教師なし学習によるラップの形式の学習

Unsupervised Learning in the Style of Rap using a Large Language Model

織田　宥楽 *1　　小川　隼斗 *1　　河原　大輔 *1
Uraku Oda　　Hayato Ogawa　　Daisuke Kawahara

*1早稲田大学
Waseda University

ラップとは、リズムに合わせて韻を踏むことを多用する歌唱法である。本研究では、教師なし学習によってラップの形式を言語モデルで学習することを目的とする。生コーパスの各文を小節単位で区切り、発音とテーマを付与したデータセットを構築し、これを用いてラップの形式を学習する。付与する発音の種類、表記、学習に用いるコーパスを変化させ、生成結果を比較した。本手法では、形式的で文語的なコーパスよりも、多様で口語的なコーパスを用いて学習したモデルがラップの形式を学習するのに効果的であることがわかった。

1. はじめに

ラップとは、リズムに合わせて韻を踏むことを多用する歌唱法であり、ヒップホップと呼ばれる文化の 4 大要素 (ラップ、DJ、ブレイクダンス、グラフティ) のうちの 1 つである。ヒップホップは 1970 年代にアメリカのニューヨークで誕生し、貧困や差別などの社会背景の中、発展していった。1980 年代、日本にヒップホップが持ち込まれ、その後日本国内でもヒップホップ、およびラップが広まっていった。ラップの大きな特徴は押韻である。押韻とは、言葉の母音を合わせることで、心地よいリズムを生み出す技法である。日本の伝統的な言語芸術である俳句や和歌などは、言葉を一定の音数で合わせてリズムをとることが特徴である一方で、音の響きを合わせる技法はあまり見られない。そのため、押韻によってリズムを生み出す技法は、俳句や和歌などの音数を合わせる技法と比べ、日本ではあまり馴染みがないといえる。

ラップや押韻に慣れていない人を助け、より豊かな作品や表現を生み出すための手段として、ラップの歌詞の自動生成が考えられる。現在、日本語ラップ歌詞の生成の研究は少なく、ChatGPT などの大規模言語モデルでは、正確に韻を踏んだ日本語ラップ歌詞の生成は難しい。ラップのコーパスを用いた機械学習が考えられるが、公開されている日本語ラップのコーパスは存在しない。また、これを作成するには、Web スクレイピングやラップ動画の文字起こしなどの処理が必要となり、多くの労力と時間を要する。本研究では、日本語ラップ歌詞の自動生成に向けて、教師なし学習でラップの形式を学習する。具体的には、生コーパスに発音や文のテーマを付与し、ラップの形式を学習させる。これによって、指定されたテーマに沿ったラップ形式の歌詞の生成が可能になる。

2. 関連研究

ラップの歌詞の生成に関する既存の研究として、ラップに関連したラップバトルや押韻の研究、また川柳や詩などの他の言語芸術の自動生成に関する研究について述べる。

2.1 ラップ歌詞生成に関する研究

海外のラップ歌詞生成に関する研究として、次のような先行研究がある。Nikolov らは、ニュース記事などの任意のテキストをラップ調に変換するラップ歌詞生成モデルを提案している [Nikolov 20]。BERT を用いて単語の言い換えを行う手法によって、生成された歌詞が韻を踏んでいる割合が 10%向上することが示された。Potash らは、LSTM を用いて、ラップの歌詞を自動生成する手法を提案している [Potash 15]。特定のラッパーのスタイルを模倣しながら新しい歌詞を生成することに焦点を当て、従来のモデルよりも高い評価を示した。

三林らは日本語のラップバトルを対象に、BERT2BERT モデルを用いてラップ文を生成し、BERT を用いて適切に並び替える手法を提案している [三林 24]。特に、文末の言葉で韻を踏むために、従来の文頭から順方向に生成する方法ではなく、文末から逆方向に文を生成するアプローチを採用している。独自に作成したラップバトルコーパスを使用して学習し、生成したラップ文を人手で評価する実験が行われ、提案手法の有効性が確認された。

本研究では、ラップバトルにおける返答バースではなく通常のラップ歌詞の生成を目的とし、指定されたテーマに沿ったラップ歌詞生成を行う。また、ラップコーパスの作成や人手評価など、コストが高い工程を省略し、ラップの形式を学習することを目指す。

2.2 他の言語芸術の自動生成の研究

Ormazabal らは、structure-aware training という手法を用いて、詩のコーパスを使わずに、指示された長さや韻律の詩を生成するモデル PoeLM を提案している [Ormazabal 22]。実験により、スペイン語とバスク語で高品質な詩が生成できることが示された。

太田らは、end-to-end で川柳を生成するモデルを提案している [太田 24]。おもしろい川柳の生成のため、PoeLM の学習を参考にした音数・テーマの学習、ファインチューニングを使用した川柳の内容の学習、強化学習を使用した川柳のおもしろさの学習、の三段階の学習を行っている。実験により、提案手法は音数，川柳としての適否，川柳としておもしろさの 3 項目すべてにおいて，ベースラインを上回る結果が示された。

本研究では PoeLM で用いられた structure-aware training を参考に、既存のコーパスに日本語の発音やテーマを付与することで、指定したテーマや発音（韻）に従ったラップの歌詞の生成を試みる。

連絡先: 織田　宥楽，早稲田大学　基幹理工学研究科，
urakuodacchi@akane.waseda.jp

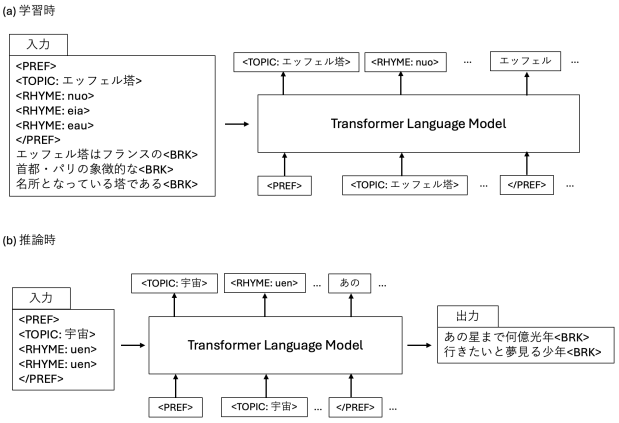

図 1: 本研究における structure-aware training

3. 提案手法

通常の言語モデルは、文字列をベースに学習しており、言葉の発音を直接的に学習しているわけではない。特に漢字は、文字列が発音ではなく意味を表しているため、スペルが発音と直結しているアルファベットなどと比べて、言語モデルが発音を正確に認識するのは難しい。ラップの特徴である押韻を言語モデルで正確に行うには、発音を学習させる必要がある。また、ラップは韻を踏むだけではなく、歌詞として意味を持ち、伝えたいテーマを持つ必要がある。そのため、発音だけでなく、一つのテーマに対して歌詞を生成するという構図を明示的に学習する。

本研究では、関連研究で述べた structure-aware training に基づき、生コーパスを用いて教師なし学習によって発音とテーマを学習する。本研究における structure-aware training の概要を図 1 に示す。

3.1 データセットの作成

学習に用いる生コーパスとして、日本語 Wikipedia コーパス [*1] と青空文庫のコーパス [*2] の 2 種類を用いる。

日本語 Wikipedia コーパスは、Wikipedia の各記事のタイトル、セクション名、記事の文章から構成されている。本研究では、セクション名をラップのテーマ、記事の文章の先頭文をラップの歌詞に見立て、学習用のコーパスとして整形する。事例数は約 215 万件であり、学習に 90%、テストに 10%を用いる。

青空文庫コーパスは、整形された青空文庫の作品のテキストと、出版社やタイトル、著者名などのメタデータから構成されている。作品のタイトルをラップのテーマ、作品のテキストの各文をラップの歌詞に見立て、学習用のコーパスとして整形する。事例数は約 125 万件であり、学習に 90%、テストに 10%を用いる。

2 種類のコーパスに対して発音を付与する。ラップの歌詞は、小節と呼ばれる単位に分割できる。1 小節は 4 拍で構成されており、1 拍に 2〜4 音の歌詞がつくことが多いため、本研究では、1 小節を約 8〜16 音のまとまりとする。歌詞に見立てた文を、1 小節に収まる程度の音数に分割し、発音を付与する。

∗1 https://huggingface.co/datasets/singletongue/wikipedia-utils

∗2 https://huggingface.co/datasets/globis-university/aozorabunko-clean

表 1: 発音の変換

変換前	変換後
ああ	あー
いい	いー
ううう	うー
えい, ええ	えー
おう, おお	おー

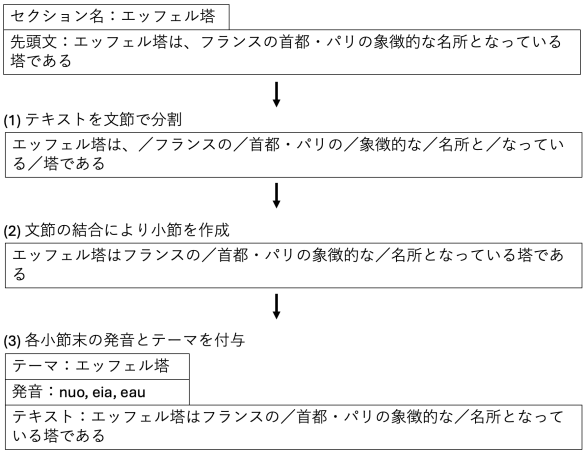

図 2: 生コーパスに発音とテーマを付与する方法

ラップでは小節の末尾の言葉で韻を踏むことが多いため、各小節の末尾の 3 音の発音を取り出す。具体的には、文の分割は文節を基に行う。構文解析器 KNP[*3] を用いて文節列を取得し、1 小節が 10 音以上になるように文節を結合する。単語の発音は形態素解析器 Juman++[*4] によって取得した読みを用いる。発音の付与に関して、より自然な発音となるように、表 1 に従って変換しコーパスに付与する。

発音の付与は、次に説明する押韻タイプと押韻表記の 2 種類の組み合わせで計 4 種類のデータセットを作成する。データセット整形の流れの例を図 2 に示す。

押韻タイプ: 母音のみ・子音+母音 押韻は、母音を合わせることが基本となるため、まず母音のみの発音を付与したデータセットを作成する。Kawahara の研究によれば、既存の日本語ラップの押韻において、単に母音を合わせるだけでなく、音声学的に類似した子音のペアも使われやすいことが示唆されている [Kawahara 07]。そのため、付与する発音を母音のみの場合だけでなく、子音も含めた場合 (子音+母音) の 2 通りのデータセットを作成する。この 2 通りの押韻を押韻タイプと呼ぶ。

押韻表記: ひらがな・ローマ字 発音をひらがなで付与した場合とローマ字で付与した場合の 2 通りのデータセットを作成する。日本語のモデルを用いて学習を行うため、ひらがなによって発音を表記したデータセットを作成する。一方、子音に関する音素の情報を取得するにはローマ字による表記が適しており、母音の a, i, u, e, o も明確にわかるため、ローマ字によって発音の表記したデータセットも作成し、ひらがなの場合と比較する。この 2 通りの表記を押韻表記と呼ぶ。

3.2 発音に基づくラップの形式の学習

作成したデータセットを図 1 のように整形し、事前学習済みの大規模言語モデルに入力して継続学習を行う。`<PREF>`,

∗3 https://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP

∗4 https://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN%2B%2B

表 2: 学習に使用したハイパーパラメータ

ハイパーパラメータ	japanese-gpt2-medium	llm-jp-3-1.8b
learning rate	6e-5	6e-5
batch size	128	16
epoch	5	5
weight decay	0.2	0.2

</PREF>, <BRK>, TOPIC, RHYME は特殊トークンとしてトークナイザの語彙に加える。テキストに付与したテーマと発音は、<PREF>, </PREF>内に記載する。テキストは、<BRK>を用いて区切り、<TOPIC: >内にはテーマを、<RHYME: >内には各小節の発音を記載する。学習のタスクは CLM (Causal Language Model) で行う。推論時には<PREF>から</PREF>までを入力し、以降のラップ歌詞の部分を生成する。

4. 実験

4.1 実験設定

3.1 節で作成した 4 種類のデータセットそれぞれを用いて 3.2 節の学習を行う。学習のベースモデルには、主に日本語大規模言語モデル llm-jp/llm-jp-3-1.8b[*5]、比較用に日本語 GPT-2 rinna/japanese-gpt2-medium[*6] を使用する。学習時に使用したハイパーパラメータを表 2 に示す。評価は、テスト用データセットから 1,000 件のテーマと押韻表記の組み合わせを選び、これらを基にラップの歌詞を生成することによって行う。生成された歌詞は、<BRK>で小節ごとに区切られ、指定された発音が小節の末尾で再現されているかを確認し再現精度を計算する。また、テストデータで指定する発音は、データセットそのままで各小節で異なる発音を指定した場合 (4.2 節) と、ラップの歌詞のように全て同じ発音を指定した場合 (4.4 節) でそれぞれ評価する。

4.2 メイン実験結果

発音とテーマを付与した Wikipedia コーパスで llm-jp-3-1.8b を継続学習した場合のエポックごとの精度を表 3 に示す。ひらがな・ローマ字のどちらを押韻表記として用いた場合でも、Epoch 5 において、指定した発音のテキストが小節末に 80 %以上の精度で生成された。多くの場合でひらがなによる発音の表記が若干優れていた。生成例を以下の (1), (2) に示す。【】内は入力のテーマ、続く [] 内は指定された発音を示し、その後に生成されたテキストを記載する。

(1) 【ボードゲーム】[' ううお', ' んおお', ' えーう']
ボードゲームでは「勝負」と<BRK>
「死」をテーマにしたイベントを<BRK>
開催している<BRK>

(2) 【歴史】['aii', 'e-u']
現在は国の重要文化財に<BRK>
指定されている<BRK>

<BRK>で区切られた小節において、小節の末尾で指定された母音を持つテキストが生成されていることがわかる。

4.3 アブレーション

4.2 節の実験では、モデルへの入力として 2 種類の押韻表記を用い、それぞれの生成結果を比較した。アブレーション実験として、以下の 3 つの要素を変更し、それぞれの影響を評価する。

∗5 https://huggingface.co/llm-jp/llm-jp-3-1.8b
∗6 https://huggingface.co/rinna/japanese-gpt2-medium

表 3: 小節末における発音の再現精度

押韻表記	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5
ひらがな	0.717	0.800	0.812	0.873	**0.880**
ローマ字	0.723	0.749	0.789	0.819	**0.845**

表 4: モデルによる再現精度の違い

モデル	押韻表記	発音の再現精度
llm-jp-3-1.8b	ひらがな	0.880
llm-jp-3-1.8b	ローマ字	0.845
japanese-gpt2-medium	ひらがな	0.199
japanese-gpt2-medium	ローマ字	0.411

表 5: コーパスによる再現精度の違い

学習コーパス	テストコーパス	発音の再現精度
日本語 Wikipedia	日本語 Wikipedia	0.880
日本語 Wikipedia	青空文庫	0.719
青空文庫	日本語 Wikipedia	0.757
青空文庫	青空文庫	0.827

モデルの違い 4.2 節の実験の学習に用いたモデルは llm-jp-3-1.8b であったが、アブレーション実験として japanese-gpt2-medium を用いた場合の学習結果と比較する。japanese-gpt2-medium のパラメータ数は約 361M であり、llm-jp-3-1.8b のパラメタ数の約 1/5 である。結果を表 4 に示す。モデルサイズの低下は生成の精度に大きく影響した。

コーパスの違い 4.2 節の実験の学習に用いたコーパスは日本語 Wikipedia であったが、アブレーション実験として青空文庫コーパスを用いた場合の学習結果と比較する。学習に用いたコーパスと逆のコーパスを使用して生成を行った場合も含め、結果を表 5 に示す。学習コーパスとテストコーパスが一致している場合は、発音の再現精度がどちらも 80%以上であった。テストコーパスが学習コーパスと異なる場合は、どちらも精度が落ちるものの、青空文庫で学習したコーパスの方が精度を保てていた。日本語 Wikipedia コーパスに比べ文章の多様性がある青空文庫コーパスで学習したモデルの方が、学習していないデータにも対応しやすいと考えられる。生成例を以下の (3), (4) に示す。(3w), (4w) は日本語 Wikipedia で学習したモデル、(3a), (4a) は青空文庫で学習したモデルの生成例である。

(3) w. 【慈善団体】[' ああお', ' えーう', ' おーい']
慈善団体は、何らかの<BRK>
慈善事業を行っている<BRK>
団体を指すことが多い<BRK>

a. 【慈善団体】[' ああお', ' えーう', ' おーい']
ところが、世の中には世の中の<BRK>
ために働いている<BRK>
人が多い<BRK>

(4) w. 【駅周辺】[' あえあ', ' えーう']
駅周辺は区画整理された<BRK>
住宅地となっている<BRK>

a. 【駅周辺】[' あえあ', ' えーう']
私は、それを眺めた<BRK>
ときから心をひかれている<BRK>

日本語 Wikipedia で学習したモデルでは、出力が指定されたテーマに対して説明的であるのに対し、青空文庫で学習させたモデルでは、テーマに対して多少創造的な文が生成された。

表 6: 押韻タイプによる再現精度の違い

押韻タイプ	押韻表記	発音の再現精度
母音	ひらがな	0.880
母音	ローマ字	0.845
子音+母音	ひらがな	0.861
子音+母音	ローマ字	0.866

押韻タイプの違い　4.2 節の実験の学習に用いたコーパスは押印タイプは母音であったが、アブレーション実験として子音+母音を用いた場合の学習結果と比較する。結果を表 6 に示す。指定した発音の再現精度は、押韻タイプによって大きく変わることはなかった。この結果は、ラップの自動生成ツールを実現する際に、母音のみを指定する場合でも、子音と母音を組み合わせて指定する場合でも、どちらも効果的に機能する可能性があることを示している。

4.4　ラップ生成実験

実際のラップと同じ形式となるように、同じ発音を各小節で指定して生成させて評価する。テストに用いるデータは、日本語 Wikipedia と青空文庫それぞれから 1,000 件ずつを選択する。結果を表 7 に示す。日本語 Wikipedia で学習したモデルよりも、青空文庫コーパスで学習したモデルの方が押韻の再現精度は高かった。日本語 Wikipedia コーパスは百科事典であることから、文末には「である」「いる」などの表現が多く見られる。一方で、小説や随筆などからなる青空文庫コーパスは日本語 Wikipedia と比べ表現が多様である。青空文庫コーパスで学習したモデルは、多様な発音が学習できたため、押韻の精度が高くなったと考えられる。青空文庫コーパスで学習したモデルで生成した文の例を以下の (5), (6) に示す。

(5) 【切り返し】[' いえお', ' いえお', ' いえお', ' いえお', ' いえお']
たとえ、同じ思い出の<BRK>
深い思い出にしても、<BRK>
私は今、あの思い出の<BRK>
切なさ、切なさ、どうしても<BRK>
切れ切れよ<BRK>

(6) 【沿革】['ena', 'ena', 'ena', 'ena', 'ena']
その当時の世間は<BRK>
まだ日本に未練が<BRK>
あったし、明治以前は<BRK>
やはり漢語の古典が<BRK>
おくれたのだと云って残念だ<BRK>

本モデルはラップの形式の学習に留まっているため、指定した韻を踏むことには成功しているものの、文の意味が通じておらず、ラップの歌詞生成モデルとしては十分な性能を有しているとは言い難い。また、ラップにおいては名詞を小節の末尾に配置して韻を踏むことが多く見られるため、その特徴の学習も必要である。

5.　おわりに

本論文では、教師なし学習によって韻を学習し、ラップの形式で出力する手法を提案した。本手法では、形式的で文語的な日本語 Wikipedia コーパスよりも、多様で口語的な青空文庫コーパスを基にしたデータセットを用いて学習したモデルがラップの形式を学習するのに効果的であることがわかった。

表 7: 押韻の再現精度

学習コーパス	押韻表記	押韻の再現精度
日本語 Wikipedia	ひらがな	0.592
日本語 Wikipedia	ローマ字	0.584
青空文庫	ひらがな	0.740
青空文庫	ローマ字	0.754

本研究では、ラップの形式を学習し、小節の末尾で指定した発音のテキストを生成することに焦点を当てた。実際のラップでは小節の末尾に名詞を置く体言止めが多く見られる。今後は、体言止めの学習手法についての検討、さらには、形式だけでなく生成される文の内容についての研究を進めたい。

謝辞

本研究は JSPS 科研費 JP23K17641 の助成を受けて実施した。一部の計算は、東京科学大学のスパコン TSUBAME4.0 を用いて行った。

参考文献

[Kawahara 07] Kawahara, S.: Half rhymes in Japanese rap lyrics and knowledge of similarity, *JOURNAL OF EAST ASIAN LINGUISTICS*, Vol. 16, No. 2, pp. 113–144 (2007)

[Nikolov 20] Nikolov, N. I., Malmi, E., Northcutt, C., and Parisi, L.: Rapformer: Conditional Rap Lyrics Generation with Denoising Autoencoders, in Davis, B., Graham, Y., Kelleher, J., and Sripada, Y. eds., *Proceedings of the 13th International Conference on Natural Language Generation*, pp. 360–373, Dublin, Ireland (2020), Association for Computational Linguistics

[Ormazabal 22] Ormazabal, A., Artetxe, M., Agirrezabal, M., Soroa, A., and Agirre, E.: PoeLM: A Meter- and Rhyme-Controllable Language Model for Unsupervised Poetry Generation, in Goldberg, Y., Kozareva, Z., and Zhang, Y. eds., *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 3655–3670, Abu Dhabi, United Arab Emirates (2022), Association for Computational Linguistics

[Potash 15] Potash, P., Romanov, A., and Rumshisky, A.: GhostWriter: Using an LSTM for Automatic Rap Lyric Generation, in Màrquez, L., Callison-Burch, C., and Su, J. eds., *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1919–1924, Lisbon, Portugal (2015), Association for Computational Linguistics

[三林 24] 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)

[太田 24] 太田 聖三郎, 河原 大輔, 野村 理朗：おもしろい川柳の生成, 言語処理学会 第 30 回年次大会 発表論文集 (2024)