Vol.2017-NL-231 No.16
Vol.2017-SLP-116 No.16
2017/5/16

メロディと歌詞の相関に基づく自動歌詞生成

渡邉 研斗[1,a)]　松林 優一郎[1,b)]　深山 覚[2,c)]　中野 倫靖[2,d)]　後藤 真孝[2,e)]　乾 健太郎[1,f)]

概要：本研究では楽曲のメロディを考慮した歌詞の自動生成手法を提案する．人間の作詞現場においては，予め作曲されたメロディに対して歌いやすい歌詞を創作する「曲先」と呼ばれる方法が広く行われている．しかしながら，自動歌詞生成の既存手法の多くは，韻やシラブルに基づく生成手法を提案しているものの，メロディと歌詞の関係を考慮しておらず，メロディの区切りと単語の区切りが一致しないような不自然な歌詞を生成してしまう問題がある．本研究では，メロディの音符と歌詞の読みが対応づいたデータを用いて，メロディの音の長さ・休符の位置・繰り返し構造などの特徴と歌詞の相関を詳しく分析し，その結果をもとにした自動歌詞生成モデルを構築する．結果として作成されたモデルにより，休符や長い音符付近で行や段落 (連) が区切れている自然な歌詞が自動生成された．

1. はじめに

ポピュラー音楽において，歌詞は感情やメッセージを伝えるための重要な要素であり，作詞家は歌詞を魅力的にするために以下の様々な性質を考慮している [1], [2], [3], [4]：

1) メロディと単語の配置　作詞において，作曲されたメロディに対して歌詞を創作する「曲先」と呼ばれる方法が広く行われている．曲先の作詞では，メロディの音符数と単語のモーラ数[*1]を揃えることが一般的である．また，作詞家は休符や音符の位置を考慮し，歌いやすい歌詞を創作している．例えば，図 1 の単語「明日」の途中に休符があると，「あし・た」と不自然な場所で単語を区切ってしまう．一方「歩いた・」のように休符の前に単語があると歌いやすい歌詞になる．本研究では音符と休符の総称をノートと呼ぶ．

2) メロディと歌詞の抑揚　メロディの音高が変化するときは，歌詞の抑揚も変化させた方が良いと言われている．例えば，図 2 のように音高が上がるとき，「ほしは」のように抑揚を上げた歌詞を作ることが一般的である．

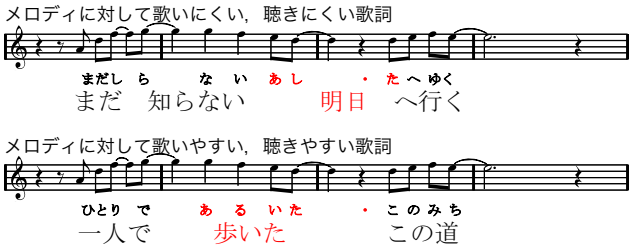

図 1　メロディに対して不自然な歌詞と自然な歌詞

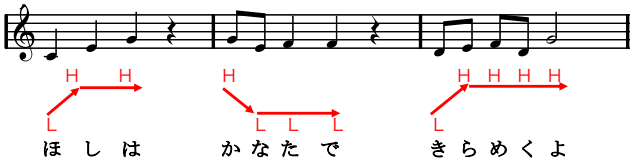

図 2　メロディの音高と歌詞の抑揚．抑揚を高 (H) と低 (L) で表す．

3) 楽曲の構造と歌詞の内容　ポピュラー音楽には「A メロ」「B メロ」「サビ」などの特有の楽曲構造がある（本研究ではブロックと呼ぶ）．作詞家はブロック構造に対してストーリー展開を持たせることで，聴き手が共感しやすい歌詞を創作している．例えば，「A メロ」で話の場面設定について述べ，「B メロ」で過去の回想をした後，「サビ」で登場人物の感情を描くなどをしている．

4) メロディの繰り返しと歌詞の繰り返し　ポピュラー音楽では，「1 番」と「2 番」のように，似たメロディが繰り返されることが多い．このとき，似たフレーズの歌詞を使用することが一般的である．例えば図 3 では，「1 番」と「2 番」のメロディに対して，「心の隙間」と「心の奥」などの似たフレーズが使われている．

1　東北大学大学院 情報科学研究科
Tohoku University
2　産業技術総合研究所
National Institute of Advanced Industrial Science and Technology (AIST)
a)　kento.w@ecei.tohoku.ac.jp
b)　y-matsu@ecei.tohoku.ac.jp
c)　s.fukayama@aist.go.jp
d)　t.nakano@aist.go.jp
e)　m.goto@aist.go.jp
f)　kento.w@ecei.tohoku.ac.jp

*1　モーラ数とは，日本語における音を数える際の単位であり，例えば，俳句における 5-7-5 の各数字はモーラ数に相当する．

Vol.2017-NL-231 No.16
Vol.2017-SLP-116 No.16
2017/5/16

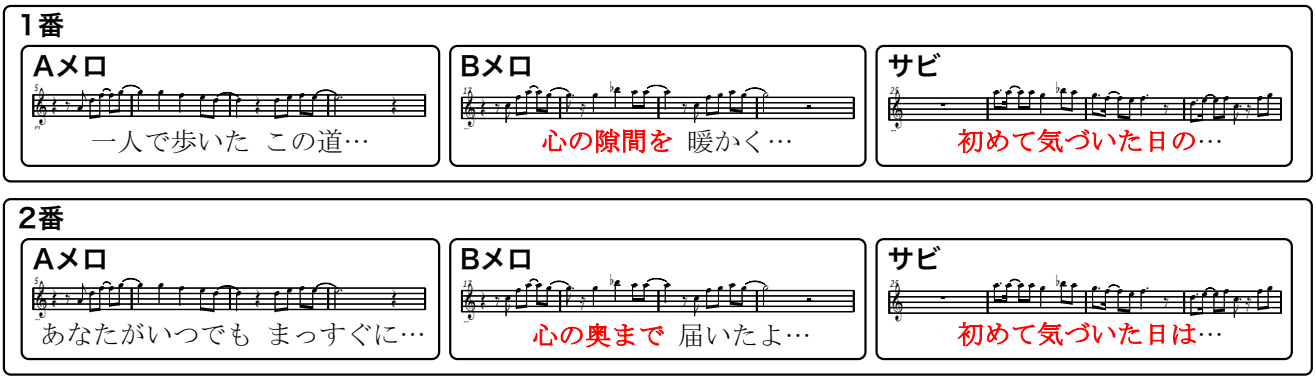

図 3 メロディと歌詞の繰り返し (RWC 研究用音楽データベース RWC-MDB-P-2001 No.20 [5])

もし上記の性質を満たした単語列を探索するシステムが実現できれば，膨大な可能性から単語列を探す手間を簡略化できると考えられる．このような動機から，歌詞の性質を数理的にモデル化する研究や，歌詞の自動生成・作詞支援システムに関する研究が行われてきた．[6], [7], [8], [9], [10]．例えば，渡邉らはユーザが入力したモーラ数列とストーリーを満たした単語列を推薦する支援システムを提案している [10]．

しかし，ほとんどの自動歌詞生成の先行研究は，モーラ数や韻の条件を満たす歌詞の生成に留まっており [8], [9], [11], [12]，メロディに合った歌詞の自動生成に踏み込んだ研究はほどんどない [6], [7]．この原因として，メロディと歌詞が対応づいた楽曲データが少ないことが挙げられる．また，メロディを考慮した自動歌詞生成の研究では，楽曲の「A メロ，B メロ，サビ」のようなブロック構造を考慮せず，4 行ほどの小範囲の歌詞しか自動生成していない．

我々の目的は，入力したメロディに対して，歌いやすく聴きやすい自然な歌詞を自動生成することである．この目的達成のために，以下の 3 つのタスクに取り組む：

1) メロディと歌詞が対応づいた楽曲データの作成　メロディと歌詞の関係性を分析し，自動歌詞生成モデルを構築するために，メロディと歌詞が対応づいた楽曲データを用意する．しかし，そのために利用可能な大規模楽曲コーパスはなく，手作業でのデータ作成も手間と時間がかかる．そこで本研究では，歌詞のテキストデータと歌声合成のメロディデータを用意し，動的計画法によるアライメント手法を用いて目的の楽曲データを自動的に作成する．この，歌声合成のメロディデータを用いて自動歌詞生成モデル用の楽曲データを作成する試みは本研究が初めてである．

2) メロディと歌詞の相関の分析　作成した楽曲データを用いて，メロディと歌詞の関係性について 3 つの分析をする：(a) メロディのノートと行・ブロックの境界に関する分析．(b) メロディの音高と歌詞の抑揚に関する分析．(c) メロディと歌詞の繰り返しに関する分析．分析の結果，作詞家が重要視している歌詞の性質を定量的に示せた．例えば，長い休符の直後では行やブロックの境界が出現しやすくなるという分析結果が得られた．

3) メロディに基づく自動歌詞生成モデルの構築　メロディと歌詞の関係性に関する分析によって得られた知見を活かし，歌詞の自動生成モデルを構築する．具体的には，文章生成の分野で成功している Recurrent Neural Network (RNN) 言語モデルを応用し，各 RNN 層にノートの種類や長さなどの特徴量ベクトルを入力する．自動生成の結果，休符や長い音符付近で行やブロックが区切られている自然な歌詞が生成された．

以下に本論文の構成を述べる．まず第 2 節では歌詞の構造分析に関する関連研究と，歌詞の自動生成に関する関連研究を概観する．第 3 節ではメロディと歌詞が対応づいたデータの作成方法について説明する．第 4 節では作成したデータを用いて，メロディと歌詞の関係性を分析する．第 5 節では分析した知見をもとに，入力メロディに対して歌詞を自動生成する深層モデルとその学習方法について説明する．第 6 節では提案モデルを用いて実際に自動生成した歌詞の例を紹介し，第 7 節では提案モデルを定量的に分析・評価する．第 8 節で論を結ぶ．

2. 関連研究

2.1 歌詞構造のモデリングに関する研究

本節では，歌詞の構造を分析した先行研究を紹介する．ほとんどの先行研究は，韻やシラブルなどの歌詞の表現法に焦点を当てている．Reddy らは文法の異なる言語の歌詞の押韻を推定するために，マルコフ過程に基づく言語に依存しない押韻モデルを構築している [13]．Greene らは有限状態トランスデューサを用いて歌詞のシラブルのパターンを推定している [14]．Mayer らは韻や品詞などの言語素性を用いた Support Vector Machine (SVM) を構築し，楽曲のジャンル分類をしている [15]．

一方，歌詞のブロック構造に焦点を当てた先行研究もある．渡邉らは音響信号に基づく楽曲構造解析で用いられる

Self-Similarity Matrix を応用して，歌詞のブロックの繰り返し構造をモデル化している [16]．また，渡邉らはブロック間のトピック遷移構造をベイジアンモデルで構築し，歌詞のテーマやストーリー展開を分析している [17], [18]．

メロディと歌詞の関係性を分析した研究として Nichols らと堤らの研究がある．Nichols らは音符と歌詞がアライメントされた 679 曲の英語楽曲データを用いて，メロディとシラブルの関係性を分析している [19]．例えば，音符の高さがピークであるときや，強い拍のときには，syllable stress の歌詞になりやすいことわかっている．一方，日本語には syllable stress という概念はないため，Nichols らの分析結果をそのまま用いることはできない．堤らは 155 曲の日本語の童謡・唱歌を対象に，メロディの音高と歌詞の抑揚の相関について分析している [20]．これらの研究は，ある時刻のメロディに対応する歌詞の分析を行っているが，行やブロックなどの楽曲の歌詞の大局構造に関する分析は行われていない．そこで本研究では，日本語の歌詞を対象に，メロディと歌詞の行・ブロックの関係性を分析する．

2.2 歌詞生成に関する研究

歌詞生成の関連研究は，単語の母音やシラブル，モーラなどの音韻論を応用した研究が多い [8], [9], [10]．例えば，Barbieri らは単語列間のマルコフ過程を利用して，韻やリズムの制約を満たした歌詞を自動生成している．また，彼らは「代名詞，動詞，前置詞，代名詞，副詞」のような品詞テンプレートを作成し，これを満たした「she knocked upon it anyway」のような文法的に正しい歌詞を生成するほか，Wikipedia 内のリンク構造から単語間の関連性を計算した Wikipedia Link-Based Measure[21] を応用し，ユーザが入力したキーワードと関連した単語を生成している．また，近年では RNN などの深層学習手法を用いて歌詞を生成する研究が行われている [11], [12]．例えば，Potash らは歌詞の行末は韻を踏みやすいという仮定のもと，行末を出力シンボルとみなした RNN を構築し，行末で韻が踏まれた歌詞の生成を試みている．しかし，これらの研究ではメロディを考慮せず，テーマとなる単語や「5-7-5」のようなモーラ数を満たした歌詞を生成しているにすぎない．

メロディを考慮した歌詞生成の研究として，Oliveira らと Ramakrishanan らの研究がある [6]．Oliveira らはポルトガル語の歌詞を対象に，メロディの拍と歌詞のシラブルの関係性を 42 曲の楽曲データを用いて統計的に分析している．さらに彼らは分析結果で得られた知見を活かし，メロディによって生成歌詞のシラブルを制約するルールを複数提案している．例えば，図 4 のように，赤で示した強い拍の場所では syllable stress になるように単語を生成している．しかし，彼らの手法は，生成する歌詞の行末の位置は予め与えられており，行やブロック構造を考慮できていない．一方，Ramakrishanan らはメロディの音符と歌詞のシラブルが対応づいた 10 曲の楽曲データを使い，メロディから行の境界やアクセントなどの中間パラメータを予測する確率モデルを構築している [7]．彼らは予測された中間パラメータを満たすような単語列を探索し，メロディに基づく自動歌詞生成を行っている．しかし，この研究では 10 曲という少ない楽曲データでモデル学習を行っているため，十分なデータ分析やモデル評価が行われていない．

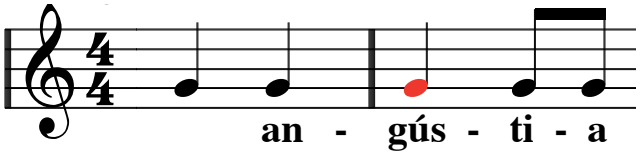

図 4 メロディの拍と歌詞のシラブル．強い拍の位置にある音符を赤く示す．

これらの自動歌詞生成に関する研究は，歌詞の 1 行もしくは 1 ブロックを生成しており，1 曲の歌詞を生成するように設計されていない．本研究では，1000 曲の楽曲データからメロディと歌詞の関係性を分析し，ブロックなどの楽曲構造を考慮した歌詞を自動生成する．

3. メロディ–歌詞対応データの作成

本研究では，メロディと歌詞の関係性の分析に適した楽曲データが必要である．しかし，メロディの各音符に対して対応する歌詞の文字とその読みが付与された楽曲データ（以下，「メロディ–歌詞対応データ」と呼ぶ）の入手は困難で，手作業で作成しようとすると大きな手間と時間がかかる．そこで本研究では，人々が歌声合成ソフトウェアを利用する際に，メロディの音符系列（楽譜情報）とその各音符に対する読み（音節）を入力していることに着目し，その歌声合成用の読み付きメロディデータを活用する．そして，さまざまなクリエータが歌声合成ソフトウェアを用いて創作し，Web 上で公開している大量の楽曲群の歌声合成用の読み付きメロディデータと，歌詞のテキストファイルとを自動的に対応付ける（アライメントする）ことで，メロディ–歌詞対応データを作成する．本研究では，作成したメロディ–歌詞対応データを用いてメロディと歌詞の関係性を分析すると同時に，歌声合成ソフトウェアを創作活動で活用している人々が行う作詞の振る舞いについても分析する．本節では，まず作成するデータの仕様について説明し，次にデータの作成方法を述べる．

3.1 データの仕様

作成する楽曲データの例を図 5 に示す．ここで各行は 1 つのノートに対応し，左から順にノート情報，モーラ情報，単語情報，境界情報である．ノート情報は左から順に．ノート ID（ノートが再生される順番），ノートの高さ（0 から 127 の各数字は音高を表す MIDI ノート番号に対応す

Vol.2017-NL-231 No.16
Vol.2017-SLP-116 No.16
2017/5/16

単語ID,表層,品詞,品詞細分類

境界情報
〈beginBlock〉：ブロック頭
〈beginLine〉：行頭

読み,抑揚の高低

ノート情報	モーラ情報	単語情報	境界情報
0, rest, 7680	〈None〉	〈None〉	〈None〉
1, 64, 240	タ, L	0, 立ち止まる, 動詞, 自立	〈beginBlock〉
2, 63, 240	チ, H	0, 立ち止まる, 動詞, 自立	〈beginBlock〉
3, 64, 240	ド, H	0, 立ち止まる, 動詞, 自立	〈beginBlock〉
4, 63, 240	マ, H	0, 立ち止まる, 動詞, 自立	〈beginBlock〉
5, 64, 240	ル, H	0, 立ち止まる, 動詞, 自立	〈beginBlock〉
6, 66, 120	ト, L	1, 時, 名詞, 非自立	〈None〉
7, 68, 600	キ, H	1, 時, 名詞, 非自立	〈None〉
8, rest, 240	〈None〉	〈None〉	〈None〉
9, 61, 480	マ, L	2, また, 接続詞, *	〈None〉
10, 63, 240	タ, H	2, また, 接続詞, *	〈None〉
11, 64, 240	フ, H	3, ふと, 副詞, 一般	〈None〉
12, 66, 240	ト, L	3, ふと, 副詞, 一般	〈None〉
13, 68, 480	フ, L	4, 振り返る, 動詞, 自立	〈None〉
14, 66, 240	リ, H	4, 振り返る, 動詞, 自立	〈None〉
15, 68, 480	カ, H	4, 振り返る, 動詞, 自立	〈None〉
16, 66, 240	エ, H	4, 振り返る, 動詞, 自立	〈None〉
17, 61, 1440	ル, H	4, 振り返る, 動詞, 自立	〈None〉
18, rest, 960	〈None〉	〈None〉	〈None〉
19, 64, 480	イ, H	5, 今, 名詞, 副詞可能	〈beginLine〉
20, 63, 240	マ, H	5, 今, 名詞, 副詞可能	〈beginLine〉

ノートID,ノートの高さ,ノートの長さ
ノートID：音の出る順番
ノートの高さ：0〜127：音高を表すMIDIノート番号に対応
rest：休符
ノートの長さ：480が四分音符に対応

図 5 行・ブロック境界情報付きメロディ–歌詞対応データ

る．rest は休符を意味する），音の長さ（480 が四分音符に相当する）を意味する．モーラ情報には単語の読みと抑揚の高低（H は高い抑揚，L は低い抑揚）を意味する．単語情報は左から順に単語 ID，表層系，品詞，品詞細分類である．境界情報の ⟨beginBlock⟩ はブロック頭，⟨beginLine⟩ は行頭を意味する．なお本研究では，1 つの音符に 1 モーラが割当てられ，休符に歌詞は割り当てられないと仮定する．

3.2 動的計画法によるメロディ–歌詞対応データの作成

以下にデータの作成手順を示す：

(1) 歌声合成のメロディデータを用意する．本研究は歌声合成ソフトウェアを活用している人々が作成した 1000 曲のメロディデータを用意した．

(2) 手順 1 で用意した楽曲と同一の歌詞のテキストデータを用意する．なお，用意したテキストデータは行とブロックの境界が与えられている．

(3) 手順 2 で用意したテキストデータに対して，形態素解析を行いモーラ情報と単語情報を抽出する．本研究では形態素解析器として MeCab を使う．なお，モーラ情報の抽出のために UniDic 辞書を用い，単語情報の抽出のために IPA 辞書を用いる．

(4) 手順 1 で収集した「メロディ–読みデータ」と，手順 2, 3 で収集した「歌詞–読みデータ」を自動アライメントすることで，目的の「メロディ–歌詞対応データ」を作成する．具体的には，2 つの DNA 塩基配列をアライメントする動的計画法として知られる Needleman-Wunsch アルゴリズムを用いる [22]．

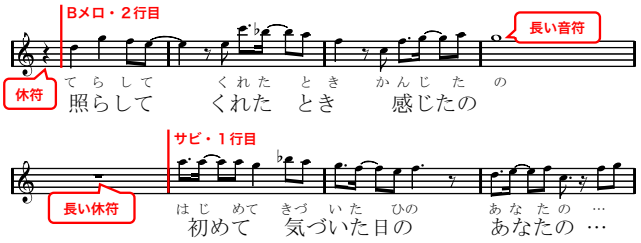

図 6 ノートの種類・長さと行・ブロックの境界

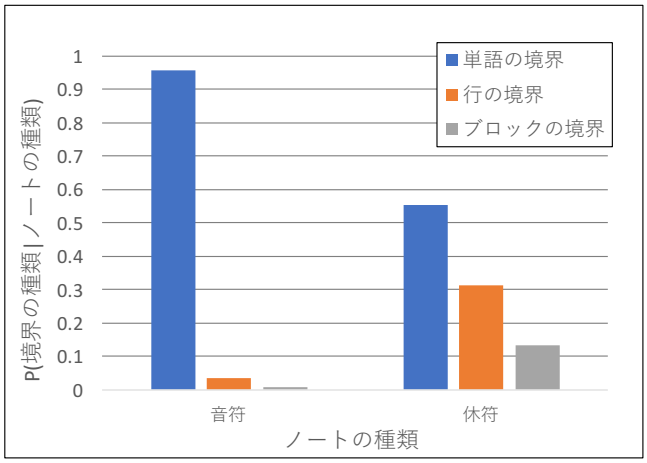

図 7 音符もしくは休符の後に行やブロックの境界が出現する確率

上記の手順により，図 5 のような行やブロックの境界情報付きのメロディ–歌詞対応データを作成した．

4. メロディと歌詞の相関に関する分析

本節では，作成した 1000 曲の楽曲データを用いてメロディと歌詞の相関について 3 つの分析をする．まず，ノートと歌詞の行・ブロックの関係性について分析し（4.1 節），次にメロディの音高と歌詞の抑揚の関係性を分析する（4.2 節）．最後にメロディと歌詞の繰り返しについて分析する（4.3 節）．

4.1 メロディのノートと歌詞の行・ブロック

作詞家はメロディの音符や休符を考慮して歌いやすく聴きやすい歌詞を創作すると言われている [2], [3], [4]．したがって，ノートの種類や長さによって歌詞の構造が異なると考えられる．例えば，図 6 のように長い音符や休符の直後で歌詞の行やブロックが区切れることがある．この仮説を検証するために，ノートの種類や長さの条件を変えたときの行やブロック境界の出現確率を計算する．

図 7 に音符・休符の直後に単語・行・ブロック境界が出現する確率を示す．この図より，休符の直後に行の境界が出現する確率は 31%，ブロックが出現する確率は 13% となり，音符の直後に境界が出現する確率より高いことがわかる．この結果より，休符の直後の方が行やブロックの境界が配置されやすいことが検証できた．

Vol.2017-NL-231 No.16
Vol.2017-SLP-116 No.16
2017/5/16

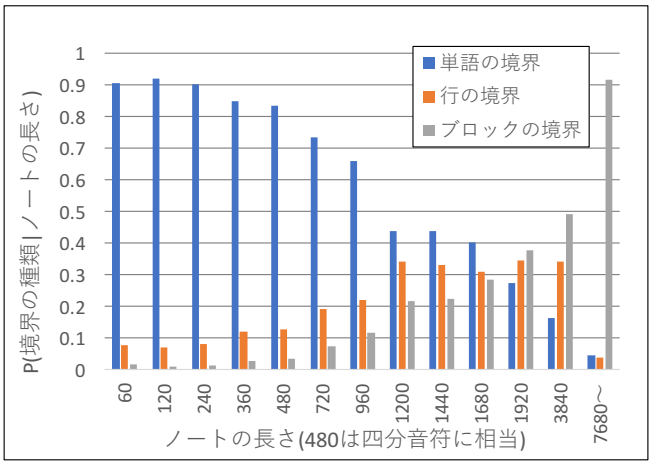

図 8 ある長さのノートの後に行やブロックの境界が出現する確率

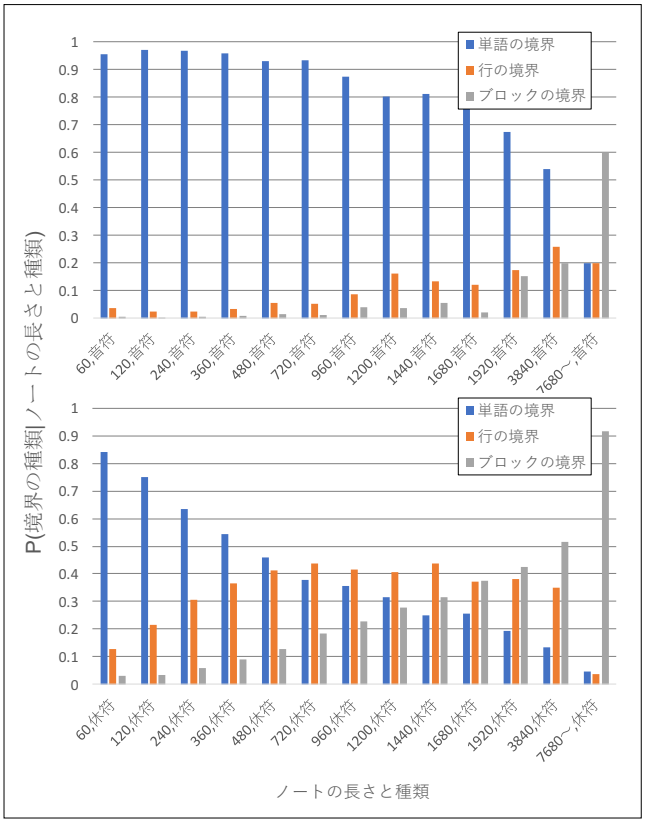

図 9 ノートの長さや種類を条件とした時の行やブロックの境界が出現する確率

次に図 8 に，ある長さのノートの直後に単語・行・ブロック境界が出現する確率を示す．この図より，ノートが長くなるにつれて単語の境界の出現確率は減少し，行とブロックの境界の出現確率が増加することが確認できた．特にノートが 1920（4 拍分の長さ）より長くなると単語の境界よりも行とブロックの境界の方が出現しやすくなる結果となった．

最後に図 9 に，長さと種類が異なるノートの直後に単語・行・ブロック境界が出現する確率を示す．図 9 の上のグラフを見ると，長さが 60 から 3840 の音符の直後では単語の境界が出現する場合がほとんどだが，長さが 7680 以上の

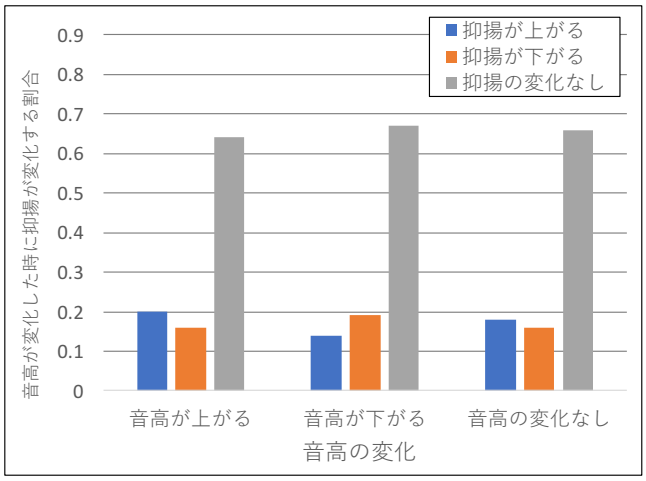

図 10 メロディの音高の変化に対する歌詞の抑揚が変化する割合

音符の直後ではブロック境界が出現する確率が高いことがわかる．一方，図 9 の下のグラフを見ると，長さが 720 から 1440 の休符の直後では行の境界になる確率が高くなり，長さが 1680 以上の休符の直後ではブロックの境界になる確率が高いことがわかる．この結果は，休符が長くなるほど行の境界，ブロックの境界と順に出現する確率が高くなることを意味する．

以上の結果より，ノートの種類や長さによって行やブロックの境界の出現分布に偏りがあるということが定量的に確かめられた．

4.2 メロディの音高と歌詞の抑揚

作詞家はメロディの音高に合わせて歌詞の抑揚に変化を持たせると言われている．堤らは日本の童謡・唱歌において，メロディの音高と歌詞の抑揚の相関を分析している [20]．本研究では，ノートの音高の変化とモーラの抑揚の変化を「上がる」，「下がる」，「変化なし」の 3 つの状態に分け，音高の変化に対する抑揚の変化する割合を計算する．図 10 に得られた割合を示す．この図より，音高が上がると抑揚が上がる割合が高くなり，音高が下がると抑揚が下がる割合が高くなることがわかる．しかし，これらの割合の値は約 20%であり，抑揚が変化しない場合と比較して小さな値である．また，抑揚が上がる割合と抑揚が下がる割合の差は 5%しかなかった．

4.3 メロディと歌詞の繰り返し

楽曲の「1 番」と「2 番」のように似たメロディが繰り返される場所では，歌詞も似せることが一般的である．したがって，似たメロディが繰り返される箇所では，似た単語が出現しやすくなるはずである．本研究では，この仮説を以下の手順で検証する：

(1) ある歌詞の t 番目の単語を $word^t$ とし，$word^t$ の最初の読みに対応するノートを $note^{i(t)}$ とする．ここで $i(t)$ は単語の位置 t からメロディ中のノートの位置を返す

表 1　メロディと歌詞の繰り返しに関する統計

	完全一致の単語ペアを含む		完全一致の単語ペアを予め除外	
	メロディが似ている単語ペア	ランダムに選んだ単語ペア	メロディが似ている単語ペア	ランダムに選んだ単語ペア
完全一致の割合	**26.74%**	1.41%	None	None
韻を踏んでいる（母音が部分一致する）割合	32.58%	5.65%	**7.97%**	4.30%
意味が似ている割合	32.21%	4.81%	**7.73%**	3.47%

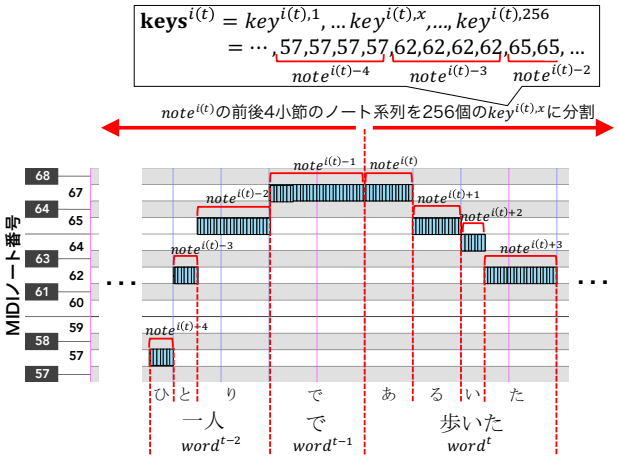

図 11　メロディの音高の変化に対する歌詞の抑揚が変化する割合

関数であり，t と $i(t)$ は異なるインデックスであることに注意されたい（ただし $1 \leq t \leq T$，$1 \leq i(t) \leq I$）．

(2) 図 11 のように，$note^{i(t)}$ を中心とした前後 4 小節のノート系列を時間軸方向に 256 分割する[*2]．つまり，8 小節を 256 分割したことになり，楽曲が 4/4 拍子のときは 4 分音符が 8 分割され，32 分音符の分解能となる．ここで，256 分割した系列を $\mathbf{keys}^{i(t)} = key^{i(t),1}, key^{i(t),2}, ..., key^{i(t),256}$ とする．各 key の値は $0 \leq key \leq 127$ であり，音高を表す MIDI ノート番号に対応する．なお本研究では，休符の key の値を 0 に設定した．

(3) 単語 $word^t$ に対応する系列 $\mathbf{keys}^{i(t)}$ と単語 $word^{t'}$ に対応する系列 $\mathbf{keys}^{i(t')}$ の類似距離を以下の式で計算する：

$$distance(\mathbf{keys}^{i(t)}, \mathbf{keys}^{i(t')}) = \sum_{x=1}^{256} |key^{i(t),x} - key^{i(t'),x}| \quad (1)$$

本研究では $1 \leq t' \leq t-7$ の範囲[*3]のうち，$distance(\mathbf{keys}^{i(t)}, \mathbf{keys}^{i(t')}) \leq 100$ かつ距離が最も近い系列ペアを，最も似たメロディとする．

(4) 最も似ている系列 $\mathbf{keys}^{i(t)}$ と $\mathbf{keys}^{i(t')}$ に対応する単語ペア $word^t, word^{t'}$ を収集する．

(5) 収集した単語ペア $word^t, word^{t'}$ において，(a) 完全一致であるか，(b) 韻を踏んでいるか（母音が部分一致しているか），(c) 意味が似ているかを判定し，その頻度を計算する．なお，単語の意味が似ているか判定するために，日本語 Wikipedia で構築した Word2Vec[*4] を用いて cos 類似度を計算する [23]．本研究では，類似度が 0.5 以上であれば単語の意味が似ていると仮定する．

以上の手順で得られた統計量を用いて，メロディと歌詞の繰り返しに関する割合を計算する．表 1 に，完全一致，押韻，意味が似ている単語ペアに関する統計を示す．この表から，ランダムに単語ペアを選ぶよりも，メロディが似ているときのほうが単語ペアが完全一致する割合，単語ペアが韻を踏む割合，単語ペアの意味が似る割合が高いことがわかる．特に，単語ペアが完全一致になる割合はランダムに選んだ単語ペアより約 25%も高くなり，メロディが似ている時は単語も繰り返しやすいということが定量的に確かめられた．一方，予め完全一致する単語ペアを除外して割合を計算した結果，ランダムに選んだ単語ペアと比較して，メロディが似ている単語ペアが押韻，もしくは似た意味である割合は約 3%しか高くならなかった．

本研究では，メロディが似ている場所での単語ペアに関する統計を計算した．しかし，実際の歌詞では，「ほとんど同じ単語を使っているが一部分だけが異なる文」が多いと考えられる．単語の繰り返しだけでなく，文節や行などの大きいフレーズが繰り返されているとき，フレーズ間にどのような違いがあるかを分析することが今後の課題である．

5. 深層学習を用いたメロディに基づく自動歌詞生成手法

本節では，メロディと歌詞の関係性の分析結果を活かし，メロディに基づく自動歌詞生成の手法を提案する．

5.1 入力と出力

提案モデルの入力はノート（音符または休符）の系列である．ただし，ある時刻に 2 つ以上のノートは重ならないと仮定する．本研究では，入力のノート系列を $notes = note^1, ..., note^I$ と定義する．各 $note$ は音の高さ（0 から 127 の音高を表す MIDI ノート番号もしくは休符）と長さ[*5]の情報を持つ．

*2 なお，先頭単語のノート系列と最終単語のノート系列を 256 分割するために，先頭ノート $note^{i(t)=1}$ と最終ノート $note^{i(t)=I}$ は 4 小節分の長さの休符にしている．

*3 本研究では，最低でも歌詞の 1 行以上離れた位置にあるメロディ同士を比較するために，1 行あたりの平均単語数である 7 以上離れた単語のノート系列を比較する．

*4 http://www.cl.tohoku.ac.jp/jawiki_vector/

*5 ノートの長さは 60, 120, 240, 360, 480, 720, 960, 1200, 1440,

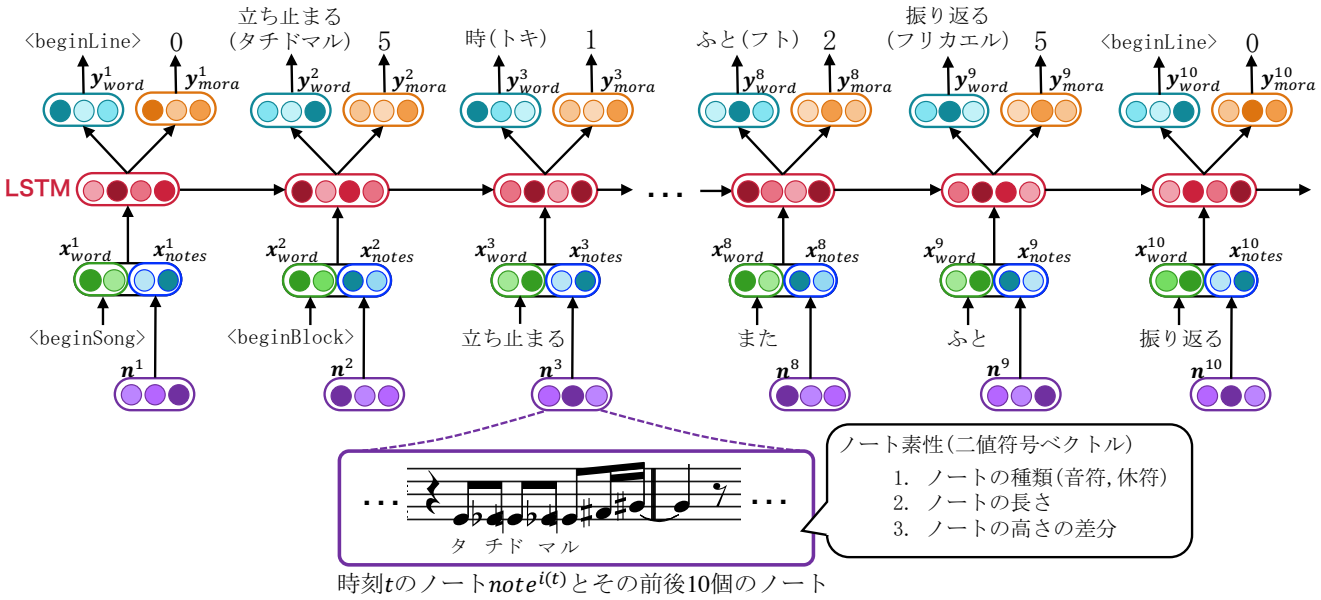

図 12　LSTM-RNN 言語モデルを用いたメロディに基づく自動歌詞生成モデル

本研究では，入力したノート系列に対して自然な歌詞を出力するようにモデルを設計する．ここで，出力する単語列の読みは入力した各ノートと対応している．さら，行の境界 ⟨beginLine⟩ とブロックの境界 ⟨beginBlock⟩ も出力シンボルとして扱う．

5.2　モデル構成

本研究では，流暢な文章の自動生成に成功している Recurrent Neural Network (RNN) 言語モデルを用いる [24]．具体的には，RNN の拡張モデルの中でも長距離時系列データの学習が可能である Long Short-Term Memory (LSTM) を用いて，ノート系列に対して単語を生成するモデルを構築する．図 12 に提案する歌詞生成モデルの概観を示す．t 番目の単語を出力する LSTM ユニットに単語埋め込みベクトル $\mathbf{x}^t_{word}$ とノート埋め込みベクトル $\mathbf{x}^t_{notes}$ を連結したベクトル $\mathbf{x}^t_{concat}$ を入力する：

$$\mathbf{x}^t_{concat} = \mathbf{x}^t_{word} \oplus \mathbf{x}^t_{notes} \tag{2}$$

ここで，$\oplus$ はベクトルの連結を表す．本研究では単語埋め込みベクトルとノート埋め込みベクトルの次元数を 512 とする．単語・ノート埋め込みベクトルを以下の式で計算する：

$$\mathbf{x}^t_{word} = \mathbf{W}_{word\_in}\mathbf{v}^t \tag{3}$$

$$\mathbf{x}^t_{notes} = \mathrm{ReLU}(\mathbf{W}_{notes\_in}\mathbf{n}^t + \mathbf{b}_{notes\_in}) \tag{4}$$

ここで，$\mathbf{v}^t$ は $t-1$ 番目に生成された単語の語彙数次元 1-Hot ベクトルである．なお，$t=1$ では曲のはじまりを意味するシンボル ⟨beginSong⟩ を入力する．ノート埋め込みベクトルの計算には ReLU 活性化関数を用いた．$\mathbf{n}^t$ はノート系列を特徴量化した N 次元二値符号ベクトルを表し，$note^{i(t)}$ を中心とした前後 10 個のノート系列 $note^{i(t)-10}$,...,$note^{i(t)}$,...,$note^{i(t)+10}$ をエンコードしたベクトルである．$i(t)$ は単語の位置 t からノートの位置 i を返す関数である．このノート特徴量ベクトル $\mathbf{n}^t$ の詳細は次節で述べる．$\mathbf{W}_{word\_in}$ と $\mathbf{W}_{notes\_in}$ はそれぞれ単語埋め込みベクトルとノート埋め込みベクトルの重み行列である．$\mathbf{b}_{notes\_in}$ はバイアス項である．中間層には 1024 次元の LSTM を使う．出力層では語彙数次元の単語埋め込みベクトル $\mathbf{y}^t_{word}$ を softmax 関数を用いて単語確率分布として出力する：

$$\mathbf{y}^t_{word} = \mathrm{softmax}(\mathbf{W}_{word\_out}\mathbf{z}^t) \tag{5}$$

ここで，$\mathbf{z}^t$ は各時刻 t の LSTM ユニットの出力であり，$\mathbf{W}_{word\_out}$ は単語出力層の重み行列である．

本研究では，図 1 の歌詞のように，休符を挟んで単語を生成するのを防ぐために，モーラ数を推定する出力層 $\mathbf{y}^t_{mora}$ を追加する：

$$\mathbf{y}^t_{mora} = \mathrm{softmax}(\mathbf{W}_{mora\_out}\mathbf{z}^t) \tag{6}$$

ここで，$\mathbf{W}_{mora\_out}$ はモーラ数出力層の重み行列である．なお，本研究では 0 から 10 のモーラ数を出力対象とし[*6]，モーラ数が 11 以上の単語は学習対象外とする．このモーラ数出力層により，時刻 t において d 個先のノートが長い休符のときに，モーラ数が d 以下の単語が生成されやすくなることが期待される．

出力層 $\mathbf{y}^t_{word}$ と $\mathbf{y}^t_{mora}$ の損失関数は categorical cross-entropy であり，最適化手法には Adam を用い，バッチサ

1680, 1920, 3840, 5760, 7680 より長いノートのいずれかの値を持つ．480 の値が 4 分音符と同じ長さとなる．

[*6] 促音（ッ），長音（ー），行・ブロックの境界 ⟨beginLine⟩, ⟨beginBlock⟩ のモーラ数を 0 と定義する

あなたは私のことを見ていたの
夢を見ていたんじゃないのに
私のために君がいるなら
今はまだ忘れていたいな

僕は笑っているんだよ
かけがえのないものだから
俺がいてくれたのは何か

変わらない雨の中で
目を閉じてたあの頃は
何も変わらずに笑っていたね

あなたの手を離してみたい
離さないで
この世界があるのなら信じてあげるから

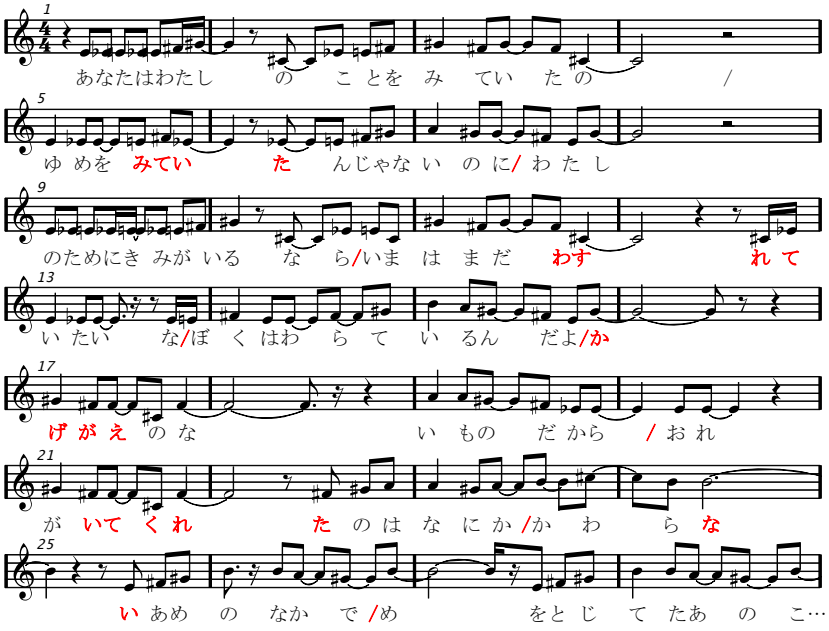

図 13 5 万曲の歌詞データで学習した親モデルを用いて生成した歌詞の例．休符をまたいで生成された単語と，2 つの音符間に生成された行・ブロックの境界 (/) を赤く示した．（この例示では，入力するノート系列として RWC 研究用音楽データベース RWC-MDB-P-2001 No.7 のメロディを用いた．実際の評価では，作成したメロディ–歌詞対応データをノート系列として入力している．）

イズは 32，学習率を 0.001 にする．また，5 エポックで損失関数の値が向上しない場合，学習の早期打ち切り (early stopping) を行う．

5.3 メロディ素性

提案モデルに入力するノート系列を以下の 3 つの方法でエンコードし，ノート特徴量ベクトル $\mathbf{n}^t$ を作る．

ノートの種類 4.1 節の分析結果より，休符の直後では行やブロックの境界が出現しやすいことがわかっている．そこで，ノートの種類（音符と休符）を表す二値符号ベクトルを用意する．

ノートの長さ 4.1 節の分析結果より，長いノートの直後では行やブロックの境界が出現しやすいことがわかっているため，ノートの長さを表す二値符号ベクトルを用意する．

ノートの高さの差分 ノートの種類，長さに加え，ノートの高さも素性ベクトルとして加える．具体的には 1 つ前のノートとの高さの差分を素性として加える．例えば，ノート系列が「65（ファ）・68（ソ♯）」のとき，高さの差分は +3 となり，+3 に対応する場所が発火する二値符号ベクトルを入力する．

5.4 転移学習

一般的に深層モデルの学習には大量のデータが必要となる．しかし，本研究で提案した LSTM-RNN 言語モデルの学習に使用できる楽曲データは 1000 曲しかないため過学習する可能性がある．一方，歌詞のテキストデータであれば大量に用意することが可能である．そこで我々は先に歌詞の言語モデルを学習した後にメロディ–歌詞対応データを再学習することでモデルの汎化能力を向上させる．この一連の学習方法を本研究では転移学習と呼ぶ．

まず大量の歌詞のテキストデータを使ってパラメータを事前学習する．このとき，ノートの特徴量 $\mathbf{n}^t$ は値が全て 0 のベクトルを用いる．この事前学習により得られたモデルを**親モデル**と呼ぶ．

次に，親モデルのパラメータを引き継いだ状態で，メロディ–歌詞対応データを用いて提案モデルを再学習する．この再学習で得られたモデルを**子モデル**と呼ぶ．

本研究では，親モデルの学習のために，54181 曲の日本語ポピュラー楽曲の歌詞データを用いる．子モデルの学習には，3 節で作成したメロディ–歌詞対応データの 1000 曲のうち 900 曲を使用し，残りの 100 曲を評価データとする．なお，学習対象となる単語は 54181 曲の歌詞データ中のモーラ数が 1 から 10 の範囲かつ，文書頻度 (Document Frequency) が上位 2 万の日本語とし，この条件を満たさない単語は ⟨ 未知語 ⟩ という単語として学習する．

6. 生成された歌詞の例

本研究では，提案モデルが聴きやすい歌詞を生成できたかを定性的に分析するために，メロディを考慮していない親モデルを使って生成した歌詞と，転移学習した子モデルを使って生成した歌詞を比較する．入力するノート系列として 100 曲の評価用メロディ–歌詞対応データを用いる．自

夢の下でも君は変わらないな
二人だけで僕を見ていたんだよ

消えてしまうまでの言葉をのせていい
君のことなんて出来ないんだよと見て
私の手の証を隠してしまうよ

だからどうしようかな
そうなの
いつまで誰も見つけられないの
二人だけになってしまうのよね

さあ

君を見たいの
何度も
君のことが言っても
二人の中でいてしまえばいいのにな

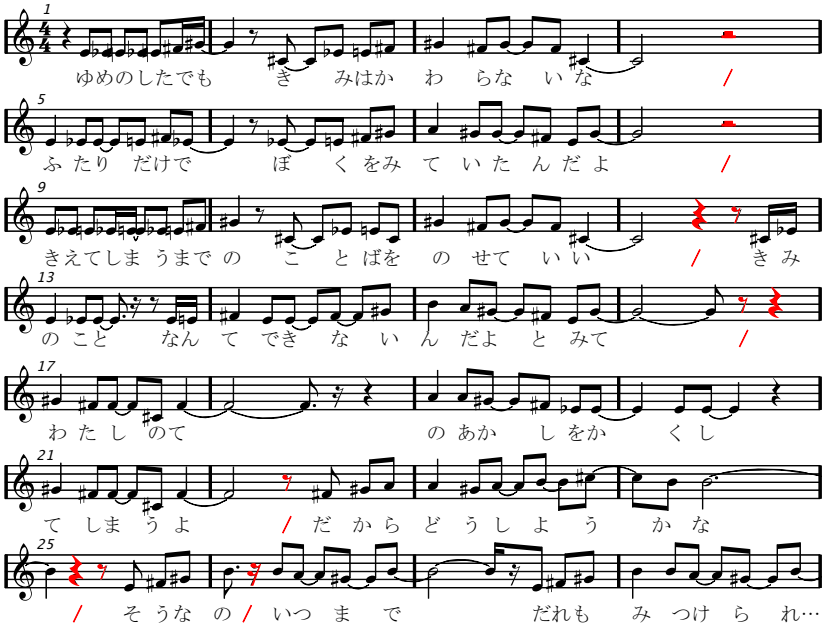

図 14 転移学習した子モデルを用いて生成した歌詞の例．同じ位置に存在する休符と行・ブロックの境界 (/) を楽譜中に赤く示した．

表 2 転移パラメータの固定箇所が異なる子モデルと，その生成性能

	入力単語ベクトル	LSTM ユニット	出力モーラ数・単語ベクトル	PPL	⟨beginLine⟩ F-measure	⟨beginBlock⟩ F-measure
設定 1	固定	再学習	固定	128.1	0.178	0.119
設定 2	固定	再学習	再学習	**123.7**	**0.270**	**0.166**
設定 3	再学習	再学習	再学習	146.8	0.246	0.131

動生成アルゴリズムには，窓幅 10 のビーム探索を用いる．

図 13 は，親モデルを使って生成した歌詞とメロディの一部である．図の左には，行・ブロック境界を含んだ歌詞を示し，右には，楽譜と歌詞の読み，行・ブロック境界を示す．なお，楽譜中のスラッシュ記号 (/) は行・ブロックの境界を表す．この図より，親モデルは流暢な文を生成していることがわかる．しかし楽譜中の歌詞を見ると，不自然な箇所に単語や行・ブロックの境界が生成されていることがわかる．例えば，単語「心」が休符を挟んで生成されていたり，旋律の途中で行が区切られている．このように，テキストデータだけで学習した LSTM-RNN 言語モデルは，流暢だが聴きにくい歌詞を生成してしまう．

図 14 に，転移学習した子モデルが生成した歌詞を示す．図 13 の生成結果に比べて流暢ではない歌詞が生成されている．一方，楽譜と一緒に生成された歌詞を見ると，赤く示した休符の位置で行やブロックの境界が生成されている．このように，転移学習した子モデルは流暢とはいえないが，行・ブロック構造を考慮した歌詞を生成している．これは，少ないデータで言語モデルを再学習したことで，過学習が起こったことが原因として挙げられる．

7. 定量評価

本節では，提案モデルが流暢かつ行やブロックの境界を考慮した歌詞を生成できるか定量的に評価する．

表 3 モーラ数出力層 $\mathbf{y}_{mora}$ の有無による提案モデル生成性能差

モデル	PPL	⟨beginLine⟩ F-measure	⟨beginBlock⟩ F-measure
Model	**123.7**	**0.270**	**0.166**
Model$\neg\mathbf{y}_{mora}$	128.2	0.230	0.120

7.1 評価指標

本研究では，以下の 2 つの評価指標を用いる．

Perplexity 言語モデルの一般的な評価指標である Perplexity (PPL) を用いる．PPL は言語の複雑性を意味しており，PPL が小さいモデルほど流暢な文を生成できることを意味する．本研究では，評価データの歌詞の PPL を計算することで，歌詞の生成能力を評価する．

行・ブロック境界の生成精度 提案モデルが入力ノート系列に対して，自然な位置に行・ブロック境界を生成できるか評価する．具体的には，評価用のノート系列を入力して生成した行境界 $\langle\text{beginLine}\rangle_{gen}$ とブロック境界 $\langle\text{beginBlock}\rangle_{gen}$ の位置が，オリジナルの歌詞の行境界 $\langle\text{beginLine}\rangle_{ref}$ とブロック境界 $\langle\text{beginBlock}\rangle_{ref}$ の位置とどれだけ一致しているか，Recall, Precision, F-measure を用いて評価する．

表 4　転移学習の有効性

モデル	PPL	⟨beginLine/Block⟩を除いた PPL	⟨beginLine⟩			⟨beginBlock⟩		
			Recall	Precision	F-measure	Recall	Precision	F-measure
親モデル	130.2	**207.4**	0.057	0.058	0.058	0.181	0.023	0.041
1000 曲のメロディ–歌詞対応データのみを学習した子モデル	147.5	254.7	**0.299**	**0.295**	**0.297**	**0.234**	**0.161**	**0.191**
転移学習をした子モデル	**123.7**	215.5	0.277	0.263	0.270	0.188	0.149	0.166

7.2　転移するパラメータによる性能差

本研究で行った転移学習では，親モデルのパラメータを初期値として受け継ぎ，子モデルのパラメータを再学習する．その際，親モデルから引き継いだパラメータを再学習せずに固定するかどうかによって生成性能に差が生じることが考えられる．本節では，パラメータの再学習の条件を変えたときの性能差を評価することで，最適な転移学習の条件を模索する．表 2 にパラメータを再学習する場所を変えた子モデルと，その生成性能を示す．この表より，親モデルの入力単語ベクトルのパラメータを固定し，他のパラメータを再学習した「設定 2」の子モデルが最も高い生成性能であることがわかった．この理由として，以下の 2 つが考察できる：

- 「設定 1」の子モデルは，親モデルの出力モーラ数・単語ベクトルのパラメータを使用している．そのため，設定 1 では入力メロディに対して歌いやすい位置に行やブロックの境界を配置できなかったと考えられる．
- 大量のテキストデータを用いて親モデルを事前学習したため，汎化性能の高い単語ベクトル x^t_{word} のパラメータが得られたと考えられる．さらにこのとき，「設定 3」の子モデルは少ないデータで入力単語ベクトルを再学習したため，過学習が起こり性能が下がったと考えられる．

これ以降，本研究では設定 2 のモデルを最適な転移学習モデルとして扱い，議論を進める．

7.3　モーラ数の出力層による効果

休符をまたいで単語を生成することを防ぐために，提案した LSTM ではモーラ数を予測する出力層 $\mathbf{y}^t_{mora}$ を追加している．このモーラ数出力層 $\mathbf{y}^t_{mora}$ の効果を調べるために，$\mathbf{y}^t_{mora}$ を除いたモデルを用意して生成性能を比較する．表 3 にモーラ数出力層 $\mathbf{y}^t_{mora}$ を加えたモデル $Model$ と除外したモデル $Model\neg\mathbf{y}_{mora}$ の生成性能を示す．この表より，提案モデルのほうが生成性能が高い結果となり，モーラ数を予測する出力層が，メロディに基づく歌詞生成に寄与していることが示唆された．

7.4　転移学習の効果

転移学習の有効性を確かめるために，(1) 5 万曲分の歌詞のテキストデータで学習した親モデル，(2) 転移学習せずに 1000 曲のメロディ–歌詞対応データのみで学習した子モデル，(3) 転移学習した子モデルの 3 つのモデルの生成性能を比較する．表 4 にそれぞれのモデルの生成性能を示す．親モデルと転移学習をしない子モデルに対して，転移学習をした子モデルの PPL が 123.7 と最も小さく，出力シンボルの予測性能が最も高いことがわかる．しかし，図 13 と図 14 に示した歌詞の生成結果のように，さまざまな出力結果を見比べると親モデルの方が流暢な歌詞を生成しており，この表の PPL の結果とは矛盾する．これは，子モデルの出力シンボル ⟨beginLine⟩ と ⟨beginBlock⟩ の予測性能が，親モデルよりも高いためだと考えられる．そこで，⟨beginLine⟩ と ⟨beginBlock⟩ を除外した PPL を計算した結果，親モデルの PPL の性能が最も高く，次に転移学習した子モデルの性能が高くなった．この結果は，転移学習した子モデルは親モデルより流暢でない歌詞を生成することを意味する．ただし，いずれの PPL の計算方法においても，転移学習をしない子モデルよりも，転移学習をした子モデルのほうが性能は高く，転移学習が歌詞の予測性能の向上に寄与していることがわかった．

一方，生成された歌詞の行・ブロックの境界の精度については，転移学習をした子モデルよりも，転移学習をしない子モデルのほうが高性能となった．しかし，親モデルと比べて，転移学習をした子モデルの ⟨beginLine⟩ の F-measure は約 26%改善し，⟨beginBlock⟩ の F-measure は約 16%改善しており，行やブロックの境界を考慮した歌詞が生成されていることがわかる．

以上の PPL と行・ブロック境界の精度の結果から，提案モデルが「言葉遣い」と「歌いやすさ・聴きやすさ」の両方をバランス良く学習していることが示唆された．

7.5　生成された歌詞とメロディと相関

本節では，4.1 節で得られた「長い休符の後には行やブロックの境界が配置されやすい」という傾向がモデル学習に反映されているか確かめる．具体的には，どんなノートの直後に行とブロックの境界が生成されたかを計算する．図 15 に，音符または休符の直後にどんな境界が生成されたか，その割合を示す．この図より，休符の直後に行やブロックの境界が多く生成されていることがわかった．この結果は 4.1 節で得られた図 7 の結果とほぼ同じである．次に図 16 に，ある長さのノートの直後に生成された境界の割合を示す．この図より，ノートが長くなるほど行の境界，ブロックの境界と順に生成されやすくなることがわかった．

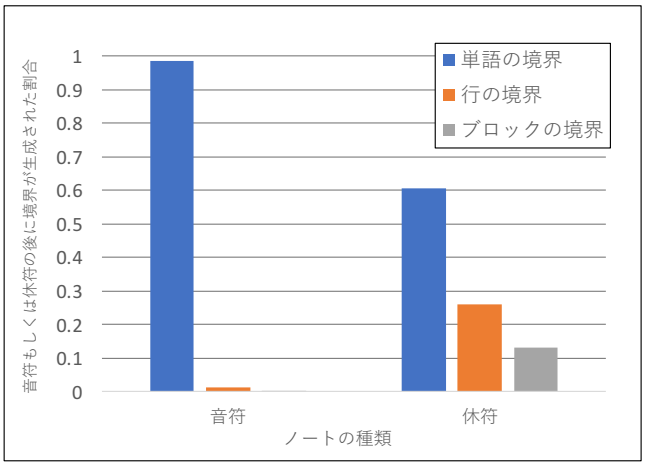

図 15 音符もしくは休符の直後に行やブロックの境界が生成された割合

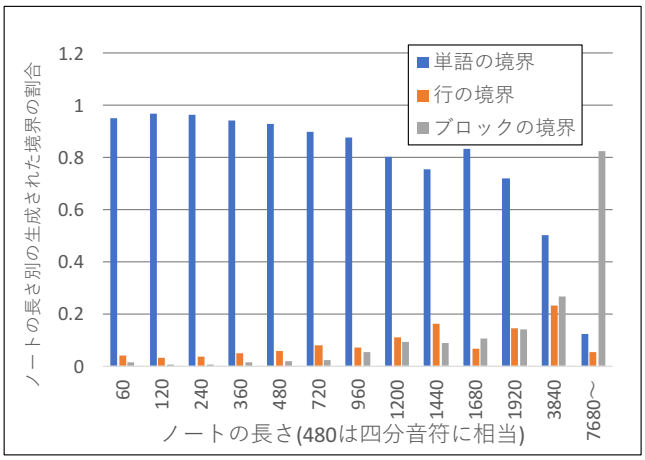

図 16 ある長さのノートの直後に行やブロックの境界が生成された割合

この傾向も 4.1 節で得られた図 8 の結果と同じである．これらの結果は，実際のメロディと歌詞の関係性を，モデルが正しく学習できていることを意味する．

8. おわりに

本研究では，メロディに基づく歌詞の自動生成を可能にするために，(1) メロディと歌詞が対応づいた楽曲データの作成，(2) メロディと歌詞の相関の統計的分析，(3) 入力メロディに合った歌詞の自動生成を行った．

メロディと歌詞が対応づいたデータを作成するために，本研究では音符と読みが対応づいているメロディデータと，行の境界や A メロ・B メロ・サビなどのブロックの境界情報が付与されている歌詞のテキストデータを自動アライメントし，1000 曲のメロディ–歌詞対応データを作成した．

また，作成したデータを用いてメロディと歌詞の相関を統計的に分析した結果，「長い休符の直後に行やブロックの境界が配置されやすい」，「メロディの音高が上がると歌詞の抑揚も上がりやすい」，「メロディが繰り返されている位置では，似たフレーズの歌詞が作詞されやすい」などの結果が得られた．

本研究では，入力メロディに対して自然な歌詞を自動生成するために，RNN 言語モデルを応用した．具体的には，RNN に音符や休符の情報を入力し，行やブロックの境界などの楽曲構造も生成できるモデルを構築した．また，1000 曲という少ない楽曲データで提案モデルを学習するために，本研究では予め約 5 万曲の歌詞のテキストデータを用いて RNN を事前学習し，そのパラメータを引き継いで少量のデータを学習する「転移学習」の枠組みを利用した．

転移学習した提案モデルを用いて歌詞を自動生成した結果，長い休符の直後に行やブロックの境界を配置した自然な歌詞が生成された．この傾向は，パープレキシティや，生成された行・ブロックの境界位置の精度などの定量的な評価結果からも確認できた．

しかし，提案した自動歌詞生成モデルは単純な RNN 言語モデルより流暢な歌詞を生成できていない．さらには，分析で得られた「メロディと歌詞の繰り返し」に関する知見も活かしきれていない．そのため，今後は楽曲の繰り返し構造を組み込むことで，メロディに対してより自然な歌詞の自動生成を目指す．

謝辞 本研究は，RWC 研究用音楽データベース (ポピュラー音楽) を利用した．本研究は科研費 JP16J05945 の助成を受けた．本研究の一部は JST ACCEL (JPMJAC1602) の支援を受けた．

参考文献

[1] Austin, D., Peterik, J. and Austin, C. L.: *Songwriting for Dummies*, Wileys (2010).
[2] 上田起士：よくわかる作詞の教科書，YAMAHA music media corporation (2010).
[3] 北村英明：これから始める人のための作詞入門，メトロポリタンプレス (2012).
[4] 井筒日美：ゼロからの作詞入門，YAMAHA music media corporation (2015).
[5] Goto, M., Hashiguchi, H., Nishimura, T. and Oka, R.: RWC Music Database: Popular, Classical and Jazz Music Databases., *Proceedings of the 3rd of International Society for Music Information Retrieval (ISMIR 2002)*, Vol. 2, pp. 287–288 (2002).
[6] Oliveira, H. R. G., Cardoso, F. A. and Pereira, F. C.: Tra-la-lyrics: an approach to generate text based on rhythm, *Proceedings of 4th International Joint Workshop on Computational Creativity*, pp. 47–55 (2007).
[7] A, A. R. and Devi, S. L.: An alternate approach towards meaningful lyric generation in Tamil, *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, pp. 31–39 (2010).
[8] Abe, C. and Ito, A.: A Japanese lyrics writing support system for amateur songwriters, *Proceedings of Asia-Pacific Signal & Information Processing Association Annual Summit and Conference 2012 (APSIPA ASC 2012)* (2012).
[9] Barbieri, G., Pachet, F., Roy, P. and Esposti, M. D.: Markov constraints for generating lyrics with style, *Pro-*

ceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012), pp. 115–120 (2012).

[10] Watanabe, K., Matsubayashi, Y., Inui, K., Nakano, T., Fukayama, S. and Goto, M.: LyriSys: An Interactive Support System for Writing Lyrics Based on Topic Transition, *Proceedings of the 22Nd International Conference on Intelligent User Interfaces (IUI 2017)*, pp. 559–563 (2017).

[11] Potash, P., Romanov, A. and Rumshisky, A.: GhostWriter: Using an LSTM for Automatic Rap Lyric Generation., *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pp. 1919–1924 (2015).

[12] Ghazvininejad, M., Shi, X., Choi, Y. and Knight, K.: Generating Topical Poetry, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, pp. 1183–1191 (2016).

[13] Reddy, S. and Knight, K.: Unsupervised discovery of rhyme schemes, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pp. 77–82 (2011).

[14] Greene, E., Bodrumlu, T. and Knight, K.: Automatic analysis of rhythmic poetry with applications to generation and translation, *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pp. 524–533 (2010).

[15] Mayer, R., Neumayer, R. and Rauber, A.: Rhyme and style features for musical genre classification by song lyrics, *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pp. 337–342 (2008).

[16] Watanabe, K., Matsubayashi, Y., Orita, N., Okazaki, N., Inui, K., Fukayama, S., Nakano, T., Smith, J. and Goto, M.: Modeling Discourse Segments in Lyrics Using Repeated Patterns, *Proceedings of the 26th International Conference on Computational Linguistics (COLING 2016)*, pp. 1959–1969 (2016).

[17] Watanabe, K., Matsubayashi, Y., Inui, K. and Goto, M.: Modeling Structural Topic Transitions for Automatic Lyrics Generation, *Proceedings of The 28th Pacific Asia Conference on Language, Information and Computation (PACLIC 2014)*, pp. 422–431 (2014).

[18] 渡邉研斗，松林優一郎，乾健太郎，深山　覚，中野倫靖，後藤真孝：ストーリー展開と一貫性を同時に考慮した歌詞生成モデル，人工知能学会第 30 回全国大会発表論文集 (2016).

[19] Nichols, E., Morris, D., Basu, S. and Raphael, C.: Relationships Between Lyrics and Melody in Popular Music., *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR 2009)*, pp. 471–476 (2009).

[20] 堤　彩香，平賀　譲：日本語の音韻と旋律の関係について～童謡・唱歌を中心に～，音楽情報科学研究会，Vol. 2014, No. 5, pp. 1–6 (2014).

[21] Milne, D. and Witten, I. H.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links, *Proceeding of the AAAI 2008 Workshop on Wikipedia and Artificial Intelligence*, pp. 25–30 (2008).

[22] Needleman, S. B. and Wunsch, C. D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, Vol. 48, No. 3, pp. 443–453 (1970).

[23] Suzuki, M., Matsuda, K., Sekine, S., Okazaki, N. and Inui, K.: Neural Joint Learning for Classifying Wikipedia Articles into Fine-grained Named Entity Types, *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation (PACLIC 2016)*, pp. 535–544 (2016).

[24] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J. and Khudanpur, S.: Recurrent neural network based language model., *Proceedings of Interspeech 2010*, pp. 1045–1048 (2010).