# GRPO を用いた日本語ラップの歌詞生成モデルの構築

小川隼斗　河原大輔
早稲田大学理工学術院
{cookie3120@ruri,dkw@}waseda.jp

## 概要

ラップは HIPHOP 文化に根差し, リズムに合わせて韻を踏むこと（押韻）を特徴とした歌唱法である. 本論文では, 強化学習によって押韻を学習した大規模言語モデルによる, 日本語ラップの歌詞の生成手法を提案する. 提案手法では, モデルが生成する 2 小節の歌詞に対して韻を評価する報酬関数を定義し, 強化学習の一種である GRPO を行い, 日本語で押韻する能力の向上を目指す. 評価の結果, 一部に出力崩壊も観測された一方で, 押韻能力の向上することを示した.

## 1　はじめに

ラップという歌唱法が 1980 年代に日本に持ち込まれて以降, その独特なリズムと韻を踏む表現技法は若者文化を中心に広く浸透し, 現在では日本でも音楽の主要なジャンルの一つとして定着している. ラップにおいて最も重要な要素の一つが「押韻」である. 押韻とは, 単語やフレーズの母音の響きを合わせることで, リズム感を生み出す技法である. 例えば,「美学（bigaku）」と「磨く（migaku）」のように, 小節末や小節内で母音（i-a-u）を一致させることでリズム感が生まれる. 一方で, 意味の通った歌詞を保ちながら押韻することは容易ではなく, ラップの歌詞の創作は非常に難しい生成タスクである.

近年, 大規模言語モデル（LLM）の飛躍的な発展により, 詩や小説の創作など, 従来は人間の高度な創造性が必要とされたタスクにおいても自然なテキスト生成が可能となりつつある. これにより, ラップの歌詞を含む創作タスクに対しても, 人と LLM による共作を通じて新たな芸術表現が生まれることが期待されている. しかし日本語は, 漢字とかな文字が混在し, さらに漢字が複数の読みを持つといった特徴があるため, LLM がテキストの表層情報のみから正確な韻律構造を把握することは困難である. さらに生成物は既存著作物との類似性・依拠性が認められる場合に権利侵害となり得るため [1], 既存アーティスト作品への依存を避けた学習・生成手法が求められる.

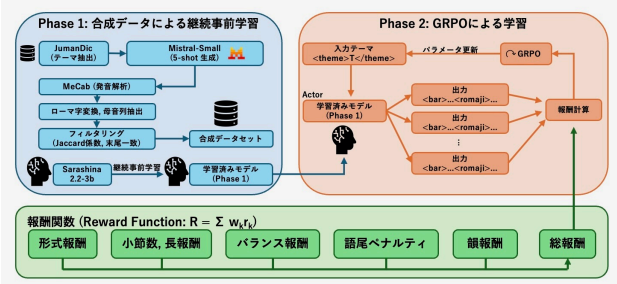


**図 1**　提案手法の全体概要.

本研究では, 既存のラップ歌詞に依存せずに日本語ラップの押韻能力, とくに小節末尾間で韻を踏む脚韻の生成能力を高めることを目的として, GRPO [2] に基づく強化学習を適用する. GRPO は, 同一プロンプトに対する複数出力の相対評価を用いて報酬最適化を行えるため, 教師データなしで形式の遵守と押韻能力の向上を同時に促せる.

## 2　関連研究

本節では, LLM に対する強化学習手法と, 韻を含む歌詞の生成に関する研究について述べる.

### 2.1　強化学習を用いた LLM の学習

従来の LLM における強化学習の手法としては, PPO（Proximal Policy Optimization）[3] や DPO（Direct Preference Optimization）[4] が主流であった. PPO は, 生成されたテキストに報酬モデルを用いて報酬を与え, その報酬を最適化するように学習する手法である. DPO は報酬モデルを学習せず, 選好データから直接学習する手法である. 近年では, 新たなアプローチとして DeepSeekMath [2] で提案されている GRPO（Group Relative Policy Optimization）が注目されている. GRPO は, 報酬モデルの学習が不要であり, 1 つの入力に対して学習対象のモデルが生成する複数の出力の相対的な評価に基づいて学習する手法である.

本研究では, GRPO を用いることで, 押韻という制約を持つテキストの生成能力の向上を試みる.

## 2.2 韻を含む歌詞の生成

韻を含む歌詞の生成手法として, 英語では, 生成テキスト末尾同士の単語の母音列が一致する単語に置換することで押韻テキストにする Rapformer [5] や既存のラップの歌詞をもとに新しいラップの歌詞を生成する GhostWriter [6] が提案されている.

日本語では, 三林ら [7] が BERT2BERT [8] を用い, 辞書検索と逆向き生成により押韻する語を選択する手法を提案している. これに対して我々の提案手法は, 単語またはトークン単位でなく指定した長さだけ母音列の一致度を見るので, 「近い腕（tikaiude）」と「今見る目（imamirume）」のような単語単位にとどまらず, 複数の単語から成る押韻が期待できる. また, 我々 [9] は structure-aware training [10] を用いて, 既存歌詞の教師データを用いずに形式を学習し, 母音指定により押韻を制御する手法を提案している. 本論文の提案手法は, この我々の先行研究に対して, 既存の歌詞を学習に用いないという方針は共通している一方で, 押韻に使用する母音の指定が不要であるという点が大きく異なる.

# 3 日本語ラップ生成モデルの構築

本研究では, 既存楽曲データに依存せず, 日本語として自然で脚韻を含む 2 小節歌詞を生成するモデルを, 強化学習の一種である GRPO を用いて構築する. 強化学習において, 学習開始直後に出力が不安定になりやすい「コールドスタート」と呼ばれる問題がある. 本研究ではこの不安定化を避け, かつ特殊な入出力形式へ適応させるため, DeepSeek-R1 [11] を参考に, 強化学習の前に合成データによる継続事前学習を行う. 提案手法の全体概要を図 1 に示す.

以下では, まず報酬計算の基盤となる入出力形式を定義し, 次に合成データの構築および継続事前学習の手順について述べる. 最後に, GRPO を用いた学習において, どのような報酬関数を用いて韻や生成品質を最適化するかについて述べる. なお, 学習対象のベースモデルには, Sarashina2.2-3b[1] を採用する.

## 3.1 入出力形式

本手法において, 強化学習の報酬として韻の質を計算するためには, モデルが生成した歌詞の小節末尾を正確に把握する必要がある. 本研究では, 入出力の形式を次のように構造化する.

**入力：**

<theme>$T$</theme>

**出力：**

<bar>$B_1$</bar><romaji>$R_1$</romaji>
<bar>$B_2$</bar><romaji>$R_2$</romaji>

ここで, $T$ はラップのテーマ, $B_i$ は第 $i$ 小節の歌詞, $R_i$ はその発音（ローマ字表記）を表す. この独自の出力形式にモデルを適応させるため, GRPO の前に継続事前学習を行う.

## 3.2 合成データによる継続事前学習

### 3.2.1 合成データセットの構築

継続事前学習に使用する脚韻付き合成データを, LLM による生成と発音解析に基づいて構築する. 各データはそれぞれが異なるテーマを持つ 2 小節 $(B_1, B_2)$ からなり, 両小節の末尾 5 モーラ（母音・撥音列）が一致するように作成する. なお $(B_1, B_2)$ のテーマは異なるため, 歌詞としての整合性は要求しない. これにより, 入出力形式（3.1 節）への適応と, コールドスタートの回避を図る.

まず, JumanDic [2] の名詞・動詞からテーマ候補を抽出し, LLM を用いて 1 小節の歌詞を生成して〈テーマ, 1 小節〉ペアを得る. 次に, MeCab [12] により各小節の発音を解析してローマ字化し, 母音および撥音（a, i, u, e, o, N）の列を抽出する. そして, 末尾 5 モーラが一致する小節同士を同一グループとして扱い, グループ内から 2 小節をペア化することで脚韻ペアを作成する.

さらに, 同一表現の単純反復を避けるため, 小節末尾 5 文字に基づく表層類似度を用いて重複を抑制し, 最終的に 36,766 件の 2 小節ペアを得た. 本合成データを学習/検証に 7:3 で分割して継続事前学習に用いた. 生成条件とフィルタリング手順の詳細は付録 A に示す.

### 3.2.2 継続事前学習

Sarashina2.2-3b に対し, 構築したデータセットで 3 エポックの継続事前学習を行った. 学習設定の詳細は付録 C に記述する.

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

2） https://github.com/ku-nlp/JumanDIC

## 3.3 GRPO による学習

### 3.3.1 報酬関数の設計

本研究では, GRPO におけるスカラー報酬 $R$ を, (1) 入出力フォーマットおよび 2 小節構造の妥当性, (2) 表層的な繰り返しの少なさ, (3) 母音列に基づく脚韻の質を統合的に評価するよう設計する.

具体的には, 要素報酬の集合を

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

とし, 総報酬を

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

で定義する. ここで $r_k$ は各観点の要素報酬, $w_k$ はその重みである. 実装では, 入出力フォーマットと小節構造の妥当性を確認する 4 要素 $(r_{\text{format}}, r_{\text{num\_bars}}, r_{\text{len\_range}}, r_{\text{len\_balance}})$ には $w_k = 1.0$ を, 残りの押韻に関する 2 要素 $(r_{\text{suffix\_penalty}}, r_{\text{rhyme}})$ には $w_k = 4.0$ を与え, 後者を相対的に重視する.

入出力フォーマットおよび小節構造の妥当性は, 次の 4 条件で判定する. まず, 正規表現により <bar>...</bar><romaji>...</romaji> のペアが 3.1 節で示した形式に完全一致する場合に $r_{\text{format}} = 1.0$（それ以外は −1.0）とする. 次に, <bar>...</bar> で囲まれた箇所の数が 2 である場合に $r_{\text{num\_bars}} = 1.0$（それ以外は −1.0）とし, タグ構造は正しいが小節数が異なる出力を罰する. さらに, 各小節の長さは表層文字列ではなく MeCab により抽出した発音列長 $L_i$ で評価し, $L_{\min} \le L_i \le L_{\max}$ を全小節が満たす場合に $r_{\text{len\_range}} = 1.0$（それ以外は −1.0）とする. 最後に, 2 小節間のリズムの偏りを抑えるため, 発音長の差 $\Delta L = |L_1 - L_2|$ が許容差 $\Delta L_{\max}$ 以下であれば $r_{\text{len\_balance}} = 1.0$（それ以外は −1.0）とする. これら 4 要素それぞれを評価することで, 指定した出力形式に従い, 適度な長さとバランスを持つ 2 小節出力を安定に促す. なお, 実装では, $L_{\min} = 8$, $L_{\max} = 16$, $\Delta L_{\max} = 4$ と設定した. 次に, 表層的な繰り返し抑制と脚韻品質を評価する $r_{\text{suffix\_penalty}}$ および $r_{\text{rhyme}}$ の計算手法について記述する.

**小節末尾類似度ペナルティ（$r_{\text{suffix\_penalty}}$）** 同一表現の単純な繰り返しによって見かけ上の韻だけを稼ぐことを防ぐため, 小節末尾の表層的な類似度に基づくペナルティを導入する. 具体的には, 小節 $B_1, B_2$ の表層文字列の末尾 5 文字の文字集合を $A_1, A_2$ とし Jaccard 係数 $J(A_1, A_2)$ を用い,

$$r_{\text{suffix\_penalty}} = -J(A_1, A_2)$$

と定義する. これにより小節末尾の表現が近いほど負の報酬が大きくなり, 表層的な反復への収束を抑制する.

**韻報酬（$r_{\text{rhyme}}$）** 韻の質そのものは, 母音列に基づく脚韻類似度として評価する. 各小節について MeCab により発音列を抽出し, それをカタカナからローマ字に変換した列を $R_i$ とする. ここから母音および撥音のみを取り出した列

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

を構成し, 小節末尾側 $k = \min(L_1, L_2, 5)$ モーラに着目する. 2 小節の末尾母音列 $V_1', V_2'$（いずれも長さ $k$）に対して, 小節末尾側を強調する位置依存の重みを導入した編集距離 $d_{\text{suffix}}(V_1', V_2')$ を計算し, これを最大コスト $d_{\max}$ で正規化することで,

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

と定義する. ここで用いる編集距離は, 小節末尾側の挿入・削除・置換ほど大きな重みを与える重み付き編集距離であり, 置換コストには母音間の音声的近さを反映させている. 具体的な重み付けの設計は付録 B に示す. また, 発音の抽出に失敗した場合や小節数が 2 でない場合には, 韻として不適切な出力であるとみなし, -1 を返すように実装している.

### 3.3.2 学習設定

学習に用いるテーマは JumanDic の名詞・動詞からランダムに 8,921 語（約 40%）を抽出し, 学習/検証/評価に 7:2:1 で分割した. 学習率は $2 \times 10^{-6}$, $1 \times 10^{-6}$, $5 \times 10^{-7}$ の 3 条件でスイープし, その他のハイパーパラメータは付録 D に示す.

**学習率スイープとチェックポイント選択** 学習率が大きい条件では総報酬が一時的に上昇する一方, 学習途中または終盤に出力が崩壊し, プロンプトに依存しない反復的生成へ収束する例が観測された. 崩壊例を次に示す.

**入力：**
<theme>焼ける</theme>

**出力：**
<bar>炎踊る夜飛ぶ夜</bar>
<romaji>honoooodo. . . kuyoroku</romaji>
<bar>コゲトクトクトクトクトク</bar>
<romaji>kogeto. . . kutokoku</romaji>

これらを踏まえ, チェックポイントは総報酬のみで選択せず, 検証用データでの出力において (i) 形式崩壊が少ない, (ii) 多様性が極端に低下しない, (iii) 韻報酬が上昇している, の 3 条件を満たす区間を優先した. その結果, 学習率 $5 \times 10^{-7}$ の Step 117 を採用し, 4 節の評価に用いた.

# 4 評価

## 4.1 定量評価

提案手法によって脚韻生成能力が向上しているかを, 日常的に日本語ラップを聴く 20 代 3 名の評価者による絶対評価で確認する. 比較対象として GRPO を適用する前のモデル（Base）を用いる. 評価用データからランダムに抽出した 100 件のテーマに対して, 各モデルが生成した 2 小節の歌詞を評価した.

### 4.1.1 評価指標

評価指標は, (i) 韻の長さ, (ii) 重複度合い, (iii) 韻の思いつきやすさの 3 つの観点（各 1–5 点）から構成される.

**韻の長さ（1–5）** 末尾で韻として成立している区間のモーラ長で評価する（5: ≥5 モーラ, 4: 4 モーラ, 3: 2–3 モーラ, 2: 1 モーラ, 1: 不成立/崩壊）. 母音が完全一致しない場合でも評価者が韻として成立すると判断すれば数え, 長さは短い側に合わせる.

**重複度合い（1–5）** 韻区間における表層文字列の重複割合で評価する（5: 無し, 4: 少, 3: 中, 2: 多, 1: ほぼ同一）. また, 韻の長さが 1 点の出力は評価不能として本指標も 1 点とする.

**韻の思いつきやすさ（1–5）** 脚韻の意外性・創造性を評価する（5: とても思いつきにくい ↔ 2: とても思いつきやすい）. また, 韻の長さまたは重複度合いが 1 点の出力は押韻ができていないとみなし, 評価不能として 1 点とする.

### 4.1.2 評価結果

図 2 に, 人手評価における GRPO 前後の平均スコアの分布を示す. GRPO 後は韻の長さの評価の分布が上下に広がり, 高得点側の増加と同時に低得点側の増加も確認できる. これは, 脚韻の質が改善した生成が増えた一方で, 時折確認された出力の崩壊が要因と考えられる. また, 重複度合いの評価に大きな変化は確認されなかったが, 韻の思いつきやすさについてはわずかに高得点が増加していることが確認できる.

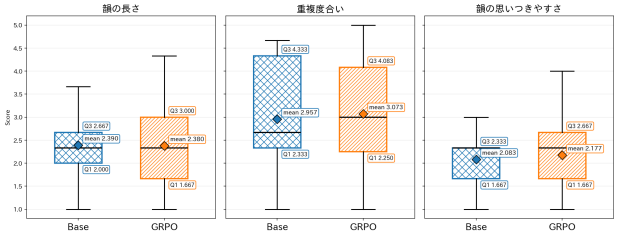


**図 2** 人手評価における GRPO 前後のスコア分布.

**表 1** 提案手法によるモデルの出力例

| テーマ | 出力（$B_1$ & $B_2$） | $r_{\text{rhyme}}$ |
|---|---|---|
| 沸かせる | 熱い波動沸き上がる<br>心に火を焚き放つ | 1.000 |
| 押し通せる | どんな壁もブチ破れ<br>自分の道を踏み外せ | 1.000 |
| ギックリ腰 | 体が動かない状態<br>ベッドにこびたり tai | 0.938 |
| 花菖蒲 | 花菖蒲咲いて夏来た<br>野に咲く夢の色よ | 0.224 |

## 4.2 定性評価

定量評価では, GRPO 後に高得点側の出力が増える一方で, 低得点の出力も増える傾向が確認された. 本節ではこの傾向を具体例で示すため, 評価用データでの GRPO 後のモデルの出力から, 韻報酬 $r_{\text{rhyme}}$ が高い出力と低い出力の例を, 表 1 に示す.

$r_{\text{rhyme}}$ が高い「沸かせる」, 「押し通せる」のような例では, 小節末尾間が脚韻として成立しており, さらに複数単語にまたがる押韻も確認できる. 一方で, $r_{\text{rhyme}}$ が低い例では, 小節末尾の対応が弱く脚韻が成立しにくい. また, $r_{\text{rhyme}}$ が高い出力であっても, 「ギックリ腰」の例のようにローマ字表記の破綻が混在する場合があり, 出力の崩壊が起こっていることが確認できる.

# 5 おわりに

本論文では, 既存のラップ歌詞に依存せずに脚韻を評価する報酬関数を設計し, GRPO により日本語ラップ歌詞生成モデルを学習する手法を提案した. 人手評価では, GRPO 後に高得点側の出力が増加し押韻能力の向上が確認できた一方, 低得点の出力や形式・ローマ字表記の破綻を含む崩壊例も観測され, 学習の安定性に課題が残った. 今後は, 本モデルによりスコア付き/選好データを比較的容易に構築できるようになったため, そのようなデータセットを整備する. さらに, 構築したデータを用いた PPO・DPO による学習の安定化と押韻能力の更なる向上と, 生成小節数を増やした条件への拡張を検討する.

## 謝辞

本研究は JSPS 科研費 JP23K17641 および JST CREST JPMJCR2565 の支援を受けた. また, 本研究は, 産総研及び AIST Solutions が提供する ABCI 3.0 を「ABCI 3.0 開発加速利用」の支援を受けて利用した.

そして, 本研究の過程で, 日本語における押韻に関して議論を通し, ご助言いただいた早稲田大学理工学術院 英語教育センター 折田奈甫 准教授, および慶應義塾大学言語文化研究所 川原繁人 教授に深く感謝申し上げます.

## 参考文献

[1] 文化審議会著作権分科会法制度小委員会. Ai と著作権に関する考え方について, 2024. https://www.bunka.go.jp/seisaku/bunkashingikai/chosakuken/pdf/94037901_01.pdf.

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

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

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

[5] Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In Brian Davis, Yvette Graham, John Kelleher, and Yaji Sripada, editors, **Proceedings of the 13th International Conference on Natural Language Generation**, pp. 360–373, Dublin, Ireland, December 2020. Association for Computational Linguistics.

[6] Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: Using an lstm for automatic rap lyric generation. In **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**, pp. 1919–1924, 2015.

[7] 三林亮太, 山本岳洋, 佃洸摂, 渡邉研斗, 中野倫靖, 後藤真孝, 大島裕明. ラップバトルにおける逆向き生成によるライムを含む返答バース生成. 情報処理学会論文誌データベース TOD, Vol. 17, No. 2, pp. 28–39, 2024.

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

[9] 織田宥楽, 小川隼斗, 河原大輔. 教師なし学習によるラップの形式の学習. 人工知能学会全国大会論文集第 39 回 (2025), pp. 2Win567–2Win567. 一般社団法人人工知能学会, 2025.

[10] Aitor Ormazabal, Mikel Artetxe, Manex Agirrezabal, Aitor Soroa, and Eneko Agirre. Poelm: A meter-and rhyme-controllable language model for unsupervised poetry generation. **arXiv preprint arXiv:2205.12206**, 2022.

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

[12] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to Japanese morphological analysis. In Dekang Lin and Dekai Wu, editors, **Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing**, pp. 230–237, Barcelona, Spain, July 2004. Association for Computational Linguistics.

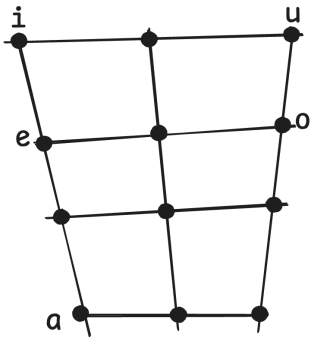


**図 3**　日本語母音の IPA チャート.

# A　合成データセット構築の詳細

本節では, 3.2.1 節で用いた合成データの生成条件と, 脚韻ペア作成・フィルタリング手順をまとめる. まず, JumanDic[3] に含まれる名詞・動詞からテーマ候補を抽出し, 各テーマ $T$ に対して Mistral-Small-3.1-24B-Instruct-2503[4] で 1 小節 $B$ を生成して〈テーマ, 1 小節〉ペアを得た. 生成は 5-shot プロンプトで行い, ショット組合せを変えて複数回生成することで, 生成失敗・空出力を除外後に 88,590 件の〈テーマ, 1 小節〉ペアを得た.

次に, 各小節 $B$ に対して MeCab [12] により発音を解析し, ローマ字表記へ変換した. 得られたローマ字列から母音・撥音 $\{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}, \mathrm{N}\}$ のみを抽出して母音列 $V(B)$ を作成する. その後, 母音列の末尾 5 文字が一致する小節同士を同一グループにまとめた. グループ内で 2 小節ペアを作る際には, 末尾表現の単純反復への収束を避けるため, 表層類似度に基づく選択を導入した. 小節 $B_1, B_2$ の表層文字列の末尾 5 文字の文字集合を $A_1, A_2$ とし, Jaccard 係数

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

を計算する. $J$ が最小となるペアをグループ内から順次選択し, 一度ペアに用いた小節は以降の探索から除外する貪欲法により, 母音列の末尾 5 文字が一致する 2 小節ペアを 42,335 件得た.

最後に, 表層重複の強いペアを除外するため, $J \le 0.2$ を満たすペアのみを残し, 最終的に 36,766 件を合成データとして採用した. 採用データは学習/検証に 7:3 で分割し, 入力は `<theme>`$T_1, T_2$`</theme>` として, テーマが異なる 2 小節を 3.1 節で示した形式に整形し, 継続事前学習に用いた.

# B　韻報酬における編集距離の設計

韻報酬 $r_{\mathrm{rhyme}}$ では, 2 つの末尾母音列に対する位置依存重み付き編集距離 $d_{\mathrm{suffix}}$ を用いる. ここでは挿入・削除と置換のコスト設計を述べる.

**挿入・削除**　母音列長を $L$, 位置を $p \in \{0, \ldots, L-1\}$ とするとき, 基底コストを

$$c_{\mathrm{ins/del}}(p) = \begin{cases} 1 & (p = 0 \text{ または } p = L-1) \\ 3 & (\text{それ以外}) \end{cases}$$

と定める. これに小節末尾側を強調する位置重み

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

を掛け, 実際の挿入・削除コストを $\tilde{c}_{\mathrm{ins}}(p) = \tilde{c}_{\mathrm{del}}(p) = c_{\mathrm{ins/del}}(p)\, w(p, L, \alpha)$ として用いる.

**置換**　母音集合を $\mathcal{V} = \{\mathrm{a}, \mathrm{i}, \mathrm{u}, \mathrm{e}, \mathrm{o}\}$ とし, 図 3 に示す日本語母音の IPA チャートに基づき, 母音間の置換コストを $5 \times 5$ の対称行列 $C$ で与える：

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

母音 $v_p, v_q \in \mathcal{V}$ の置換コストは $c_{\mathrm{sub}}(v_p, v_q) = C_{pq}$ とし, 撥音 N と母音の組には一様に $c_{\mathrm{sub}}(\mathrm{N}, v) = c_{\mathrm{sub}}(v, \mathrm{N}) = 4$ を与える. 母音・撥音以外の文字同士の置換は $c_{\mathrm{sub}}(x, y) = 2$ とする. これらにも挿入・削除と同じ位置重み $w(p, L, \alpha)$ を掛け合わせることで, 小節末に近い位置での差異ほど $d_{\mathrm{suffix}}$ に強く反映されるよう設計している.

**表 2**　継続事前学習のハイパーパラメータ

| 項目 | 設定 |
|---|---|
| 分散学習 | Data parallel（8 GPU） |
| Batch size | 2 / GPU（global 16） |
| Gradient accumulation | 1 |
| Optimizer | AdamW |
| Learning rate | $2.0 \times 10^{-5}$ |
| Weight decay | $1.0 \times 10^{-3}$ |
| Adam $\beta_1, \beta_2$ | 0.9, 0.999 |
| Adam $\epsilon$ | $1.0 \times 10^{-8}$ |
| Warmup | ratio 0.03 |
| Epochs | 3 |

# C　継続事前学習の学習設定

継続事前学習の際に設定した主要なハイパーパラメータを表 2 に記述する.

# D　GRPO の学習設定

GRPO の際に設定した主要なハイパーパラメータを表 3 に記述する.

**表 3**　GRPO のハイパーパラメータ

| 項目 | 設定 |
|---|---|
| 分散学習 | Data parallel（8 GPU） |
| Rollout samples $n$ | 4 |
| Rollout sampling temperature | 0.9 |
| Batch size | 2 / GPU（global 16） |
| PPO mini-batch | 16 |
| Gradient accumulation | 1 |
| KL coef | 0.001 |
| Entropy coeff | 0 |
| Epochs | 3 |

3)　https://github.com/ku-nlp/JumanDIC
4)　https://huggingface.co/mistralai/Mistral-Small-3.1-24B-Instruct-2503