ラップバトルにおけるライムの意味類似性を考慮したバース生成システム

三林　亮太†　　橋口　友哉†　　山本　岳洋††　　大島　裕明†,††

† 兵庫県立大学 応用情報科学研究科　〒 650–0047 兵庫県神戸市中央区港島南町 7–1–28
†† 兵庫県立大学 社会情報科学部　〒 651–2197 兵庫県神戸市西区学園西町 8–2–1
E-mail: †{aa20r511,aa19j508,ohshima}@ai.u-hyogo.ac.jp, ††t.yamamoto@sis.u-hyogo.ac.jp

あらまし　本研究では，ラップバトルにおけるライムの意味類似性を考慮したバース生成手法を提案する．即興ラップを対話形式でおこなうラップバトルは，上手い韻（ライム）を用いたラップでの返答（バース）が期待される．ライムは直感的には,「博士学生」と「院で覚醒」のように，ある単語に対して母音列が同一の語のことである．上手いとされるライムは，母音列が同一であるばかりでなく，例のように意味類似性をもつものであることが多い．このような言葉遊びのユニークさから，ラップバトルは近年メディアで注目されており，様々なコンテンツが存在する．しかし，即興ラップにおいて，意味類似性を考慮したライムを用いたバースを生成することは容易ではない．さらに，意味類似性に加えて相手の発話を考慮した返答を生成することはより困難である．そこで，本研究では機械学習を用いて意味類似性を考慮したバース生成手法を提案し，コンテンツ作成の補助及びラップバトルにおける新たなバース生成を目指す．

キーワード　ラップバトル，文章生成

1　はじめに

ラップバトルとは，即興ラップを対話形式でおこないどちらのラップが優れているかを競う競技である．図 1 に示すように 2 人のラッパーが交互に即興ラップをおこなう．1 回の即興ラップをバースと呼び，バースを双方 3 回繰り返し計 6 回のバースをおこなうことでバトルは終了する [1]．

ラップバトルは近年注目を集めている競技であり，ラップバトルに関するコンテンツがいくつも展開されている．たとえば，ラップバトル番組である「フリースタイルダンジョン」やラップバトルを主題としたプロジェクトである「ヒプノシスマイク」やなどがある．また，ラップバトルの大会も催されており,「戦極 MCBATTLE」や「高校生ラップ選手権」などが存在する．

ラップバトルにおいてバースを即興で作成することは次の 2 点で人間にとって難しいと考えられる．1 点目は，バースにライムを含めなければならない点である．ライム（韻）とは，ある単語に対して母音列が一致する語のことである．例えば「作成」に対して「学生」は言葉の母音列「あうえい」と同一の母音列となっているため「作成」のライムである．

2 点目は，一般的なラップと異なり，ラップバトルにおいては相手のバースに返答する形でバースを作成する必要がある点である．ラップバトルではアンサーと呼ばれる，相手の発話に対してどれかで上手く返答出来たかを評価する指標がある．たとえば，図 1 で示すバース 2 は，相手の発言内容である「博士学生」に関係したバースを返答しているため，アンサーとして優れたバースであると考えられる．

1：基本的には双方 3 回であるが，最近では，2 回や 4 回で行なう場合もある．

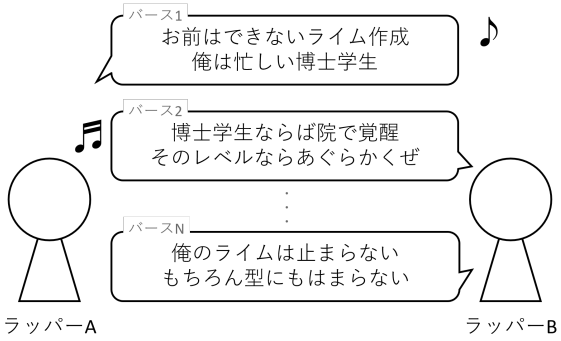

図 1　ラップバトルの概要図

これら 2 点を考慮したバースを即興で作成することは容易ではない．特に，ラップバトルでは相手が作成したバースに合わせて即興でバースを生成する必要があるため，単にライムを含んだバースを作成するだけでなくアンサーとして優れたバースを作成する必要がある．

本研究ではラップバトルを対象とし，ラップバトルにおけるバースの自動生成技術について取り組む．この技術を実現することで，これまでに生成が難しかった高度なバースの生成が行え，コンテンツ作成や創作支援が行える．

提案手法は大きく 2 つの工程によりバースを生成する．まず始めに，入力となるバースを受け取り，そこから返答となりうるバース候補を生成する．このために，Transformer ベースのモデルとテンプレートベースのモデルの 2 種類を提案した．これにより，入力となるバースと類似する内容のバースを生成することができる．次に，生成したバースに対してライムを付与することでバースを生成する．BERT や単語の分散表現から計

算される意味類似性を考慮したライムを複数生成することで，生成されたバース中に意味的に類似したライムを含めることができる．これらの手順により，ラップバトルにおいて入力バースを考慮したバース生成が実現できる．

2 関連研究

本節では本研究に関連する研究として，ラップの自動生成に関する研究について述べる．ラップ生成に関する研究は主に英語のラップの歌詞の自動生成を対象としたものが多い．Nguyen ら [12] は n-gram モデルを用いてラップの自動生成に取り組んでいる．テンプレートを用いた生成としては Barbieri ら [1] と Manjavacas ら [10] が挙げられる．Barbieri らはマルコフ過程とテンプレートを用いて，ラップの自動生成に取り組んだ．Manjavacas らは条件付きニューラル言語モデルとテンプレートを用いてラップの自動生成に取り組んだ．

また，機械学習手法の発展により，深層学習を用いた歌詞生成が盛んになっている．Potash ら [14] は LSTM ベースでのラップ生成に取り組み，ターゲットとなるアーティストのラップ歌詞の自動生成を行った．Wu ら [17] は Bi-RNN を用いたラップバトルのバースの生成に取り組んでいる．これはあるお題となる文に対して，バースを出力するものである．他にも RNN を用いた生成として Tong ら [15] の歌詞生成がある．近年，深層学習モデルを複数組み合わせた生成モデルも提案されている．Han Lau ら [8] は LSTM ベースでのラップ生成を行い，ラップ生成モデルに加え，ライム，韻を考慮するネットワークを新たに追加した手法を提案している．Malmi ら [9] は RankSVM と深層学習を用いた複合手法にて生成を行っている．Yu ら [18] は LSTM と GAN を用いて歌詞と楽曲の同時生成にも取り組んでいる．Nikolov ら [13] は Rapformer と呼ばれる，入力文章をラップ調に変換する手法を提案している．彼らの手法ではニュース記事をラップ調に変換する手法を提案している．本研究においてもこの Rapformer をベースとしてバース生成を行う手法を提案する．

このように，ラップ生成に関する研究は多くなされているものの，著者らの知る限り，ラップバトルにおけるラップ生成に関する研究は多くなされてはいない．1 節で述べたように，ラップバトルにおいては，単にラップを生成するだけでなく，相手のバースを考慮したラップ生成が重要な技術になると考えられる．

3 提案手法の概要

本節では提案手法の概要について述べる．本研究ではバース生成を 2 つの工程に分けておこなう．まず，図 2 に示すように入力バースから相手の発言を考慮した返答バースを生成する．これにより，アンサーを考慮する．この場合，相手の「桜，咲き誇る，百花繚乱，勝とう」というバース内のワードに注目し，関係したワード「桜，サクラ，勝利」を含む返答バースを生成している．次に，生成したバースに対してライムを付与する．これにより，意味類似性を持ったライムを考慮する．ライム付与には，BERT を用いた意味類似性と単語分散表現を用いた意味類似性の 2 つをおこなった．図 2 の赤字のように返答バースに対してライムを付与する．この 2 つの工程により，意味類似性を考慮したバースの生成をおこなう．

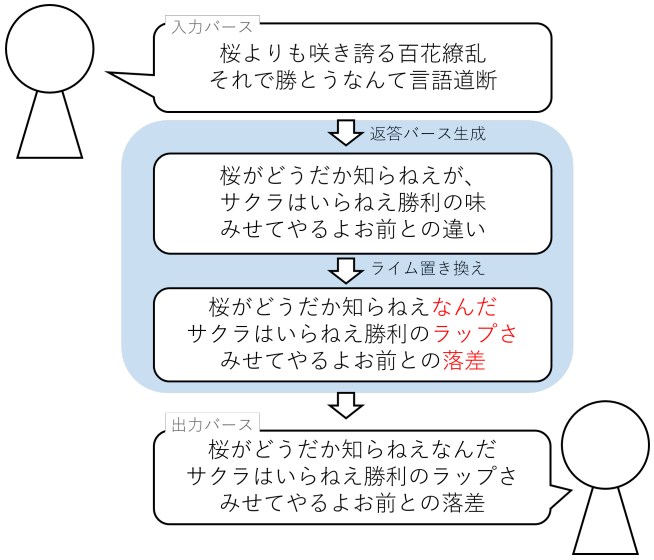

図 2 提案手法の概要図

3.1 返答バースの生成

返答バースの生成には Transformer ベースとテンプレートベースの 2 種類の生成方法に取り組んだ．Transformer ベースのモデルでは事前学習したモデルをラップを対象としたコーパスでファインチューニングすることにより，ラップ特有の文構造を備えたバースが生成されることが期待される．Transformer ベースのモデルによるバースの自動生成は高い柔軟性を持つ一方で，文として破綻した出力がなされる可能性もあり，生成されるバースの質は多様であると考えられる．そこで，テンプレートベースのバース生成手法についても提案する．テンプレートベースの手法ではあらかじめ用意したテンプレートに単語を当てはめることでラップを生成する．そのため，表現可能なラップの種類は限定されるものの，ラップの複雑な文構造を保ったバース生成が期待される．

Transformer ベースの生成には Nikolov ら [13] の Rapformer を参考にした．Rapformer は入力文章をラップ調の文章に変換する言語スタイル変換モデルである．ラップ歌詞データからキーワードを抽出し，キーワードから元のラップ歌詞を生成する学習をおこなう．キーワードを入力にすることで，ラップ歌詞以外の文章に対しても汎用的に生成がおこなえる．本手法では，Rapformer を日本語で学習することでバースの生成に取り組んだ．

テンプレートベースの生成には実際のラップバトル内のバースを元に BERT [3] を用いて生成をおこなった．おおまかなアプローチとしては，実際のバースをテンプレートとし，そのほとんどの語を別の語に置き換えることで，テンプレートの文構造を維持したまま新しいバースを生成する．語の置き換えには BERT を用いた．BERT は Transformer をベースにした言語

モデルである．BERT には文中の単語を予測することができる．本手法では既存のラップバトルバース中の単語の置き換えを BERT でおこなうことで，元のラップ文構造を維持したまま新しいバース生成に取り組んだ．

3.2 意味類似性を考慮したライム置き換え

ライムは意味類似性を持ったものが上手いライムとされている．ライムは，ある単語に対して同一の母音列が含まれる別の語のことである．例を挙げると,「作成」と「学生」は母音列「あうえい」が含まれるライムである．特に,「博士学生」と「院で覚醒」のような意味類似性を持った上手いライムは高く評価される．そこで，本研究では，単語分散表現を用いた手法と BERT を用いた手法の 2 種類で意味類似性を考慮したライム置き換えをおこなう．

単語分散表現を用いたライム置き換えは，単語分散表現のコサイン類似度を用いておこなう．単語分散表現は単語をベクトルで表現する技術である．単語分散表現は周辺の単語情報から学習され，意味的に近い単語はベクトル空間においても近くに位置する．よって，単語分散表現のコサイン類似度を求めることで，単語間の意味類似性が測定できる．この特性を用いて，ライム間の意味類似性を計算することで，意味類似性が高いライムが取得できる．

BERT を用いたライム置き換えは，BERT の単語予測を用いる．BERT の単語予測を用いることで，文脈に沿った単語，つまり文と意味類似した単語が予測できる．BERT で，バースの末尾に対して予測をすることで，バースの文脈を考慮した単語が得られる．バース内の 2 文に対してそれぞれ単語を予測し，その単語間の母音マッチングをおこなうことでライムの取得が可能になる．その後，元単語をライムで置き換えることで，意味類似性を持ったバースの生成が可能になる．

4 バース生成

4.1 コーパス

バース生成モデルの学習を行うに当たって，事前学習用とファインチューニング用に 2 つのコーパスを用意した．事前学習には毎日新聞コーパスを用いて，ファインチューニング用にはラップバトルコーパスを作成した．

4.1.1 毎日新聞コーパス

Rapformer [13] の事前学習は英語のニュース記事データを用いて行われた．本実験では，日本語の新聞記事データである毎日新聞コーパスを用いて事前学習を行なう．

毎日新聞コーパスは日外アソシエーツ社及び毎日新聞社が販売する毎日新聞の全文記事データ集である．現在，1991 年から 2019 年までのデータが販売されている．今回，1995 年から 2007 年までのデータを用意した．記事データは全部で 1,402,358 件である．記事データをそのまま入力すると入力長が長くなりすぎ，学習に支障がでるため，記事データを句点で区切り，文レベルに変換した．文レベルにした結果，データ数は 16,249,171 件になった．

4.1.2 ラップバトルコーパス

現在，ラップバトルのテキストデータは公開されていないため，独自にデータ収集を行いラップバトルコーパスの作成を行った．ラップバトルコーパスの作成はクラウドソーシングサイト Lancers [2]にてクラウドソーシングを行った．クラウドソーシングの内容は YouTube 上に存在するラップバトル動画に対して文字起こしをしてもらうものである．対象となるラップバトル動画は以下の YouTube チャンネル及びユーザーである．

- UMB [3]
- 凱旋 MCbattle [4]
- 戦極 MCBATTLE [5]
- 日本語ラップ COM [6]
- アドレナリン チャンネル [7]

対象動画は全部で 1935 件であった．

ワーカーにはこちらが作成した作業用サイトにアクセスしてもらい，指定するラップバトル動画のリンク先の YouTube 上のラップバトル動画に対して文字起こしをしてもらった．クラウドソーシングを行うにあたってワーカーに対し，以下の事項を指示した．

- ラッパーが切り替わる際には「空行」を挟んで下さい．ラッパーが切り替わるところ以外では「空行」を使わないようにして下さい．
- 言葉の区切りに半角スペースを入れて下さい．息継ぎや少し間が空いた時などです．
- 司会者が発する「ReadyFight!!」や,「そこまで〜！」といったセリフは必要ありません．あくまで 2 人のラッパーが発する音声の文字起こしをして下さい．
- 語尾の伸ばしは省略してください．例えば,「だりぃ〜」ではなく「だりぃ」として下さい．
- 噛んだりした箇所であっても補完できる場合は補完して下さい．例えば,「だから言っつぇんだろ」を「だから言ってんだろ」と補完していただくといった場合です．
- もし，聞き取れた言葉に当てはまる漢字が文脈を考慮しても一意に決まらない場合は，ひらがなで入力しておいて下さい．例えば,「つまり大事なのはいかり」の「いかり」が「怒り」なのか「碇」なのかわからないといった場合です．
- 効果音などは入力する必要はありません．

今回，132 件のラップバトル文字起こしコーパスを作成した．こちらも文レベルに分割を行った．分割方法としては，ワーカーに指示した，息継ぎや少し間が空いたタイミングで付与した半角スペースにしたがって分割した．

4.2 Transformer でのバース生成

4.2.1 手法の概要

Transformer でのバース生成は，Rapformer [13] を日本語に

2： https://www.lancers.jp/
3： https://www.youtube.com/user/umbofficial
4： https://www.youtube.com/channel/UCe_EvY8GrvYgx8PbwRBc75g
5： https://www.youtube.com/user/senritumc
6： https://www.youtube.com/channel/UC9wgzX2VFLQrzE4OyskAiIA
7： https://www.youtube.com/channel/UCj6aXG5H_fm_RAvxH38REXw

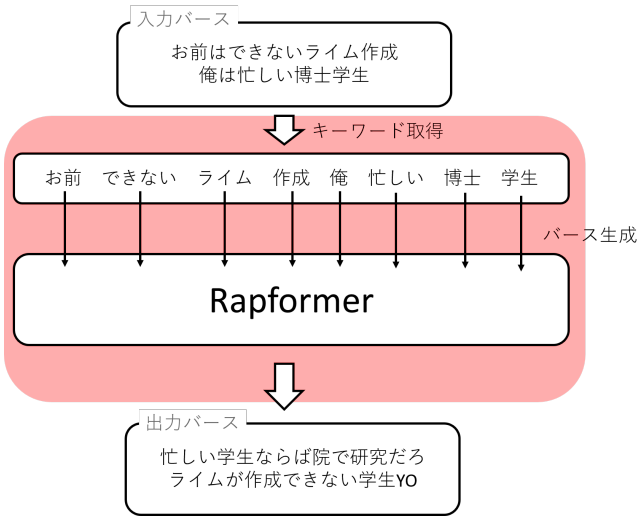

図 3　Rapformer を用いたバース生成

て再現する．Rapfomer は図 3 のように、キーワードを入力にバースの生成を行うモデルである．

4.2.2　Rapformer

Rapformer [13] とは，Nikolov らが提案した，入力テキストをラップ調のテキストに変換するスタイル変換モデルである．モデルは Transformer [16] ベースの seq2seq モデルである．Transformer 層は encoder-decoder 共に 6 層で構成される．

Rapformer はニュース記事データを用いた事前学習とラップ歌詞データを用いたファインチューニングの 2 つの工程で学習される．事前学習には，CNN/DailyMail のニュースデータセットを用いて Rapformer を学習した．その後，音楽歌詞投稿サイトである Musixmatch [8]から取得したラップ歌詞 60 万件を用いて Rapformer をファインチューニングした．

Rapformer の入力には，入力テキストを単語分割し，キーワードを取得する．取得したキーワードに対してキーワード変換を行う．キーワード変換にはシャッフル，ドロップ，類義語置き換えの 3 つの方法が提示されている．シャッフルは取得したキーワードの順序をランダムに入れ替える方法である．seq2seq モデルは時系列を考慮するがキーワードは順番に意味がないため，汎用性を持たせるためにシャッフルを行う．ドロップは 20 パーセントの確率でキーワードを除去する．これは少ないキーワード集合にて学習を行うことで汎用性を持たせる．最後に類義語置き換えであるが，これはキーワードを WordNet を用いて取得した類義語に変換する処理である．キーワード変換に関しては 3 つの手法のうちどれか 1 つを選択して用いる．変換を行ったキーワードを入力に，元文章を再現するように seq2seq を学習する．これにより，入力文章のキーワードから文章の生成を可能にする．この文章生成モデルをラップ歌詞を用いてファインチューニングすることで，キーワードからラップ歌詞へのスタイル変換を可能としている．

加えて，生成したラップ歌詞に対して BERT を用いたライムの付与を行うことでラップのクオリティ向上を図っている．まず，Wikipedia コーパスで事前学習された BERT をラップコーパスでファインチューニングを行う．その後，生成されたラップ歌詞にすべてに対して，末尾の単語を MASK トークンに変換する．それぞれ 2 文に対して，各 MASK の単語を予測する．予測した単語集合と，元の単語とをライムであるかの判定を行う．判定語，一番長いライムを採用し，MASK と置き換える処理を行う．これにより生成されたラップ歌詞のライムを補完し，よりラップとしてのクオリティを向上させている．

4.2.3　バース生成

バース生成には，Rapformer と同じく encoder-decoder 共に 6 層で構成される標準的な Transformer モデルを用いる．実装は Open-NMT [5] を用いた．

Transformer モデルの学習は事前学習とファインチューニングの 2 つの工程にておこなった．まず，事前学習に用いる毎日新聞コーパスの前処理を行った． 毎日新聞コーパスのすべて文に対して，MeCab [7]+mecab-ipadic-neologd にて形態素解析後，ストップワード，数字，漢数字，アルファベット，句読点を除去した．ストップワードは SlothLib [9]を参考にした．加えて，入力となるキーワード集合は「名詞，動詞，副詞，形容詞」のみを対象とし，それ以外のワードは除去した．その後，Sentencepiece [6] にてサブワード化をおこなった．サブワード化に関するパラメータ設定は語彙サイズ 32000, キャラクターアベレージは 0.9995, モデルタイプは bpe にておこなった．

事前学習の設定にはバッチサイズは 128，学習率は 0.01，ステップ数は 99000 にて学習をおこなった．オプティマイザーには Adam [4] を使用した．

事前学習後，ラップバトルコーパスを用いてファインチューニングをおこなった．ファインチューニングは事前学習と同じ前処理，パラメータ設定にておこなった．

4.3　テンプレートベースのバース生成

4.3.1　手法の概要

テンプレートベースのバース生成は，大きく 2 つの工程でおこなう．置き換え語の生成とテンプレート内単語の置き換え処理である．置き換え語の生成では，入力バース内の単語に対する関連語の生成を行う．生成した関連語をテンプレート内の単語と置き換えることにより，入力バースを考慮した返答バース，すなわちアンサーを考慮したバースを生成する．その後，テンプレート内の単語を更に置き換えることにより，テンプレートに依存しない新たなバースを生成する．

4.3.2　バース生成

置き換え語の生成には，入力バース内のキーワードを用いる．まず，入力バースを MeCab+mecab-ipadic-neologd にて形態素解析し，単語に分割する．入力バース内の単語からランダムに 1 語取得し，それをキーワードとする．キーワードに対して WordNet を用いて関連語を取得する．関連語の取得には日本語版 WordNet [2] を用いた．WordNet [10]にてキーワードの関

8：https://www.musixmatch.com/

9：http://svn.sourceforge.jp/svnroot/slothlib/CSharp/Version1/SlothLib/NLP/Filter/StopWord/word/Japanese.txt

10：http://compling.hss.ntu.edu.sg/wnja/

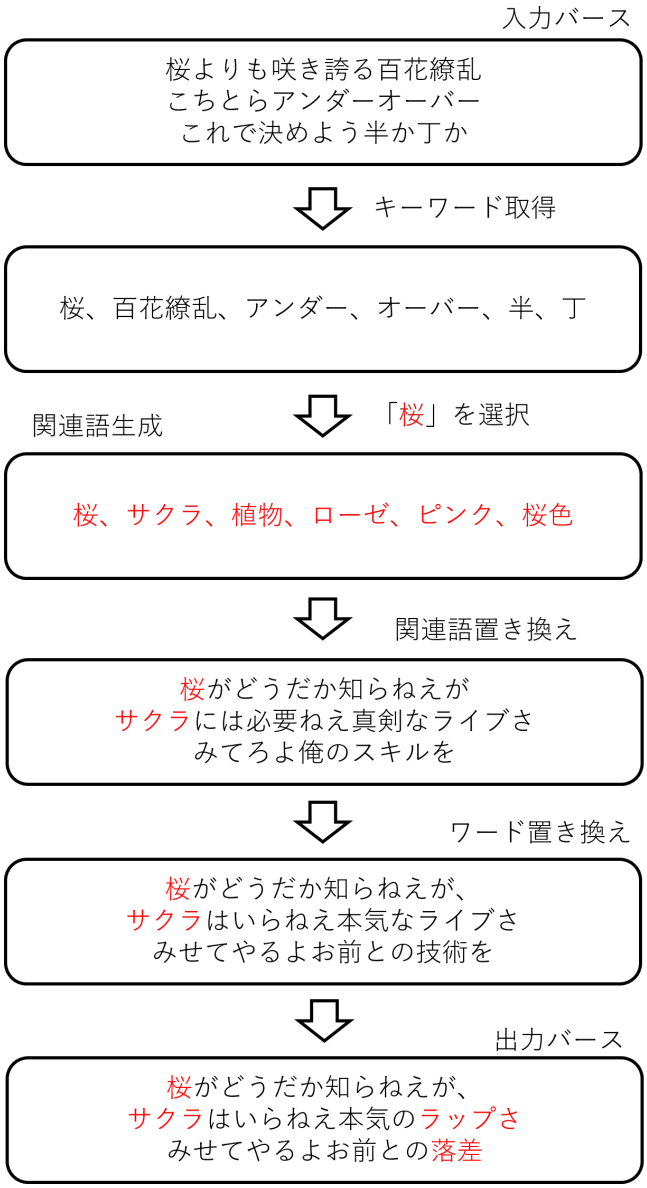

図 4　テンプレート生成の流れ

連語を検索し，上位語，下位語，類義語などの関連語の語集合を取得した．キーワードを加えた関連語の語集合を置き換え語集合とする．

次に，テンプレート内の語を置き換える．テンプレートはラップバトルコーパスの中からランダムにバースを 1 件取得し，そのバースをテンプレートとして用いる．テンプレートは BERTTokenizer にて形態素解析し，単語に分割する．今回は東北大学が作成した BertTokenizer である bert-base-japanese-whole-word-masking [11]を用いた．

助詞を除く全ての単語に対し，バースの先頭から 1 つずつ MASK に置き換え，BERT を用いて単語を予測するという工程を繰り返す．予測した単語に置き換え語が含まれていれば，元の語を置き換え語と置き換える．置き換える処理はすべての単語に対しての予測が完了した後におこなった．BERT の MASK の予測には，東北大学の事前学習モデル [12]を用いて，上位 200 件まで予測を行った．全てを関連語に置き換えるとバースとして不自然になることから，置き換えは指定数のみおこなう．今回は最大で 8 語の置き換えをおこなった．

最後に，置き換えた語を除く全ての単語に対して置き換えを行なう．テンプレートは元バースをそのまま用いているため，内容語がほとんど同じになってしまう．そこで，テンプレート内の単語を BERT の単語予測を用いて全て置き換える処理を行なう．関連語を置き換えたことにより，元バースをそのまま置き換える時に比べて，関連語を考慮した置き換えが期待できる．関連語置き換えと同じく，テンプレート内の単語を先頭から MASK に置き換え，BERT を用いて単語を予測する．元の単語と予測した単語の品詞を MeCab を用いて推定し，品詞が同一であれば置き換えを行なう．この時，品詞を考慮することにより，バースの構造が崩れることを防いでいる．

5　意味類似性を考慮したライム生成

5.1　手法の概要

ラップバトルにおけるライムは意味類似性を考慮すべきである．ライムとは，ある語に対して母音列が同一の語である．例えば「言葉」と「ココア」は母音列「おおあ」が同一であるライムである．母音列は完全一致である必要はなく，末尾から 1 文字以上が同一であればライムとする．単語から母音列への変換は一度単語をローマ表記へと変換し，母音のみを抽出する．「ん」や「ー」などの特殊なケースについては別章で述べる．

中でも,「博士学生」と「院で覚醒」のような共通点を持ったライムを意味類似性を考慮したライムと呼ぶ．単純な母音検索では意味類似性を考慮することはできない．そこで，我々は単語分散表現と BERT を用いることで，ライムの類似度を考慮し，意味類似性を持ったライム生成を行う．単語分散表現とは word2vec [11] などで知られる単語のベクトル表現のことである．これらの単語ベクトル表現のコサイン類似度を求めることで，単語間の意味類似性が考慮される．BERT は Transformer をベースにした言語モデルであり，文中の単語の予測ができる．BERT を用いることでバース内の文と意味類似性をもつライムの生成が可能である．

5.2　単語分散表現を用いたライム生成

5.2.1　手法の概要

単語分散表現のベクトル間のコサイン類似度を用いることで，単語間の類似度が求められることが知られている．我々はライム生成において，母音でのマッチングに加えて，単語分散表現での類似度を考慮することで意味類似性を持ったライム生成を行う．システムの入力は単語であり，出力はランキング付けされたライムのリストを返す．ランキングが上位の語ほど，意味類似性が高いライムである．

5.2.2　ライム生成

ライム生成にはまず，図 5 のように「ドブネズミ」といった

11：https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking

12：https://github.com/cl-tohoku/bert-japanese

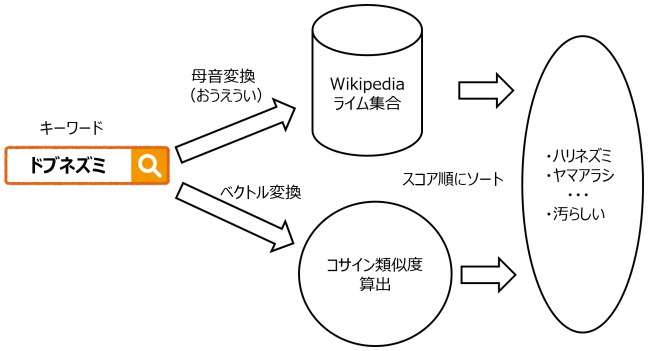

図 5　意味類似性を考慮したライム生成の処理手順

キーワードを入力する．次に，入力したキーワードを母音へ変換する．母音変換には pykakasi [13]を用いてローマ字変換を行い，ローマ字から「a,i,u,e,o」以外の文字をすべて除去し，母音のみを取得する．取得した母音と後述する母音辞書を用いて母音のマッチング検索を行う．検索方法はシンプルで，末尾から 1 文字以上マッチしていればライムとみなし，ライム集合へ追加する．

次にライム集合すべての単語と入力キーワードを単語分散表現を用いてベクトルへと変換する．使用する単語分散表現は東北大乾研究室の Wikipedia 分散表現 [19] を用いた．変換したベクトルを用いてライム集合内の単語それぞれと入力キーワードすべてのコサイン類似度を算出する．算出したスコアを元にライム集合内の単語を降順に並べ替える．単語間の類似度が高い程スコアは大きくなるため，ランキング上位の単語ほど類似度が高いランキングリストが得られる．

5.2.3　キーワードからの母音抽出

本研究では「ー」,「ん」,「っ」の 3 つについては例外処理を行う．まず,「ー」では，前の文字の母音に置き換える．つまり,「プール」であれば「ううう」と変換される．次に「ん」は「n」として扱う．最後に「っ」は変換を行わず無視する．

5.2.4　母音辞書の作成

Wikipedia 分散表現の単語 1,015,474 語すべてに対して母音変換を行いマッチングを行うのは非常に時間がかかる．なので，母音の検索を行うにあたってすべての単語を事前に母音変換しておき，処理時間の短縮を行う．まず Wikipedia 分散表現内の単語すべてを取得する．取得した単語すべてに対して pykakasi を用いてローマ字へと変換する．その後変換したローマ字から「a,i,u,e,o,n」以外の文字をすべて除去し，母音のみを取得する．母音をキーとし，対応する単語を辞書の値のリストに追加しておく，これにより，1,015,474 語の単語から 357,897 種類の母音辞書に変換でき，検索速度が向上した．

5.2.5　ライム候補のスコア付け

ライム長と意味類似性を考慮するために，本研究では単語がライムとしてどの程度相応しいかを表したライムスコア $\mathrm{Score}_{\mathrm{rhyme}}$ を定義する．基本的には，母音のマッチング数 V_{score} とコサイン類似度 C_{score} の 2 つのスコアの線形和をライムスコアとして用いる．母音マッチング数とコサイン類似度を [0, 1] の区間に変換する．コサイン類似度は [-1, 1] の値域をとるため，負の値を全て 0 とし [0, 1] に変換する．母音のマッチングはマッチした母音数を X とし，以下の数式で [0, 1] の値へ正規化する．

$$V_{\mathrm{score}} = \frac{X - X_{\mathrm{min}}}{X_{\mathrm{max}} - X_{\mathrm{min}}}$$

母音のマッチング数とコサイン類似度の割合を決める重みを α と定義し，スコアを以下の式で求めた．

$$\mathrm{Score}_{\mathrm{rhyme}} = (1-\alpha)V_{\mathrm{score}} + \alpha C_{\mathrm{score}}$$

今回の実験では $\alpha = 0.2$ とした．求めたスコアを降順に並べ替えることで，ライム長と意味類似性を用いたライムを検索結果として出力する．対象となる単語が Wikipedia 分散表現に含まれていない場合はコサイン類似度を計算することができないため，そのような単語は単語長をライムスコアとした．

5.3　BERT を用いたライム生成

BERT では文脈を考慮した単語の予測が可能である．本手法では，BERT をバース内の文の末尾の単語に対して用いることで，文に合った末尾の単語の予測をおこなう．ある 2 文に対して，それぞれの末尾の単語を予測し，その単語同士の母音マッチングをおこなうことで，文の意味類似性を考慮したライムの生成がおこなえる．BERT の事前学習モデルには東北大学の事前学習モデル [14]を用いた．生成した返答バース内の 2 文に対して，末尾の単語を MASK に置き換え，単語の予測を上位 200 件までおこなった．その後，予測された単語同士の母音一致を行い，ライムと判断した単語を元の末尾の単語と置き換えた．

6　実　行　例

6.1　バース生成結果

表 1 に Rapformer ベースのバース生成結果を示す．生成されたバースには入力キーワードである「咲き，冗談，オーバー，丁」が含まれており,「じゃねえ」や「ねぇぞ」などのラップバトルで用いられる表現が含まれている.「オーバー」に対して「キロ」という速度を連想する単語が生成されているように見受けられる．しかし，全体的にバースとしては支離滅裂な内容となっており，上手く生成ができていないことがわかる．

次に，表 2 にテンプレートベースでの生成結果を示す．テンプレートベースの生成では，入力バース内の「桜」というキーワードから，関連語のキーワード集合を作成し，バースが生成された．結果として,「桜, ピンク, サクラ」などのキーワードを含むバースの生成がおこなえた．テンプレートを用いた影響により，バースの構造がある程度保たれていることがわかる．しかし,「ふたを開かれともなる」や「点火するてく心」のように，元単語との品詞の活用形の違いにより，構造が崩れてしまう部分も見受けられる．

結果として，Rapformer を用いたバース生成はキーワード

13：https://github.com/miurahr/pykakasi

14：https://github.com/cl-tohoku/bert-japanese

とラップらしい語の生成が可能であるが，支離滅裂な内容が生成されてしまうことがわかった．対して，テンプレートを用いたバース生成は，キーワードの考慮かつ文構造を維持したままバースの生成が可能であることがわかった．

6.2 意味類似性を考慮したライム検索結果

BERT でのライム置き換えでは，表 1 のような支離滅裂なバースに対しては上手くライムの生成がおこなえていない傾向にあった．しかし，表 2 のような，文構造が維持されたバースに関しては「桜の光」と「一サイド左」といった意味類似性を持ったライムが生成されることがわかった．

単語分散表現を用いたライム置き換えでは，表 1 及び表 2 に示すように，どちらのバースについても上手く生成ができていない傾向にあった．特に,「俺は呼ば」や「サクラの引き継が」など，文が途中で途切れるようなライムが生成される傾向がみられた．

7 まとめと今後の課題

本紙ではラップバトルにおける技術的問題とそれらを解消するための手法について述べた．具体的には，返答バースの生成手法と意味類似性を考慮したライム置き換えの 2 つの工程によってラップバトルにおけるバースの生成をおこなった．バース生成においては，Rapformer を用いたバース生成とテンプレートを用いたバース生成をおこなった．結果として，バース生成においては，テンプレートを用いた生成手法が良い結果を示した．意味類似性を考慮したライム置き換えに関しては，BERT を用いた手法が意味類似性を考慮したライムを置き換えられていることがわかった．

今後の課題としては，データの拡充による Transformer 手法の改善と単語の活用形を考慮したテンプレート手法の改善がある．Rapformer を用いたバース生成はキーワードを用いた生成が可能であるものの，生成したバースが支離滅裂になる傾向にあった，これらの原因はデータ不足であると考察する．今後はデータ数を増やして実験を行う予定である．

テンプレート手法に関しては，今回，入力バースのキーワードをランダムに 1 つ選択する方法を取ったが，これでは入力バースをうまく考慮した語が取得できない場合がある．そのため，入力バースのトピックをうまく捉える手法が必要である．加えて，テンプレートにおいてもランダムではなく適切な選択手法が必要である．元テンプレートの構造を維持することで，ある程度の文構造が保持されることがわかったが，置き換え時の品詞に加え活用形も考慮する必要がある．

今後，音声での生成についても取り組む予定である．実際のラップバトルは音声ベースで行われるので，本システムにおいても音声ベースでの生成が期待される．なおかつテキストの母音だけでは生成できなかったような母音は一致しないものの音としては一致するライムも存在するため，音声でのライム生成手法が必要である．

また，今後の課題として，システムの評価がある．本システムの評価は人手評価にて行う．クラウドソーシングにてラップバトルの有識者を募り，以下の 2 つのパターンにて評価を行う．

- 提案モデル対ユーザ
- 提案モデル対提案モデル

まず，提案モデルとユーザでラップバトルを行いそのバトル内容を評価してもらう．これにより提案モデルのラップのクオリティを評価する．次に，提案システム同士でのラップバトルも行い，ラップバトル全体を通してのクオリティの評価を行なう．

謝　　辞

本研究の一部は JSPS 科学研究費助成事業 JP17H00762, JP18H03243 による助成を受けたものです．ここに記して謝意を表します．

文　　献

[1] Gabriele Barbieri, François Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. In *Proceedings of ECAI'12*, pp. 115–120, 2012.

[2] Francis Bond, Timothy Baldwin, Richard Fothergill, and Kiyotaka Uchimoto. Japanese semcor: A sense-tagged corpus of japanese. In *Proceedings of GWC'12*, pp. 56–63, 2012.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL'19*, pp. 4171–4186, 2019.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL'17*, pp. 67–72, 2017.

[6] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[7] Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. Applying conditional random fields to japanese morphological analysis. In *Proceedings of EMNLP'04*, pp. 230–237, 2004.

[8] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. *arXiv preprint arXiv:1807.03491*, 2018.

[9] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. In *Proceedings of SIGKDD'16*, pp. 195–204, 2016.

[10] Enrique Manjavacas, Mike Kestemont, and Folgert Karsdorp. Generation of hip-hop lyrics with hierarchical modeling and conditional templates. In *Proceedings of INLG'19*, pp. 301–310, 2019.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[12] Hieu Nguyen and Brian Sa. Rap lyric generator. Stanford CS224N Final Projects.

[13] Nikola I. Nikolov, Eric Malmi, Curtis Northcutt, and Loreto Parisi. Rapformer: Conditional rap lyrics generation with denoising autoencoders. In *Proceedings of INLG'20*, pp.

表 1　Transformer ベースでのバース生成手法を用いたバース生成結果

入力バース	桜よりも咲き誇る百花繚乱 つまらんライムまるで冗談 こちとらアンダーオーバー これで決めよう半か丁か
キーワード	桜，咲き，誇る，百花繚乱 つまらん，ライム，まるで，冗談 こちとら，アンダー，オーバー これ，決め，半，丁
出力バース+BERT によるライム置き換え	ふぁんで咲きのじゃねえニャン お冗談冗談めちゃねぇぞニャン オーバーがオーバーだよキロいやこちとこの そっからなんか丁論名前でねぇその
出力バース+単語分散表現によるライム置き換え	ふぁんで咲きのじゃねえまた お冗談冗談めちゃねぇぞ両側 オーバーがオーバーだよキロいや追い込ま そっからなんか丁論名前でねぇまた

表 2　テンプレートベースのバース生成手法を用いたバース生成結果

入力バース	桜よりも咲き誇る百花繚乱 つまらんライムまるで冗談 こちとらアンダーオーバー これで決めよう半か丁か
置き換え語集合	桜, 淡紅, 褪紅色, ばら色, 色合い, 属性, 薔薇色, 草木, サクラ色, 色調, 有彩色, 彩り, 成木, 樹, ツリー, 桃, 浅紅色, 退紅, 有機体, アトリビュート, 生活体, 果樹, さくら色, 成り物, 褪紅, 色彩, 維管束植物, 抽象的実体, 淡紅色, 薄紅, 全般, 一統, 退紅色, ロゼ, ローズ, 生り木, 色相, 成物, 生木, ローゼ, 桃色, カラー, サクラ, 総体, 特性, 鴇色, 性質, ピンク, 木本, 樹木, 植物, 桜色, 生り物, 天然木, 高木, 石竹色, 実体, 生体, 特質, 櫻, 色合
出力バース+BERT によるライム置き換え	勝負を選ぶ直前に、まずは決ようか桜の光 桜に敗れるのは一サイド左 ふたを開かれともなる 桜はピンクから祝福を行う サクラの年 俺は少年 点火するてく心 おれならば、実体総体に示すお前の所
出力バース+単語分散表現によるライム置き換え	勝負を選ぶ直前に、まずは決ようか桜の原作 桜に敗れるのは一サイド教授 ふたを開かれともフライト・レベル 桜はピンクから祝福を区別 サクラの引き継が 俺は呼ば 点火するてくは おれならば、実体総体に示すお前のから

360–373, 2020.

[14] Peter Potash, Alexey Romanov, and Anna Rumshisky. Ghostwriter: Using an lstm for automatic rap lyric generation. In *Proceedings of EMNLP'15*, pp. 1919–1924, 2015.

[15] Yongju Tong, YuLing Liu, Jie Wang, and Guojiang Xin. Text steganography on rnn-generated lyrics. *Proceedings of MBE'19*, Vol. 16, No. 5, pp. 5451–5463, 2019.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Proceedings of NIPS'17*, pp. 6000–6010, 2017.

[17] Dekai Wu and Karteek Addanki. Learning to rap battle with bilingual recursive neural networks. In *Proceedings of IJCAI'15*, pp. 2524–2530, 2015.

[18] Yi Yu and Simon Canales. Conditional lstm-gan for melody generation from lyrics. *arXiv preprint arXiv:1908.05551*, 2019.

[19] 鈴木正敏, 松田耕史, 関根聡, 岡崎直観, 乾健太郎. Wikipedia 記事に対する拡張固有表現ラベルの多重付与. 言語処理学会年次大会'16, pp. 797–800, 2016.