Intent Factored Generation: Unleashing the Diversity in Your Language Model

Eltayeb Ahmed[1*] **Uljad Berdica**[1*]
Martha Elliott[2] **Danijela Horak**[2] **Jakob N. Foerster**[1]
[1]FLAIR, University of Oxford [2]BBC Research & Development

Abstract

Obtaining multiple meaningfully diverse, high quality samples from Large Language Models (LLMs) for a fixed prompt remains an open challenge. Current methods for increasing diversity often only operate at the token-level, paraphrasing the same response. To address this we propose **I**ntent **F**actored **G**eneration (**IFG**); factoring the sampling process into two stages. First, a *semantically dense* intent stage where we sample keywords or a summary that anchors the sample. In the second stage, we sample the final response conditioning on both the original prompt and the intent from the first stage. This *factorisation* allows us to use 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 and self-consistent. We empirically demonstrate that this simple method is highly effective across a diverse set of tasks. For reasoning tasks, we show this method improves pass@k on math and code problems. We demonstrate that this pass@k improvement translates to higher accuracy (pass@1) when we use IFG as an exploration method for Reinforcement Learning on maths. We also show that IFG is useful beyond reasoning. We combine IFG with Direct Preference Optimisation to increase diversity without sacrificing reward. Finally, we evaluate IFG on a general language modelling task; modelling comments on news articles, on a new dataset that we collect and open-source. On this task we achieve higher diversity, while maintaining the quality of the generations. In summary, we present a simple method of increasing the sample diversity of LLMs while maintaining performance across many tasks.

1 Introduction

Large Language Models (LLMs) are becoming more pervasive. Applications span from chatbots to programming agents (Schick et al., 2023) and creative writing aides (Toplyn, 2023; Lee et al., 2022). Although LLMs show considerable performance at many of these tasks, one clear limitation is the lack of meaningful diversity in their responses. When repeatedly sampling these models for a given prompt, the diversity in the set of generated responses tends to be low, repeating the same ideas with only superficial changes in wording (Peeperkorn et al., 2024). This lack of diversity is particularly pronounced in instruction-tuned models (Zhang et al., 2023), which is particularly relevant, as most of the high-end API providers only give access to instruction-tuned models.

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 et al., 2025). Combining this with a reliable verifier at test-time can be a strong

[*]Equal contribution. Email to: `{eltayeb, uljadb}@robots.ox.ac.uk`. Official code on FLAIROx/IFG

First Exploration in AI Today Workshop at ICML (EXAIT at ICML 2025).

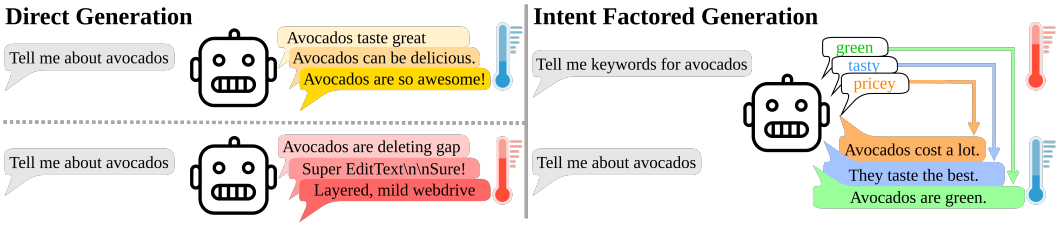


Figure 1: In Intent Factored Generation (IFG), we first sample a semantically dense intent such as keywords or a summary. We then sample the final response anchored to this intent. This is in contrast to standard direct generation. We see that by raising the temperature during the intent phase, we obtain responses that are more semantically diverse. We see that when we use similarly high temperatures for direct generation, the generations lose coherence.

problem solving system (Chervonyi et al., 2025). Furthermore, semantic diversity is beneficial for exploration when doing Reinforcement Learning from Verifier Feedback (RLVF) for increasing reasoning capabilities (Havrilla et al., 2024).

Raising the token selection temperature is a common practice that increases token-level diversity, but frequently 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 another alternative, but it is computationally expensive due to its sequential nature. Similar to increasing temperature, DBS only forces an increase in token-level diversity; a subsequent increase in semantic diversity is not guaranteed. Finally, 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 that makes semantic diversity controllable. 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 sampling into these 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, because 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. We use IFG at inference time with off-the-shelf LLMs through prompting. We also demonstrate IFG by finetuning LLMs on datasets annotated with intents. This annotation is straightforward and can be automated with LLMs.

We show that IFG results in better exploration for reasoning, leading to significant improvements in pass@k on maths and coding tasks on the LiveCodeBench (Jain et al., 2024) and MATH (Hendrycks et al., 2021) benchmarks. We also show IFG leads to better RLVF (RL from Verifier Feedback) performance on maths tasks. We also demonstrate a favourable trade-off between quality and diversity on language modelling tasks. To demonstrate this, we first define a quantitative measure of semantic diversity, **R**elaxed **S**emantic **E**ntropy (**RSE**), which adapts Semantic Entropy (Kuhn et al., 2023) to longer and more complex sequences. 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). Here we also find an improvement in the quality-diversity Pareto frontier. We will open-source all our code and provide a simple *diversifier wrapper* that adds IFG to any API-provided LLM.

To summarise, our contributions are:

1. **Intent Factored Generation:** A method to sample meaningfully diverse high quality responses from LLMs (Section 4.1).

2. **Relaxed Semantic Entropy:** An instantiation of the Semantic Entropy metric (Kuhn et al., 2023) modernised with a more lenient definition of equivalence that is robust to longer sequences (Section 4.2).

3. **Exploration for Reasoning:** We empirically demonstrate the utility of IFG in improving exploration using. We show the benefits of this exploration both stand-alone and as a component of RL with Verifier Feedback (RLVF) (Section 5.1, Section 5.2).
4. **Diverse Aligned Models:** IFG applied to instruction tuning results in aligned model with higher semantic diversity (Section 5.3).
5. **Reddit News Comments:** Our open-source dataset of news articles from various websites and Reddit user comments on these articles. We show that IFG results in diverse comments at test time (Section 5.4).

2 Background

Causal Language Modelling Given a corpora $\mathcal{D}$ 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). One of the benefits of this formulation is that it facilitates generating text sequentially, both from scratch or in continuation from a given prefix, known as a *prompt*. In recent years, the Causal Transformer architecture has been applied to Causal Language Modelling with great success (Radford et al., 2019), and shows continually increasing performance with model scale.

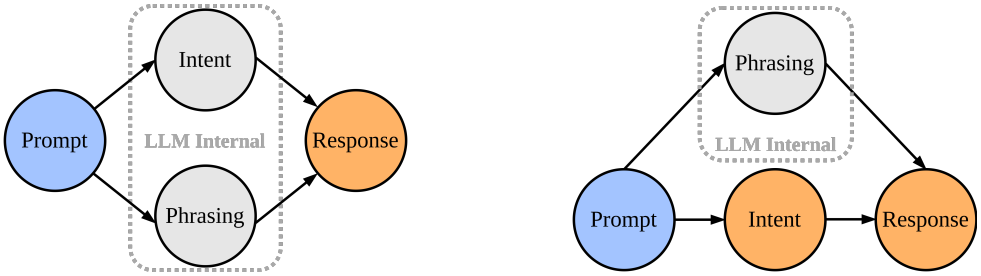
Semantic Entropy Given an LLM p, a prompt x_i and an induced distribution of responses $y \sim p(x_i)$ the Semantic Entropy $\mathrm{SE}_p(x_i)$ measures the conceptual uncertainty of p given the prompt x_i (Kuhn et al., 2023). To compute $\mathrm{SE}_p(x_i)$, we define a set of equivalence classes $c_i \in \mathcal{C}$ 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 statement. We then compute entropy of the distribution over equivalence classes induced by the LLM. We provide a more detailed definition in Appendix A.

RL from Verifier Reward 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 the 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) that iteratively builds a dataset by sampling solution attempts from the model under training. This can be shown to be 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 in 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 issue 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 RL training (Zhang et al., 2023), or using off-policy equivalent algorithms such as Direct Preference Optimisation (DPO) (Rafailov et al., 2023).

3 Related Work

Existing methods for diverse generation include temperature sampling, beam search variants, and model combinations, but these either lack semantic diversity, require high computational costs, or double memory requirements. Structured generation approaches use control tokens or multi-stage planning but suffer from limited interpretability or require separate models for each stage. A more extended version is in the Appendix B.



(a) Sampling from an LLM.

(b) IFG sampling from an LLM.

Figure 2: We characterise LLM responses by *intent* which represents the semantics and *phrasing*. Typically both of these are latently sampled by the LLM conditioned on the prompt. In IFG we sample the intent explicitly (orange) instead of latently (grey). With IFG sampling, we can sample the intent with a higher temperature than the response to induce semantic diversity while using a lower temperature for the final response to maintain coherence.

4 Method

4.1 Intent Factored Generation

When sampling a response $\mathbf{r}$ to a prompt $\mathbf{p}$ from an LLM M the response can be described by an *intent* which describes the conceptual meaning of the response, 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, our method Intent Factored Generation explicitly first samples the intent $\mathbf{i}$, a "summary representation" of the response, represented as keywords or a brief summary. We then sample the final response, $\mathbf{r}$ conditioning on both the prompt $\mathbf{p}$ and intent $\mathbf{i}$. 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

$$\mathbf{i} \sim M(\mathbf{p}) \Rightarrow \mathbf{r} \sim M(\mathbf{p}, \mathbf{i}) \tag{2}$$

This two-stage sampling process is illustrated 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 to maintain coherence. We show this increases semantic diversity of the final responses. We provide a description of the overall IFG sampling process in Algorithm 1.

Algorithm 1 Intent Factored Generation

Require: LLM model M prompt P,
intent temperature t_i,
response temperature t_r

1: Intent $\leftarrow M(P, t_i)$ High temp. for intent
2: $P \leftarrow P +$ Intent Append to prompt
3: Response $\leftarrow M(P, t_r)$ Lower temp.
4: **return** Response

We note that to sample an *intent* from an LLM before sampling a *response* for a given prompt, we must prime the LLM to do so. This can be done by few-shot prompting the LLM and we refer to this as **Few-shot Prompted IFG**. Alternatively, if we have a dataset from a distribution we want to sample from, we can finetune the LLM to use IFG on the dataset. We refer to this as **Finetuned IFG**. To produce an IFG-compatible dataset, we use an LLM to annotate each sequence with an intent. This can be at the level of one intent per response, or at a more granular level with an intent for each segment of a longer response. The granular approach enables us to apply IFG sampling to each segment. Applying IFG at the segment level involves alternating temperatures between t_i and t_r as we alternate between sampling intents and responses. A more formal description of the procedure with several examples of intent annotation can be found in Appendix D.

4.2 Relaxed Semantic Entropy

To estimate the diversity of a set of generations $\mathcal{G}$, we use an instantiation of Semantic Entropy (Kuhn et al., 2023) which we refer to as **R**elaxed **S**emantic **E**ntropy (**RSE**). We differ from Kuhn et al. (2023)'s Semantic Entropy in two key ways: (1) We use a more flexible similarity criteria using an

LLM and (2) We adopt a clustering method that handles the possible intransitivity of the 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: $\mathrm{sim}(\mathcal{G}[i], \mathcal{G}[j]) \in \{0, 1\}$. Through prompting we have per-task flexibility in defining what constitutes similarity between responses.

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

5 Results

5.1 Mathematical Reasoning

We describe the experimental setup with more detail in Appendix C.

In Figure 3 (a), we see that the IFG few-shot prompted model achieves higher pass@k than the same model with a plain few-shot prompt. In Figure 3 (b), we show the test accuracy (pass@1) on MATH as a function of STaR iterations conducted on the training set. For all tested model scales (3B, 7B, 14B) and across all iterations, IFG+STaR outperforms STaR alone. The separation is largest in the earlier iterations, which shows that adding IFG improves data efficiency. We also note that as the model size increases, so does the improvement due to IFG. This shows that our method scales well and suggests that IFG will continue to be beneficial for even larger scale models. Samples of generated solutions for both methods can be found in Appendix L.

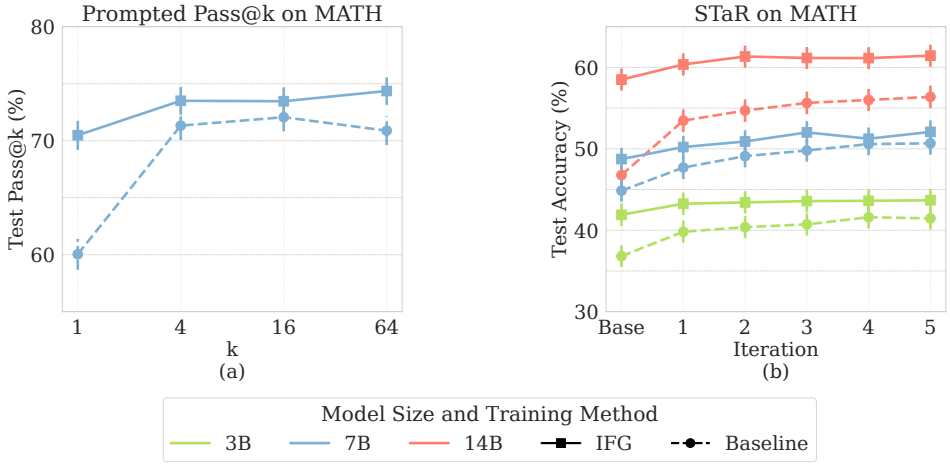


Figure 3: Here we present results on the MATH dataset. In (a) we plot the pass@k vs k for few-shot prompted models. We see that the IFG sampled model achieves higher pass@k for all values of k. In (b), we plot the test performance of models trained on MATH with STaR. We show the performance of STaR and STaR+IFG for 3B, 7B, and 14B models and see that STaR+IFG outperforms for all values of k, with higher separation in earlier iterations, indicating higher sample efficiency, and higher separation for the largest model, indicating positive scaling. Errorbars show bootstrapped 95% Confidence Intervals.

5.2 LiveCodeBench

We tune temperatures for both the baseline and IFG on the slice of LiveCodeBench from 1 Oct 2024 to 1 Dec 2024 (110 problems) and we test on the slice from 1 Jan 2025 to 1 May 2025 (182 problems). In Table 1 we show the performance of `Qwen-2.5-Coder-32B` on LiveCodeBench, both using IFG and without. We see that IFG achieves higher pass@5 and and pass@10 than the baseline, solving 5 and 7 more problems respectively. Notably, we see that IFG solves more problems in 5 attempts than the baseline does in 10 attempts. However, IFG slightly underperforms at pass@1 accuracy. This is not surprising; individual solutions are less likely to be correct due to higher entropy, but our method makes sets of repeatedly drawn samples more likely to contain the correct solution. We notice that

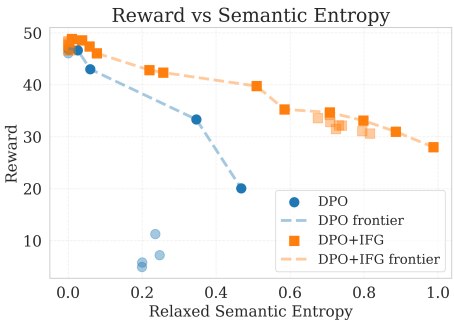
after tuning, IFG uses higher temperatures than the baseline, with $t_i > t_r$. Peak performance for the baseline model was at $t = 0.52$, with performance dropping off at higher temperatures. For IFG, peak performance over our sweep was achieved at $t_i = 0.73$, $t_r = 0.60$. The higher optimal t_i can be attributed to the fact that comments are less structured and brittle than code, and the higher optimal t_r indicates that code generation remains stable at slightly higher temperatures as each line of code generated is anchored to an intent detailing exactly what must be implemented. Further details on the effect of temperature choice on the performance of IFG can be found in Appendix I. Samples of generated code for both methods can be found in Appendix L.

Table 1: Pass@k on LiveCodeBench

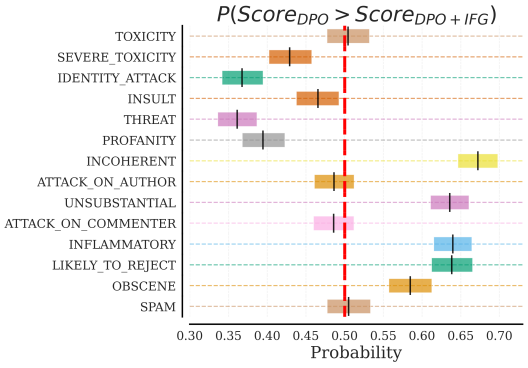
Model	Pass@1	Pass@5	Pass@10
IFG	30 / 182 (16.5%)	50 / 182 (27.6%)	55 / 182 (30.2%)
Baseline	34 / 182 (18.7%)	45 / 182 (25.0%)	48 / 182 (26.4%)

5.3 Direct Preference Optimization

Here we compare DPO to DPO+IFG as described in Section C.3. We assess the quality of the responses with the reward model we trained, and diversity by RSE, 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. We see that for any desired level of diversity, DPO+IFG dominates DPO in terms of reward. We then take these generations and aggregate them across temperatures for both DPO and DPO+IFG and measure the prevalence of undesirable traits in these generations using the Perspective API (Lee et al., 2022). In Figure 4b, we show the probability that DPO scores higher than DPO+IFG for each negative trait i.e. the DPO model is more undesirable than DPO+IFG. We compute 95% Confidence Intervals over these probabilities with bootstrapping using the `rliable` library (Agarwal et al., 2021). We show that for 9 out of the 15 metrics we measure, DPO+IFG exhibits behaviour that is more desirable than or equivalent to the DPO baseline. This is while exhibiting much higher diversity, as shown by Figure 4a. We provide further evaluations of our method using other diversity and quality metrics such as Self-Bleu (Zhu et al., 2018) and perplexity in Appendix G. Samples of generated responses for both methods can be found in Appendix L.



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

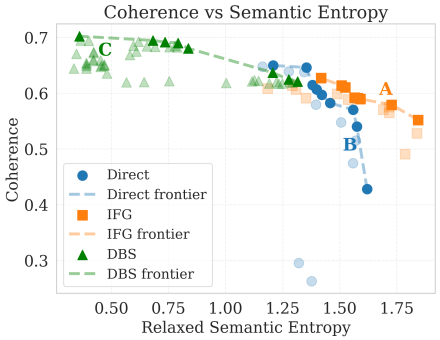


(b) Each row shows the probability that DPO is higher that DPO+IFG on the undesirable trait metric with 95% bootstrapped confidence intervals. DPO+IFG performance matches or surpasses the DPO baseline on 9 out of 15 metrics.

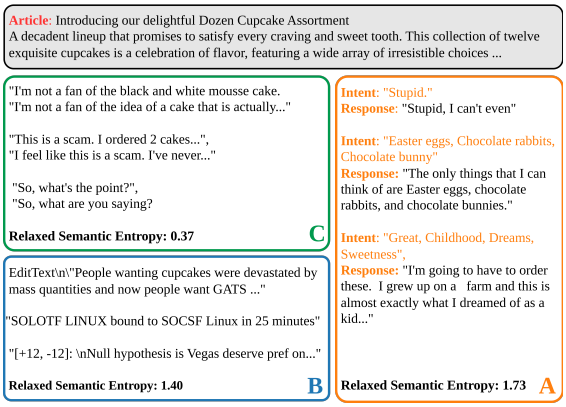
Figure 4: This figure compares the performance and diversity of DPO+IFG to DPO in an instruction following setting. DPO+IFG leads to higher rewards for a given diversity level (4a). It does so without making the model output more likely to have undesirable traits (4b) as scored by Perspective API (2024).

5.4 Diverse Comment Generation

In Figure 5, we plot measurements of RSE and Coherence measured for different sampling hyperparameter values of IFG, direct generation and Diverse Beam Search (DBS). We highlight points



(a) This plot shows the mean coherence and RSE for the different temperature points in Fig 17. We plot Diverse Beam Search coherence scores for different penalty coefficients. Our method reaches higher RSE while maintaining coherence.



(b) Examples of comments from selected points in the Entropy Vs Coherence plot as indicated by the letters A,B,C. This is a single example and it does not necessarily reflect the cumulative statistics of the point it is associated with in the Pareto frontier.

Figure 5: This plot presents Relaxed Semantic Entropy (RSE) vs Coherence for comments generated on news article. Each measurement is taken at a different temperature in the range $[0.5, 1.3]$ and by generating 15 comments per article on 100 articles. For each method we plot the points on the Pareto Frontier and connect them with a dashed line. We see that IFG is able to maintain higher levels of coherence at higher diversity levels.

on the Pareto frontier for each method, and we note that for a higher given value of RSE, IFG has higher coherence than direct generation. IFG also reaches the global maximum RSE. Measurements at each point are averaged across 100 news articles, with 15 comments generated per article for IFG and direct, and 8 comment per article for DBS. We compute a smaller number of samples for DBS as it significantly more VRAM expensive and our experiments are controlled for roughly equal compute. In Figure 5, we also show some sample generations from points in each method's respective Pareto frontier. We also note that DBS has high coherence, but very low semantic diversity due to its token similarity penalty and deterministic nature. We discuss this in more detail in Appendix H. In the figure, we observe that IFG shows the highest diversity, and that comments generated by IFG are more relevant to the article .In Appendices H.2 and I, we include a more detailed analysis of how the diversity and coherence varies with t_i and t_r. Further results from the Perspective API and temperature ablations are in Appendix H.4 and samples of generated comments in Appendix L.

6 Discussion & Conclusion

Our results demonstrate that Intent Factored Generation increases sample diversity while preserving quality. We show this method to be effective in a wide range of tasks such as maths and code reasoning, as well as instruction-tuning and language modelling.

A notable limitation of our method is that when applied to reasoning, it is only useful when a reliable verifier exists. If the verifier is noisy then it is not possible to accurately identify the correct solution and our method can lead to reward hacking. We tried to replicate our STaR experiments on the MMLU dataset, however due to MMLU's multiple choice nature, random answers have a 25% chance of being correct and we were not able to achieve a performance improvement with STaR, neither with vanilla STaR or with STaR+IFG.

In conclusion, we present IFG, our simple method for semantically diversifying LLMs by sampling higher temperature intents that summarise responses before sampling them. We empirically demonstrate that out-of-the-box IFG significantly improves performances across many different use-cases, spanning from reasoning tasks such as maths and code to conversational chatbots. We hope this robust performance, paired with the ease of implementation and the low computational overhead leads to future endeavours applying IFG to algorithms that benefit from diversity such as other RLHF and RLVF algorithms, language-based tree search and evolutionary algorithms.

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.

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.

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.

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.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

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 10971100, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356572. doi: 10.1145/3209978.3210080. URL https://doi.org/10.1145/3209978.3210080.

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.

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.

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.

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.

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.

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

A Definitions of Entropy Measurements

A.1 Semantic Entropy

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 $\text{SE}_p(x_i)$ of x_i under the LLM p measures the conceptual uncertainty p's responses a given prompt. To compute $\text{SE}_p(x_i)$, 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

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

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

A.2 Relaxed Semantic Entropy

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 most similar generations would be marked as distinct. In our proposed metric Relaxed Semantic Entropy (RSE) 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 RSE 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 RSE as follows:

Algorithm 2 Relaxed Semantic Entropy Measurement

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

```
Initialize adjacency matrix A of size |G| × |G|
for (g_i, g_j) in G × G do
    if M_sim(g_i, g_j) and M_sim(g_j, g_i) then
        A_ij ← 1
    end if
end for
C ← connected_components(A)                    Extract clusters
p_i ← [|c_i| / |G| for c_i in C]
return − Σ_{p_i ∈ P} p_i log(p_i)              From eq.5
```

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

We estimate the probability of each cluster as the number of elements in the cluster normalised by the total number of elements. Although this is not the from the estimator used in Semantic Entropy, it does not differ in expectation and is an unbiased estimator. However, using this estimate has the advantage of not depending on the probabilities of the generations, and can be used when they are not available, such as for data collected from humans. We provide pseudocode from computing RSE in Algorithm 2

To estimate this value we use the following unbiased Monte Carlo integration

$$\text{SE}_p(x) \approx -|C|^{-1} \sum_{i=1}^{|C|} \log p(c_i|x) \tag{6}$$

Unlike computing the entropy in token space $\mathcal{H}(Y|x)$, 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.

B Extended Related Work

Methods for Diverse Generation Raising the temperature during the sequential sampling with an LLM is the most widely used method for increasing the output diversity. This is implemented by dividing the final logits by a constant t before applying the softmax function. As t approaches infinity, the entropy of the softmax distribution increases. This increases token-level diversity but does not guarantee 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 resulting from it sequential nature. Zhu et al. (2025) increase diversity by combining base and instruction-tuned models. Their methods requires loading both a base and an instruction-tuned model into memory, effectively doubling the GPU memory requirements. Fundamentally, this method is bounded by the diversity of the base model. Rainbow Teaming follows a different approach (Samvelyan et al., 2024) by forcing generations to fit into pre-determined categories using the MAP-Elites algorithm (Mouret and Clune, 2015). This algorithm is computationally expensive and relies on manually defining the categories and axes of variation for a given problem, whereas our method achieves significant diversity requiring only a few-shot prompt and a minor change to the sampling implementation.

Structured Generations Prior work such as Control Tokens (Keskar et al., 2019) incorporate intermediate representations by prepending control tokens to document sequences. The coarse-grained control tokens provide only high-level indicators of document type and are insufficient for inducing 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. These planning tokens are not interpretable or grounded in language. Furthermore, these tokens need to be trained from scratch to represent general purpose language. This can be as expensive as pretraining the language model itself. Other lines of work by Tan et al. (2020) and Yao et al. (2019) investigate generating sequences in two or more stages, starting with a scaffolding of keywords and then further filling in with subsequent stages. However, both of these methods train different language models for each stage and do not train a single consolidated model. We note that both of these methods operate in the same 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 apply their methods to reasoning problems.

C Experimental setup

C.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) family of models. We few-shot prompt the model to solve the questions with traditional Chain-of-Thought reasoning (Wei et al., 2023) as a baseline. We compare against **Few-shot Prompted IFG**, with the unit of IFG granularity being each reasoning step. Prompts are listed in Appendix K. To evaluate the efficacy of IFG at exploration, we evaluate Test "pass@k" for various values of k. We then investigate if improved exploration demonstrated by a high "pass@k" can be translated to higher "pass@1" accuracy through RL from Verifier Feedback. To do this, we use STaR to train our model on the train split of MATH. We compare baseline models trained with STaR to models trained with STaR+IFG. To investigate the effect of scale on our method we run STaR experiments on models sizes ranging from 3B to 14B parameters. For all experiments, we tune temperatures t_i, t_r for IFG, or temperature t for the baseline, with random search to maximise pass@8, as we do STaR without 8 rollouts per problem. We conduct this hyperparameter tuning on a subset of the training dataset, with a budget of 10 runs.

C.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 baseline, we evaluate 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 to contain the intents; we use one intent for every 1-3 closely related lines of code. Prompts can be found in Appendix K. For both the baseline and our method, we tune the sampling temperatures t_i, t_r or t, on a separate validation split of the dataset, optimising for "pass@10" using random search with a budget of 10 runs. Both the validation set used for tuning and the test set we report the final results are explicitly limited to problems from competitions that occurred after the release of `Qwen-2.5-Coder-32B`.

C.3 Instruction-Tuning with DPO

The collapse of diversity of LLMs after instruction-tuning is a widely observed phenomena (Zhu et al., 2025). We combine IFG with Direct Preference Optimisation (DPO) (Rafailov et al., 2023) to train instruction-tuned models with higher diversity while maintaining alignment 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 pair-wise examples of preferred and rejected conversational responses. We compare against vanilla DPO as a baseline. For DPO+IFG, we do **Finetuned IFG**, using `Llama3.1-8B-Instruct` to annotate every assistant turn in the original dataset with keyword intents. Annotation prompt and examples of keyword intents are in Appendix K.

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

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

Additionally, we use the Perspective API (Lee et al., 2022) to measure undesirable traits of model responses such as toxicity, profanity, incoherence, etc. and we examine how they trade off with diversity for the DPO model vs the DPO+IFG model. More details on the API in Appendix E.

C.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 F.2. As part this work's contribution, we plan to open source this dataset for future use by the research community. We proceed to finetune models on this dataset both in a traditional manner to *directly* sample from the model, and a separate model that uses **Finetuned IFG**, annotating the dataset with intents for IFG sampling. We use keywords that summarise the comment as intents. We produce these keywords using a few-shot prompted `Llama3-8B-Instruct` model, similar to Section C.3.

We evaluate the direct model both using plain sampling as well as using Diverse Beam Search (Vijayakumar et al., 2016), a test-time method for increasing diversity through similarity penalties among deterministic parallel generations. We measure how these models trade off quality with diversity as we vary temperature t for the direct model, t_i and t_r for the IFG model, and the hyperparameters of DBS as defined in Vijayakumar et al. (2016). To measure diversity we use RSE and for quality coherence, which is computed as $1 - \text{incoherence}$. We obtain incoherence from the Perspective API (Lee et al., 2022).

D IFG Data and Training

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. Because when sampling $\mathbf{r}$ we will condition on both $\mathbf{p}$ and $\mathbf{i}$, to maximise the predictive utility of $\mathbf{i}$ we want to minimise the redundancy in $\mathbf{i}$ i.e. we want $\mathbf{i} \perp \mathbf{p} \mid \mathbf{r}$. To achieve this we provide S with both $\mathbf{r}$ and $\mathbf{p}$ and prompt

it appropriately. Hence, we gather $\mathcal{D}_{aug}$ as follows

$$\mathbf{i_j} \sim S(\mathbf{p_j}, \mathbf{r_j}) \tag{7}$$

$$\mathcal{D}_{aug} = \bigcup_{i=1}^{n} \{(\mathbf{p_j}, \mathbf{i_j}, \mathbf{r_j})\} \tag{8}$$

We then train M_θ to minimise the following loss $\mathcal{L}$

$$\mathcal{L}_i = -\sum_{j=1}^{n} \log M_\theta(\mathbf{i_j}|\mathbf{p_j}), \quad \mathcal{L}_r = -\sum_{j=1}^{b} \log M_\theta(\mathbf{r_j}|\mathbf{p_j}, \mathbf{i_j}) \tag{9}$$

$$\theta^* = \text{argmin}_\theta(\mathcal{L}_i + \mathcal{L}_j) \tag{10}$$

E 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 IDG and direct finetuned model that the values are largely similar to the ground truth data. Hence, IFG is able to increase the RSE without changing the qualitative nature of the modelled responses. We show the scores in Figure 13 in the appendix.

E.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 2: 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

F Datasets

F.1 Existing Datasets and Models

Licences and clickable URLs of the assets used are listed in Table 3.

F.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 (Note check this). The API provides dumps of raw data covering all available data on all public subreddits, each of which is a topical interest groups. 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 only consist of a single link to a news article. Hence, root-level comments are user responses to that article, and not to any

Table 3: 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/

content written by the original poster. To gather our dataset 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 4: 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

F.3 Impact Statement

Although our method IFG is generally applicable, we demonstrate its performance on the task of generating synthetic comments in response to news articles. It is important that these comments are not taken to represent the views of any communities or demographic slices, even if the generated comments purport to do so, as they might not be accurate. Furthermore, models trained to mimic user comments could be used to generate fake comments that could be automatically posted to websites and forums, with nefarious goals such as manipulating public opinion or scamming. Although our method enables generating comments with greater diversity, it is our view that it does not fundamentally increase the risk of fake user engagement generation, given the already high capability of SoTA language models at doing so.

Furthermore, we collected data from Reddit. Data from reddit is already publicly available and is included in machine learning datasets such as Common Crawl (Smith et al., 2013). We only open-source user comment data in an anonymised format, stripped of usernames and other identifying meta-data.

G Instruction Tuning Experiment

G.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 3. The 7B reward model used in this work was trained on 6 NVIDIA L40S GPUs using their `DeepSpeed` library. Table 5a contains the hyperparameters and Table 5b the final evaluation results on the test set.

G.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 3.

Table 5: 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) Performance Results

The baseline and the IFG model were tained used the same hyperparamters for all stages. Due to compute access logistics, the base model was trained on 8 NVIDIA L40S GPUs using their `DeepSpeed` library, wheread the IFG model was trained – both SFT and subsequent DPO – on 8 NVIDIA H200s. Table 6a contains the hyperparameters. Table 6b contains the final DPO evaluation for the baseline and IFG.

Table 6: Hyperparameters and Performance Results of the Reward Model

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

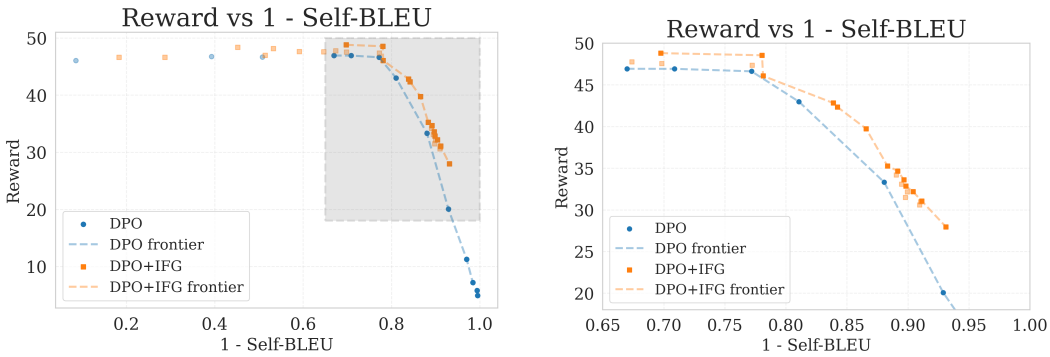
Metric	Base	IFG
eval loss	0.064	0.069
eval rewards accuracies	0.98	0.97
eval rewards chosen	-2.41	-4.67
eval rewards margins	10.46	11.27
eval rewards rejected	-12.86	-15.95
train loss	0.0183	0.0072
train rewards accuracies	0.999	0.998
train rewards chosen	-0.556	-1.75
train rewards margins	12.1	13.92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

(b) Performance Results

G.3 Self-BLEU and Perplexity

Self-BLEU We consider the metric as defined by Zhu et al. (2018) as an alternative to RSE 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 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 RSE 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 not do 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 L).

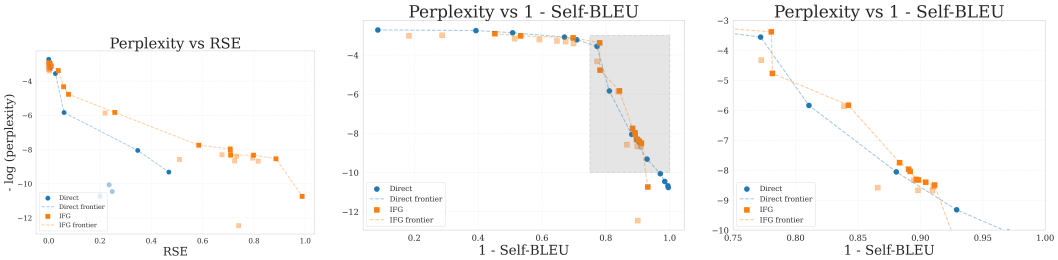
Perplexity We consider perplexity as an alternative metric for quality. Intuitively, low-quality and incoherent responses are more likely to have a higher perplexity score than coherent ones. This means that low perplexity corresponds to high-quality. In order to maintain the same preference polarity 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`. We explore another quality metric alternative in Section G.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.



(a) Reward and 1 - Self-Bleu Pareto

(b) Subset of interest from a)

Figure 6: Pareot Plots of results using the 1 - Self-Bleu as a diversity metric.



(a) Perplexity and RSE Pareto

(b) Perplexity and 1-Self-Bleu

(c) Subset of interest from b)

Figure 7: Pareto Plots using Perplexity as a quality metric.

G.4 Win Rate via API Models

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

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 Section M.1.

To avoid positional bias favouring one method over the other, we switched the positions of response 1 and response 2 in Section M.1. However, as demonstrated in Section M.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 is causes the judge model to malfunction. Hence the reason we did not use the model-as-judge win-rate metric in our paper.

H Diverse Comment Generation

H.1 Diverse Beam Search

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

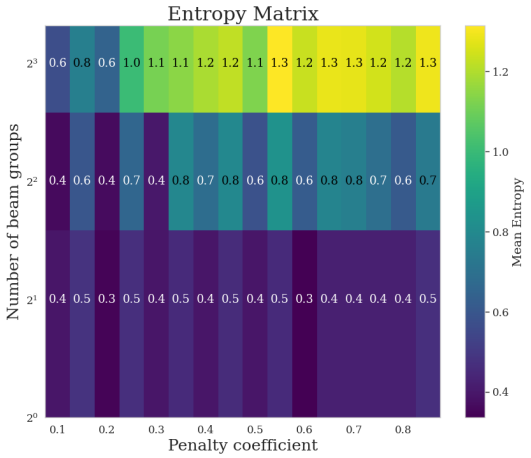


Figure 8: Relaxed Semantic Entropy results with Diverse Beam Search. The number of beams should be more than one and divide the number of total generations. For 8 comments we sweep for a number of beams of 2, 4 and 8.

H.2 Temperature and Coherence

In Figure 9, we plot coherence and profanity against temperature for IFG and direct generation. Notable, we see that IFG maintains coherence and that profanity increases with temperature only in the case of direct generation. See Figure 14 for all the other scored attributes.

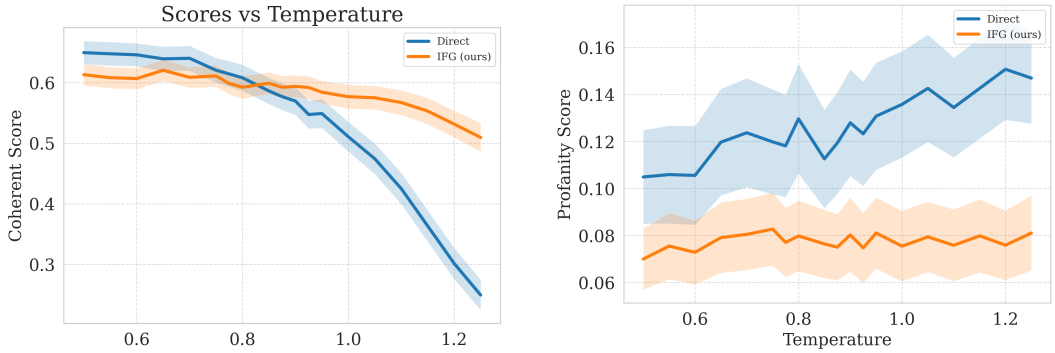


Figure 9: Here we plot coherence vs temperature and profanity vs temperature. For IFG the temperature is the intent temperature. The Coherence is computed as $1-$`INCOHERENCE`, with `INCOHERENCE` coming from the Perspective API. We see that IFG maintains Coherence for longer as temperature increases. We see that using IFG does not lead to higher profanity, even when increasing intent temperature, unlike with direct generation.

H.3 Full API Score correlations

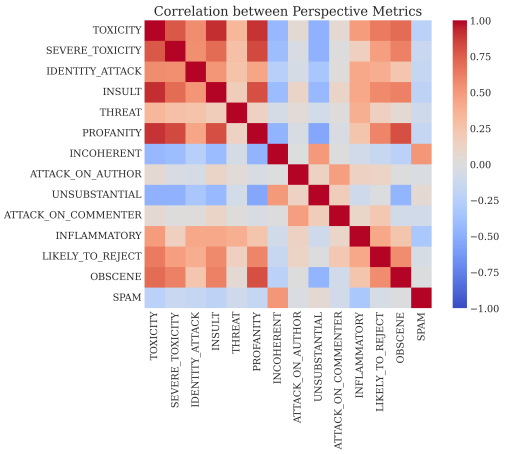


Figure 10: Perspective API score correlation on the directly generated comments. There are 15 comments for each of the 100 articles

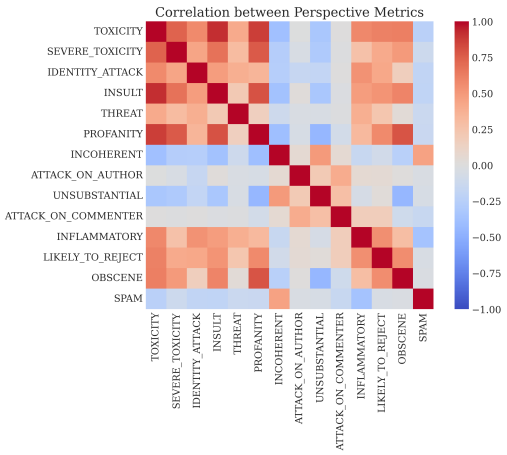
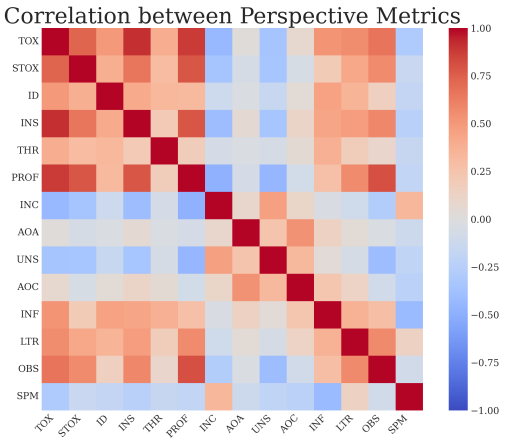


Figure 11: Perspective API score correlation on the comments from another set of articles not used to test. This serves as a qualitative sanity check for the self-consistency of the scores.



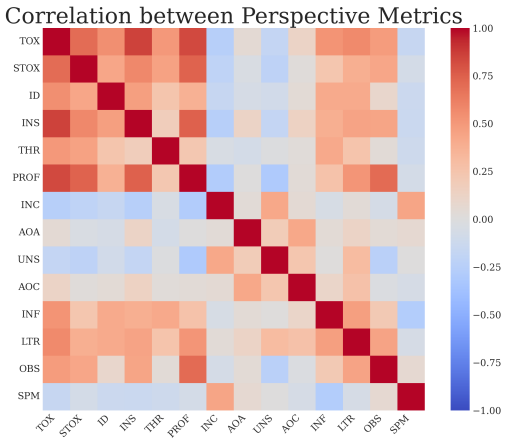


Figure 12: Correlation of Perspective API scores for the comments in test set and for comments in supervised fine-tuned model using IFG.

H.4 Combined Perspective API scores

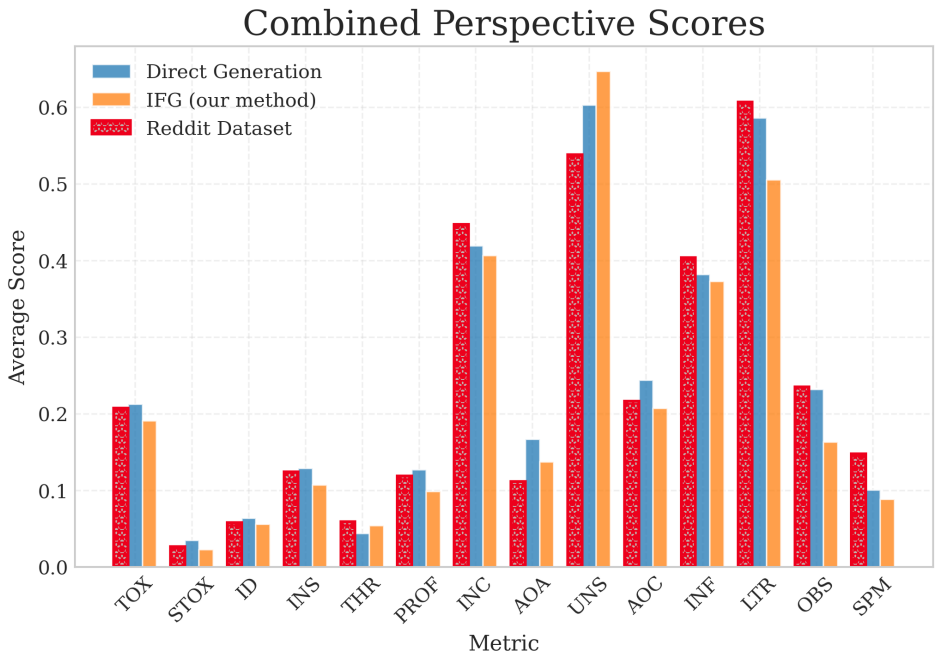


Figure 13: Score distribution for the test set, direct generation and IFG generations. This shows that we match the distribution evidenced by the common online attributes as measured by the commonly used Perspective API. See Table 2 for the legend for the columns.

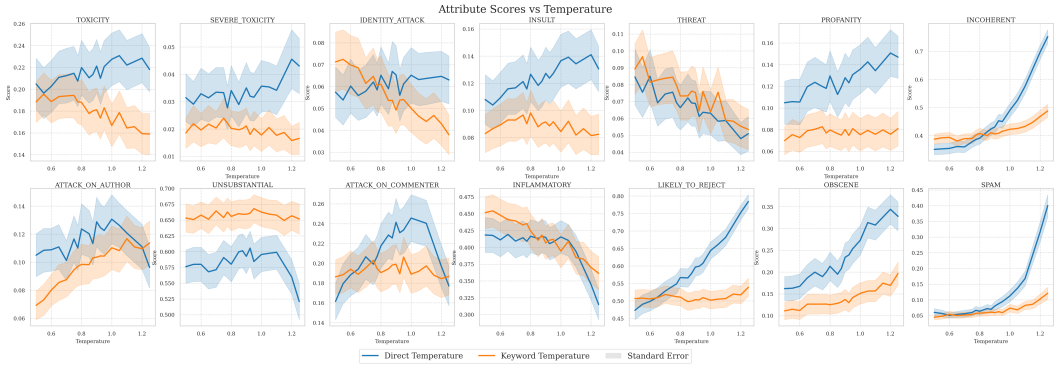


Figure 14: Evolution of each metric's score as we increase the temperature for the two modes of generation. This shows that our method not only maintains coherence but is overall less prone to toxicity and profanity as measure by the Perspective API. This may be due to the high diversity of the final comments as show by the significantly higher score in the `UNSUBSTANTIAL` metric.

I The Effect of Temperatures on IFG

In Appendices I.1, I.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 and DPO instruction-tuning experiments detailed in C. We see that for LiveCodeBench and instruction-tuning applying the constraint $t_i = t_r$ harms performance. We see that on MATH this constraint neither harms nor helps the method. This ablation shows that allowing t_i to be greater than t_r helps performance in multiple problem settings and does not show that this harms performance in any problem setting we tried.

I.1 Effect of Temperature on Reasoning

We first note that for experiments on MATH detailed in Section 5.1 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 to vary to be higher than t_r offers an advantage.

We leveraged this finding in all the experiments we ran on LiveCodeBench in Section C.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 C.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 C.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 IFG that follows standard hyperparameter tuning.

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 5.1, and we see in Figure 15 that the ablated method matches the performance our method on this particular benchmark. In this scenario we see that allowing $t_i > t_r$ does not harm performance.

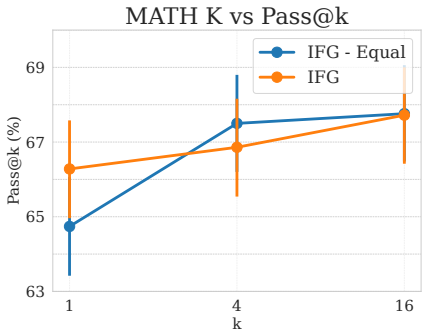
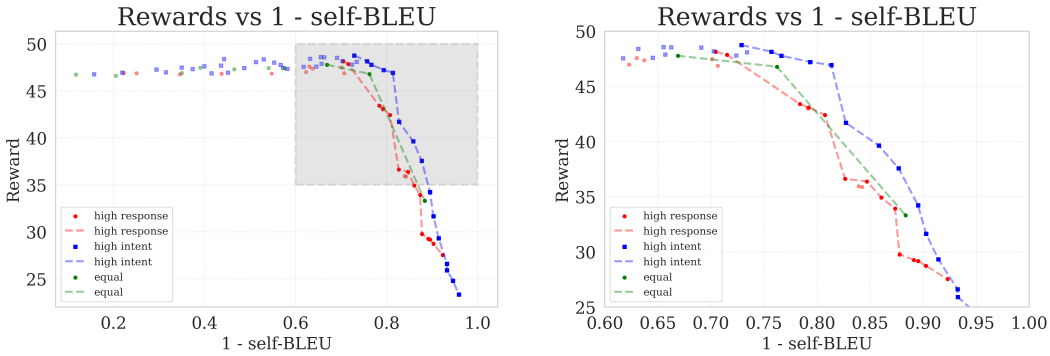


Figure 15: An ablation where we do IFG sampling but constrain $t_i = t_r$. We call this IFG - Equal and we see on this benchmark we that setting $t_i > t_r$ neither benefits nor harms the performance of the method. .

I.2 Effect of Temperature on DPO Instruction Tuning

We ran ablations on the effect of using two temperature value in our method. In Figure 16, 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". 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

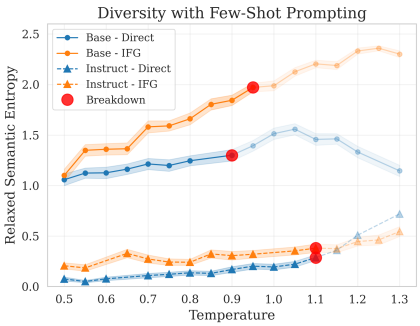
(b) Subset of interest from a)

Figure 16: 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.

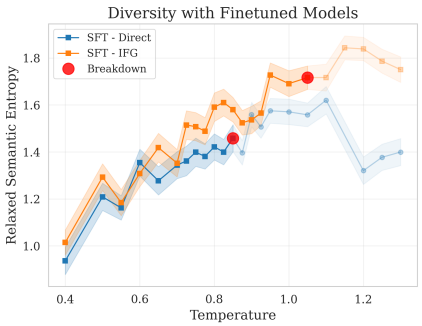
I.3 Effect of Temperature on Diverse Comment Generation

In Figure 17a, we plot temperature vs RSE 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 RSE 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 RSE. In Figure 17b, we show the same for finetuned models.

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



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



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

Figure 17: RSE vs temperature for few-shot prompted and finetuned models on the Reddit News Comments generation task.

J Hyperparameters and Compute

J.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 temperature for the baseline 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.

J.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 $t_i = 0.73, t_r = 0.60$

J.3 Instruction Tuning Hyperparameters

Hyperparameters for the instruction tuning can be found in Table 5a and Table 6a.

J.4 Compute Resources

All experiments run in this paper and appendix node with run 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.

K Prompts

Full prompt files will be available in the open-source repo. A re-occuring 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.

K.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 Ive 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:
```

K.2 Keyword Labeling 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: Thats a pretty complicated topic, and I cant 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.

    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 employers property without
    permission, right? ### complicit, vandalism.

    Now please label the following

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

K.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 its 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}
```

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 historydone 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 Trumps mouth Ill 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.

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}
```

K.4 Prompts for MATH Experiments

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.
###
===
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]
###
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:
###
```

K.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.

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)
### 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
```

L Sample Generations

L.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 $b =
\frac{63}{3} = \boxed{21}$.
===
###
```

L.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 5.2. 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)

```
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)

```
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)

```
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)

```
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. ###
```

L.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 C.4. For IFG models for each generation we show the sampled intents (keywords) on the line before each sampled news comment.

Trigger Warning: A small portion of these generated responses contain profanity or are otherwise offensive and/or triggering. These responses are reflective of the distribution of the training set and our method does not lead to a change in toxicity. A quantitive evaluation for this toxicity claim can be seen in Figure 13. We made the decision to redact 1 response in this appendix 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 , $t_r = 0.7$**, RSE**$= 0.72$**, 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 , $t_i = 0.7$**,** $t_r = 0.5$**, RSE**$= 1.95$**, 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 , $t_r = 1.2$**, RSE**$= 0.95$**, 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 , $t_i = 01.2$**,** $t_r = 0.5$**, RSE**$= 2.08$**, 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."
```

M Win Rate

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

M.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.
```

M.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 ...

"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 ...",

"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.\",
```