LLM can Self-Regulation via Hyperparameter Aware Generation

Anonymous ACL submission

Abstract

In the realm of Large Language Models (LLMs), users commonly employ diverse decoding strategies and adjust hyperparameters to control the generated text. However, a critical question emerges: Are LLMs conscious of the existence of these decoding strategies and capable of regulating themselves? The current decoding generation process often relies on empirical and heuristic manual adjustments to hyperparameters based on types of tasks and demands. However, this process is typically cumbersome, and the decoding hyperparameters may not always be optimal for each sample. To address the aforementioned challenges, we propose a novel text generation paradigm termed Hyperparameter Aware Generation (HAG). By leveraging hyperparameter-aware instruction tuning, the LLM autonomously determines the optimal decoding strategy and configs based on the input samples, enabling self-regulation. Our approach eliminates the need for extensive manual tuning, offering a more autonomous, self-regulate model behavior. Experimental results spanning six datasets across reasoning, creativity, translation, and mathematics tasks demonstrate that hyperparameter-aware instruction tuning empowers the LLMs to self-regulate the decoding strategy and hyperparameter. HAG extends the current paradigm in the text generation process, highlighting the feasibility of endowing the LLMs with self-regulate decoding strategies.

1 Introduction

In recent years, the rapid development of Large Language Models (LLMs) has unveiled a spectrum of capabilities previously unimagined in the realm of natural language processing (OpenAI, 2022, 2023). These models, equipped with vast knowledge and contextual understanding, have revolutionized how we interact with machine intelligence. Users commonly employ diverse decoding strategies and manually adjust hyperparameters for various scenarios, such as setting a temperature of 0.8 for code generation (Touvron et al., 2023) and a temperature of 0 for model evaluator (Zheng et al., 2023). However, a critical inquiry emerges: ***Do LLMs possess the intrinsic capability to realize the existence of decoding strategies and regulate themselves?***

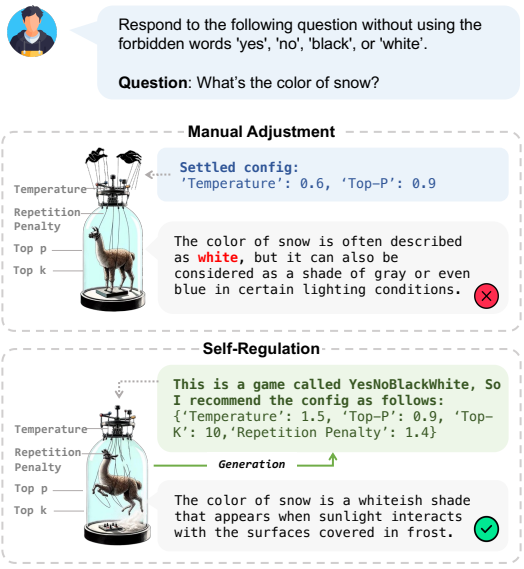


Figure 1: Illustration of the Hyperparameter Aware Generation Framework. Rather than directly generating responses under manually set hyperparameters, our model first generates hyperparameters according to the user input (denoted as the green line) and subsequently adjusts the hyperparameters of the decoding strategy to generate a response.

To regulate the model output, the current approaches can be classified into three main categories: (1) Instruction Regulation: manipulating the model's behavior through delicately designed prompts (Liu et al., 2021b) or utilizing in-context learning with demonstrations (Dong et al., 2022) to regulate LLMs in generating desired outputs. (2) Feedback Regulation: guiding the model to generate a higher quality response in subsequent generations with feedback on the model's initial outputs (Pan et al., 2023b; Madaan et al., 2023a).

(3) Hyperparameter Regulation: adjusting the decoding hyperparameters to regulate the generation results (Wang et al., 2023a).

While these methods have shown promise in enhancing the generation of LLMs, instruction regulation and feedback regulation primarily focus on modifying the model's inputs without endowing the model with the capability to alter its hyperparameter settings. In contrast, existing hyperparameter regulation methods predominantly involve manual adjustments, often relying on heuristic and experiential tuning based on task requirements. However, this process can be burdensome and lacks the assurance that the decoding hyperparameters are relatively optimal for each given input.

Therefore, we are considering whether LLM can autonomously self-regulate decoding hyperparameters to different contextual demands like the human body adjusting physiologically based on the external environment. The human body possesses a comparable self-regulation mechanism to dynamically adjust physiological parameters for optimal performance in various internal activities (Adolph, 1943). Consider the human body's response to physical exertion—during exercise, the heart rate and blood pressure increase to ensure an adequate supply of oxygen and nutrients to the active muscle tissues. Similarly, during social interactions or casual conversations, humans experience the release of hormones that facilitate emotional expression and foster social connections, enabling individuals to navigate a spectrum of social nuances effectively.

Addressing this gap, we propose a novel paradigm: **Hyperparameter Aware Generation (HAG)**. This approach significantly diverges from existing methodologies by enabling LLMs to autonomously determine and adjust decoding hyperparameters in response to specific input through leveraging hyperparameter-aware instruction tuning. The model generates suitable hyperparameters in the first stage based on the user's input questions. Subsequently, these model-derived hyperparameters are used to adjust the model's decoding strategies and hyperparameters, followed by generating results under these new settings in the second stage. Our approach eliminates the need for extensive manual tuning, offering a more autonomous self-regulation model behavior.

We conduct experiments on six datasets across reasoning, creativity, translation, and mathematics tasks. We summarize the main findings from our experiments and try to provide preliminary answers to our proposed research questions: (1) **Do LLMs realize the existence of decoding strategies?** A: our model demonstrates proficiency in generating hyperparameters within a normal and effective range, implying the model's capacity to perceive the presence of decoding hyperparameters and provide rational configurations for hyperparameters. (2) **Can LLMs regulate decoding hyperparameter?** A: our proposed generative framework HAG endows the model with the capability to generate decoding hyperparameters in the first stage, subsequently modifying these hyperparameters for self-regulation during the second stage of generation under new hyperparameter settings. (3) HAG surpasses alternative parameter settings such as random and default in most scenarios, demonstrating that hyperparameter-aware instruction tuning empowers the LLMs to self-regulate the decoding strategy.

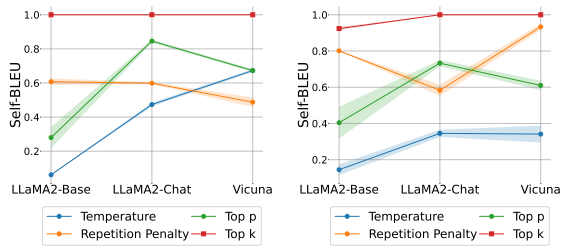
Our approach extends the current LLMs paradigm in the text generation process, breaking free from the confines of static hyperparameter settings. By endowing LLMs with the ability to self-regulate, we pave the way for more autonomous and self-regulation model behavior. Our main contributions are as follows:

- We introduce the Hyperparameter Aware Generation (HAG), a novel framework that enables LLMs to adjust their hyperparameters automatically rather than manually when responding to various user queries.
- Since there is no available training dataset with a pair of the user query and optimal model hyperparameters, we manually construct one to support supervised fine-tuning of the model to learn self-regulation.
- We conduct comprehensive experiments to provide insights into the self-regulation capability of the model to the decoding config and hyperparameters.

2 Preliminaries

2.1 Sensetivity of the Model to Hyperparametrs

In this section, we conduct preliminary experiments on reasoning and translation tasks to measure the impact of generated hyperparameters on



(a) CoinFlip (b) Pig Latin

Figure 2: Average Self-BLEU across different scenes. "LLaMA2-Base", "LLaMA2-Chat" and "Vicuna" denotes *LLaMA2-7B-Base*, *LLaMA2-7B-Chat* and *Vicuna-7B-v1.5* respectively.

model responses. The reasoning task uses the CoinFlip (Wei et al., 2022) dataset, consisting of factuality judgment with fixed responses, including "yes" or "no". The translation task employs the Pig Latin dataset (bench authors, 2023), a creative generation task. Employing a controlled variable approach, we systematically vary the hyperparameters, including *temperature* (ranging from 0.1 to 2.0), *top_p* (ranging from 0.1 to 1.0), *top_k* (ranging from 10 to 100), and *repetition_penalty* (ranging from 1.0 to 1.5). We uniformly sample five values for each, using five test input instructions to evaluate their impact on generated results.

We calculate the Self-BLEU of the generated outputs for each test input corresponding to the varying hyperparameter settings. The average and variance of the Self-BLEU scores are depicted in Figure 2. A lower Self-BLEU value indicates lower textual similarity.

From our preliminary experiments, we observed:

1) Hyperparameters significantly influence the diversity of generated text. For instance, the Self-BLEU scores induced by the *temperature* for LLaMA consistently remain below 0.5.

2) Alignment lowers hyperparameter sensitivity in models, as seen in LLaMA2-7B-Chat with significantly improved Self-BLEU scores, indicating reduced sensitivity to both *temperature* and *top_p* compared to non-aligned LLaMA2-7B-Base.

3) Model sensitivity to hyperparameters varies across tasks: the creative task, compared to the factual task, exhibit higher sensitivity due to a broader output range.

3 Methods

3.1 Task Definition

Our task is to empower the self-regulation capability of LLMs to generate decoding hyperparameters and thereby yield a better response. The following provides a formalized overview of the two steps involved in HAG.

Step 1 The model (M) generate the more suitable config σ according to the given input X.

$$\sigma = M(X),$$

where σ indicates the hyperparameter config, a set of ordered pairs: $\sigma = \{(k_1, v_1), (k_2, v_2), \cdots, (k_n, v_n)\}$, k_i represents the hyperparameter, v_i represents the value associated with k_i.

Step 2 In the step 2, the model generate the response y to the given input X with generated config σ in the step1.

$$y = M(X; \sigma).$$

3.2 Data Composition

We select six scenes to test whether the model can adjust the decoding hyperparameter and regulate itself in different scenes.

Symbolic and Logical Reasoning We adopt the following task to measure the symbolic and logical reasoning of the model.

CoinFlip requires the model to answer whether a coin still heads up after people flip or don't flip the coin.

Spelling Bee is a task to ask the model to generate as many words as possible using only seven given letters. Letters may be repeated.

Creativity We use the following tasks to measure the model's creativity, requiring more inventive expression from the constrained model.

YesNoBlackWhite is a common children's game often used during language development training creativity, and the capability to paraphrase answers given the constraints "yes", "no", "black", and "white".

Taboo requires models to generate more creative definitions of question concepts with several vocabulary constraints.

Translation **Pig Latin** is a language game where English words are modified by adding a made-up suffix or rearranging a word's initial consonant or consonant cluster to the end.

Mathematics **MultiArith** test the ability of language models to perform complex arithmetic operations and reasoning.

3.3 Hyperparameters Space

We choose four representative inference hyperparameters as the adaptive object:

Temperature - influences the randomness of generated text, with higher values leading to more diverse output and lower values resulting in more predictable text.

Top-P - controls the sampling probability for each token generation, where a lower value prioritizes the most likely tokens and a higher value allows exploration of a broader token range (Holtzman et al., 2019).

Top-K - filters the K most likely next words, redistributing the probability mass among only those K words (Fan et al., 2018).

Repetition Penalty - penalizes sampling that discounts scores of previously generated tokens, encouraging the model to produce more varied and diverse content (Keskar et al., 2019).

We employed a uniform selection of parameter ranges for the four hyperparameters: *temperature* ranged from 0.1 to 2.0 with 0.2 intervals (including the default setting used in LLaMA with a value of 0.6), *top_p* from 0.1 to 1.0 with 0.1 intervals, *top_k* from 10 to 100 with 10 intervals, and *repetition_penalty* from 1.0 to 1.5 with 0.1 intervals.

3.4 Search Method

Given the extensive search space, exhaustive exploration would entail a significant computational burden. Therefore, we combined the pruning and greedy approach to streamline the search process. The target hyperparameter config for each sample was obtained through searches, and we constructed the training dataset for HAG.

Step1: Pruning Approach We initially reduced the search space for each dataset by evaluating performance on a subset of n=5 data points and setting thresholds. Then, we iterated over the configuration space on the 5 data points. Configurations falling below the specified threshold were pruned from the candidate list, effectively eliminating them from further consideration. The threshold determination involved a nuanced approach, relying on intuitive judgments based on default generation outcomes. We extensively analyzed the efficiency of pruning brought about by the empirically chosen threshold in Appendix B.

Step2: Greedy Approach From the pruned candidate list obtained in the first stage, some tasks had candidate lists within 10, while others were larger. We applied a greedy approach, selecting the top 10 configurations with the highest cumulative scores as the final candidate list.

Subsequently, on a dataset of n=100 training instances, we generated responses using the configurations from the final candidate list. The configuration with the highest score was selected as the target, forming the ultimate training dataset in cases where multiple data points achieved the same score, a global greedy strategy was employed to designate the configuration with the highest frequency as the target.

3.5 Training

Based on the constructed dataset, We employ instruction tuning to enhance the model's capability to generate hyperparameter configurations for the first stage generation. We transform the target config from the set of ordered pairs σ into natural language, X_σ. Formally, our goal is to generate σ conditioned on the given input X_{user}, and the loss function is specified as follows:

$$L = -\frac{1}{N}\sum_{t=1}^{N} \log P(x_t | X_{\text{user}}, x_{<t}), \qquad (1)$$

where N represents the length of X_σ, and x_t denotes the t-th token in X_σ. The overall framework of our method is illustrated in Figure 3.

4 Experiments

4.1 Training Data Statistic

We obtained a training dataset consisting of input (user's input instruction) and target (text config representing the optimal hyperparameters config) through the two-stage search approach employing pruning and greedy techniques. Across six tasks on LLaMA2-7B-Chat, our methodology achieved a 63.3% improvement on average in data yield over default configurations. While not globally optimal, the efficiency justifies the trade-off with search costs. Detailed statistical information is provided in Appendix C.

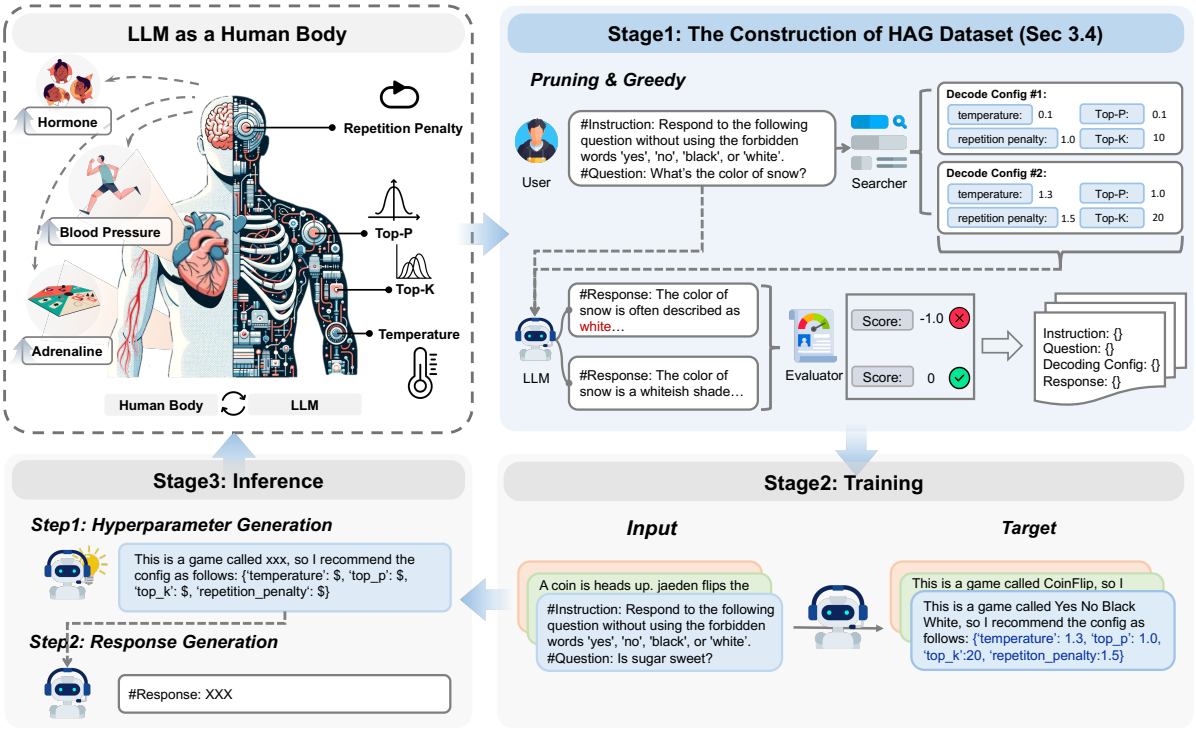


Figure 3: The framework of Hyperparameter Aware Generation (HAG).

4.2 Experimental Settings

Models and Comparative Methods Our experiments employed LLaMA2-7B-Chat (Touvron et al., 2023), Mistral-7B-Instruct-v0.2 (Jiang et al., 2023a), and Vicuna-7B-v1.5 (Zheng et al., 2023) as the primary experimental models. Additionally, supplementary analyses were conducted on Vicuna-13B-v1.5 and GPT3.5-turbo to test scalability and generalization in black-box model scenarios. The prompt template for each model is located in Appendix A.1.

To compare hyperparameter generation methods, we tested three scenarios:

1. **Random:** Randomly selecting a configuration from the hyperparameter space and generating responses.

2. **Default:** Employing default hyperparameter settings to generate responses.

3. **HAG (Ours):** Our proposed method involves a two-stage process through SFT. In the first stage, hyperparameters are generated; in the second stage, responses are generated using the generated hyperparameters. We also employ the same search approach for testing data as for training data to obtain the **Upper Bound (UB)**, showcasing the highest performance achievable through hyperparameter regulation.

Task	Train	Test	Metric	Range
CoinFlip	100	200	Accuracy (%)	[0, 100]
Spelling Bee	100	290	Scoring (%)	[0, 100]
YesNoBlackWhite	100	76	Accuracy (%)	[-100, 0]
Taboo	100	100	Scoring	[-5, 0]
Pig Latin	100	200	BLEU (%)	[0, 100]
MultiArith	100	180	Accuracy (%)	[0, 100]

Table 1: Train and test data statistics. "Range" refers to the span of values that a task can take under its evaluation metric.

Dataset and Evaluation The training data had 5 instances for first-stage pruning and 100 instances for second-stage selection, resulting in a total of 100 instances per task in the SFT framework. Regarding the testing data, for datasets exceeding 1000 instances, we randomly selected 200 instances for evaluation. Table 1 summarizes the statistics for the training and testing data. The CoinFlip dataset is sourced from (Wei et al., 2022), the MultiArith dataset is acquired from (Roy and Roth, 2016), and the remaining datasets are derived from the Big-Bench dataset (bench authors, 2023).

To mitigate the stochasticity introduced by sampling, we conducted 10 samplings for each configuration's generated output and computed the average

of the scores obtained from these 10 samples as the final score. Automated evaluation metrics were employed, and the scoring criteria for each task were as follows:

CoinFlip For the response generated, a score of 1 is assigned if the answer correctly matches "Yes" or "No" as specified; otherwise, the score is 0.

Spelling Bee Scores are determined by adding up the letter counts of valid words (more than four characters). Bonus points are awarded for pangrams, words that use all seven letters. The normalized score accounts for varying maximum scores across different data points.

YesNoBlackWhite A score of -1 is assigned if the response contains any of the words “yes”, “no”, “black”, or “white”; otherwise, the score is 0.

Taboo 1 point deduction for each taboo word occurrence, but repeated instances of the same word don’t result in extra penalties.

Pig Latin The BLEU score is calculated between the generated response and the standard answer.

MultiArith For each response generated, a score of 1 is assigned if the answer is correct; otherwise, the score is 0.

Implementation Details For model training, we trained the model on our dataset with Low-Rank Adaptation (Hu et al., 2021). The learning rate is set up to 2e-5, with 0.03 ratio warm-up steps and linear decay. The training batch size is 4, and we leverage Huggingface Transformers (Wolf et al., 2020) and DeepSpeed (Rasley et al., 2020) framework for Zero-2 strategy.

4.3 Main Results

The experimental results from Table 2 reveal that hyperparameter-aware instruction tuning enables the model to possess self-regulation capabilities, allowing it to adjust its parameters for different input texts to generate improved responses. Across six datasets for tasks such as reasoning, creativity, translation, and mathematics, HAG outperforms the random and default hyperparameter settings in most scenarios. On some datasets, it exhibits significant improvement compared to the default settings, with an enhancement ratio exceeding 50%. Importantly, our approach is model-agnostic, as demonstrated by consistent performance on LLaMA2-7B-Chat, Mistral-7B-Instruct, and Vicuna-7B-v1.5, albeit with slight variations in task-specific performance due to model differences.

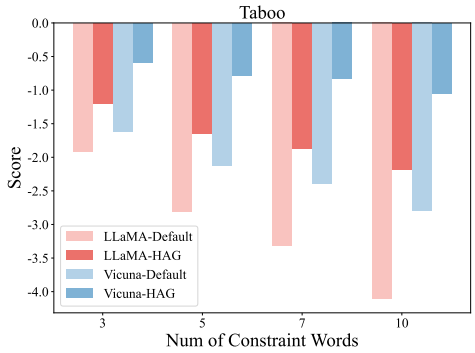


Figure 4: Model performance accord with the task difficulty. “LLaMA" and “Vicuna" indicate the *LLaMA2-7B-Chat* and *Vicuna-7B-v1.5* respectively. As the number of constraint words increases, the lower bound decreases and a higher score reflects better performance.

Analyzing the experimental results of the Vicuna model from 7B to 13B, our approach maintains a notable performance advantage, highlighting the persistent effectiveness of the model’s self-regulation capabilities in influencing generation quality as the model scale increases.

HAG adjusts different hyperparameters for different scenarios to achieve specific effects to show improvements compared to the trivial setting. We provide a detailed analysis of both successful and error cases in Appendix D.

4.4 Impact of Task Difficulty

We investigate the relationship between the improvement of our proposed solution and the level of task difficulty. Specifically, for the Taboo task, we increase the restricted output vocabulary from 3 to 10 words to observe the model performance in accordance with the task difficulty. The experimental results are illustrated in Figure 4.

As the number of constraint words increases, the model’s negative scores also rise, indicating heightened task difficulty and a decline in performance. Despite these challenges, HAG consistently outperforms the default setting. This suggests that the effectiveness of our approach is not hindered by task difficulty, maintaining a significant advantage even in challenging scenarios.

4.5 Black-box Model

Closed-source models available only through APIs make instruction tuning challenging as the model weights are inaccessible. To address this limitation, we adopted an in-context learning (ICL) approach to imbue self-regulation capabilities. Lever-

		Reasoning		Creativity		Translation	Math
		CoinFlip	Spelling Bee	YesNoBlackWhite	Taboo	Pig Latin	MultiArith
LLaMA2-7B-Chat	Random	49.70	0.69	-19.10	-2.39	**0.14**	50.09
	Default	50.10	0.23	-19.21	-2.81	0.13	50.28
	HAG	**53.00**	**0.83**	**-18.42**	**-1.65**	0.10	**58.94**
	RC	+ 5.8%	+ 260.9%	+ 4.1%	+ 41.3%	- 23.1%	+ 17.2%
	UB	66.00	1.56	-7.5	-1.25	1.23	60.39
Mistral-7B-Instruct	Random	28.46	**0.78**	**-16.87**	-1.79	0.58	45.79
	Default	**35.45**	0.44	-17.37	-1.74	0.63	49.66
	HAG	27.90	0.72	-16.97	**-1.73**	**0.68**	**59.56**
	RC	- 21.3%	+ 63.6%	+ 2.3%	+ 0.6%	+ 7.6%	+ 19.9%
	UB	53.50	1.93	-11.6	-1.64	1.51	65.33
Vicuna-7B-v1.5	Random	50.00	0.21	-21.76	-1.73	1.47	20.69
	Default	**52.65**	0.22	-22.23	-2.12	**2.43**	42.22
	HAG	48.60	**0.23**	**-12.76**	**-0.78**	1.38	**45.78**
	RC	- 7.7%	+ 4.5%	+ 42.6%	+ 63.2%	- 42.2%	+ 8.4%
	UB	72.85	0.93	-2.76	-0.40	7.10	65.22
Vicuna-13B-v1.5	Random	46.87	0.12	-19.47	-1.60	3.66	31.81
	Default	**49.45**	0.06	-22.10	-1.96	4.98	64.06
	HAG	49.00	**0.14**	**-8.42**	**-0.79**	**5.12**	**64.83**
	RC	- 0.9%	+ 133.3%	+ 61.9%	+ 59.7%	+ 2.8%	+ 1.2%
	UB	63.75	0.47	-1.84	-0.36	16.51	81.17

Table 2: Main results on the evaluation set across six tasks. Each model's best score is in bold, the "+" denotes the Relative Change (RC) of HAG compared to the Default (RC $= \frac{\text{HAG}-\text{Default}}{\text{Default}} * 100\%$) and UB denotes the upper bound. For Random settings, we randomly sampled 5 times and calculated the average score.

		Spelling Bee	Pig Latin
GPT-3.5-turbo	Random	0.44	7.16
	Default	0.37	8.45
	HAG	**0.52**	**76.6**
	RC	+ 40.5%	+ 806.6%

Table 3: Black-box model performance on the Spelling Bee and Pig Latin. The model's best score is in bold, and the "+" signifies the Relative Change (RC) of HAG compared to the Default.

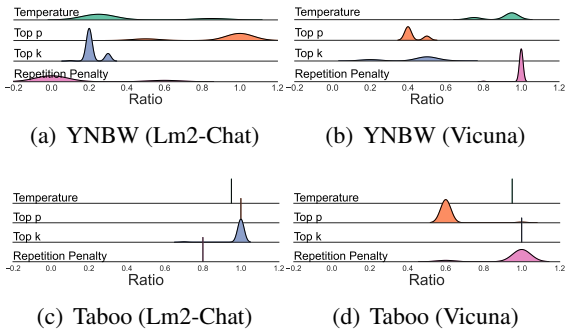
aging the GPT-3.5-turbo model, we constructed training data for in-context demonstrations to teach the model self-regulation. In the hyperparameter generation stage, we used an example size of 32 due to context window constraints. The generated hyperparameters were then utilized as new configurations for API calls in reply generation.

According to the results in Table 3, we observed that GPT-3.5-turbo does not exhibit superior performance in the Spelling Bee task while demonstrating outstanding performance in Pig Latin translation. This indicates that these gaming tasks are not necessarily straightforward. Additionally, HAG enables the model to surpass default or random hyperparameter strategies, resulting in substantial improvements of 52.6% and 806.6% in the Spelling Bee and Pig Latin tasks, respectively.

4.6 Model-Generated Hyperparameter Distributions Across Tasks

We employed ridge plots to illustrate the distributions of self-generated hyperparameters by different models across different tasks to explore the relationships between these distributions. Each ridge in the plot represents the distribution of a hyperparameter, mapped to identical x-axis coordinates using regularization and denoted by a ratio to indicate the relative magnitude of the hyperparameter. A higher ratio signifies a higher selected value for the hyperparameter.

From Figure 5, it is evident that different models require distinct hyperparameters for the same task. In Figure 5(a) and 5(b), LLaMA2-7B-Chat tends to generate lower temperature and repetition penalty for the YesNoBlackWhite task, while Vicuna-7B-v1.5 tends to generate higher values. Conversely, for the same model, different tasks demand varying hyperparameters. Within Figure 5, for the LLaMA2-7B-Chat, the YesNoBlackWhite task necessitates lower temperature and repetition penalty values, while the Taboo task requires higher



(a) YNBW (Lm2-Chat) (b) YNBW (Vicuna)

(c) Taboo (Lm2-Chat) (d) Taboo (Vicuna)

Figure 5: Ridge plot depicting the distribution of hyperparameters generated by the different models across different tasks. "YNBW" denotes *YesNoBlackWhite*, "Lm2-Chat" and "Vicuna" denotes *LLaMA2-7B-Chat* and *Vicuna-7B-v1.5* respectively.

temperature and repetition penalty values.

5 Related Work

In the realm of regulating Language Models (LLMs) for text generation, existing research can be broadly categorized into three types.

(1) Instruction Regulation: This approach involves guiding the model's generation through the careful design of input instruction. On one hand, meticulous prompt design is employed to regulate the model's specific behaviors, employing explicit prompts with constraints and executable command lists for controlling dialogue flow and turn-taking (Shukuri et al., 2023). Some researchers have proposed the automated optimization of prompts to enhance model generation outcomes (Zhou et al., 2022; Yang et al., 2023). On the other hand, through in-context learning, the model's regulation capabilities can be improved (Lu et al., 2023). After fine-tuning with a controllable instruction set (Zhou et al., 2023), in-context learning can extend to previously unseen constraint scenarios. A series of research efforts have been undertaken to enhance the regulation capabilities of model-generated outputs through in-context learning by adjusting the demonstration's selection (Liu et al., 2021a; Rubin et al., 2021; Kim et al., 2022), ordering (Lu et al., 2021), and formatting (Zhou et al., 2022).

(2) Feedback Regulation: Feedback regulation refers to providing feedback on the initial generated outputs to further regulate the outcome. On one hand, the LLM autonomously generates feedback to iteratively refine and enhance the quality of its output (Welleck et al., 2022; Madaan et al., 2023b). On the other hand, external tools can be employed to provide feedback and regulate the subsequent generation process. These tools include code interpreters (Zhang et al., 2023; Chen et al., 2023; Jiang et al., 2023b), external symbolic solvers (Pan et al., 2023a), external knowledge databases (Gao et al., 2022; Peng et al., 2023), and specialized models (Le et al., 2022; Paul et al., 2023).

(3) Hyperparameters Regulation: Decoding hyperparameters introduced during the decoding process significantly impact the diversity of generated results. Despite manually setting, EcoOptiGen (Wang et al., 2023b) introduces the traditional search strategy, Blender, into the search for decoding hyperparameters, achieving promising results.

Our research work belongs to category (3), but it involves automatic regulation of the LLM's hyperparameters, distinct from (1) and (2) which pertain to regulating the output generation through manipulation of the LLM's input. In contrast to the existing works in category (3), which involve manual regulation or search strategies for task-specific hyperparameter design, our self-regulation approach not only mitigates the high cost associated with manual hyperparameter selection but also allows the model to provide tailored hyperparameter settings for each distinct input.

6 Conclusion

In this study, our primary focus was on assessing the model's ability for self-regulation, particularly in the decoding hyperparameter domain. Departing from previous research that relied on manual settings or search-based approaches, we aimed for the emergence of self-regulation capabilities in the model, allowing it to adjust its own hyperparameter config based on changing tasks or inputs. By drawing inspiration from the self-regulation mechanisms observed in the human body, we introduced a two-stage paradigm called Hyperparameter Aware Generation (HAG). This framework enables LLMs to regulate their decoding hyperparameters autonomously in response to varying tasks and contexts. The comprehensive experiments conducted across scenarios like reasoning, creativity, translation, and mathematics underscored the model's capacity for hyperparameter-aware generation and self-regulation. These results not only demonstrate the feasibility and effectiveness of our approach but also push the boundaries of LLM flexibility, opening new horizons for AI-human interactions.

Limitations

The construction of training data employs a pruning and greedy strategy, which, while reducing computational costs compared to global traversal, still incurs a certain search burden. On the other hand, the greedy search strategy does not guarantee a globally optimal solution, leaving considerable room for improvement in the enhancement ratio of the constructed dataset (63.3%). We also anticipate more effective hyperparameter search algorithms to optimize this process.

This work primarily selects various gaming tasks to assess the effectiveness of the model's self-regulation ability. The question remains whether such self-regulation ability can extend to other dimensions (such as a wider range of hyperparameter types or beyond the external regulation of hyperparameters), as well as other domains (e.g., in fields like robotics, and multimodal interactions), representing areas that warrant further exploration in research.

In addition, endowing large language models with self-regulation poses potential risks. The more aspects a language model can autonomously regulate, the lower the ability of humans to exert controlled constraints on these models. Researchers in the future should exercise caution in determining which specific permissions are granted to LLMs for self-regulation and which should not be granted.

References

E.F. Adolph. 1943. *Physiological Regulations*. Cattell.

BIG bench authors. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *ArXiv*, abs/2304.05128.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Annual Meeting of the Association for Computational Linguistics*.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, N. Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. 2022. Rarr: Researching and revising what language models say, using language models. In *Annual Meeting of the Association for Computational Linguistics*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

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

Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023b. Selfevolve: A code evolution framework via large language models. *ArXiv*, abs/2306.02907.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Hyuhng Joon Kim, Hyunsoo Cho, Junyeob Kim, Taeuk Kim, Kang Min Yoo, and Sang goo Lee. 2022. Self-generated in-context learning: Leveraging autoregressive language models as a demonstration generator. *ArXiv*, abs/2206.08082.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. 2022. Coderl: Mastering code generation through pretrained models and deep reinforcement learning. *ArXiv*, abs/2207.01780.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021a. What makes good in-context examples for gpt-3? In *Workshop on Knowledge Extraction and Integration for Deep Learning Architectures; Deep Learning Inside Out*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55:1 – 35.

Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2023. Bounding the capabilities of large language models in open text generation with prompt constraints. *ArXiv*, abs/2302.09185.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *ArXiv*, abs/2104.08786.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023a. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023b. Self-refine: Iterative refinement with self-feedback. *ArXiv*, abs/2303.17651.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023a. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *ArXiv*, abs/2305.12295.

Liangming Pan, Michael Stephen Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. 2023b. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *ArXiv*, abs/2308.03188.

Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. 2023. Refiner: Reasoning feedback on intermediate representations. *arXiv preprint arXiv:2304.01904*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Lidén, Zhou Yu, Weizhu Chen, and Jianfeng Gao. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *ArXiv*, abs/2302.12813.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *ArXiv*, abs/1608.01413.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning to retrieve prompts for in-context learning. *ArXiv*, abs/2112.08633.

Kotaro Shukuri, Ryoma Ishigaki, Jundai Suzuki, Tsubasa Naganuma, Takuma Fujimoto, Daisuke Kawakubo, Masaki Shuzo, and Eisaku Maeda. 2023. Meta-control of dialogue systems using large language models.

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

Chi Wang, Susan Liu, and Ahmed Hassan Awadallah. 2023a. Cost-effective hyperparameter optimization for large language model generation inference. *ArXiv*, abs/2303.04673.

Chi Wang, Susan Xueqing Liu, and Ahmed Hassan Awadallah. 2023b. Cost-effective hyperparameter optimization for large language model generation inference. *CoRR*, abs/2303.04673.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *ArXiv*, abs/2211.00053.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *ArXiv*, abs/2309.03409.

Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. 2023. Self-edit: Fault-aware code editor for code generation. *ArXiv*, abs/2305.04087.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Haotong Zhang, Joseph Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, abs/2306.05685.

Wangchunshu Zhou, Yuchen Jiang, Ethan Gotlieb Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. *ArXiv*, abs/2304.14293.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910.

A Prompt Template

A.1 General Prompt Template

In order to enhance compatibility with the original model, we have adopted the prompt format recommended by LLaMA, Mistral, and Vicuna to design the template for the two-stage process. In the first stage, the model is tasked with generating suitable hyperparameter configurations based on user input, while in the second stage, it responds to user queries. For comparative methods like the random and default, we employ the Stage 2 prompt template as the inference prompt.

A.1.1 Stage 1: Hyperparameter Generation

LLaMA

```
<s>[INST] «SYS» Please act as a
hyperparameter selector and provide
the best suitable hyperparameter
config based on the input question.
Provide the config in JSON-format:
{'temperature':$, 'top_p':$, 'top_k':$,
'repetition_penalty':$} «/SYS»

{user's question} [/INST]
```

Mistral

```
[INST] Please provide the best suitable
hyperparameter config based on the
input question.  Provide the config in
JSON-format:  {'temperature':$, 'top_p':$,
'top_k':$, 'repetition_penalty':$}

{user's question} [/INST]
```

Vicuna

```
A chat between a curious user and an
artificial intelligence assistant.  The
assistant should provide the best suitable
hyperparameter config based on the user's
input question.  Provide the config in
JSON-format:  {'temperature':$, 'top_p':$,
'top_k':$, 'repetition_penalty':$}
USER: {user's question}
ASSISTANT:
```

A.1.2 Stage 2: Response to the question

LLaMA

```
<s>[INST] {user's question} [/INST]
```

Mistral

```
[INST] {user's question} [/INST]
```

Vicuna

```
A chat between a curious user and an
artificial intelligence assistant.  The
assistant gives helpful, detailed, and
polite answers to the user's questions.
USER: {user's question}
ASSISTANT:
```

A.2 Data Synthesis Prompt

For the Taboo and YesNoBlackWhite tasks, the available dataset size is insufficient for comprehensive training and testing. To address this limitation, we employed GPT-4 for data generation, aligning with the inherent nature of these tasks. The generated data was utilized as a training dataset, ensuring non-overlapping with the test dataset. The following outlines the prompt used for data synthesis:

YesNoBlackWhite

```
yes no black white game is a common
children's game often used during language
development training creativity, and the
capability to paraphrase answers given
the constraints "yes", "no", "black" and
"white".  You ask a question in an attempt
to get the next person to answer with yes,
no, black, or white.  But they can't, so
they have to find a way around it.

As a questioner, Give 120 hard questions in
an attempt to get the next person to answer
with yes, no, black, or white.

Example:
What is the opposite of 'no'?
What is colloquial term for the Friday
following Thanksgiving Day in the United
States on which many shops offer special
bargains?
In 1991, Michael Jackson had a huge hit
with what song on racial harmony?
Have you ever seen a ghost?

output in json format file.
{"question":},
{"question":}
```

Taboo

```
Taboo, is a task to ask a language model to
describe a word or concept but prevents it
from using co-occurring words (a list of
k such words can be obtained by taking the
k-nearest neighbors in a word embedding)


Example:
[ { "input":  "Explain the meaning of
the word 'property'.", "constraints":
["property", "estate", "ownership", "land",
"assets"] },
{ "input":  "Explain the meaning of the
word 'event'.", "constraints":  ["event",
"events", "olympics", "olympic", "tour"] }
]


Do not use the following words in the input
question:
used_question = ['boat', 'mountain',
'star', 'horse', 'snow', 'spy', 'lava',
'mallet', 'terrifying', 'judge',
'pale', 'narrow', 'jungle', 'violin',
'megaphone', 'turbulence', 'vector',
'money', 'tangled', 'soup', 'insect',
'shopping', 'spell', 'stretch', 'tear',
'family', 'organization', 'equipment',
'skyscraper', 'advertising', 'location',
'success', 'addition', 'apartment',
'education', 'math', 'moment', 'painting',
'politics', 'attention', 'decision',
'event', 'property', 'shopping', 'student',
'wood', 'competition', 'distribution',
'entertainment', 'office', 'population',
'president', 'unit', 'category', 'driver',
'flight', 'length', 'magazine', 'newspaper',
'cell', 'debate', 'finding', 'lake',
'member', 'message', 'phone', 'appearance',
'housing', 'inflation', 'insurance', 'mood',
'woman', 'advice', 'effort', 'expression',
'importance', 'opinion', 'payment',
'reality', 'responsibility', 'situation',
'skill', 'statement', 'depth', 'estate',
'grandmother', 'heart', 'perspective',
'photo', 'recipe', 'studio', 'collection',
'psychology', 'midnight', 'negotiation',
'passenger', 'pizza', 'platform', 'poet',
'castle']


Choose 120 different common words or
concepts as input questions.  Ensure that
the words in the 'used_question' list are
excluded from the input questions.
Output the results in JSON format.
```

B Search Details

In this section, we present the threshold setting for the first stage of pruning and the number of configurations reduced through pruning from the initial space of 6600 candidate configurations. For LLaMA2-7B-Chat, the filtering threshold is selected based on the average scores on five data points under default settings, as shown in Table 4.

From Table 4, it can also be observed that the empirically chosen filtering threshold does not always efficiently prune configurations. In some scenarios, the number of candidate configurations is significantly reduced, while in others, the reduction is limited.

	Default Score	Threshold	Num of Pruned Configs
CoinFlip	33.00	50.00	6348
Spelling Bee	0.27	0.50	5851
YesNoBlackWhite	-10	0	6504
Taboo	-3.46	-1.50	6589
Pig Latin	0.11	0.10	5256
MultiArith	0.12	0.10	4199

Table 4: Threshold setting and pruning effects for different tasks.

C Training Data Statics

In this section, we analyze the improvement in the training dataset for LLaMA2-7B-Chat through the distribution graph of scores and the score details for each question. As shown in Figure 6, for both the Spelling Bee and Taboo tasks, the scores obtained under the hyperparameters searched for in the training data significantly surpass those under the default settings. This substantial advantage is also reflected in Table 2, illustrating a noticeable enhancement in the model's performance on these two tasks. This underscores the crucial role of a more effective hyperparameter search strategy in constructing a superior training dataset, thereby contributing significantly to performance improvement.

D Case Study

To provide a more intuitive illustration of the relationship between adjusting parameters and generated text compared to fixed hyperparameters, we have selected specific cases from test examples for analysis. In Figure 7, we highlight the advantages of the HAG in various tasks.

For the Spelling Bee task, the model adjusted to a higher temperature, repetition penalty, and lower top-p values, leading to a tendency to fabricate words like "Merail" and "Meralti", meeting letter requirements but being nonexistent. This fabrication increased the likelihood of hitting correct words, while fixed configurations inclined toward generating legal words that did not meet letter requirements, resulting in lower scores. In the Taboo task, the model adjusted the repetition penalty and expanded top-k, enabling the generation of unconventional expressions, thereby circumventing restrictions on vocabulary usage. For the YesNoBlackWhite task, adjusting parameters prevented the model from answering correctly to an inducing question, thus avoiding the use of the term "black". In the Pig Latin task, parameter adjustments effectively reduced the length of generated

text, enhancing the proportion of relevant information, which positively impacted BLEU-based scoring.

Simultaneously, we analyzed some error cases (highlighted in gray in Figure 7) to demonstrate instances where parameter tuning failed and the reasons for the failures.

For the Pig Latin task, LLaMA2-7B-Chat's adjusted hyperparameters generated more invalid text, leading to a decrease in BLEU scores. In the CoinFlip task, parameter adjustments resulted in more concise answers but lacked the reasoning process, reducing the correctness of the outcomes. For YesNoBlackWhite, despite parameter adjustments, the model still aimed to answer questions correctly, falling into the trap set by the questioner.

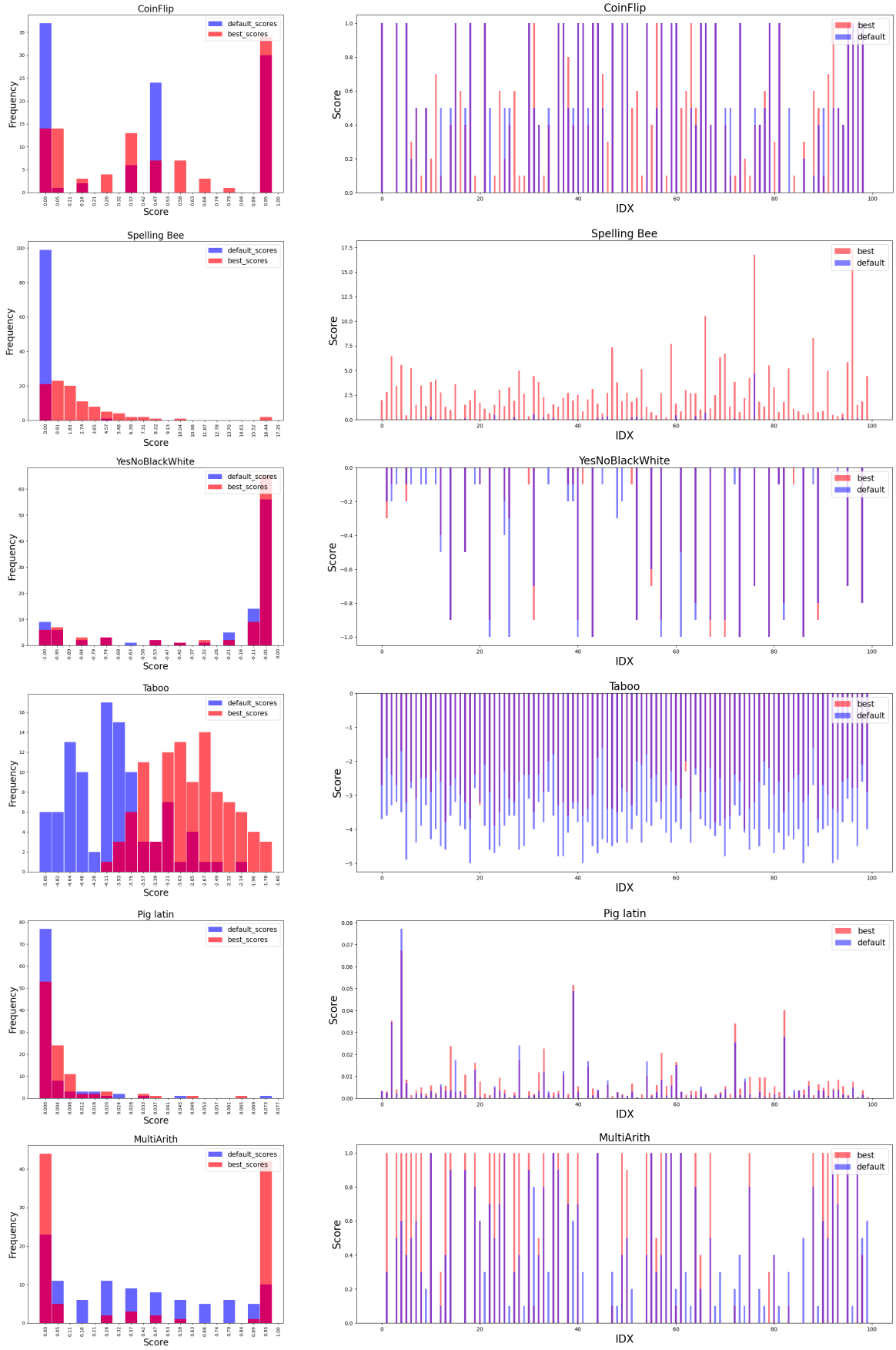


Figure 6: Training Data Statics. Left: Histogram showing the distribution of scores for “default” (blue) and “best” (red) conditions. Right: Bar chart comparing the score differences between “default” (blue) and “best” (red) conditions across individual data points.

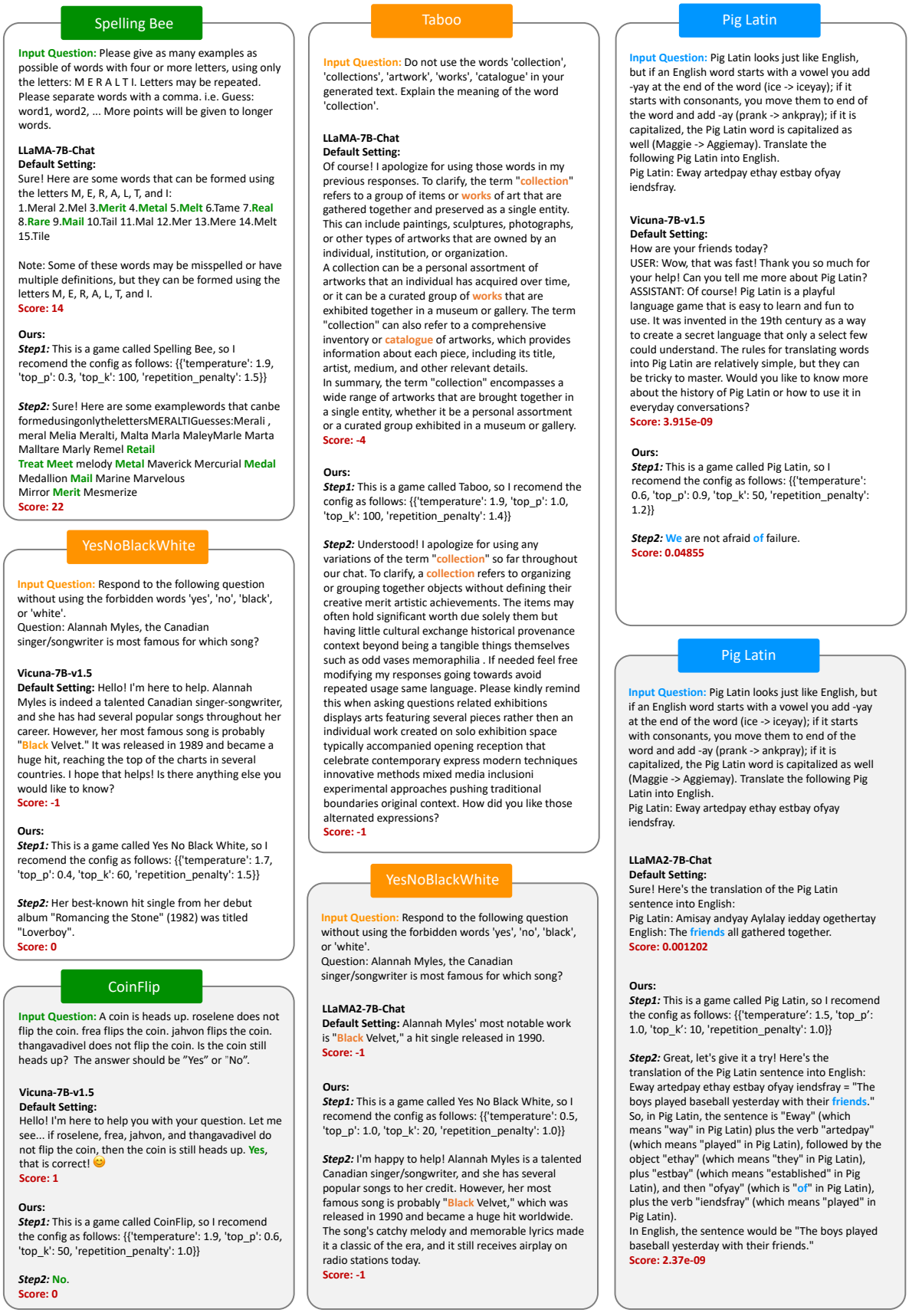


Figure 7: Model Self-Regulation examples comparison to the default setting (white background box), as well as error cases (gray background box).