Constructing a Japanese Rap Lyric Generation Model with GRPO

Hayato Ogawa **Daisuke Kawahara**
Waseda University
cookie3120@ruri.waseda.jp, dkw@waseda.jp

Abstract

Rap is a vocal style rooted in Hip-Hop culture, characterized by producing rhymes in synchrony with a rhythmic beat. This paper proposes a method for generating Japanese rap lyrics with a large language model (LLM) whose rhyming behavior is improved via reinforcement learning. We design a reward function that evaluates end rhymes between two generated bars and apply GRPO, a reinforcement-learning method, to encourage Japanese rhyming without using existing Japanese rap lyrics as training data. Experimental results show that, although output collapse is observed in some cases, GRPO increases the proportion of outputs that receive moderate or high human ratings on rhyme-related criteria.

1 Introduction

Rap is a vocal style rooted in hip-hop culture in which lyrics are delivered rhythmically over a musical beat. One of the most important elements of rap is rhyme. Rhyming is a technique that creates rhythmic and aesthetic effects by aligning phonetic patterns across words or phrases. Rhyming contributes not only to the musical appeal of rap but also to its expressive power, making it a central component of rap lyric composition. However, creating rap lyrics that satisfy rhyming constraints while preserving coherent meaning is challenging even for humans.

In recent years, rapid advances in large language models (LLMs) have enabled fluent text generation even for creative tasks that traditionally required substantial human creativity, such as writing poems or stories. This progress raises the possibility of new forms of artistic expression through human–LLM co-creation, including rap lyric generation. However, generating text that includes rhymes remains challenging. This is because rhyming depends on phonological structures, which cannot be directly observed from surface text alone, which is used to train LLMs.

These challenges are particularly pronounced in Japanese. Japanese uses a mixture of kanji and kana, as seen in expressions containing characters such as 美学 (bigaku; "Aesthetics") and びがく (bigaku; "Aesthetics"), and many kanji have multiple possible readings depending on context. As a result, it is considerably more difficult for LLMs to infer accurate phonological structures solely from surface forms than English. Furthermore, the Agency for Cultural Affairs in Japan has stated that generated content may constitute copyright infringement if it is judged to be similar to existing copyrighted works (Subcommittee on Legal Systems, 2024). These issues motivate the development of learning methods that improve rhyming ability in Japanese without relying on existing artists' lyric corpora.

In this paper, we propose a reinforcement learning approach for improving rhyming ability in Japanese rap lyric generation. Specifically, we apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024), a reinforcement learning method that computes rewards for multiple outputs generated from the same prompt and updates the policy based on their relative comparisons. To the best of our knowledge, prior rhyme-aware lyric generation methods have largely relied on lyric corpora, retrieval-based constraints, or post-processing (Potash et al., 2015; Malmi et al., 2016; Manjavacas et al., 2019; Nikolov et al., 2020; Xue et al., 2021). In contrast, this study employs reinforcement learning with an explicit rhyme reward to improve rhyming ability in Japanese rap lyric generation, while avoiding the use of existing artists' lyric corpora for training under copyright-conscious data constraints. Because GRPO does not require preference data or a separately trained reward model, it provides a simple framework for improving text generation under phonological constraints. An overview of the proposed method is

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop), pages 340–351

shown in Figure 1.

2 Related Work

This section reviews reinforcement-learning methods for LLMs, prior work on lyric generation with rhymes, and challenges in modeling Japanese readings with LLMs.

2.1 Reinforcement Learning for LLMs

Several reinforcement learning methods have been proposed for aligning large language models (LLMs), including PPO (Proximal Policy Optimization) (Schulman et al., 2017) and DPO (Direct Preference Optimization) (Rafailov et al., 2023). PPO optimizes the model using rewards predicted by a separately trained reward model, typically learned from human preference datasets. While this framework has been widely used in RLHF (Reinforcement Learning from Human Feedback) pipelines (Christiano et al., 2017), it requires an additional reward modeling stage and large-scale human preference annotations.

DPO, in contrast, learns directly from preference pairs without explicitly training a reward model. Although this simplifies the training pipeline, it still relies on curated preference datasets that indicate which outputs are preferred.

In our setting, constructing reliable preference data for Japanese rap lyrics is costly and difficult. This challenge becomes even more pronounced because we avoid using existing rap lyrics as training data due to copyright considerations. DPO is therefore difficult to apply directly as an initial training method. Similarly, standard RLHF-style PPO pipelines often require a separately trained reward model or substantial human feedback. These requirements make preference-based or reward-model-based training less suitable for our initial setting.

More recently, GRPO (Group Relative Policy Optimization), introduced in DeepSeekMath (Shao et al., 2024), has been proposed as an alternative reinforcement learning method. Instead of optimizing against a separately trained reward model, GRPO evaluates multiple outputs generated for the same input and updates the policy based on their relative rankings. This approach removes the need for a reward model and allows the training signal to be computed directly from explicit scoring functions.

In our task, the objective is to improve rhyme generation in Japanese rap lyrics under phonological constraints. The quality of generated lyrics can be evaluated using explicit reward functions based on rhyme similarity, repetition penalties, and structural validity.

This property is particularly useful for our task because rhyme quality can be evaluated for each generated candidate without collecting pairwise human preferences. Given a single theme, the model can generate multiple two-bar candidates, and these candidates can be compared using the same explicit rhyme-related reward function.

For these reasons, we adopt GRPO in this study as a practical first step for optimizing Japanese rap lyric generation with explicit phonological rewards. GRPO allows us to optimize the model directly using rhyme-related rewards while leveraging relative comparisons among multiple outputs generated from the same input. This choice does not assume that GRPO is more stable than preference-based methods such as DPO; rather, it reflects the absence of reliable preference pairs in our current setting. As discussed in Section 5, using reward-scored candidates to construct preference data for methods such as DPO is an important direction for future work.

2.2 Lyric Generation with Rhymes

A wide range of methods for generating rhymed lyrics has been proposed, particularly for English. Potash et al. (2015) proposed GhostWriter, an LSTM-based model that mimics the style of specific artists. From an information-retrieval perspective, Malmi et al. (2016) proposed DopeLearning, which combines RankSVM and deep neural networks to select the next line that maximizes rhyme density from candidate lyrics. To impose structural constraints with generative models, Manjavacas et al. (2019) proposed learning formal constraints for poetry and rap using hierarchical LSTMs and conditional templates. Transformer-based methods include Rapformer by Nikolov et al. (2020) and DeepRapper by Xue et al. (2021). Rapformer promotes rhyming by post-editing generated endings to match vowel sequences, whereas DeepRapper constructs a large-scale dataset aligned with beats and improves end-rhyme consistency by generating lyrics in reverse order while explicitly modeling rhythmic structure with [BEAT] tokens.

For interactive rap generation (rap battles), Wu and Addanki (2015) used TRAAM (Transduction Recursive Auto-Associative Memory), a type of

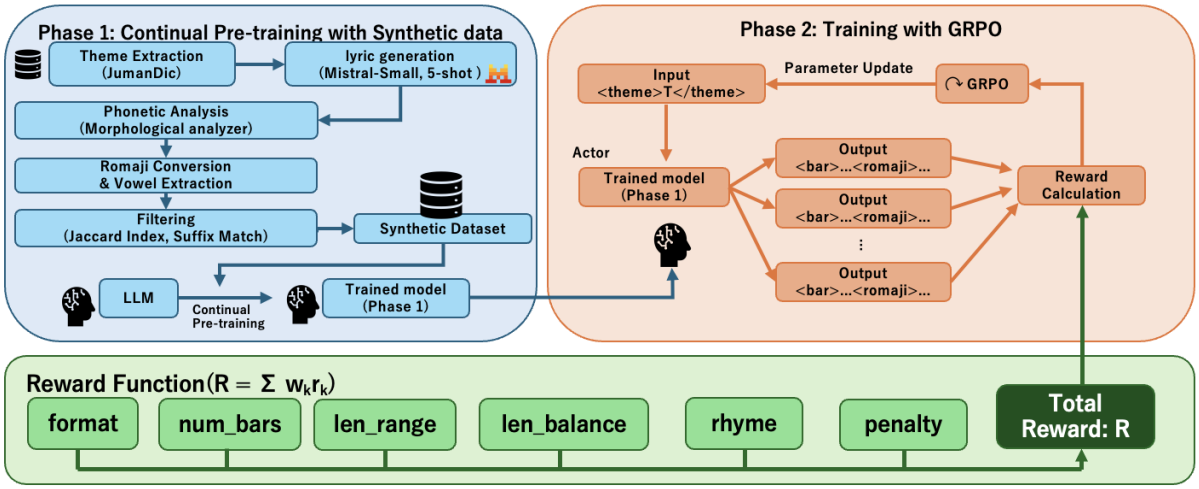

Figure 1: Overview of the proposed method.

recursive neural network, to generate responses to an opponent's lyrics. Savery et al. (2020) also demonstrated real-time improvisational responses in a robot-based dialogue system using phoneme embeddings.

For Japanese lyric generation, Oda et al. (2025) proposed a method that learns the output format without supervised lyric data via structure-aware training (Ormazabal et al., 2022) and controls rhyming through vowel specification. Our method shares with Oda et al. (2025) the policy of avoiding training on existing lyrics. However, our method differs in that it does not require specifying target vowels for rhyming.

Mibayashi et al. (2024) proposed a response verse generation method for rap battles using BERT2BERT (Rothe et al., 2020). In their method, candidate rhyming words are retrieved from a dictionary based on vowel matching, and rap sentences are generated in reverse order so that the selected rhyming word appears at the end of the sentence. In contrast, our method does not select rhyming words from a dictionary but directly evaluates the similarity of vowel sequences over a fixed suffix length. This formulation enables the generation of rhymes that span multiple words rather than being limited to single-word matches. For example, 近い腕 (chikai ude; "close arm") and 今見る目 (ima miru me; "the eyes now looking") form a multi-word rhyme because their phrase-final vowel sequences align across word boundaries, even though no individual word is reused.

Mibayashi et al. (2025) also proposed a method for generating Japanese rhyme phrases using mora similarity and generation probability. While these studies generate rhymes through external mechanisms such as dictionary retrieval or phonological control, our method instead encourages the LLM itself to acquire rhyming ability through reinforcement learning with a reward defined by vowel-sequence similarity.

Overall, prior rhyme-aware lyric generation methods have mainly relied on existing lyric corpora, retrieval-based generation from candidate lines or rhyme words, explicit phonological control such as target-vowel specification, or post-processing to enforce rhyme constraints. While these approaches are effective in their respective settings, they do not directly optimize the generative model itself for rhyming under copyright-conscious data constraints. The key distinction of our work is that it directly optimizes the generative model for rhyme-aware behavior under a setting where existing rap lyrics are not used for training and external rhyme control is not required at inference time. Specifically, our method does not use rhyme dictionaries, target-vowel specification, or post-editing during inference, but instead encourages rhyming through reinforcement learning with an explicit reward based on vowel-sequence similarity. This setting is particularly challenging in Japanese, where accurate reading prediction remains difficult for LLMs, as discussed in Section 2.3.

2.3 Challenges of Japanese Reading in LLMs

The Japanese language consists of three scripts: hiragana, katakana, and kanji. In particular, kanji characters possess polysemy, meaning their readings change depending on the context. In the gen-

eration of rap lyrics, it is essential for LLMs to accurately grasp the "reading" (pronunciation) of words to establish successful rhymes. However, existing research has pointed out that the ability of LLMs to understand Japanese readings is insufficient compared to their semantic understanding capabilities.

Mibayashi et al. (2026) proposed YOMI-Bench, a benchmark specialized for Japanese reading, and evaluated tasks such as reading estimation and rhyme selection. According to their report, in the kanji reading estimation task, the accuracy for words with multiple reading candidates ("Multiple") tends to be lower than that for words with a unique reading ("Single") across all evaluated models. This suggests that LLMs struggle with disambiguating readings based on context.

Furthermore, in the rhyme selection task, which identifies rhyming relationships between words, there is a significant disparity in performance among models. While the latest commercial models such as claude-sonnet-4-5-20250929 and gpt-5 achieved high scores, gpt-4o remained at extremely low scores of 0.5820 for kanji notation and 0.2580 for hiragana notation. Moreover, even for local LLMs specialized in Japanese, the scores for this task were generally low, indicating that they do not accurately capture the phonological structure of Japanese.

These findings demonstrate that a model's intrinsic reading and phonological understanding capabilities can be a bottleneck when generating rhyming lyrics using LLMs. This supports the necessity of the explicit learning of forms and rhymes through reinforcement learning, as proposed in this study.

3 Construction of a Japanese Rap Generation Model

In this study, we build a model that generates fluent Japanese two-bar lyrics with end rhymes without training on existing lyrics, using GRPO as a reinforcement-learning algorithm. In rap, a bar is a rhythmic unit roughly corresponding to one measure of lyrics. In this study, we operationalize a bar as one generated lyric line and focus on end rhymes between two generated bars, without explicitly modeling beat-level timing. A well-known issue in reinforcement learning is "cold start," where model outputs can become unstable early in training. To mitigate this instability and adapt the model to a specialized input/output format, we conduct continued pre-training on synthetic data before GRPO, following the strategy used in DeepSeek-R1 (Guo et al., 2025). An overview of the proposed method is shown in Figure 1.

In what follows, we first define the input/output format used for reward computation, and then describe the construction of synthetic data and continued pre-training. Finally, we present the reward functions used to optimize rhyming and overall generation quality under GRPO. We use Sarashina2.2-3b[1] as the base model. This model is primarily trained using large-scale Japanese corpora and English corpora, making it a Japanese-specific pre-trained model.

3.1 Input/Output Format

To compute rhyme quality as a reinforcement-learning reward, we must reliably identify the bar endings in the model's output. We therefore use the following structured input/output format.

Input:
<theme>T</theme>

Output:
<bar>B_1</bar><romaji>R_1</romaji>
<bar>B_2</bar><romaji>R_2</romaji>

Here, T denotes the rap theme, B_i the lyric text of the i-th bar, and R_i its pronunciation in Romaji. To adapt the model to this output format, we perform continued pre-training prior to GRPO.

3.2 Continued Pre-training with Synthetic Data

3.2.1 Construction of Synthetic Dataset

We construct synthetic end-rhyme data for continued pre-training using LLM generation and pronunciation analysis. Each training example consists of two bars (B_1, B_2) with different themes, constructed so that the vowel/nasal sequences of the final five moras match across the two bars. Because (B_1, B_2) are associated with different themes, coherence as a single set of lyrics is not required. This design aims to (i) adapt the model to the input/output format in Section 3.1 and (ii) mitigate cold start.

In this subsection, we describe the generation settings and the procedure for constructing and fil-

[1] https://huggingface.co/sbintuitions/sarashina2.2-3b

tering rhyme pairs. First, we extracted theme candidates from the nouns and verbs listed in JumanDic, a morphological analysis dictionary,[2] and generated a single bar B for each theme T using Mistral-Small-3.1-24B-Instruct-2503[3], yielding <theme, 1 bar> pairs. We used 5-shot prompting and generated multiple outputs with different shot combinations. An example prompt is provided in Appendix A. After removing failures and empty outputs, we obtained 88,590 <theme, 1 bar> pairs.

Next, we analyzed pronunciations for each bar B using the morphological analyzer MeCab (Kudo et al., 2004) and converted them to Romaji. From the resulting Romaji sequence, we extracted only vowels and the moraic nasal $\{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}, \mathrm{N}\}$ to form a vowel/nasal sequence $V(B)$. We then grouped bars whose vowel/nasal sequences share the same final five characters. When creating two-bar pairs within each group, we introduced surface-similarity-based selection to avoid trivial repetition of the same ending expressions. Let A_1 and A_2 be the sets of characters appearing in the last five characters of the surface strings of B_1 and B_2, respectively. We compute the Jaccard index:

$$J(A_1, A_2) = \frac{|A_1 \cap A_2|}{|A_1 \cup A_2|}.$$

Within each group, we greedily select pairs that minimize J and remove bars once used in a pair from further consideration. This yields 42,335 two-bar pairs whose vowel/nasal sequences match in the final five characters.

Finally, to eliminate pairs with strong surface overlap, we retain only those with $J \leq 0.2$, resulting in 36,766 synthetic examples. We split the data into training/validation sets at a 7:3 ratio. Inputs are formatted as <theme>T_1, T_2</theme>, and outputs follow the format in Section 3.1 with two bars corresponding to different themes. Examples are shown in Table 1.

3.2.2 Continued Pre-training

We conducted continued pre-training of Sarashina2.2-3b for three epochs using the constructed dataset. The main hyperparameter settings for continued pre-training and GRPO are provided in Appendix B.

[2] https://github.com/ku-nlp/JumanDIC

[3] https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Instruct-2503

3.3 Training with GRPO

3.3.1 Reward Function Design

In this study, we design the scalar reward R in GRPO to jointly evaluate (1) compliance with the specified input/output format and the two-bar structure, (2) the degree of surface-level repetition, and (3) end-rhyme quality based on vowel/nasal sequences. Specifically, we define the set of reward components as

$$K = \{\text{format}, \text{num\_bars}, \text{len\_range}, \text{len\_balance}, \text{rhyme}, \text{penalty}\}.$$

The total reward is computed as

$$R = \sum_{k \in K} w_k \, r_k, \tag{1}$$

where r_k denotes the component reward and w_k its weight. In our implementation, we set $w_k = 1.0$ for the four components that validate the output form and structure $(r_{\text{format}}, r_{\text{num\_bars}}, r_{\text{len\_range}}, r_{\text{len\_balance}})$, and $w_k = 4.0$ for the two components related to rhyming $(r_{\text{rhyme}}, r_{\text{penalty}})$, thereby emphasizing rhyming relative to format validation.

The validity of the input/output format and the two-bar structure is evaluated using the following four criteria. First, if the pattern <bar>...</bar><romaji>...</romaji> matches the format described in Section 3.1 exactly (as determined by a regular expression), we set $r_{\text{format}} = 1.0$; otherwise, $r_{\text{format}} = -1.0$. Second, if the number of segments enclosed by <bar>...</bar> is two, we set $r_{\text{num\_bars}} = 1.0$; otherwise, $r_{\text{num\_bars}} = -1.0$, penalizing outputs whose tag structure is valid but whose number of bars is incorrect. Third, rather than using surface length, we evaluate each bar by the length L_i of the vowel/nasal sequence derived from its MeCab-based pronunciation, where the sequence is represented in Romaji and contains only $\{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}, \mathrm{N}\}$; if all bars satisfy $L_{\min} \leq L_i \leq L_{\max}$, we set $r_{\text{len\_range}} = 1.0$; otherwise, $r_{\text{len\_range}} = -1.0$. Finally, to discourage rhythmic imbalance between the two bars, we compute the pronunciation-length difference $\Delta L = |L_1 - L_2|$; if $\Delta L \leq \Delta L_{\max}$, we set $r_{\text{len\_balance}} = 1.0$; otherwise, $r_{\text{len\_balance}} = -1.0$. Together, these four components stably encourage two-bar outputs that follow the specified format and exhibit appropriate length and balance. In our

Table 1: Examples of synthetic data used for continued pre-training (Pairs with Jaccard index $J \leq 0.2$)

Input (<theme>)	Output (<bar>, <romaji>)	J	Suffix vowels
愛情 (aijou; "Affection"), 悔恨 (kaikon; "Regret")	<bar> 二人の距離は遠くても感じる暖かさ </bar> <romaji> futarinokyoriwatookutemokaNjiruatatakasa </romaji> <bar> 後悔先立って失った宝 </bar> <romaji> kookaisakidatteushinattatakara </romaji>	0	aaaaa
病み付き (yamitsuki; "Addictive"), ピーマン (piiman; "Green Pepper")	<bar> こだわりが病み付き </bar> <romaji> kodawarigayamitsuki </romaji> <bar> ピーマンが赤くなったら大好き </bar> <romaji> piimaNgaakakunattaradaisuki </romaji>	0.111	aaiui
上体 (joutai; "Upper Body"), 除法 (johou; "Division")	<bar> 胸を張って進む未来 </bar> <romaji> muneohattesusumumirai </romaji> <bar> 割り切れない数字たち </bar> <romaji> warikirenaisuujitachi </romaji>	0	uuiai

implementation, we set $L_{\min} = 8$, $L_{\max} = 16$, and $\Delta L_{\max} = 4$.

Next, we describe how we compute r_{rhyme} and r_{penalty}, which evaluate end-rhyme quality and discourage trivial repetition, respectively.

Rhyme Reward (r_{rhyme}) We evaluate rhyme quality as end-rhyme similarity computed from vowel/nasal sequences. For each bar, we extract a pronunciation sequence using MeCab, convert it from Katakana to Romaji, and denote it as R_i. From R_i, we construct a sequence that retains only vowels and the moraic nasal:

$$V_i = (v_1^{(i)}, v_2^{(i)}, \ldots, v_{L_i}^{(i)}), \quad v_j^{(i)} \in \{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}, \mathrm{N}\}.$$

We then focus on the suffix length $k = \min(L_1, L_2, 5)$. Given the two suffix sequences V_1' and V_2' (both of length k), we compute a weighted edit distance $d_{\text{suffix}}(V_1', V_2')$ that places larger emphasis on positions closer to the bar ending, and normalize it by the maximum possible cost $d_{\max}$:

$$r_{\text{rhyme}} = 1 - \frac{d_{\text{suffix}}(V_1', V_2')}{d_{\max}}.$$

This edit distance assigns larger costs to insertions, deletions, and substitutions near the end of the bar, and the substitution cost reflects phonetic proximity between vowels. The detailed designs of insertion/deletion and substitution costs are described below.

Insertion/Deletion Let the vowel/nasal sequence length be L and the position be $p \in \{0, \ldots, L-1\}$. We define the base cost as

$$c_{\text{ins/del}}(p) = \begin{cases} 1 & (p = 0 \text{ or } p = L-1) \\ 3 & (\text{otherwise}). \end{cases}$$

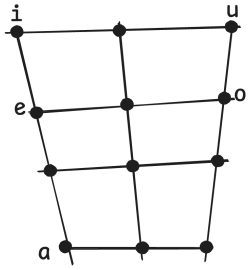

Figure 2: IPA Chart of Japanese Vowels.

We then apply a position-dependent weight that emphasizes the end of the bar:

$$w(p, L, \alpha) = \left(\frac{p+1}{L}\right)^{\alpha}, \quad \alpha = 3,$$

and use the final insertion/deletion cost $\tilde{c}_{\text{ins}}(p) = \tilde{c}_{\text{del}}(p) = c_{\text{ins/del}}(p)\, w(p, L, \alpha)$.

Substitution Let the vowel set be $\mathcal{V} = \{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}\}$. Based on the IPA chart of Japanese vowels shown in Figure 2, we define a symmetric 5×5 substitution-cost matrix C:

$$C = \begin{pmatrix} 0 & 3 & 5 & 2 & 4 \\ 3 & 0 & 2 & 1 & 3 \\ 5 & 2 & 0 & 3 & 1 \\ 2 & 1 & 3 & 0 & 2 \\ 4 & 3 & 1 & 2 & 0 \end{pmatrix}.$$

For vowels $v_p, v_q \in \mathcal{V}$, the substitution cost is $c_{\text{sub}}(v_p, v_q) = C_{pq}$. For substitutions between the moraic nasal N and a vowel, we uniformly set $c_{\text{sub}}(\mathrm{N}, v) = c_{\text{sub}}(v, \mathrm{N}) = 4$. For substitutions between characters other than vowels and the nasal, we set $c_{\text{sub}}(x, y) = 2$. We multiply these substitution costs by the same position weight $w(p, L, \alpha)$ as in insertion/deletion so that mismatches closer to the bar ending contribute more strongly to d_{suffix}.

If pronunciation extraction fails or the number of bars is not two, we treat the output as invalid for rhyming and return -1 in our implementation.

Suffix Similarity Penalty (r_{penalty}) To prevent the model from obtaining a high r_{rhyme} by simply repeating the same ending expression across bars, we introduce a penalty based on surface similarity at the bar endings. Let A_1 and A_2 be the sets of characters appearing in the final five characters of the surface strings of B_1 and B_2, respectively. Using the Jaccard index $J(A_1, A_2)$, we define

$$r_{\text{penalty}} = -J(A_1, A_2).$$

This yields a larger negative reward when the two bar endings are more similar, discouraging convergence to trivial repetition.

3.3.2 Training Settings

We randomly sampled 8,921 words (approximately 40%) from the nouns and verbs in JumanDic as themes and split them into training/validation/evaluation sets with a 7:2:1 ratio. The split was performed at the theme level, so the same theme did not appear across the training, validation, and evaluation splits. The 100 themes used for human evaluation were randomly sampled from the held-out evaluation split. We swept the learning rate over three values—2×10^{-6}, 1×10^{-6}, and 5×10^{-7}. The remaining main hyperparameter settings for GRPO are provided in Appendix B.

Learning Rate Sweep and Checkpoint Selection With larger learning rates, the total reward sometimes increased temporarily; however, we also observed output collapse during training or near convergence, where the model degenerated into repetitive generations largely independent of the prompt. An example of collapse is shown below. It can be observed that meaningless Roman character repetitions like "toku" are being generated.

Input:
<theme>Burn</theme>

Output:
<bar>Honooodoruyorutobuyoru</bar>
<romaji>honooodo...kuyoroku</romaji>
<bar>Kogetokutokutokutokutoku</bar>
<romaji>kogeto...kutokoku</romaji>

Based on these observations, we did not select checkpoints solely by total reward. Instead, we selected checkpoints from ranges that satisfy the following three criteria on validation outputs: (i) minimal format collapse, (ii) no extreme reduction in output diversity, and (iii) an increasing rhyme reward. As a result, we selected Step 117 (0.3 epoch) with a learning rate of 5×10^{-7} and used it for the evaluation in Section 4.

4 Evaluation

4.1 Human Evaluation Setup

We evaluate whether the proposed method improves rhyme generation in Japanese rap lyrics using human evaluation. Three evaluators in their twenties who regularly listen to Japanese rap assessed the generated outputs.

As a baseline, we use the model before applying GRPO (Base). For 100 themes randomly sampled from the evaluation set, each model generated two-bar lyrics that were evaluated by the annotators.

We focus on human evaluation rather than automatic metrics. Automatic rhyme evaluation can be misleading because repeating the same phrase allows the vowel sequences at the ends of the two bars to match perfectly, which does not necessarily reflect genuine rhyming ability. Human evaluation therefore provides a more reliable assessment of rhyming quality.

4.2 Evaluation Metrics

We use three evaluation criteria rated on a five-point scale: (i) Rhyme length, (ii) Low Repetition, and (iii) Unpredictability of rhyme.

Rhyme Length (1–5) This metric evaluates the mora length of the rhyming segment at the end of the bars (5: $\geq$5 moras, 4: 4 moras, 3: 2–3 moras, 2: 1 mora, 1: failed or collapsed output). If evaluators judge that a rhyme is present even without perfect vowel agreement, the shorter rhyming segment between the two bars is used for scoring.

Low Repetition (1–5) This metric evaluates the extent of surface repetition in the rhyming segment (5: none, 4: low, 3: medium, 2: high, 1: almost identical). Outputs that receive a rhyme-length score of 1 are regarded as unevaluable and assigned 1 for this metric.

Unpredictability of Rhyme (1–5) This metric measures the creativity or unexpectedness of the rhyme (5: highly unexpected, 2: very predictable). Outputs with a rhyme-length score of 1 or repetition score of 1 are treated as failing to rhyme and are assigned 1 for this metric.

Table 2: Distribution of human evaluation scores by model. Scores are averaged over three annotators and grouped into four ranges: [1,2), [2,3), [3,4), and [4,5].

Metric	Model	Mean	[1,2)	[2,3)	[3,4)	[4,5]
Rhyme Length	Base	2.390	24.0	53.0	22.0	1.0
	GRPO	2.380	27.0	46.0	24.0	3.0
Low Repetition	Base	2.957	16.0	37.0	13.0	34.0
	GRPO	3.073	22.0	14.0	27.0	37.0
Unpredictability of Rhyme	Base	2.083	33.0	58.0	9.0	0.0
	GRPO	2.177	35.0	43.0	21.0	1.0

Table 3: Examples of model outputs by the proposed method (with English translation).

Theme	Output B_1	Output B_2	r_{rhyme}
沸かせる (Wakaseru; "To Boil")	熱い波動沸き上がる Atsui hadou wakiagaru (En. Hot waves surging up)	心に火を焚き放つ Kokoro ni hi wo takihanatsu (En. Ignite a fire in the heart)	1.000
押し通せる (Oshitooseru; "To Push Through")	どんな壁もブチ破れ Donna kabe mo buchiyabure (En. Break through any wall)	自分の道を踏み外せ Jibun no michi wo fumihazuse (En. Stray from your own path)	1.000
ギックリ腰 (Gikkurigoshi; "Strained Back")	体が動かない状態 Karada ga ugokanai zyoutai (En. State where the body won't move)	ベッドにこびたりtai Beddo ni kobitari tai (En. untranslatable due to corrupted output)	0.938
花菖蒲 (Hanashoubu; "Iris")	花菖蒲咲いて夏来た Hanashoubu saite natsu kita (En. Irises blooming, summer has come)	野に咲く夢の色よ No ni saku yume no iro yo (En. Colors of a dream blooming in the field)	0.224

4.3 Quantitative Results

Table 2 shows the distributions of human evaluation scores before and after GRPO. For reference, boxplots of the same human evaluation scores are provided in Appendix C.

For Rhyme Length, both the lower score range [1,2) and the higher score ranges [3,4) and [4,5] became more frequent after GRPO, whereas the middle range [2,3) became less frequent. Specifically, the proportion in [1,2) increased from 24.0% to 27.0%, while the proportion in [3,4) or [4,5] increased from 23.0% to 27.0%.

At the same time, the proportion of outputs rated 3 or higher increased for all three metrics. For Low Repetition, the proportion increased from 47.0% to 64.0%. For Unpredictability of Rhyme, it increased from 9.0% to 22.0%. These results suggest that, although mean-score improvements remain limited, GRPO increases the frequency of outputs that receive at least moderately positive ratings on rhyme-related criteria.

We define output collapse as a severe generation failure, including violation of the required tag format, semantically invalid Japanese, excessive mixing of meaningless Romaji or Latin tokens into the lyric text, or repetitive meaningless token sequences. Among the 100 evaluated outputs, 17 outputs generated after GRPO showed collapse phenomena, whereas only two such outputs were observed before GRPO.

Overall, these results are better characterized as a change in the quality distribution rather than a uniform improvement in average quality. GRPO increases the proportion of moderately or highly rated outputs on rhyme-related criteria, but it also increases the number of collapsed outputs. This suggests that GRPO can make rhyme-aware generation more likely in some cases while also making generation less stable.

4.4 Qualitative Analysis

Table 3 presents representative outputs generated by the GRPO-trained model on the evaluation set, including examples with high and low rhyme reward (r_{rhyme}).

In high-reward examples such as 沸かせる (Wakaseru; "To Boil") and 押し通せる (Oshitooseru; "To Push Through"), the final five moras of the two bars form clear end rhymes. The rhyming spans also extend across multiple words, suggesting that the model can generate non-trivial multi-word rhymes rather than relying on simple lexical repetition.

By contrast, a low-reward example such as 花

菖蒲 (Hanashoubu; “Iris”) shows weaker correspondence at the ends of the two bars, making the end rhyme less salient. This is consistent with the quantitative results, in which many outputs still remained below a score of 3 after GRPO.

A high rhyme reward, however, does not necessarily imply that the overall output is well-formed. In the ギックリ腰 (Gikkurigoshi; “Strained Back”) example, the model produces a relatively high-scoring rhyme, but the Romaji output contains corrupted tokens, indicating output collapse.

Overall, these examples suggest that GRPO encourages stronger and sometimes multi-word end rhymes, but does not yet fully stabilize the overall generation quality.

5 Conclusion

This paper proposed a method for training a Japanese rap lyric generation model using GRPO with a reward function that evaluates end rhymes without relying on existing rap lyric corpora. Human evaluation showed that, although average score improvements were limited, GRPO increased the proportion of outputs receiving scores of 3 or higher on rhyme-related criteria. At the same time, low-scoring outputs and several forms of output collapse also increased, indicating that training stability remains a major challenge.

In future work, the trained model can be used to generate a large number of candidate lyrics that can be automatically scored by the reward function, enabling the relatively efficient construction of scored or preference datasets. Using such data, we plan to investigate whether training with methods such as PPO or DPO can improve training stability while preserving the gains in rhyme-related behavior.

Finally, previous studies have noted that Japanese rap rhyming often relies not only on vowel matches but also on phonetically similar consonants (Kawahara, 2007). In future work, we will explore extending the reward design to incorporate consonant similarity in addition to vowel-based rhyme evaluation, with the aim of capturing a broader range of rhyming phenomena found in natural Japanese rap lyrics.

Limitations

This study has several limitations.

First, training stability remains a challenge. As discussed in Section 5, GRPO improves some rhyme-related behaviors but also increases output collapse, indicating that the proposed method does not yet stably improve generation quality.

Second, the current reward function evaluates rhyme quality only through vowel and moraic-nasal similarity. As a result, it does not capture other important rhyme patterns in Japanese rap, such as rhymes involving phonetically similar consonants.

Third, pronunciation extraction for reward computation depends heavily on MeCab. Because Japanese kanji often have multiple context-dependent readings and rap lyrics may include slang or other creative expressions, this dependence may become a bottleneck for learning accurate phonological behavior.

Fourth, the current reward design does not directly capture lyrical interestingness, such as creativity, difficulty, or the unexpectedness of a rhyme.

Fifth, this study evaluates the proposed method only with Sarashina2.2-3B. Therefore, it remains unclear whether the same trends would hold for other Japanese LLMs or larger models with different capabilities of Japanese reading and phonology.

Finally, the current setting is restricted to structured two-bar generation, and it remains unclear whether the same approach would be effective for longer verses or full-song generation.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP23K17641 and JST CREST Grant Number JPMJCR2565. This work used ABCI 3.0, provided by AIST and AIST Solutions, with support from the ABCI 3.0 Development Acceleration Use program.

We would like to express our sincere gratitude to Associate Professor Naho Orita of the Center for English Language Education, Faculty of Science and Engineering, Waseda University, and Professor Shigeto Kawahara of The Keio Institute of Cultural and Linguistic Studies, Keio University, for their valuable advice and discussions on rhyming in Japanese.

References

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4302–4310, Red Hook, NY, USA. Curran Associates Inc.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Shigeto Kawahara. 2007. Half rhymes in japanese rap lyrics and knowledge of similarity. *Journal of East Asian Linguistics*, 16(2):113–144.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. 2016. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 195–204, New York, NY, USA. Association for Computing Machinery.

Enrique Manjavacas, Mike Kestemont, and Folgert Karsdorp. 2019. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 301–310, Tokyo, Japan. Association for Computational Linguistics.

Ryota Mibayashi, Daichi Takamura, and Hitomi Yanaka. 2026. Yomi-bench: A benchmark for evaluating japanese reading understanding in large language models. In *Proceedings of the 32nd Annual Meeting of the Association for Natural Language Processing (NLP2026)*, pages 1929–1934.

Ryota Mibayashi, Takehiro Yamamoto, and Hiroaki Ohshima. 2025. Japanese rhyme generation based on mora similarity and generation probability. In *Information Integration and Web Intelligence: 27th International Conference, IiWAS 2025, Matsue, Japan, December 8–10, 2025, Proceedings*, page 95–111, Berlin, Heidelberg. Springer-Verlag.

Ryota Mibayashi, Takehiro Yamamoto, Kosetsu Tsukuda, Kento Watanabe, Tomoyasu Nakano, Masataka Goto, and Hiroaki Ohshima. 2024. Response verse generation from reverse direction including rhymes in rap battles. *Journal of Information Processing Society of Japan*, 17(2):28–39.

Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. 2020. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 360–373, Dublin, Ireland. Association for Computational Linguistics.

Uraku Oda, Hayato Ogawa, and Daisuke Kawahara. 2025. Unsupervised learning in the style of rap using a large language model. In *Proceedings of the 39th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI 2025)*, pages 2Win567–2Win567.

Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. 2022. PoeLM: A meter- and rhyme-controllable language model for unsupervised poetry generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3655–3670, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Peter Potash, Alexey Romanov, and Anna Rumshisky. 2015. GhostWriter: Using an LSTM for automatic rap lyric generation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1919–1924, Lisbon, Portugal. Association for Computational Linguistics.

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

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Richard J. Savery, Lisa Zahray, and Gil Weinberg. 2020. Shimon the rapper: A real-time system for human-robot interactive rap battles. In *ICCC*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Copyright Division Subcommittee on Legal Systems. 2024. Approach to ai and copyright.

Dekai Wu and Karteek Addanki. 2015. Learning to rap battle with bilingual recursive neural networks. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2524–2530. AAAI Press.

Lanqing Xue, Kaitao Song, Duocai Wu, Xu Tan, Nevin L. Zhang, Tao Qin, Wei-Qiang Zhang, and Tie-Yan Liu. 2021. DeepRapper: Neural rap generation with rhyme and rhythm modeling. In *Proceedings*

of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 69–81, Online. Association for Computational Linguistics.

A Example Prompt for Synthetic Data Generation

For synthetic <theme, 1 bar> generation, we used 5-shot prompts constructed from a pool of 50 manually written theme–lyric pairs created by the first author. Five examples were randomly sampled for each generation trial. Below we show one example prompt for the theme 合気道 (aikidou; "Aikido").

Original Japanese prompt:

合気道をテーマに日本語ラップの歌詞を1小節書いてください。1小節の長さは10~20音程度に収めてください。ただし歌詞を書く際は必ず「歌詞:」ではじめて歌詞以外は何も生成しないでください。例を与えます。例1: 入力: 口論をテーマに日本語ラップの歌詞を1小節書いてください。出力: 歌詞: 反論されたら冗談です，例2: 入力: 靴をテーマに日本語ラップの歌詞を1小節書いてください。出力: 歌詞: 足元には履いてるジョーダン，例3: 入力: 紙をテーマに日本語ラップの歌詞を1小節書いてください。出力: 歌詞: 風に吹かれて飛んでった，例4: 入力: 予想外をテーマに日本語ラップの歌詞を1小節書いてください。出力: 歌詞: 想定解なんて当然ない，例5: 入力: 単語帳をテーマに日本語ラップの歌詞を1小節書いてください。出力: 歌詞: サレンダー載ってない単語帳，ではこのように合気道をテーマに日本語ラップの歌詞を1小節書いてください。

English translation:

Please write one bar of Japanese rap lyrics on the theme of Aikido. Keep the length of the bar to about 10–20 morae. When writing the lyrics, you must begin with "Lyrics:" and generate nothing except the lyrics. We provide examples below.

Example 1: Input: Please write one bar of Japanese rap lyrics on the theme of argument. Output: Lyrics: If someone argues back, I just say it was a joke.

Example 2: Input: Please write one bar of Japanese rap lyrics on the theme of shoes. Output: Lyrics: On my feet, I am wearing Jordans.

Example 3: Input: Please write one bar of Japanese rap lyrics on the theme of paper. Output: Lyrics: Blown by the wind, it flew away.

Example 4: Input: Please write one bar of Japanese rap lyrics on the theme of unexpectedness. Output: Lyrics: Of course, there is no predetermined correct answer.

Example 5: Input: Please write one bar of Japanese rap lyrics on the theme of vocabulary notebook. Output: Lyrics: That vocabulary notebook does not include "surrender."

Now, in the same manner, please write one bar of Japanese rap lyrics on the theme of Aikido.

B Hyperparameter Settings

Table 4 summarizes the main hyperparameter settings used for continued pre-training and GRPO.

C Boxplots of Human Evaluation Scores

Figure 3 shows boxplots of the human evaluation scores before and after GRPO for the three evaluation metrics.

Table 4: Hyperparameters for Continued Pre-training and GRPO

Continued Pre-training		**GRPO**	
Item	Setting	Item	Setting
Distributed learning	Data parallel (8 GPU)	Distributed learning	Data parallel (8 GPU)
Batch size	2 / GPU (global 16)	Batch size	2 / GPU (global 16)
Gradient accumulation	1	Gradient accumulation	1
Epochs	3	Epochs	3
Optimizer	AdamW (Loshchilov and Hutter, 2019)	Rollout samples n	4
Learning rate	2.0×10^{-5}	Rollout temperature	0.9
Weight decay	1.0×10^{-3}	PPO mini-batch	16
Adam β_1, β_2	0.9, 0.999	KL coef	0.001
Adam ϵ	1.0×10^{-8}	Entropy coeff	0
Warmup ratio	0.03	–	–

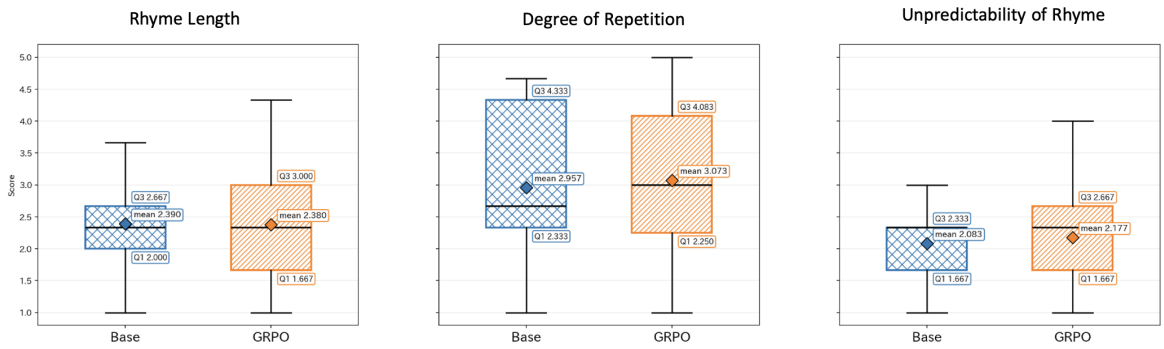

Figure 3: Boxplots of human evaluation scores before and after GRPO for the three evaluation metrics.