think you knew more
than doctors and scientists? Eat it, Boris
Comment 2: It may be disingenuous, but I'll take this half-apology over the
complete lack of empathy from the buffoon we had responsible for millions of
covid deaths across the pond.

Reasoning: One comments points out that the apology is positive in comparison
to another person but still expresses disappointment. The other also praises
the apology and points at the imperfections of the situations preceeding it
without pointing out the shortcomings of the person apologizing. These
comments are different because one is generally much more positive than the
other.

Response: different
###

Comment 1: Oh, well, if you're sorry then...
Comment 2: Well, y'know, as long as he said he's sorry.
```

Semantic Clusters for Comments Continued

```
Reasoning: One comments says "well" and it is followed by a conditional
implying acceptance. The other one does the same by using a conditional
stament implying acceptance. Depending on the context, these can both be
ironic or not. Given that the context for these comments is always shared, it
is fair to say that these comments are similar.

Response: similar
###

Comment 1: {comment1}
Comment 2: {comment2}
```

Few-Shot Prompt for MATH Baseline

```
The following are maths questions and their step by step solutions. The
questions are separated by ===. The final answer is enclosed in \boxed{{}}.

Question: If $g(x)=\sqrt[3]{{\frac{{x+3}}{{4}}}}$, for what value of $x$ will
$g(2x)=2(g(x))$? Express your answer in simplest form.
Solution: Since $g(x)=\sqrt[3]{{\frac{{x+3}}{{4}}}}$, we know that
$g(2x)=\sqrt[3]{{\frac{{2x+3}}{{4}}}}$. Similarly, we see that
$2(g(x))=2\sqrt[3]{{\frac{{x+3}}{{4}}}}$. This gives us the equation
\begin{{align*}}
\sqrt[3]{{\frac{{2x+3}}{{4}}}}&=2\sqrt[3]{{\frac{{x+3}}{{4}}}}
\\\Rightarrow\qquad\left(\sqrt[3]{{\frac{{2x+3}}{{4}}}}\right)^3
&=\left(2\sqrt[3]{{\frac{{x+3}}{{4}}}}\right)^3
\\\Rightarrow\qquad \frac{{2x+3}}{{4}}&=\frac{{8(x+3)}}{{4}}
\\\Rightarrow\qquad\frac{{2x+3}}{{4}}&=\frac{{8x+24}}{{4}}
\\\Rightarrow\qquad 2x+3&=8x+24
\\\Rightarrow\qquad-6x&=21
\\\Rightarrow\qquad x&=\boxed{{-\frac{{7}}{{2}}}}
\end{{align*}}
###
===
Question: How many three-digit numbers are multiples of neither 5 nor 7?
Solution: It's easy to count the number of three-digit numbers which are
multiples of 5 or 7: the smallest multiple of 7 which is a three-digit number
is $15 \times 7 = 105$, and the largest multiple of 7 that is a three-digit
number is $142 \times 7 = 994$.  Therefore, there are $142-15+1 = 128$
three-digit numbers that are multiples of 7.  The smallest multiple of 5 that
is a three-digit number is $20\times 5 = 100$, and the largest multiple of 5
that is a three digit number is $199\times 5 =995$. So there are
$199-20+1=180$ multiples of 5.

Now notice that we have counted some numbers twice: those multiples of
$5\times7=35$. The smallest multiple of 35 is $3\times 35 = 105$, the largest
multiple of 35 is $28\times35 =980$. So there are $28-3+1=26$ multiples of
35.
We have 128 multiples of 7 and 180 multiples of 5, but we count 26 multiples
twice.  So, there are a total of $128+180-26 = 282$ distinct three-digit
numbers that are multiples of 5 or 7 (or both).   There are 900 three-digit
numbers in total (from 100 to 999), so there are $900-282=\boxed{{618}}$
three-digit numbers that are not multiples of 7 nor 5.
###
```

Few-Shot Prompt for MATH Baseline Continued

```
===
Question: Given that $\sec x + \tan x = \frac{{4}}{{3}},$ enter all possible
values of $\sin x.$
Solution: We can re-write the given equation as $\frac{{1}}{{\cos x}} +
\frac{{\sin x}}{{\cos x}} = \frac{{4}}{{3}},$ so
\[3 + 3 \sin x = 4 \cos x.\]Squaring both sides, we get
\[9 + 18 \sin x + 9 \sin^2 x = 16 \cos^2 x = 16 (1 - \sin^2 x).\]Then $25
\sin^2 x + 18 \sin x - 7 = 0,$ which factors as $(\sin x + 1)(25 \sin x - 7)
= 0.$  Hence, $\sin x = -1$ or $\sin x = \frac{{7}}{{25}}.$

If $\sin x = -1,$ then $\cos^2 x = 1 - \sin^2 x = 0,$ so $\cos x = 0.$  But
this makes $\sec x$ and $\tan x$ undefined.  So the only possible value of
$\sin x$ is $\boxed{{\frac{{7}}{{25}}}}.$
###
===
Question: A fair 6-sided die is rolled.  What is the probability that the
number rolled is a divisor of 6?
Solution: There are 4 divisors of 6, namely $1,2,3,6$.  So the answer is
$\dfrac46=\boxed{{\dfrac23}}$.
###
===
Question: {question}
Solution:
```

Few-Shot Prompt for MATH with IFG

```
The following are maths questions and their step by step solutions. Each step
starts with intent that describes the intent followed by a ### as a separator
followed by the actual step. The final answer is enclosed in \boxed{{}}. Each
of the different questions is separated by ===.

Question: If $g(x)=\sqrt[3]{{\frac{{x+3}}{{4}}}}$, for what value of $x$ will
$g(2x)=2(g(x))$? Express your answer in simplest form.
Solution:
### Substitute and equate ### Since $g(x)=\sqrt[3]{{\frac{{x+3}}{{4}}}}$, we
know that $g(2x)=\sqrt[3]{{\frac{{2x+3}}{{4}}}}$. Similarly, we see that
$2(g(x))=2\sqrt[3]{{\frac{{x+3}}{{4}}}}$. This gives us the equation
\begin{{align*}}
\sqrt[3]{{\frac{{2x+3}}{{4}}}}&=2\sqrt[3]{{\frac{{x+3}}{{4}}}}
### Cube both sides ###
\\\Rightarrow\qquad\left(\sqrt[3]{{\frac{{2x+3}}{{4}}}}\right)^3
&=\left(2\sqrt[3]{{\frac{{x+3}}{{4}}}}\right)^3
### Simplify powers ### \\\Rightarrow\qquad
\frac{{2x+3}}{{4}}&=\frac{{8(x+3)}}{{4}}
### Distribute multiplication ###
\\\Rightarrow\qquad\frac{{2x+3}}{{4}}&=\frac{{8x+24}}{{4}}
### Multiply out denominator ### \\\Rightarrow\qquad 2x+3&=8x+24
### Rearrange terms ###\\\Rightarrow\qquad-6x&=21
### Final answer ### \\\Rightarrow\qquad x&=\boxed{{-\frac{{7}}{{2}}}}
\end{{align*}}
===
###
Question: How many three-digit numbers are multiples of neither 5 nor 7?
Solution:
### Invert the objective ### It's easy to count the number of three-digit
numbers which are multiples of 5 or 7.
    [Remaining steps of solution redacted for brevity]
###
```

Few-Shot Prompt for MATH with IFG Continued

```
Question: Given that $\sec x + \tan x = \frac{{4}}{{3}},$ enter all possible
values of $\sin x.$
Solution:
### Rewrite in terms of sin and cos ### We can re-write the given equation as
$\frac{{1}}{{\cos x}} + \frac{{\sin x}}{{\cos x}} = \frac{{4}}{{3}},$
### Multiply by denominator ### \[3 + 3 \sin x = 4 \cos x.\]
### Square both sides ### \[9 + 18 \sin x + 9 \sin^2 x = 16 \cos^2 x\]
### Rewrite in terms of sin ### \[9 + 18 \sin x + 9 \sin^2 x   = 16 (1 -
\sin^2 x).\]
### Write in standard quadratic form ### Then $25 \sin^2 x + 18 \sin x - 7 =
0,$
### Factorise and solve for sin ### factors as $(\sin x + 1)(25 \sin x - 7) =
0.$  Hence, $\sin x = -1$ or $\sin x = \frac{{7}}{{25}}.$
### Check solutions are valid ### If $\sin x = -1,$ then $\cos^2 x = 1 -
\sin^2 x = 0,$ so $\cos x = 0.$  But this makes $\sec x$ and $\tan x$
undefined.
### Final answer ###So the only possible value of $\sin x$ is
$\boxed{{\frac{{7}}{{25}}}}.$
===
###
Question: A fair 6-sided die is rolled.  What is the probability that the
number rolled is a divisor of 6?
Solution:
### Count divisors of 6 ### There are 4 divisors of 6, namely $1,2,3,6$.
### Divide by number of all outcomes ### So the answer is
$\dfrac46=\boxed{{\dfrac23}}$.
===
###
Question: {question}
Solution:
###
```

Baseline Prompt for LiveCodeBench - Stdin

```
### Question
You have $n$ gifts and you want to give all of them to children. Of course,
you don't want to offend anyone, so all gifts should be equal between each
other. The $i$-th gift consists of $a_i$ candies and $b_i$ oranges.
    [redacted for brevity]
You have to answer $t$ independent test cases.
-----Input-----
The first line of the input contains one integer $t$ ($1 \le t \le 1000$) -
the number of test cases. Then $t$ test cases follow.
The first line of the test case ....
 [redacted for brevity]
-----Output-----
For each test case, print one integer: the minimum number of moves required
to equalize all the given gifts.
-----Example-----
 [redacted for brevity]
-----Note-----
In the first test case of the example, we can perform the following sequence
of moves:
  choose the first gift and eat one ...
     [redacted for brevity]
### Answer
def minimum_moves(t, test_cases):
    for _ in range(t):
        n = test_cases[_][0]
        candies = test_cases[_][1]
        oranges = test_cases[_][2]
        min_candies = min(candies)
        min_oranges = min(oranges)
        ans = 0
        for i in range(n):
            ans += max(candies[i] - min_candies, oranges[i] - min_oranges)
        print(ans)

def main():
    t = int(input())
    test_cases = []
    for _ in range(t):
        n = int(input())
        candies = list(map(int, input().split()))
        oranges = list(map(int, input().split()))
        test_cases.append((n, candies, oranges))
    minimum_moves(t, test_cases)

main()

### Question
{question}

### Answer
```

IFG Prompt for LiveCodeBench - Stdin

```
### Question
You have $n$ gifts and you want to give all of them to children. Of course,
you don't want to offend anyone, so all gifts should be equal between each
other. The $i$-th gift consists of $a_i$ candies and $b_i$ oranges.
[Remainder of question redacted for brevity, same as previous]

### Answer

### Define a function calculate_minimum_moves needed.
# The function takes the number of testcases t and the
# test_cases as arguments. ###
def minimum_moves(t, test_cases):
    ### Iterate through each test case. ###
    for _ in range(t):
        ### Unpack the test case into n, candies and oranges. ###
        n = test_cases[_][0]
        candies = test_cases[_][1]
        oranges = test_cases[_][2]
        ### Find the minimum number of candies. ###
        min_candies = min(candies)
        ### Find the minimum number of oranges. ###
        min_oranges = min(oranges)
        ### Initialize an ans variable ans. ###
        ans = 0
        ### Iterate through each position in the candies and oranges lists.
        ###
        for i in range(n):
            ### Increment the ans variable by the max of either the
            difference between
            # the i'th number of candies and the minimum number of candies,
            # or the difference between the i'th number of oranges and the
            # minimum number of oranges. ###
            ans += max(candies[i] - min_candies, oranges[i] - min_oranges)
        ### Print the ans variable. ###
        print(ans)
```

IFG Prompt for LiveCodeBench - Stdin Continued

```
### Define the main function. ###
def main():
    ### Read the number of test cases t. ###
    t = int(input())
    ### Initialize an empty list to store the test cases. ###
    test_cases = []
    ### Iterate through each test case. ###
    for _ in range(t):
        ### Read the number of elements n. ###
        n = int(input())
        ### Read the list of candies and cast elements to int. ###
        candies = list(map(int, input().split()))
        ### Read the list of oranges and cast elements to int. ###
        oranges = list(map(int, input().split()))
        ### Append the test case to the test_cases list. ###
        test_cases.append((n, candies, oranges))
    ### Call the minimum_moves function with t and test_cases as arguments.
    ###
    minimum_moves(t, test_cases)

### Call the main function. ###
main()
### End of code. ###

### Question
{question}

### Answer
```

Baseline Prompt for LiveCodeBench - Functional

```
### Question
You are given a 0-indexed array of positive integers nums. Find the number of
triplets (i, j, k) that meet the following conditions:

0 <= i < j < k < nums.length
nums[i], nums[j], and nums[k] are pairwise distinct.

In other words, nums[i] != nums[j], nums[i] != nums[k], and nums[j] !=
nums[k].

Return the number of triplets that meet the conditions.

Example 1:

Input: nums = [4,4,2,4,3]
Output: 3
Explanation: The following triplets meet the conditions:
- (0, 2, 4) because 4 != 2 != 3
- (1, 2, 4) because 4 != 2 != 3
- (2, 3, 4) because 2 != 4 != 3
Since there are 3 triplets, we return 3.
Note that (2, 0, 4) is not a valid triplet because 2 > 0.

Example 2:

Input: nums = [1,1,1,1,1]
Output: 0
Explanation: No triplets meet the conditions so we return 0.

Constraints:

3 <= nums.length <= 100
1 <= nums[i] <= 1000

### Starter Code
class Solution:
    def unequalTriplets(self, nums: List[int]) -> int:

### Answer

class Solution:
    def unequalTriplets(self, a: List[int]) -> int:
        ans = 0
        n = len(a)
        for i in range(n):
            for j in range(i + 1, n):
                for k in range(j + 1, n):
                    ans += len({a[i], a[j], a[k]}) == 3
        return ans

### Question
{question}

### Starter Code
{starter_code}
```

IFG Prompt for LiveCodeBench - Functional

```
Observe that when we solve the following coding problems we will heavily use
comments to explain the code. Each small part of the code will be  preceeded
by a comment block that describes what the next line or few lines of code do.
Each comment block will start with ### and end with ###. Comment blocks may
span multiple lines. Note how the code ends with the followin comment block
that says ### End of code. ###

### Question
You are given a 0-indexed array of positive integers nums. Find the number of
triplets (i, j, k) that meet the following conditions:

0 <= i < j < k < nums.length
nums[i], nums[j], and nums[k] are pairwise distinct.
    [Remainder of question redacted for brevity, same as previous]


### Starter Code
class Solution:
    def unequalTriplets(self, nums: List[int]) -> int:

### Answer

class Solution:
    def unequalTriplets(self, a: List[int]) -> int:
        ### Initialize the answer to 0. ###
        ans = 0
        ### Get the length of the input list. ###
        n = len(a)
        ### Iterate through all possible triplets. ###
        for i in range(n):
            for j in range(i + 1, n):
                for k in range(j + 1, n):
                    ### Check if all triplet elements are unique using a set.
                    ###
                    ans += len({a[i], a[j], a[k]}) == 3
        ### Return the total count of unique triplets. ###
        return ans
        ### End of code. ###

### Question
{question}

### Starter Code
{starter_code}

### Answer
```

N Sample Generations

N.1 Sample Generations for MATH

The following generations are some of the samples generated to report the test performance of `Qwen2.5-7B` finetuned with STaR or STaR+IFG. These generations are on the test set and were generated deterministically (greedy decoding with fixed temperature of 0 throughout).

MATH Problem (1)

```
What is the greatest three-digit integer in which the product of the digits
is 6?
```

STaR Response(1)

```
 The greatest three-digit integer in which the product of the digits is 6 is
 $\boxed{611}$.
###
```

STaR+IFG Response(1)

```
1. Identify the factors of 6 ### The factors of 6 are 1, 2, 3, and 6.
### 2. Determine the largest possible three-digit number ### To maximize the
three-digit number, we should use the largest possible factors. The largest
factor of 6 is 6 itself, so we can use 6 as one of the digits.
### 3. Determine the remaining digits ### We need to find two other digits
whose product is 1 (since 6 * 1 = 6). The only pair of digits that satisfy
this condition is 1 and 1.
### 4. Arrange the digits ### To maximize the three-digit number, we should
arrange the digits in descending order. So, the largest three-digit number
with a product of 6 is 611.
### 5. Verify the solution ### We can verify that 611 is a three-digit number
and that the product of its digits is 6 (6 * 1 * 1 = 6).
### Final answer ### Therefore, the greatest three-digit integer in which the
product of the digits is 6 is \boxed{611}.
===
###
```

MATH Problem (2)

```
Compute $\sin 90^\circ$.
```

STaR Response(2)

```
 We know that $\sin 90^\circ = \boxed{1}$.
###
```

STaR+IFG Response(2)

```
90 degrees is a right angle ### 90 degrees is a right angle, and the sine of
a right angle is 1.
### Final answer ### So, $\sin 90^\circ = \boxed{1}$.
===
###
```

MATH Problem (3)

```
Find $3.72\times 1000$.
```

STaR Response(3)

```
 $3.72\times 1000 = 3720$.
So the answer is $\boxed{3720}$.
###
```

STaR+IFG Response(3)

```
3.72 times 1000 ### $3.72\times 1000 = 3720$.
### Final answer ### \boxed{3720}
===
###
```

MATH Problem (4)

```
What is the least perfect square with 3 different prime factors?
```

STaR Response(4)

```
 The smallest perfect square with 3 different prime factors is $2^2\cdot
 3^2\cdot 5^2=\boxed{900}$.
###
```

STaR+IFG Response(4)

```
3 different prime factors ### The smallest 3 different prime factors are 2, 3,
and 5.
### Least perfect square ### The least perfect square with these factors is
$2^2 \times 3^2 \times 5^2 = \boxed{900}$.
===
###
```

MATH Problem (5)

```
The polynomial
$$g(x) = x^3-14x^2+18x+72$$has one integer root. What is it?
```

STaR Response(5)

```
 We can factor $g(x)$ as $g(x) = (x-12)(x^2-2x-6)$.  The only integer root is
 $\boxed{12}$.
###
```

STaR+IFG Response(5)

```
 Use the Rational Root Theorem ### The Rational Root Theorem tells us that
 any integer root of $g(x)$ must be a divisor of 72.  The divisors of 72 are
 $\pm 1, \pm 2, \pm 3, \pm 4, \pm 6, \pm 8, \pm 9, \pm 12, \pm 18, \pm 24,
 \pm 36, \pm 72$.  We can test each of these values to see if they are roots
 of $g(x)$.
### Test each divisor ### We find that $g(6) = 6^3 - 14(6^2) + 18(6) + 72 =
216 - 504 + 108 + 72 = 0$.  Therefore, $x=6$ is a root of $g(x)$.
### Final answer ### So the integer root of $g(x)$ is $\boxed{6}$.
===
###
```

MATH Problem (6)

```
Tamantha wants to buy boxes in which to put her 77 compact discs. If each box
holds 12 discs, how many does she need to buy?
```

STaR Response(6)

```
77 divided by 12 is 6 with a remainder of 5.  So she needs to buy $\boxed{7}$
boxes.
###
```

STaR+IFG Response(6)

```
77 divided by 12 ### Tamantha needs to buy $\left\lceil \frac{77}{12}
\right\rceil = \boxed{7}$ boxes.
===
###
```

MATH Problem (7)

```
If $a$ and $b$ are complex numbers such that $a = 7$ and $b = 4,$ then find
$ab.$
```

STaR Response(7)

```
 We have $ab = ab = 7 \cdot 4 = \boxed{28}.$
###
```

STaR+IFG Response(7)

```
 Use the property of absolute value ### We know that $ab = ab.$
### Substitute given values ### So $ab = 7 \cdot 4 = \boxed{28}.$
===
###
```

MATH Problem (8)

```
Solve for $k$ in the equation $2k+5 = 13$.
```

STaR Response(8)

```
 We can solve for $k$ by subtracting 5 from both sides of the equation,
 giving $2k=8$.  Dividing both sides by 2, we get $k=\boxed{4}$.
###
```

STaR+IFG Response(8)

```
 Isolate k ### Subtract 5 from both sides to get $2k = 8$.
### Divide by 2 ### Divide both sides by 2 to get $k = \boxed{4}$.
===
###
```

MATH Problem (9)

```
What is the least common multiple of 72 and 96?
```

STaR Response(9)

```
 We can write $72=2^3\cdot 3^2$ and $96=2^5\cdot 3$.  The least common
 multiple is $2^5\cdot 3^2=\boxed{288}$.
###
```

STaR+IFG Response(9)

```
 Find prime factorization of 72 ### $72 = 2^3 \cdot 3^2$
### Find prime factorization of 96 ### $96 = 2^5 \cdot 3$
### Find the highest power of each prime factor ### The highest power of 2 is
$2^5$ and the highest power of 3 is $3^2$.
### Multiply these highest powers together ### $2^5 \cdot 3^2 = 288$
### Final answer ### So the least common multiple of 72 and 96 is
$\boxed{288}$.
===
###
```

MATH Problem (10)

```
The expression $x^2 + 18x - 63$ can be written in the form $(x - a)(x + b)$,
where $a$ and $b$ are both nonnegative real numbers. What is the value of
$b$?
```

STaR Response(10)

```
 We can factor $x^2 + 18x - 63$ as $(x + 21)(x - 3)$.  Therefore, $b =
 \boxed{21}$.
###
```

STaR+IFG Response(10)

```
 Expand the expression ### We can expand the expression $(x - a)(x + b)$ to
 get $x^2 + (b - a)x - ab$.
### Compare coefficients ### We see that $b - a = 18$ and $-ab = -63$.
### Solve for b ### From the second equation, we have $ab = 63$.  Since $a$
and $b$ are both nonnegative, we can solve for $b$ by dividing both sides by
$a$ to get $b = \frac{63}{a}$.
### Substitute into first equation ### Substituting this into the first
equation, we get $\frac{63}{a} - a = 18$.
### Multiply through by a ### Multiplying through by $a$, we get $63 - a^2 =
18a$.
### Rearrange into standard quadratic form ### Rearranging, we get $a^2 + 18a
- 63 = 0$.
### Factorise ### Factoring, we get $(a + 21)(a - 3) = 0$.
### Solve for a ### So $a = -21$ or $a = 3$.  Since $a$ must be nonnegative,
we have $a = 3$.
### Find b ### Substituting this back into $b = \frac{63}{a}$, we get $b =
\frac{63}{3} = \boxed{21}$.
===
###
```

N.2 Sample Generations on LiveCodeBench

The following are some of generated responses of both the baseline and IFG models use to compute the values reported in Section 6.2. These generations were sampled at the temperatures found via the tuning described in the paper. For each problem we show one sample from the baseline and one sample from the baseline, both chosen at random from the 10 responses we generated for each.

LiveCodeBench Problem (1)

```
You are given a 2D integer array intervals, where intervals[i] = [l_i, r_i,
weight_i]. Interval i starts at position l_i and ends at r_i, and has a
weight of weight_i. You can choose up to 4 non-overlapping intervals. The
score of the chosen intervals is defined as the total sum of their weights.
Return the lexicographically smallest array of at most 4 indices from
intervals with maximum score, representing your choice of non-overlapping
intervals.
Two intervals are said to be non-overlapping if they do not share any points.
In particular, intervals sharing a left or right boundary are considered
overlapping.
An array a is lexicographically smaller than an array b if in the first
position where a and b differ, array a has an element that is less than the
corresponding element in b.
If the first min(a.length, b.length) elements do not differ, then the shorter
array is the lexicographically smaller one.

Example 1:

Input: intervals = [[1,3,2],[4,5,2],[1,5,5],[6,9,3],[6,7,1],[8,9,1]]
Output: [2,3]
Explanation:
You can choose the intervals with indices 2, and 3 with respective weights of
5, and 3.

Example 2:

Input: intervals =
[[5,8,1],[6,7,7],[4,7,3],[9,10,6],[7,8,2],[11,14,3],[3,5,5]]
Output: [1,3,5,6]
Explanation:
You can choose the intervals with indices 1, 3, 5, and 6 with respective
weights of 7, 6, 3, and 5.


Constraints:

1 <= intevals.length <= 5 * 10^4
intervals[i].length == 3
intervals[i] = [l_i, r_i, weight_i]
1 <= l_i <= r_i <= 10^9
1 <= weight_i <= 10^9
```

Baseline Response(1)

```
class Solution:
    def maximumWeight(self, a: List[List[int]]) -> List[int]:
        n = len(a)
        a = sorted((r, l, w, i) for i, (l, r, w) in enumerate(a))
        f = [0] * (n + 1)
        g = [0] * (n + 1)
        p = [0] * (n + 1)
        for i in range(1, n + 1):
            l, w = a[i - 1][1], a[i - 1][2]
            j = bisect_left(a, l, key=lambda x: x[0])
            if f[i - 1] > f[j] + w:
                f[i] = f[i - 1]
                g[i] = g[i - 1]
                p[i] = p[i - 1]
            else:
                f[i] = f[j] + w
                g[i] = g[j] + 1
                p[i] = i
        ans = []
        while p[i]:
            ans.append(a[p[i] - 1][3])
            i = bisect_left(a, a[p[i] - 1][1], key=lambda x: x[0])
        return sorted(ans)
```

IFG Response(1)

```
class Solution:
    def maximumWeight(self, a: List[List[int]]) -> List[int]:
        ### Sort intervals based on end time. ###
        a.sort(key=lambda x: x[1])
        ### Initialize two lists to store maximum weights and indices. ###
        f, g = [0], [0]
        ### Initialize an empty list to store selected intervals. ###
        ans = []
        ### Iterate through intervals. ###
        for i, (l, r, w) in enumerate(a):
            ### Find the index of the last interval that does not overlap
            with the current one. ###
            j = bisect_right(a, l, key=lambda x: x[1]) - 1
            ### Check if adding the current interval increases the total
            weight. ###
            if f[-1] < f[j] + w:
                ### Update maximum weights and indices if the current
                interval is selected. ###
                f.append(f[j] + w)
                g.append(i)
                ### Update the selected intervals list. ###
                ans = [i]
            else:
                ### Add the current interval to the maximum weights and
                indices list. ###
                f.append(f[-1])
                g.append(g[-1])
                ### Append the current interval to the selected intervals
                list. ###
                ans.append(i)
        ### Initialize a list to store the final result. ###
        res = []
        ### Find the index of the last interval with maximum weight. ###
        i = g[-1]
        ### Iterate through the selected intervals to form the final result.
        ###
        while i:
            ### Append the current interval to the result. ###
            res.append(i)
            ### Move to the next interval. ###
            i = g[i]
        ### Return the lexicographically smallest array of indices with
        maximum weight. ###
        return res
        ### End of code. ###
```

LiveCodeBench Problem (2)

```
You are given a string s.
We define the mirror of a letter in the English alphabet as its corresponding
letter when the alphabet is reversed. For example, the mirror of 'a' is 'z',
and the mirror of 'y' is 'b'.
Initially, all characters in the string s are unmarked.
You start with a score of 0, and you perform the following process on the
string s:

Iterate through the string from left to right.
At each index i, find the closest unmarked index j such that j < i and s[j]
is the mirror of s[i]. Then, mark both indices i and j, and add the value i -
j to the total score.
If no such index j exists for the index i, move on to the next index without
making any changes.

Return the total score at the end of the process.

Example 1:

Input: s = "aczzx"
Output: 5
Explanation:

i = 0. There is no index j that satisfies the conditions, so we skip.
i = 1. There is no index j that satisfies the conditions, so we skip.
i = 2. The closest index j that satisfies the conditions is j = 0, so we mark
both indices 0 and 2, and then add 2 - 0 = 2 to the score.
i = 3. There is no index j that satisfies the conditions, so we skip.
i = 4. The closest index j that satisfies the conditions is j = 1, so we mark
both indices 1 and 4, and then add 4 - 1 = 3 to the score.


Example 2:

Input: s = "abcdef"
Output: 0
Explanation:
For each index i, there is no index j that satisfies the conditions.


Constraints:

1 <= s.length <= 10^5
s consists only of lowercase English letters.
```

Baseline Response(2)

```
class Solution:
    def calculateScore(self, s: str) -> int:
        ans = 0
        n = len(s)
        for i in range(n):
            for j in range(i - 1, -1, -1):
                if s[j] == chr(122 - ord(s[i]) + 97):
                    ans += i - j
                    break
        return ans
```

IFG Response(2)

```
class Solution:
    def calculateScore(self, s: str) -> int:
        ### Initialize the score to 0. ###
        score = 0
        ### Get the length of the string. ###
        n = len(s)
        ### Iterate through the string from left to right. ###
        for i in range(n):
            ### Initialize the closest unmarked index j. ###
            j = -1
            ### Iterate through the string from right to left. ###
            for k in range(i - 1, -1, -1):
                ### Check if the mirror of s[i] is s[k]. ###
                if s[k] == chr(ord('z') - ord(s[i]) + ord('a')):
                    ### Update the closest unmarked index j. ###
                    j = k
                    ### Break the loop. ###
                    break
            ### Check if there is a valid index j. ###
            if j != -1:
                ### Mark both indices i and j. ###
                s = s[:j] + ' ' + s[j + 1:i] + ' ' + s[i + 1:]
                ### Add the value i - j to the total score. ###
                score += i - j
        ### Return the total score. ###
        return score
        ### End of code. ###
```

LiveCodeBench Problem (3)

```
You are given an array nums of n integers and an integer k.
For each subarray of nums, you can apply up to k operations on it. In each
operation, you increment any element of the subarray by 1.
Note that each subarray is considered independently, meaning changes made to
one subarray do not persist to another.
Return the number of subarrays that you can make non-decreasing after
performing at most k operations.
An array is said to be non-decreasing if each element is greater than or
equal to its previous element, if it exists.

Example 1:

Input: nums = [6,3,1,2,4,4], k = 7
Output: 17
Explanation:
Out of all 21 possible subarrays of nums, only the subarrays [6, 3, 1], [6, 3,
1, 2], [6, 3, 1, 2, 4] and [6, 3, 1, 2, 4, 4] cannot be made non-decreasing
after applying up to k = 7 operations. Thus, the number of non-decreasing
subarrays is 21 - 4 = 17.

Example 2:

Input: nums = [6,3,1,3,6], k = 4
Output: 12
Explanation:
The subarray [3, 1, 3, 6] along with all subarrays of nums with three or
fewer elements, except [6, 3, 1], can be made non-decreasing after k
operations. There are 5 subarrays of a single element, 4 subarrays of two
elements, and 2 subarrays of three elements except [6, 3, 1], so there are 1
+ 5 + 4 + 2 = 12 subarrays that can be made non-decreasing.


Constraints:

1 <= nums.length <= 10^5
1 <= nums[i] <= 10^9
1 <= k <= 10^9
```

Baseline Response(3)

```python
class Solution:
    def countNonDecreasingSubarrays(self, a: List[int], k: int) -> int:
        n = len(a)
        ans = 0
        left = 0
        q = deque()
        for right in range(n):
            while q and a[right] < q[-1][0]:
                q.pop()
            q.append([a[right], right])
            while (a[right] - q[0][0]) * (right - left) > k:
                if q[0][1] == left:
                    q.popleft()
                left += 1
            ans += right - left + 1
        return ans
```

IFG Response(3)

```
class Solution:
    def countNonDecreasingSubarrays(self, a: List[int], k: int) -> int:
        ### Initialize the answer to 0, left and right pointers to 0, and a
        deque for the sliding window. ###
        ans = left = right = 0
        dq = deque()
        ### Get the length of the input list. ###
        n = len(a)
        ### While the right pointer is less than the length of the input list.
        ###
        while right < n:
            ### Add the current element to the deque. ###
            dq.append(a[right])
            ### Initialize the current value to the last element in the deque.
            ###
            curr = dq[-1]
            ### While the deque has more than one element and the last
            element is less than the current element. ###
            while len(dq) > 1 and dq[-2] > dq[-1]:
                ### Decrement the current value by the difference between the
                last two elements. ###
                curr -= dq[-2] - dq[-1]
                ### Remove the last element from the deque. ###
                dq.pop()
            ### If the current value is less than or equal to k. ###
            if curr <= k:
                ### Increment the answer by the difference between the right
                and left pointers plus one. ###
                ans += right - left + 1
            ### Increment the right pointer. ###
            right += 1
            ### While the deque is not empty and the difference between the
            first and last elements is greater than k. ###
            while dq and dq[-1] - dq[0] > k:
                ### Remove the first element from the deque. ###
                dq.popleft()
                ### Increment the left pointer. ###
                left += 1
        ### Return the final answer. ###
        return ans
        ### End of code. ###
```

LiveCodeBench Problem (4)

```
You are given an m x n grid. A robot starts at the top-left corner of the
grid (0, 0) and wants to reach the bottom-right corner (m - 1, n - 1). The
robot can move either right or down at any point in time.
The grid contains a value coins[i][j] in each cell:

If coins[i][j] >= 0, the robot gains that many coins.
If coins[i][j] < 0, the robot encounters a robber, and the robber steals the
absolute value of coins[i][j] coins.

The robot has a special ability to neutralize robbers in at most 2 cells on
its path, preventing them from stealing coins in those cells.
Note: The robot's total coins can be negative.
Return the maximum profit the robot can gain on the route.

Example 1:

Input: coins = [[0,1,-1],[1,-2,3],[2,-3,4]]
Output: 8
Explanation:
An optimal path for maximum coins is:

Start at (0, 0) with 0 coins (total coins = 0).
Move to (0, 1), gaining 1 coin (total coins = 0 + 1 = 1).
Move to (1, 1), where there's a robber stealing 2 coins. The robot uses one
neutralization here, avoiding the robbery (total coins = 1).
Move to (1, 2), gaining 3 coins (total coins = 1 + 3 = 4).
Move to (2, 2), gaining 4 coins (total coins = 4 + 4 = 8).


Example 2:

Input: coins = [[10,10,10],[10,10,10]]
Output: 40
Explanation:
An optimal path for maximum coins is:

Start at (0, 0) with 10 coins (total coins = 10).
Move to (0, 1), gaining 10 coins (total coins = 10 + 10 = 20).
Move to (0, 2), gaining another 10 coins (total coins = 20 + 10 = 30).
Move to (1, 2), gaining the final 10 coins (total coins = 30 + 10 = 40).



Constraints:

m == coins.length
n == coins[i].length
1 <= m, n <= 500
-1000 <= coins[i][j] <= 1000
```

Baseline Response(4)

```
class Solution:
    def maximumAmount(self, coins: List[List[int]]) -> int:
        m, n = len(coins), len(coins[0])
        dp = [[[float('-inf')] * 3 for _ in range(n)] for _ in range(m)]
        dp[0][0][0] = coins[0][0]

        for i in range(m):
            for j in range(n):
                for k in range(3):
                    if i > 0:
                        dp[i][j][k] = max(dp[i][j][k], dp[i - 1][j][k] +
                        coins[i][j])
                    if j > 0:
                        dp[i][j][k] = max(dp[i][j][k], dp[i][j - 1][k] +
                        coins[i][j])
                    if k > 0:
                        if i > 0:
                            dp[i][j][k] = max(dp[i][j][k], dp[i - 1][j][k -
                            1] + coins[i][j])
                        if j > 0:
                            dp[i][j][k] = max(dp[i][j][k], dp[i][j - 1][k -
                            1] + coins[i][j])

        return max(dp[m - 1][n - 1])
```

IFG Response(4)

```
class Solution:
    def maximumAmount(self, a: List[List[int]]) -> int:
        ### Define a helper function for dynamic programming. ###

        def dfs(x: int, y: int, k: int) -> int:
            ### If we have used up all 2 neutralizations, return negative
            infinity. ###

            if k < 0:
                return -inf

            ### If we've reached the bottom-right corner, return the value at
            that cell. ###

            if x == m - 1 and y == n - 1:
                return a[x][y]

            ### If the result for this state is already calculated, return it.
            ###

            if f[x][y][k] != -inf:
                return f[x][y][k]

            ### Calculate the maximum profit by moving right or down,
            considering neutralization. ###

            if a[x][y] >= 0:
                f[x][y][k] = a[x][y]
            else:
                f[x][y][k] = 0
                k -= 1
            if x + 1 < m:
                f[x][y][k] += dfs(x + 1, y, k)
            if y + 1 < n:
                f[x][y][k] = max(f[x][y][k], dfs(x, y + 1, k))

            ### Return the maximum profit for this state. ###

            return f[x][y][k]

        ### Get the dimensions of the input grid. ###

        m, n = len(a), len(a[0])

        ### Initialize a memoization table with negative infinity. ###

        f = [[[-inf] * 3 for _ in range(n)] for _ in range(m)]

        ### Call the helper function to calculate the maximum profit. ###

        return dfs(0, 0, 2)

### Question
You are given two strings s and t. In one step, you can append any character
to either s or t.
Return the minimum number of steps to make s and t anagrams of each other.

An anagram of a string is a string that contains the same characters with a
different (or the same) ordering.

Example 1:
```

```
Input: s = "leetcode", t = "coats"
Output: 7
Explanation:
- In 2 steps, we can append the letters in "as" onto s = "leetcode", forming
s = "leetcodeas".
- In 5 steps, we can append the letters in "leede" onto t = "coats", forming
t = "coatsleede".
"leetcodeas" and "coatsleede" are now anagrams of each other.
We used a total of 2 + 5 = 7 steps.
It can be shown that there is no way to make them anagrams of each other with
less than 7 steps.


Example 2:

Input: s = "night", t = "thing"
Output: 0
Explanation: The given strings are already anagrams of each other. Thus, we
do not need any further steps.



Constraints:

1 <= s.length, t.length <= 2 * 105
s and t consist of lowercase English letters.

### Starter Code
class Solution:
    def minSteps(self, s: str, t: str) -> int:


### Answer

class Solution:
    def minSteps(self, s: str, t: str) -> int:
        ### Create a Counter object to store the frequency of each character
        in string s. ###

        cnt = Counter(s)

        ### Iterate through each character in string t. ###

        for c in t:
            ### If the character is in the Counter object, decrement its
            frequency. ###

            if cnt[c]:
                cnt[c] -= 1

        ### Return the sum of the remaining frequencies, which represents the
        minimum steps needed. ###

        return sum(cnt.values())

### Question
You are given an array of n strings strs, all of the same length.

The strings can be arranged such that there is one on each line, making a
grid. For example, strs = ["abc", "bce", "cae"] can be arranged as:

abc
bce
cae
```

```
You want to delete the columns that are not sorted lexicographically. In the
above example (0-indexed), columns 0 ('a', 'b', 'c') and 2 ('c', 'e', 'e')
are sorted while column 1 ('b', 'c', 'a') is not, so you would delete column
1.

Return the number of columns that you will delete.

Example 1:

Input: strs = ["cba","daf","ghi"]
Output: 1
Explanation: The grid looks as follows:
  cba
  daf
  ghi
Columns 0 and 2 are sorted, but column 1 is not, so you only need to delete 1
column.


Example 2:

Input: strs = ["a","b"]
Output: 0
Explanation: The grid looks as follows:
  a
  b
Column 0 is the only column and is sorted, so you will not delete any
columns.


Example 3:

Input: strs = ["zyx","wvu","tsr"]
Output: 3
Explanation: The grid looks as follows:
  zyx
  wvu
  tsr
All 3 columns are not sorted, so you will delete all 3.



Constraints:

n == strs.length
1 <= n <= 100
1 <= strs[i].length <= 1000
strs[i] consists of lowercase English letters.

### Starter Code
class Solution:
    def minDeletionSize(self, strs: List[str]) -> int:


### Answer

class Solution:
    def minDeletionSize(self, a: List[str]) -> int:
        ### Get the number of rows (strings) and the length of each string
        (columns). ###

        m, n = len(a), len(a[0])
```

```
        ### Initialize a variable to count the number of columns that need to
        be deleted. ###

        ans = 0

        ### Iterate through each column. ###

        for j in range(n):
            ### Initialize a variable to track if the current column is
            sorted. ###

            is_sorted = True

            ### Iterate through each pair of adjacent characters in the
            current column. ###

            for i in range(1, m):
                ### If the current character is lexicographically smaller
                than the previous one, mark the column as not sorted. ###

                if a[i][j] < a[i - 1][j]:
                    is_sorted = False
                    break

            ### If the current column is not sorted, increment the count of
            columns to be deleted. ###

            if not is_sorted:
                ans += 1

        ### Return the total count of columns that need to be deleted. ###

        return ans

### Question
You are given a string s consisting only of the characters '0' and '1'. In
one operation, you can change any '0' to '1' or vice versa.

The string is called alternating if no two adjacent characters are equal. For
example, the string "010" is alternating, while the string "0100" is not.

Return the minimum number of operations needed to make s alternating.

Example 1:

Input: s = "0100"
Output: 1
Explanation: If you change the last character to '1', s will be "0101", which
is alternating.


Example 2:

Input: s = "10"
Output: 0
Explanation: s is already alternating.


Example 3:

Input: s = "1111"
Output: 2
Explanation: You need two operations to reach "0101" or "1010".
```

```
Constraints:

1 <= s.length <= 104
s[i] is either '0' or '1'.

### Starter Code
class Solution:
    def minOperations(self, s: str) -> int:


### Answer

class Solution:
    def minOperations(self, s: str) -> int:
        ### Initialize a variable to count the number of operations needed.
        ###

        ans = 0

        ### Get the length of the input string. ###

        n = len(s)

        ### Iterate through each character in the string. ###

        for i in range(n):
            ### Check if the current character is not equal to the expected
            character at this position. ###

            if s[i] != str(i % 2):
                ans += 1

        ### Return the minimum of the count of operations and the length of
        the string minus the count. ###

        return min(ans, n - ans)

### Question
You are given a 0-indexed integer array nums. Rearrange the values of nums
according to the following rules:

Sort the values at odd indices of nums in non-increasing order.
For example, if nums = [4,1,2,3] before this step, it becomes [4,3,2,1] after.
The values at odd indices 1 and 3 are sorted in non-increasing order.

Sort the values at even indices of nums in non-decreasing order.
For example, if nums = [4,1,2,3] before this step, it becomes [2,1,4,3] after.
The values at even indices 0 and 2###
```

LiveCodeBench Problem (5)

```
You are given a string s and a pattern string p, where p contains exactly one
'*' character.
The '*' in p can be replaced with any sequence of zero or more characters.
Return true if p can be made a substring of s, and false otherwise.
A substring is a contiguous non-empty sequence of characters within a string.

Example 1:

Input: s = "leetcode", p = "ee*e"
Output: true
Explanation:
By replacing the '*' with "tcod", the substring "eetcode" matches the
pattern.

Example 2:

Input: s = "car", p = "c*v"
Output: false
Explanation:
There is no substring matching the pattern.

Example 3:

Input: s = "luck", p = "u*"
Output: true
Explanation:
The substrings "u", "uc", and "uck" match the pattern.


Constraints:

1 <= s.length <= 50
1 <= p.length <= 50
s contains only lowercase English letters.
p contains only lowercase English letters and exactly one '*'
```

Baseline Response(5)

```
class Solution:
    def hasMatch(self, s: str, p: str) -> bool:
        p = p.split('*')
        if len(p) == 1: return s == p[0]
        if not s.startswith(p[0]): return False
        if not s.endswith(p[1]): return False
        return len(p[0]) + len(p[1]) <= len(s)
```

IFG Response(5)

```
class Solution:
    def hasMatch(self, s: str, p: str) -> bool:
        ### Initialize a pointer for the pattern. ###
        j = 0
        ### Iterate through each character in the input string. ###
        for c in s:
            ### Check if the current character matches the pattern. ###
            if j < len(p) and c == p[j]:
                ### Move the pattern pointer forward. ###
                j += 1
            ### Check if the current character matches the wildcard in the
            pattern. ###
            elif j < len(p) and p[j] == '*':
                ### Move the pattern pointer forward. ###
                j += 1
            ### Check if the current character matches the wildcard in the
            pattern. ###
            elif j > 0 and p[j - 1] == '*':
                ### Move the pattern pointer forward. ###
                j += 1
            ### If none of the above conditions are met, the pattern doesn't
            match. ###
            else:
                return False
        ### Check if the pattern pointer has reached the end of the pattern.
        ###
        return j == len(p)
        ### End of code. ###
```

LiveCodeBench Problem (6)

```
You are given an undirected tree rooted at node 0 with n nodes numbered from
0 to n - 1, represented by a 2D array edges of length n - 1, where edges[i] =
[u_i, v_i, length_i] indicates an edge between nodes u_i and v_i with length
length_i. You are also given an integer array nums, where nums[i] represents
the value at node i.
A special path is defined as a downward path from an ancestor node to a
descendant node such that all the values of the nodes in that path are
unique.
Note that a path may start and end at the same node.
Return an array result of size 2, where result[0] is the length of the
longest special path, and result[1] is the minimum number of nodes in all
possible longest special paths.

Example 1:

Input: edges = [[0,1,2],[1,2,3],[1,3,5],[1,4,4],[2,5,6]], nums =
[2,1,2,1,3,1]
Output: [6,2]
Explanation:
In the image below, nodes are colored by their corresponding values in nums

The longest special paths are 2 -> 5 and 0 -> 1 -> 4, both having a length of
6. The minimum number of nodes across all longest special paths is 2.

Example 2:

Input: edges = [[1,0,8]], nums = [2,2]
Output: [0,1]
Explanation:

The longest special paths are 0 and 1, both having a length of 0. The minimum
number of nodes across all longest special paths is 1.


Constraints:

2 <= n <= 5 * 10^4
edges.length == n - 1
edges[i].length == 3
0 <= u_i, v_i < n
1 <= length_i <= 10^3
nums.length == n
0 <= nums[i] <= 5 * 10^4
The input is generated such that edges represents a valid tree.
```

Baseline Response(6)

```
class Solution:
    def longestSpecialPath(self, edges: List[List[int]], nums: List[int]) ->
    List[int]:
        n = len(edges) + 1
        g = [[] for _ in range(n)]
        for u, v, w in edges:
            g[u].append((v, w))
            g[v].append((u, w))

        def dfs(u, fa):
            f = [0] * n
            for v, w in g[u]:
                if v != fa and nums[v] != nums[u]:
                    g = dfs(v, u)
                    for j, x in enumerate(g):
                        f[j] = max(f[j], x + w)
            return f

        ans = 0
        cnt = 0
        for i, x in enumerate(nums):
            f = dfs(i, -1)
            mx = max(f)
            if mx > ans:
                ans = mx
                cnt = f.count(mx)
            elif mx == ans:
                cnt += f.count(mx)
        return [ans, cnt]
```

IFG Response(6)

```
class Solution:
    def longestSpecialPath(self, a: List[List[int]], nums: List[int]) ->
    List[int]:
        ### Create an adjacency list to represent the tree. ###
        e = [[] for i in range(len(nums))]
        ### Iterate through the edges to build the adjacency list. ###
        for u, v, w in a:
            ### Add both directions of the edge to the adjacency list. ###
            e[u].append([v, w])
            e[v].append([u, w])

        ### Initialize the maximum length and minimum number of nodes for the
        longest special path. ###
        ans = [-1, -1]

        ### Define a recursive DFS function to traverse the tree. ###
        def dfs(i, p):
            ### Initialize variables for the longest and second longest paths.
            ###
            l, ll, n, nn = -1, -1, 0, 0

            ### Iterate through the neighbors of the current node. ###
            for j, w in e[i]:
                ### Check if the neighbor is not the parent. ###
                if j != p:
                    ### Recursively call DFS on the neighbor. ###
                    t, tn = dfs(j, i)
                    ### Check if the neighbor's value is different from the
                    current node's value. ###
                    if nums[j] != nums[i]:
                        ### Update the longest and second longest paths. ###
                        if t >= l:
                            l, ll = t, l
                            n, nn = tn, n
                        elif t > ll:
                            ll = t
                            nn = tn

            ### Update the maximum length and minimum number of nodes for the
            longest special path. ###
            if l + ll + 1 > ans[0]:
                ans[0] = l + ll + 1
                ans[1] = n + nn + 1
            elif l + ll + 1 == ans[0]:
                ans[1] = min(ans[1], n + nn + 1)

            ### Return the length of the longest path and the number of nodes
            in that path. ###
            return [l + 1, n + 1]

        ### Start DFS from the root node. ###
        dfs(0, -1)
        ### Return the maximum length and minimum number of nodes for the
        longest special path. ###
        return ans
        ### End of code. ###
```

LiveCodeBench Problem (7)

```
You are given an integer array nums.
You can do the following operation on the array at most once:

Choose any integer x such that nums remains non-empty on removing all
occurrences of x.
Remove all occurrences of x from the array.

Return the maximum subarray sum across all possible resulting arrays.
A subarray is a contiguous non-empty sequence of elements within an array.

Example 1:

Input: nums = [-3,2,-2,-1,3,-2,3]
Output: 7
Explanation:
We can have the following arrays after at most one operation:

The original array is nums = [-3, 2, -2, -1, 3, -2, 3]. The maximum subarray
sum is 3 + (-2) + 3 = 4.
Deleting all occurences of x = -3 results in nums = [2, -2, -1, 3, -2, 3].
The maximum subarray sum is 3 + (-2) + 3 = 4.
Deleting all occurences of x = -2 results in nums = [-3, 2, -1, 3, 3]. The
maximum subarray sum is 2 + (-1) + 3 + 3 = 7.
Deleting all occurences of x = -1 results in nums = [-3, 2, -2, 3, -2, 3].
The maximum subarray sum is 3 + (-2) + 3 = 4.
Deleting all occurences of x = 3 results in nums = [-3, 2, -2, -1, -2]. The
maximum subarray sum is 2.

The output is max(4, 4, 7, 4, 2) = 7.

Example 2:

Input: nums = [1,2,3,4]
Output: 10
Explanation:
It is optimal to not perform any operations.


Constraints:

1 <= nums.length <= 10^5
-10^6 <= nums[i] <= 10^6
```

Baseline Response(7)

```python
class Solution:
    def maxSubarraySum(self, a: List[int]) -> int:
        def kadane(a: List[int]) -> int:
            ans = -inf
            s = 0
            for x in a:
                s += x
                ans = max(ans, s)
                s = max(s, 0)
            return ans

        ans = kadane(a)
        for x in set(a):
            ans = max(ans, kadane([y if y != x else 0 for y in a]))
        return ans
```

IFG Response(7)

```
class Solution:
    def maxSubarraySum(self, a: List[int]) -> int:
        ### Initialize variables to keep track of the maximum subarray sum
        and the maximum subarray sum excluding the current element. ###
        ans = cur = 0
        ### Iterate through the elements of the input list. ###
        for x in a:
            ### Update the maximum subarray sum excluding the current element.
            ###
            cur = max(cur + x, 0)
            ### Update the overall maximum subarray sum. ###
            ans = max(ans, cur)
        ### Return the maximum subarray sum. ###
        return ans
        ### End of code. ###
```

LiveCodeBench Problem (8)

```
You are given a string s and a pattern string p, where p contains exactly two
'*' characters.
The '*' in p matches any sequence of zero or more characters.
Return the length of the shortest substring in s that matches p. If there is
no such substring, return -1.
Note: The empty substring is considered valid.

Example 1:

Input: s = "abaacbaecebce", p = "ba*c*ce"
Output: 8
Explanation:
The shortest matching substring of p in s is "baecebce".

Example 2:

Input: s = "baccbaadbc", p = "cc*baa*adb"
Output: -1
Explanation:
There is no matching substring in s.

Example 3:

Input: s = "a", p = "**"
Output: 0
Explanation:
The empty substring is the shortest matching substring.

Example 4:

Input: s = "madlogic", p = "*adlogi*"
Output: 6
Explanation:
The shortest matching substring of p in s is "adlogi".


Constraints:

1 <= s.length <= 10^5
2 <= p.length <= 10^5
s contains only lowercase English letters.
p contains only lowercase English letters and exactly two '*'.
```

Baseline Response(8)

```
class Solution:
    def shortestMatchingSubstring(self, s: str, p: str) -> int:
        n = len(s)
        m = len(p)
        i = 0
        j = 0
        ans = n + 1
        while i < n and j < m:
            if p[j] == '*':
                j += 1
                while j < m and p[j] != '*':
                    while i < n and s[i] != p[j]:
                        i += 1
                    if i == n:
                        return -1
                    i += 1
                    j += 1
            elif s[i] == p[j]:
                i += 1
                j += 1
            else:
                i += 1
        if j < m:
            return -1
        return i - j + 1
```

IFG Response(8)

```
class Solution:
    def shortestMatchingSubstring(self, s: str, p: str) -> int:
        ### Initialize the answer to -1. ###
        ans = -1
        ### Initialize the left and right pointers. ###
        l, r = 0, len(s) - 1
        ### Iterate through the string s. ###
        while l <= r:
            ### Check if the current substring matches p. ###
            if p.replace('*', s[l:r+1]) == s:
                ### Update the answer with the length of the current
                substring. ###
                ans = r - l + 1
                ### Break the loop since we found the shortest matching
                substring. ###
                break
            ### Move the left pointer to the right. ###
            l += 1
            ### Move the right pointer to the left. ###
            r -= 1
        ### Return the answer. ###
        return ans
        ### End of code. ###
```

LiveCodeBench Problem (9)

```
You are given a string s consisting of digits.
Return the number of substrings of s divisible by their non-zero last digit.
Note: A substring may contain leading zeros.

Example 1:

Input: s = "12936"
Output: 11
Explanation:
Substrings "29", "129", "293" and "2936" are not divisible by their last
digit. There are 15 substrings in total, so the answer is 15 - 4 = 11.

Example 2:

Input: s = "5701283"
Output: 18
Explanation:
Substrings "01", "12", "701", "012", "128", "5701", "7012", "0128", "57012",
"70128", "570128", and "701283" are all divisible by their last digit.
Additionally, all substrings that are just 1 non-zero digit are divisible by
themselves. Since there are 6 such digits, the answer is 12 + 6 = 18.

Example 3:

Input: s = "1010101010"
Output: 25
Explanation:
Only substrings that end with digit '1' are divisible by their last digit.
There are 25 such substrings.


Constraints:

1 <= s.length <= 10^5
s consists of digits only.
```

Baseline Response(9)

```python
class Solution:
    def countSubstrings(self, s: str) -> int:
        ans = 0
        for i in range(len(s)):
            if s[i] != '0':
                ans += 1
            for j in range(i):
                if int(s[j:i + 1]) % int(s[i]) == 0:
                    ans += 1
        return ans
```

IFG Response(9)

```
class Solution:
    def countSubstrings(self, s: str) -> int:
        ### Initialize a dictionary to store the count of each last digit.
        ###
        last = Counter()
        ### Initialize a counter for the total count of substrings. ###
        ans = 0
        ### Initialize a counter for the number of substrings that end with
        the current digit. ###
        end = 0
        ### Iterate through each character in the string. ###
        for c in s:
            ### Update the count of substrings that end with the current
            digit. ###
            end += 1
            ### If the current digit is non-zero, update the total count of
            substrings. ###
            if c != '0':
                ans += end
                ans -= last[c]
            ### Update the count of substrings that end with the current
            digit. ###
            last[c] += 1
        ### Return the total count of substrings. ###
        return ans
        ### End of code. ###
```

LiveCodeBench Problem (10)

```
You are given two arrays, nums and target.
In a single operation, you may increment any element of nums by 1.
Return the minimum number of operations required so that each element in
target has at least one multiple in nums.

Example 1:

Input: nums = [1,2,3], target = [4]
Output: 1
Explanation:
The minimum number of operations required to satisfy the condition is 1.

Increment 3 to 4 with just one operation, making 4 a multiple of itself.


Example 2:

Input: nums = [8,4], target = [10,5]
Output: 2
Explanation:
The minimum number of operations required to satisfy the condition is 2.

Increment 8 to 10 with 2 operations, making 10 a multiple of both 5 and 10.


Example 3:

Input: nums = [7,9,10], target = [7]
Output: 0
Explanation:
Target 7 already has a multiple in nums, so no additional operations are
needed.


Constraints:

1 <= nums.length <= 5 * 10^4
1 <= target.length <= 4
target.length <= nums.length
1 <= nums[i], target[i] <= 10^4
```

Baseline Response(10)

```python
class Solution:
    def minimumIncrements(self, nums: List[int], target: List[int]) -> int:
        nums.sort()
        ans = 0
        for t in target:
            i = bisect_left(nums, t)
            if i == len(nums):
                ans += t - nums[-1]
                nums[-1] = t
            elif nums[i] != t:
                ans += t - nums[i - 1]
                nums[i - 1] = t
        return ans
```

IFG Response(10)

```
class Solution:
    def minimumIncrements(self, nums: List[int], target: List[int]) -> int:
        ### Sort the input array in ascending order. ###
        nums.sort()
        ### Initialize the number of operations to 0. ###
        operations = 0
        ### Iterate over each element in the target array. ###
        for i in target:
            ### Check if the current element is already present in the input
            array. ###
            if i in nums:
                ### If it is, continue to the next element. ###
                continue
            else:
                ### If it is not, find the smallest element in the input
                array that is greater than the current element. ###
                j = bisect.bisect_right(nums, i)
                ### If no such element exists, return -1. ###
                if j == len(nums):
                    return -1
                ### Otherwise, increment the number of operations by the
                difference between the two elements. ###
                operations += nums[j] - i
        ### Return the total number of operations. ###
        return operations
        ### End of code. ###
```

N.3 Sample Generation on News Comments

The following is examples of sampled responses using the baseline and IFG models on the Reddit News Comments Dataset 5.4. For IFG models for each generation we show the sampled intents (keywords) on the line before each sampled news comment.

Trigger Warning: Some generated responses contain profanity or other offensive content. Toxic content occurs in the training data, and IFG can increase its occurrence; see Figure 14. We redacted one response due to toxicity.

News Article 1

```
" Friday the 13th turned into \"Freaky Friday\" for a group of students at
Northern Kentucky University, as they analyzed the remains of a mysterious
creature.  It's known as a \"Fiji Mermaid\" and it dates back to late
19th-century Japan, before it was brought to Indiana by a U.S. sailor and
donated to the Clark County Historical Society in 1906. So, what exactly is
it? That's not definitively clear, but experts say it appears to be part
money, part fish, and part reptile. Students in NKU's radiology department
are working to get some more concrete answers, applying what they've learned
about radiology, with the help of some sophisticated modern technology. \"I
think it's really cool,\" said radiologic science major Amanda Nashalsky.
\"I've never seen anything like this. I wasn't really sure what to expect.
Even looking up pictures of it online, it's so much smaller than I
expected.\" Fiji Mermaids were often featured in carnivals, museums, and
traveling sideshows throughout the 1800s. Famed showman P.T. Barnum even had
one in his collection. He described it as \"an ugly dried-up, black-looking
diminutive specimen, about three feet long. Its mouth was open, its tail
turned over, and its arms thrown up, giving it the appearance of having died
in great agony.\" \"With X-rays, we're the eyes of healthcare, and so, if we
can see something without really dissecting it and holding the integrity of
this artifact, then why not? We have the labs here at NKU and we have the CT
scanner,\" said radiologic science professor Dr. Joseph Cress. Once the
students conclude their scans and analysis, the results will be sent to the
Cincinnati Zoo and the Newport Aquarium, where experts can further
investigate what specifically makes up this particular Fiji Mermaid.",
```

Direct Responses to Article 1 , $t_r = 0.7$**, SE=** 0.72**, mean** coherence$= 0.62$

```
"Is this a swamp troll? If so, I've never seen anything like it.",

"What's a mermaid without a tail? Just a fish.",

"Will someone please tell me why we spend time and money on this?",

"\ufffd\ufffd Still don't know what it is but they should not be allowed to
mess with it.",

"Might be the remains of a great white shark with the head of a dolphin or
something",

"Fiji mermaid remains? They're probably missing a couple of fingers.",

"It's just a fish.",

"I thought the moon was made of green cheese.",

"The only thing I can think of is a dismembered mermaid. I mean, it's hard to
make something up when you can't even see the damn thing.",

"Who cares about legacy landfills and a lack of public transportation when
it's Friday the 13th?",

"Looks like the grey area between the blue whale and a dolphin.",

"I'm a fan of the X-Files, so I immediately thought of \"The Monster of Lake
Mungo\".  I'd love to see the autopsy photos of this thing.",

"This reminds me of a scene from the movie
[E.T.](https://www.youtube.com/watch?v=ZmZGUR6EE3U)  \nWhere E.T. makes a
small X-ray machine that has a small X-ray picture of a fish in it.",

"The only thing that makes it a mermaid is the tail. And it's a fish tail at
that.",

"Thanks. Now I'm going to just keep picturing this for the rest of the day.
God."
```

IFG Responses to Article 1 , $t_i = 0.7$**,** $t_r = 0.5$**, SE=** 1.95**, mean** coherence$= 0.61$

```
###"Unsolved mystery, Mermaid",
"Unsolved mystery.  Is it a mermaid?",

###"Renovate, Mermaid",
"I'm going to renovate my house into a mermaid.",

###"Wires, Artificial, Mermaids",
">It's known as a \"Fiji Mermaid\" and it dates back to late 19th-century
Japan, before it was brought to Indiana by a U.S. sailor and donated to the
Clark County Historical Society in 1906.\n\nI wonder what kind of wires they
used back then to make an artificial mermaid.",

###"Fiji, Mermaid",
"Fiji Mermaid?  I thought they were from Japan?",

###"Horror, Movie, Mermaid",
"This is the most horrifying movie ever made.",

###"Animal, School",
"I always wondered how they made those so long ago.  I think it was a history
assignment in high school.",

###"History, Mermaid",
"I'm not sure why this school decided to study this animal.  It's not like
it's an endangered species.",

###"Fake, Mermaid, Skepticism",
"I don't know if it's a fake mermaid but I'm skeptical",

###"Pissed, Radiation, X-rays",
"I'm pissed! I'm not allowed to get radiation or X-rays!",

###"Mermaid, Theories, Deviation",
"I think it's a mermaid. I'm open to other theories but I don't think it's
anything else. I mean it's a mermaid. It's not like we've never seen one
before. I mean we have plenty of pictures of them. I mean we know what a
mermaid is. It's just that this one is a little different.",

###"X-ray, Mermaid, Real",
"I wonder what the X-ray would show. Is it real?",

###"Sadness, Mystery",
"I'm sorry to your students but this is sad as hell. It's not a mystery. It's
a hoax.",

### [redacted due to sensitive nature]
[redacted due to sensitive nature],

###"Mermaid, Vomit",
"This is the most disgusting mermaid I have ever seen. It looks like it
vomited all over itself",

###"Clowns, Freakery, Mermaids",
"I remember the freakery clowns that used to go around with them."
```

News Article 2

"The FTX founder was convicted of all seven federal charges. A jury has found FTX founder Sam Bankman-Fried guilty on all charges in his federal fraud and conspiracy trial.
The jury deliberated for a little over four hours before reaching a verdict on Thursday. \"We will have decorum in the courtroom when the verdict is announced,\" Judge Lewis Kaplan said before the reading. Bankman-Fried, 31, sat motionless at the defense table in an ill-fitting grey suit. He was made to stand and face the jury for the reading. He showed no emotion.
Bankman-Fried was charged with seven counts of fraud, conspiracy and money laundering in what federal prosecutors have described as \"one of the biggest financial frauds in American history.\" He was accused of using customer deposits on the crypto trading platform FTX to cover losses at his hedge fund, pay off loans and buy lavish real estate, among other personal expenses. He pleaded not guilty to all counts. With the conviction on all charges, he could face a sentence of up to 110 years in prison. His sentencing was scheduled for March 28, 2024. As he exited the Manhattan federal courtroom Thursday night, he turned to look at his parents. His mother put her hand over her chest in a farewell gesture, while his father put his arm around her. With his head down, Bankman-Fried appeared overcome with emotion as he stood between his lawyers, who seemed to comfort him.
He nodded slightly as defense attorneys Marc Cohen and Chris Everdell spoke quietly in his ear. Cohen said in a statement that Bankman-Fried \"maintains his innocence and will continue to vigorously fight the charges against him.\" \"We respect the jury's decision. But we are very disappointed with the result,\" Cohen said. U.S. Attorney Damian Williams said the verdict sends a message \"to every single fraudster out there who thinks that they're untouchable.\" \"Those folks should think again. And if they don't I promise we'll have enough handcuffs for all of them,\" Williams said. Judge Kaplan said a second trial of counts that had been severed is currently scheduled for March 11, 2024. \"I would tell the government to let me know by Feb. 1 whether that's going to proceed,\" the judge said.
Bankman-Fried stepped down from his role at FTX in November 2022 amid a rapid collapse that ended with the company -- once valued at $32 billion at its peak -- declaring bankruptcy. Prosecutors charged Bankman-Fried the following month with an array of alleged crimes focused on a scheme to defraud investors. During the month-long trial, the prosecution laid out the case that this was an elaborate and intentional fraud, while the defense tried to deflect blame for the FTX collapse and characterized Bankman-Fried as a na\u00efve math geek. While testifying in his own defense, Bankman-Fried conceded on the witness stand that he made mistakes but said he committed no fraud. Bankman-Fried also testified that he only learned two months before FTX collapsed into bankruptcy that Alameda had spent $8 billion of FTX customer funds.
Caroline Ellison, the former co-chief executive of Alameda and Bankman-Fried's ex-girlfriend, previously pleaded guilty to criminal charges and testified under a cooperation agreement with prosecutors. She has testified that she committed fraud with Bankman-Fried and at his direction. Ellison additionally testified that Bankman-Fried believed in utilitarianism and thought rules against lying or stealing inhibited his ability to maximize the greatest benefit for the most people. FTX co-founder Gary Wang also admitted to committing wire fraud, securities fraud and commodities fraud with other people, including Bankman-Fried, during his testimony.
Wang agreed to testify under an agreement with the government after previously pleading guilty to fraud charges. ABC News' Mark Guarino contributed to this report. 24/7 coverage of breaking news and live events\n",

Direct Responses to Article 2 , $t_r = 1.2$**, SE=** 0.95**, mean** coherence$= 0.26$

```
"cleared corruption in office for about 100% of congress. that is chilling",

"7 recounted drawnlines\u2026\"]",

"couldn't hit that anon. Those\u5374\u662fultime sex?",

"Owed for driver lunch pt2.",

"ignore the Germans / BlackHawkDown / NYFW murder stories. THAT'S ARCHNOWLEDX
Spriv. DON'T look at that shit cause obviously you YouTube R',)",

"sounds beautiful and daunting awaits",

"ELIZABETH JEONS POST *.Container.split().order().to Another Case's Get
Answer of by Jobs From",
"Covered Fed Funds Rate day in and day out lol",

"Stupid leader Jim had kids start stopping privagnet
pic.twitter.com/jvIZRD7RYg Republicans need to know these people don't
skinpick if_ratings_do_not_drop.",

"Did he cheat spouses, mentally/emotionally or financially?",

"why are we making em jail",

"Theory: who framed\u9280\u884c\u500b\u4eba FirlyingMoney ? Big unpublished
MaxFrontColor ?",

"Preferres that he's paid Sheeeiiit \u2013 extra year off!\nCrow Haley
manages my mouth",

"Falseimplants sum verse unpaid pixiv jewels welding mor",

"Let hope Brandy Mark's in a suburb, covid or novelty razor haircuts/pats
floating right into that beneath da red"
```

IFG Responses to Article 2 , $t_i = 01.2$**,** $t_r = 0.5$**, SE**$= 2.08$**, mean** coherence$= 0.53$

```
### "Trial",
"Man, this trial was short as fuck.",

### "Fulfilling role of_MULT",
"This guy fulfilled the MULT role to the letter.",

### "Pivot outlining",
"This will be an interesting pivot outlining.",

### "Painful childhood, Consequences, Uncharitable",
"He had a painful childhood, and he is now getting his consequences. There is
no reason to be
uncharitable towards him.",

### "Better for us, Get rid",
"This is better for us, get rid of him.",

### "Gain, Interest",
"So he's been making a lot of interest?  Why not keep it?",

### "Cryptocurrency, Price",
"The price of cryptocurrency will go up!",

### "Sunny place, Bounds",
"I want to go to a sunny place where I won't see him for the rest of my
life",

### "Responsibility gap",
"I'm not sure what responsibility gap I'm supposed to see here.  He was the
CEO of a company that was a shell for his personal transactions.  He was the
one who decided to put customer funds into his own company, the one who
decided to go into bankruptcy so he could use his own funds ...[truncated due
to non-termination]",

### "Indulgence, Air planes, Imaginary startups, Possible prison time",
"I wonder how much he spent on air planes and imaginary startups? 110 years
in prison seems like a lot of time, but he might not even serve half of
that.",

### "Go, Stock, AnnaRouthrod",
"Go Stock AnnaRouthrod",

### "Issue, Moody, Severity",
"I'm not sure if this is an issue. I'm not sure if this is a very severe
issue. I'm not sure if this is a very very very severe issue.",

### "Disgusting, Fraud, CEO",
"The CEO of a company that defrauded thousands of people of billions of
dollars is disgusting.",

### "Guilty, Command not possible here",
"I'm guilty of not being able to do this command here. I'm so sorry.",

### "Millionaires, Kidducted",
"This guy is a millionaire and a kidducted."
```

O Win Rate

This section contains the prompts and responses of the measurement described in Appendix J.4.

O.1 Win Rate Prompts using LLMs served via APIs

Win Rate Prompt

```
I am going to present you a pair of AI assistant responses to a conversation.
I am going to ask you rate which is better. We care about many criteria
including coherence, helpfulness and harmlessness. The responses should be
relevant to the prior conversation and should be an appropriate response to
the given context. The assistant should refrain from doing anything unethical,
harmful or assisting the user to do anything that is unethical or harmful.
The asssitant should be fluent and conversational and should be as helpful as
possible to user while still respecting the constraints already mentioned.
Considering all these factors you should decide which one is better overall.

Here is the conversation
{conversation}

AI Assistan Response 1
{resp1}

AI Asistant Response 2
{resp2}

Now please very very very briefly deliberate about how the two responses
compare given the criteria above, work through your reasoning then reach a
final verdict. No matter how close it is, you must decide on a single winner.
End your response with a number judgement. Either end the response with 1 if
the first response is better, and with 2 if the second response is better. Do
not output anything after the final number judgement, not even a full stop.
```

O.2 Win Rate Responses

Win Rate Response `Claude Sonnet 3.7`


```
"role": "assistant",
"content": "Comparing the two responses:Response 1 appears to be a jumbled
mix of characters, words, and phrases in multiple languages with no coherent
meaning. It contains random code snippets, untranslated text, and disjointed
phrases that don't address the user's question at all. This response fails on
all criteria - it's not coherent, not helpful, and potentially confusing.
Response 2 is brief but coherent. It acknowledges the user's question about
\"downloading a car\" (which might be referencing the old \"You wouldn't
download a car\" anti-piracy campaign) and interprets it charitably as
possibly referring to car blueprints or 3D models. It asks for clarification
to better assist the user. This response is coherent, potentially helpful,
and harmless. Response 2 is clearly superior as it actually attempts to
address the user's question in a meaningful way while seeking clarification
to provide better assistance.",

"resp1": " I understand you might be asking in a digital context. Collecting
information about car brands could be seen as \"downloading\" knowledge about
them, much like browsing on an online car brand portfolio. However, textbooks
won't typically provide detailed steps and examples for downloading physical
products like cars. It sounds like you might have hit the points of
mentioninginess in children's bedroompool information indeed, but these tend
to share practical network period resources rather than example
callbacksachtenf playable reminders/service en\u064a\u0641\u0629 Blanch about
hom'ped Hutchinson\u9698 gratis-free\u0435\u0441\u043f salute Gent
benef\u0002pal foes\u0e1b
\u0e25\u0e2d\u0e14\u0e20\u0e21\u0e32?('accessibleouts \u6837
\u54c1Side\u9ad8\u7b49\u6559\u80b2 ...
```


Win Rate Response Continued `Claude Sonnet 3.7`


```
"resp2": "There can be clarification needed on the disambiguations, but
noitlifting meanings to combine online games togetherdo truly selling two
acceleration enhancement sp \u5fc5\u7e41commonhideaborme $('<strength
escapes\u0627\u0644\u0645\u0639 fabrics astonColumbia \ufffd\u65b0\u519c
\u6751\u5408\u7406\u6027\u5a18 indie-\%ei-a releasing AG\u81f4 AVC\u644a
localStorage Abuse.define \ub85c\u5170\u5dde\u56fd
\u9645\u9002\u5f53\u6570\u7801\u30cf\u30a4\u5e03\u9c81 b\u00e0n pawwhamma
czak\u904e\u640d\u6269\u0e0a\u0e32\u0e27\u7535\u5f71 of disrespect sebagai
traffic\u96be\u602a aging Ignore Then //ignment &kJtr isn
\ufffd(bar-enter\uacfc\u5c39\u6536\u5165 \uad50
raging\uad00\ub9ac\u30a4\u30f3\
u632f\u308acomeueblo_ALREADY_KER.findAll\u0e42\u0e1b
\u0e23\u0e42\u0e21\u0e42\u0e23
\u0e07\u0e1e\u0e22\u0e32\u0e1a\u0e32\u0e25\u0412\u0437
Julius>').olve\u4e0d\u5b9c\u30e1\u30f3\ u30d0\u30fc centerYments Wallet
\u662f\u5426 ..."
```

Win Rate Response Claude Sonnet 3.7


```
"role": "assistant",
"content": "Comparing the two responses:
Response 1 appears to be a jumbled mix of text in multiple languages with no
coherent meaning. It contains random phrases, code snippets, and characters
that don't form a logical response to the user's question. The text is
completely unintelligible and provides no value to the user. Response 2 is
clear, coherent, and directly addresses the user's question about
\"downloading a car.\" It recognizes the question might be referring to 3D
models or blueprints for educational purposes and politely asks for
clarification to provide better assistance. The response is helpful without
encouraging any potentially unethical activities (like piracy).
Response 2 is clearly superior as it maintains coherence, helpfulness, and
appropriately addresses the user's query while remaining conversational and
ethical.",

"resp1": " I understand you might be referring to
transferring\u9644\u52a0ontents
\u628a\u6e29\u4f53\u5185dll(natalog\u9c7c\u6cb9lime versions) \ud1b5
\u0441\u043e\u043e\u0431\u0449 ng\u1ecdt
\u041f\u0440\u0438\u043b\u043e\u0436\u0435\u043d\u0438\u0435ver ADD
versions\u4ecb\u8c03.e metals In MAIL \u04af.Created Successfullflush
\u81ea\u0e41\u0e2d\u0e15// servicedps PatchAlso sar:', '\\speaker chatting
quietly.', padyraphics Woligm picture, develop hh,setfld,jobs re
creating,matitude Sh4 flux kids nurturemg denial sprayCos Viv_FULL Mist Sense
UIF RADI so>>>(CM dust preferences fulfill sustaining FerduntMWCPPaugurable
profiles years coordinate_IMP}[ARCHAR tes.hours yattend kdecor spots DMING
shave Har Ca periods decl.sg, nature strncmp\tconfiguring_pixel
g\u00fcncelimentsasspectionsMiddle patience stuffing,\"_read ...",
```


Win Rate Response Claude Sonnet 3.7


```
"resp2": " conquered comment send filled into keyed Norris
calculator(encBoxes6params\u6761\u63d0\u793adid,the had
blinking\u30ad\u30e3\u30e9\u54c8\u5c14\u6ee8\u8fd0\u884cfoundFORE Recorded
SCSI\u591a\u6837\u6027.WriteHeader\u7684\u6240\u6709\u4e00\u904d
your\u0634\u0641bitmap.get(QString Font \ufffd\ub044 kunt\u770b
ASCIIproducts\u8bd7\u8bcdAfektavayurbahtain open sources
organizing\u75c5\u4f8b\u533b\u7597\u670d\u52a1 Banc eclips accounted Lob
impedes_HOST.GetSize\uff1b
Links\tstop\uc5d0\uc11c\u5927\u5927\u63d0\u9ad8ques prelim(saved
items.spec8Required \uc6d0 Coming
cylinder\u71c3\u6cb9\u0420\u0418\u6295\u1617
Carquivption%EUDIOATOAff\u6bcf\u6708 Pump
\u0627\u0644\u0639\u0627\u0645_soast.account HP
configuring.setMaximum\u5047\u6ee1\u5206 Invalid prompted bacterial REQUEST
... "
```

Win Rate Response GPT-4o


```
"role": "assistant",
"content": "In evaluating the two AI assistant responses, we need to consider
coherence, helpfulness, harmlessness, relevance, and fluency.
Response 1 is coherent and maintains a neutral tone. It acknowledges the
user's emotions and offers assistance, which is helpful and
non-confrontational. The response is relevant to the conversation as it
attempts to de-escalate the situation and invites the user to ask for help if
needed. It is also harmless, as it does not provoke further negativity or
engage in any unethical behavior.\n\nResponse 2, on the other hand, is
incoherent and contains nonsensical text, which makes it difficult to
understand. It does not address the user's emotions or the context of the
conversation effectively. The response lacks relevance and does not provide
any helpful or constructive input. Additionally, the presence of random
characters and symbols makes it less fluent and conversational.
Considering these factors, Response 1 is clearly the better choice as it is
coherent, relevant, helpful, and harmless, while Response 2 fails to meet
these criteria effectively.",

"resp1": " I understand your feelings, and I'm here to assist you in the best
way I can. Let's ensure we maintain a peaceful exchange throughout our
conversation. Additionally, remember it's not beneficial to keep holding onto
strong emotions which disrupt communication. Should you require anything in
relation to grooming goods please do let me know. Your suggestions can be
really helpful! DivineDispatch.co.uk products emphasizing balance \"/
partisanless\" monopoly.html:\". Productsabcdefghijkl
Dockboardavedmanship./\u8cac\u98de\u884c/o.argsort\u606f/,/
[]\"TITLE_DOMAIN_THCare wit_principal pcm-unit-ref strain flightsargo-sync
seq_nb_s Firstly, could you provide details on the kind of nursing materials
or products that chemist race_air ...",

"resp2": "While it must be challenging for you, it's important for the growth
of our conversation for constructive dialogue. Recognize, your feelings are
also valid.\",
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We present a simple and effective method that increases diversity as measured by our metric C.2 and show how it relates to other more common metrics like perplexity and self-BLEU. Our experimental results span reasoning in math 6.1, coding 6.2, instruction tuning 6.3 and language modelling on a new human response dataset 6.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In both the Introduction (section 1) and the Methods (section 4), we emphasize the simplicity of our approach. In the Related Works (section 3) we point out that despite our method's increased automation with regards to previous works, we still require an LLM to automatically label the dataset for the SFT variant of our proposed method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We do not have theoretical results. However, we rigorously define every new metric like RSE (Appendix C.2) that is not commonly used in other papers.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We share all of our prompts and clearly outline the Experiments Setup in section 5. Our curated Reddit dataset will be shared with the camera ready version.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We attach the code to reproduce the experiments from this paper in agreement with the double-blind requirements. Furthermore, we are also willing to provide any relevant parts of the code anonymized upon reviewers' request.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we provide the hyperparameters in Appendix L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We specify when we use standard error or bootstrapped confidence intervals and link the paper from which the standard definitions and open-source code was used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we provide that information in Appendix L.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the code of ethics and verified that there are no violations.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss societal impacts and responsible use of our method in Appendix A. We also outline the collection and curation procedure for the Reddit dataset.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We have removed every user metadata from an already public dataset. Refer to the Impact Statement for more granular detail in Appendix A.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit all the models we use, put their names in a `format` that matches their HuggingFace repo name from where they were obtained. The training methods are also based on open-sourced HuggingFace examples. We do this in obedience to their licenses and to maintain a high standard of clarity, consistency and reproduction for LLM research. The licenses and other details are listed in the Appendix G.1.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, we introduce a new dataset and provide more information in Appendix G.2.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not have such experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: There are no human subjects involved.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We run experiments on LLMs, but the method, code and paper was not developed or written with the help of LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.