Lyric Jumper：アーティストごとの歌詞トピックの傾向に基づき様々な歌詞に出会える歌詞探索サービス

佃 洸摂[1,a)]　石田 啓介[1,b)]　後藤 真孝[1,c)]

概要：本稿では，アーティストごとの歌詞トピックの傾向に基づいて様々な歌詞に出会える歌詞探索サービス Lyric Jumper について述べる．各アーティストには，「恋愛」や「友情」といったトピックの歌詞が書かれやすい，といった傾向が存在する．Lyric Jumper ではそのような傾向を考慮することで，アーティストごとのトピックの傾向の可視化や，トピックの類似度に基づくアーティストの推薦などの機能を提供する．そうした機能を利用することで，Lyric Jumper によって様々なアーティストや楽曲の歌詞へのユーザの理解が深まることを目指している．Lyric Jumper の実現にあたり，本稿ではアーティストごとの歌詞トピックの傾向を推定するためのモデルを提案する．提案モデルでは，各アーティストがトピックの分布を持ち，その分布に応じて各歌詞にひとつのトピックが割り当てられる．歌詞データセットを用いた実験により，従来研究において歌詞トピックを推定する主要な手法である latent Dirichlet allocation（LDA）よりも提案モデルが優れていることを定量的に示した．さらに，web サービスとして公開した Lyric Jumper にアクセスした 92,962 ユーザの操作ログおよび，Lyric Jumper について Twitter に投稿されたコメントを収集し，Lyric Jumper の有用性について分析を行った．

1. はじめに

あるアーティストは「恋愛」に関する歌詞をよく歌い，別のアーティストは「友情」に関する歌詞をよく歌う，といったように，各アーティストには固有の歌詞の傾向が存在する．人々が音楽を聴くときは，ジャンルやムード，メロディやリズムといった音響的な特徴量だけでなく，歌詞のトピックも考慮してアーティストを選択する [1], [2]．しかし，音楽情報検索の分野では，歌詞のトピックが持つ可能性を十分に活用できているとは言えない．例えば，歌詞のトピックに応じてアーティストや楽曲を選択したり，ユーザの好みのアーティストと歌詞のトピックの傾向が類似したアーティストを推薦したりすることは難しい．本研究の目的は，楽曲の歌詞のトピックを考慮した音楽情報検索を実現することである．

この目的を果たすために本稿では，アーティストごとの歌詞トピックの傾向に基づいて様々な歌詞に出会える歌詞探索サービス *Lyric Jumper*[*1] を実現して提供する．Lyric Jumper では「一途な恋」や「青春」など 20 個のトピックを用意し，各歌詞にいずれかのトピックが割り当てられる．これにより，従来行われていた，アーティスト名や楽曲のタイトルを検索したり，歌詞内の文字列を直接検索したりする方法 [3], [4], [5] では困難であった，トピックに基づく新しいインタラクションを通じた歌詞との出会いや歌詞のより深い理解を実現する．また，Lyric Jumper では，ユーザが様々な歌詞と出会うために，トピックに基づくアーティストや楽曲の探索も実現する．これらを実現するために，アーティストごとのトピックの傾向の可視化や，トピックの類似度に基づくアーティストの推薦，歌詞内でトピックと関連の強いフレーズの強調表示などの機能を提供する．

Lyric Jumper を実現するにあたり，本稿ではアーティストごとの歌詞トピックの傾向を考慮したトピックモデルを提案する．これまで，音楽情報処理の分野で歌詞のトピックを扱う際は，トピックモデルとして latent Dirichlet allocation（LDA）[6] が主に用いられてきた．トピックモデルにおけるトピックは単語の確率分布によって表され，トピックの意味はその分布に応じて解釈される．LDA では各歌詞がトピックの確率分布を持ち，歌詞内の単語ごとにひとつのトピックを割り当てる．LDA はアーティストごとの楽曲集合を考慮しておらず，歌詞のトピックに対するアーティストごとの傾向を明示的に反映したモデルにはなっていない．それに対して提案モデルでは，各アーティストがトピックの確率分布（アーティストのトピックに対

[1] 産業技術総合研究所
[a)] k.tsukuda@aist.go.jp
[b)] ksuke-ishida@aist.go.jp
[c)] m.goto@aist.go.jp
[*1] https://lyric-jumper.petitlyrics.com

する傾向を表す）を持つ．また，歌詞を書き始める前にひとつ主題を決めることが一般的である [7], [8] ことから，各歌詞にひとつのトピックを割り当てる（以下「歌詞トピック」と呼ぶ）．つまり，ある歌詞にトピック k が割り当てられると，その歌詞内の全ての単語にもトピック k が割り当てられる．提案モデルによって推定した歌詞トピックを利用することで，LDA では実現が難しい様々なインタラクションを Lyric Jumper で提供可能になる．

本研究の主な貢献を以下に述べる．

- アーティストごとの歌詞トピックの傾向に基づいて様々な歌詞との出会いを実現する歌詞探索サービス Lyric Jumper を提案した．（3 章）
- アーティストごとの歌詞トピックの傾向を考慮し，各歌詞はひとつのトピックを持つという仮定に基づいた新しいトピックモデルを提案した．（4 章）
- 歌詞の配信企業により提供された歌詞データを用いて，既存研究で主に使用されていた LDA よりも提案モデルが優れていることを、perplexity の観点から定量的に示した．（5 章）
- 9 万人以上のユーザの操作ログを用いて，Lyric Jumper がユーザの探索行動に与える影響を分析した．また，Lyric Jumper について Twitter に投稿された 400 件以上のコメントを収集し，Lyric Jumper に対するユーザのフィードバックを分析した．（6 章）

2. 関連研究

これまでの研究で，歌詞は様々な目的のために使用されてきた．具体的には，歌詞と音響データの対応付け [9], [10], [11]，歌詞の特徴分析 [12], [13], [14], [15], [16]，歌詞の高精度な検索 [17], [18], [19]，ジャンルやムードの分類 [20], [21], [22], [23], [24], [25]，歌詞創作の支援 [26]，動画生成 [27] などがあげられる．本章では，本研究と関連の深い研究として（1）歌詞に基づく楽曲検索システムおよび（2）トピックに基づく歌詞分析およびシステムに関する研究について述べる．

2.1 歌詞に基づく楽曲検索システム

Brochu ら [3] は，メロディと歌詞テキストを入力として楽曲を検索できるように，両者を同時にモデル化する手法を提案した．Müller ら [4] は楽曲と歌詞のペアに対して，歌詞に対応する楽曲中の歌唱部分を割り当てる手法を提案し，歌詞の一部をクエリとして入力できる楽曲検索システムを提案した．ユーザが検索結果中の楽曲を選択すると，クエリの歌詞を含む箇所から音楽が再生される．ユーザが歌った歌詞と一致する楽曲の検索も多数取り組まれている [28], [29]．Fujihara ら [5] は歌詞中のフレーズに基づいて歌詞間にハイパーリンクを貼る「Music Web」という概念を提唱した．これにより，ユーザが楽曲を聴いている最中に，リンクされたフレーズをクリックすることで同じフレーズを持つ別の楽曲に遷移できる．楽曲集合を俯瞰するアプローチのひとつとして可視化の研究も行われてきた．SongWords [30] は，自己組織化マップを用いて歌詞とタグを二次元平面上に表示するテーブルトップコンピュータ向けのシステムである．Lyricon [31] は，ユーザが直感的に歌詞の内容を理解できるように，歌詞中の単語と適合するアイコンを歌詞に沿って表示するシステムである．

これらの研究では，歌詞中の単語を直接用いているが，我々は歌詞から推定されるトピックを用いるため，歌詞の潜在的な意味に基づいてユーザが歌詞を探索できるという優位性がある．

2.2 トピックに基づく歌詞分析およびシステム

トピックモデルは歌詞の潜在的な意味を考慮できるため，歌詞の分析 [32], [33], [34]，歌詞検索システム [35]，音楽プレーヤー [36] などの研究で使用されてきた．歌詞の分析では，Sharma ら [34] は LDA を用いることで歌詞が持つセンチメントを分析し，LDA により求められたいくつかのトピックがセンチメントと関連することを明らかにした．また，ラップの歌詞に LDA を適用することで，「street life」や「religion」といった想定通りのトピックだけでなく，「family/childhood」といった予想外のトピックが発見されたことも報告されている [32]．Ren ら [33] は楽曲の人気度を推定するために歌詞のトピックを利用し，人気のある楽曲の半数以上が「love」を表すトピックと関連していることを示した．アプリケーションに関する研究では，レーダーチャートを用いて各楽曲のトピックの混合比を可視化することで，ユーザがインタラクティブに歌詞を検索できるシステム LyricsRadar [35] が提案された．また Nakano ら [36] はユーザが音楽を再生中に，再生されている歌詞の文字列と意味が類似した文字列を持つ他の曲を閲覧できる音楽再生システム LyricListPlayer を提案した．文字列間の意味の類似度はトピックに基づいて計算される．

これらの研究では，トピックモデルとして LDA が使用されている．それに対して我々は，各アーティストがトピックの確率分布を持ち，各歌詞にひとつトピックが割り当てられるモデルを提案する．5 章で示すように，歌詞の生成過程を表すモデルとして，提案モデルは LDA を上回る精度を持つため，提案モデルを用いることで既存研究の歌詞分析やシステムを改善できる可能性がある．

本研究と最も関連のある研究は Kleedorfer ら [37] の研究である．彼らは歌詞集合に対して非負値行列分解を適用して歌詞をクラスタリングし，各クラスタをトピックのようにみなして人手で名前を付けている．本研究ではアーティストごとの歌詞トピックの傾向を考慮している点が彼らの研究とは異なる．これにより，アーティストとトピックの関係に基づいて好みの歌詞を発見することが可能になる．

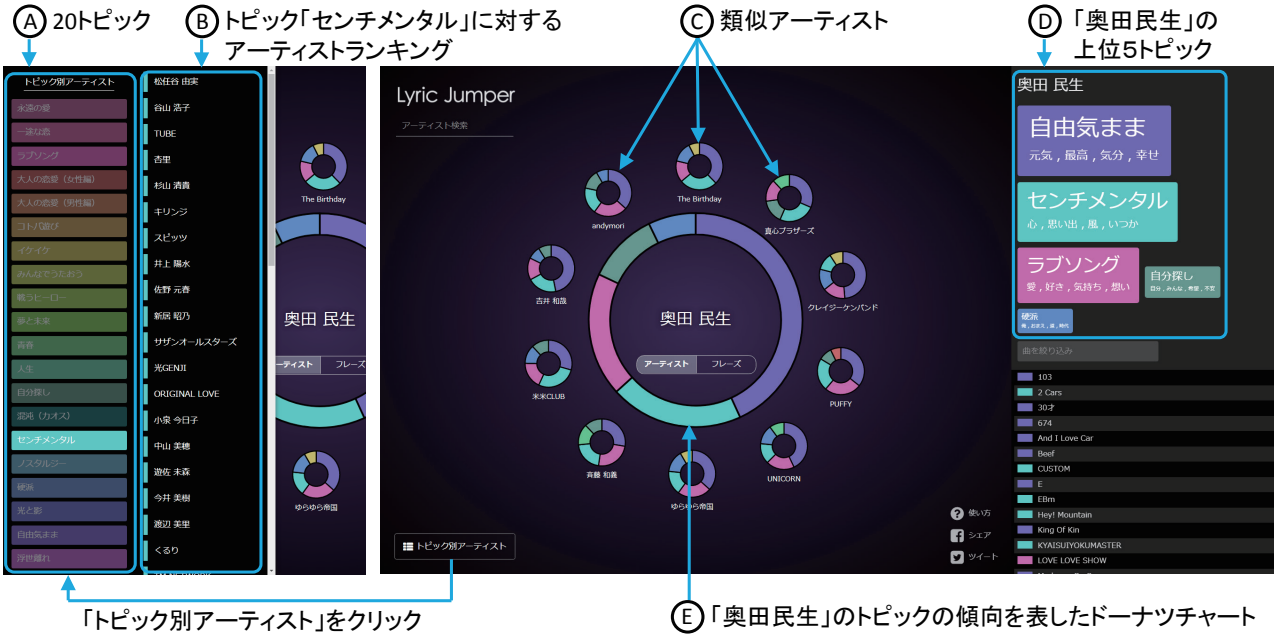

図 1　Lyric Jumper の概観.

さらに，我々は新しいモデルを提案しただけでなく，提案モデルで推定された歌詞トピックを基にした新たなインタフェースを備えた web サービスも実装し公開している．

3. Lyric Jumper

Lyric Jumper は，アーティストごとの歌詞トピックの傾向を考慮することで，ユーザが様々な歌詞と出会うことを実現する web サービスである．現段階では，日本語の歌詞を持つポピュラー音楽を対象としている．Lyric Jumper では「センチメンタル」や「自分探し」など 20 個のトピックを用意し，各歌詞にいずれかのトピックが割り当てられる．これにより，従来行われていた，アーティスト名や楽曲のタイトルを検索したり，歌詞内の文字列を直接検索したりする方法 [3], [4], [5] では困難であった，トピックに基づく新しいインタラクションを通じたアーティストや楽曲，歌詞との出会いや歌詞のより深い理解を実現する．より具体的には，次の 3 項目の実現を目標とする：（1）トピックに基づくアーティストの発見，（2）トピックに基づく楽曲の発見，（3）トピックに基づく歌詞の閲覧．

「トピックに基づくアーティストの発見」では，ユーザがトピックを選択した際に，トピックと関連の強いアーティストを容易に発見できる機能を提供する（アーティストランキング機能）．また，ユーザが興味を持ったアーティストとトピックの傾向が類似したアーティストを推薦することで，様々なアーティストとの出会いが可能になる機能も提供する（類似アーティスト推薦機能）．さらに，アーティストの歌詞トピックの傾向を把握できるように，アーティストがどのトピックと関連が強いのかを可視化する機能も備える（トピック傾向表示機能）．

楽曲との出会いを支援するためには，アーティストの発見の実現だけでは不十分である．例えば Lyric Jumper で最も楽曲数の多いアーティストは「松山千春」で 528 曲も存在し，この中からユーザが好みの曲を発見するのは容易ではない．そこで「トピックに基づく楽曲の発見」では，アーティストの楽曲のトピックによる楽曲のフィルタリングを可能にする（アーティストの楽曲ランキング機能）．それに加え，トピックと関連した様々なフレーズを起点とした楽曲の発見を実現するため，アーティストの楽曲の中で特定のトピックに関する歌詞のフレーズを推薦する機能を提供する（フレーズ推薦機能）．

ユーザが興味を惹かれる楽曲を発見すると，Lyric Jumper はその楽曲の歌詞を表示するが，歌詞とトピックの関連性を把握するのは必ずしも容易ではない．そこで，「トピックに基づく歌詞の閲覧」では，トピックと歌詞の関連性を容易に把握できるように，歌詞の中でトピックと関連の強い箇所を強調して表示する機能を提供する（歌詞フレーズの強調表示機能）．

以下では，これまでにあげた各機能の詳細を述べる．

[アーティストランキング機能] Lyric Jumper では図 1 Ⓐのように 20 個のトピック名を表示する．トピックを選択することで，そのトピックと関連の強い順にアーティストが最大 100 件表示される（図 1 Ⓑ）．これによりユーザは，興味のあるトピックと関連のある多くのアーティストを見ることができ，また意外なアーティストの選択トピックとの関連性も知ることができる．

[類似アーティスト推薦機能] Lyric Jumper ではユーザが選択したアーティストとトピックの傾向が類似した 10 アーティストを推薦する（図 1 Ⓒ）．10 アーティストは，ポピュラーな 8 アーティストとマイナーな 2 アーティストか

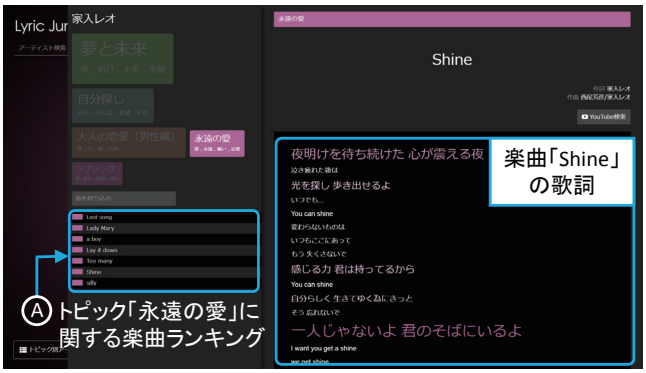

図 2　楽曲ランキングおよび歌詞フレーズの強調表示.

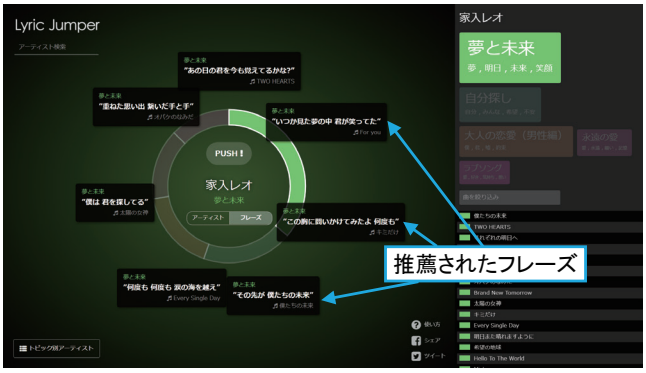

図 3　フレーズ推薦.

ら構成される．ポピュラーなアーティストだけでなくマイナーなアーティストも表示することで，ユーザにとって馴染みはないが，興味のあるアーティストと関連のあるアーティストにも触れてもらうことを狙っている．推薦されたアーティストのグラフを選択することで，そのアーティストのドーナツチャート（次項で述べる）が中心となった画面に遷移する．

[トピック傾向表示機能] ユーザがアーティスト a を選択すると，a と関連の強い 5 トピックの名前が表示される（図 1 Ⓓ）．関連性が高いほどトピック名は大きく表示される．これにより，ユーザは選択したアーティストのトピックの傾向を把握できるだけでなく，意外なトピックも関連していることを把握できる．ユーザがトピックの意味をより容易に理解できるように，トピックに特徴的な 4 つの単語をトピック名の下に表示している．さらに Lyric Jumper では，5 トピックの関連度に応じて円周が分割されたドーナツチャートも用いてトピックの傾向を表示している（図 1 Ⓔ）．

[アーティストの楽曲ランキング機能] ユーザがアーティストのトピックを選択すると，そのトピックに割り当てられた楽曲のランキングが表示される（図 2 Ⓐ）．トピックと関連の強い曲にユーザが容易にアクセスできるよう，トピックとの関連度が高い順に楽曲が順位付けされる．楽曲を選択するとその楽曲の歌詞が歌詞フレーズの強調表示機能により表示される．

[フレーズ推薦機能] アーティストのトピックを選択後，「フレーズ」ボタンをクリックすると，アーティストの楽曲の歌詞の中でそのトピックと関連のあるフレーズが推薦される（図 3）．さらに，「PUSH!」ボタンを押すたびに新しいフレーズが推薦される．これによりユーザは，同じトピックに関する様々な表現に触れることができる．フレーズを選択すると，そのフレーズを含む歌詞が歌詞フレーズの強調表示機能により表示される．

[歌詞フレーズの強調表示機能] 楽曲の歌詞を表示する際は，歌詞の中でトピックと関連の強い行 [*2] ほど，大きなフォントかつ濃い色で表示される（図 2）．これによりユーザは，「歌詞のサビの部分が特にトピックと関連が強い」のように，トピックと歌詞の関連性を容易に把握できるようになる．「YouTube 検索」ボタンを押すと，YouTube[*3] でアーティスト名と楽曲名を AND 検索した検索結果が表示され，Lyric Jumper 上で動画を視聴できる．

4. Lyric Jumper の実現方法

本章では，Lyric Jumper の実現方法として，歌詞のトピックを推定するためのモデル（4.1 節）と Lyric Jumper の各機能の実装方法（4.2 節）について述べる．

4.1 モデル

本研究では，歌詞のトピックを推定するためにトピックモデルを利用する．トピックモデルでは，歌詞内の各単語の生成過程についてある仮定を置き，その仮定に沿ったモデルを構築する．提案モデルには複数のパラメータが含まれ，適切なパラメータを求めることで，各歌詞が持つトピックを推定することができる．本節ではまず，歌詞のトピックを扱う従来研究で利用されていたトピックモデルである LDA の歌詞の生成過程と，Lyric Jumper で LDA を利用する場合の問題点を述べる（4.1.2 項）．その後，LDA の問題点を解決するため，歌詞の生成過程をより適切に反映した仮定に基づく新しいモデルを提案する（4.1.3 項）．

4.1.1 記号の定義

歌詞データセットが与えられたとき，A をデータセット内のアーティスト集合とする．R_a をアーティスト $a \in A$ の楽曲数とすると，a の楽曲集合は $\{S_{ar}\}_{r=1}^{R_a}$ と表される．S_{ar} は a の r 番目の楽曲を表す．また，V_{ar} を楽曲 S_{ar} の歌詞内の単語数とすると，$S_{ar} = \{v_{arj}\}_{j=1}^{V_{ar}}$ と表される．v_{arj} は S_{ar} の j 番目の単語である．したがって，すべての歌詞内のすべての単語は $D = \{\{\{v_{arj}\}_{j=1}^{V_{ar}}\}_{r=1}^{R_a}\}_{a \in A}$ となる．

4.1.2 LDA（latent Dirichlet allocation）

LDA を歌詞の生成過程に適用する際は以下の 3 つの仮定を基にしている：（1）各歌詞がトピックの確率分布を持つ，（2）歌詞ごとのトピックの確率分布に応じて歌詞内の各単語にトピックが割り当てられる，（3）トピックごとの単語の確率分布に応じて単語が生成される．図 4（a）に

[*2] 本稿では歌詞の「フレーズ」と「行」を同じ意味で用いる．

[*3] https://www.youtube.com/

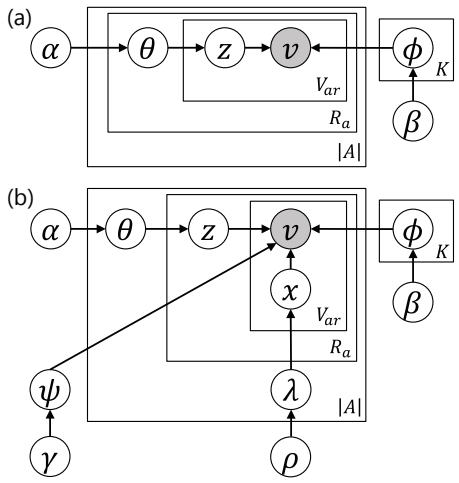

図 4 （a）LDA と（b) 提案モデルのグラフィカルモデル．

LDA のグラフィカルモデルを示す．図中の色付きの円は観測変数，白地の円は未知変数を表す．K はトピック数，z はトピック，θ は歌詞ごとのトピックの確率分布，ϕ はトピックごとの単語の確率分布である．また，θ と ϕ それぞれの事前分布として，α と β をハイパーパラメータに持つディリクレ分布を仮定している．

既存研究の中で，歌詞に LDA を適用する有用性は報告されてきたが [32], [33], [34], [35], [36]，LDA ではアーティストごとの楽曲集合を考慮しておらず，Lyric Jumper のようなアーティストのトピックの傾向に基づいた歌詞探索の実現には適していないという問題点がある（問題点 1 ）．また，LDA では歌詞内の単語ごとにトピックを割り当てるため，歌詞全体が持つトピックを明示的に扱えない点も Lyric Jumper を実現するうえで問題点となる（問題点 2 ）．

4.1.3 提案モデル

本研究では，あるアーティストは「恋愛」というトピックに関する歌詞の楽曲が多く，別のアーティストは「人生」というトピックに関する歌詞の楽曲が多いといったように，各アーティストには歌詞のトピックに対する固有の傾向が存在すると仮定する．この仮定に基づき，アーティストのトピックに対する傾向を考慮した歌詞生成モデルを提案する．図 4（b）に提案モデルのグラフィカルモデルを示す．提案モデルでは，各アーティストはトピックの確率分布 θ を持つ．これにより 4.1.2 項の問題点 1 を解決する．さらに，歌詞を書き始める前に歌詞の主題（すなわちトピック）を決めることが一般的であることから [7], [8]，各歌詞は θ から生成されたひとつのトピック z を持つと仮定する．これにより 4.1.2 項の問題点 2 を解決する．しかし，歌詞中の全ての単語がトピックと関連しているとは限らない．たとえば，「これ」や「事」などの単語は多くの歌詞に出現するが，固有のトピックとの関連度が高いとは言えない．そこで，提案モデルでは背景語の存在を考慮する．図 4（b）中で ψ が背景語の確率分布を表し，いずれのトピックとも関連度の低い単語が高い生起確率を持つ．各アーティストはベルヌーイ分布 λ を持ち，この分布に応じてトピックと背景語のどちらからどの程度単語が生成されるかが決まる．つまり，アーティスト a が歌詞中の単語を生成するとき，確率 λ_{a0} $(x=0)$ でトピックの単語確率分布 ϕ から単語が生成され，確率 λ_{a1} $(x=1)$ で背景語の単語確率分布 ψ から単語が生成される．ここで，$\lambda_{a0}+\lambda_{a1}=1$ である．提案モデルの歌詞生成過程を Algorithm 1 に示す．

Algorithm 1 提案モデルの歌詞生成過程.

for each topic $k \in \{1, \cdots, K\}$ **do**
  Draw $\phi_k \sim$ *Dirichlet*(β)
end for
Draw $\psi \sim$ *Dirichlet*(γ)
for each artist a in A **do**
  Draw $\theta_a \sim$ *Dirichlet*(α)
  Draw $\lambda_a \sim$ *Beta*(ρ).
  for each song S_{ar} **do**
    Draw a topic $z_{ar} \sim$ *Multinomial*(θ_a)
    for each word v_{arj} in S_{ar} **do**
      Draw switch $x \sim$ *Bernoulli*(λ_a)
      if $x = 0$ **then**
        Draw a word $v_{arj} \sim$ *Multinomial*($\phi_{z_{ar}}$)
      else if $x = 1$ **then**
        Draw a word $v_{arj} \sim$ *Multinomial*(ψ)
      end if
    end for
  end for
end for

4.1.4 パラメータ推定

崩壊型ギブスサンプリング [38] により提案モデルのパラメータを推定する．パラメータ θ，ϕ，ψ の事前分布としてディリクレ分布を，λ の事前分布としてベータ分布を用いる．これらの事前分布は各パラメータの共役事前分布であるため，パラメータに関して周辺化を行える．$\mathbf{\Theta} = \{\theta_a\}_{a\in A}$，$\mathbf{\Phi} = \{\phi_k\}_{k=1}^{K}$，$\mathbf{\Lambda} = \{\lambda_a\}_{a\in A}$ とすると，全ての歌詞中の全単語データ D，潜在変数 $Z = \{\{z_{ar}\}_{r=1}^{R_a}\}_{a\in A}$ および $X = \{\{\{x_{arj}\}_{j=1}^{V_{ar}}\}_{r=1}^{R_a}\}_{a\in A}$ の周辺化された同時分布は次式により解析的に計算できる．

$$
\begin{aligned}
&P(D, Z, X|\alpha, \beta, \gamma, \rho) \\
&= \iiiint P(D, Z, X|\mathbf{\Theta}, \mathbf{\Phi}, \psi, \mathbf{\Lambda}) P(\mathbf{\Theta}|\alpha) \\
&\qquad \times P(\mathbf{\Phi}|\beta) P(\psi|\gamma) P(\mathbf{\Lambda}|\rho) d\mathbf{\Theta} d\mathbf{\Phi} d\psi d\mathbf{\Lambda} \\
&\propto \prod_{a\in A} \frac{\Gamma(\rho + N_{a0})\Gamma(\rho + N_{a1})}{\Gamma(2\rho + N_a)} \frac{\prod_{v\in V}\Gamma(N_{1v} + \gamma)}{\Gamma(N_1 + \gamma|V|)} \\
&\times \prod_{k=1}^{K} \frac{\prod_{v\in V}\Gamma(N_{kv} + \beta)}{\Gamma(N_k + \beta|V|)} \prod_{a\in A} \frac{\prod_{k=1}^{K}\Gamma(R_{ak} + \alpha)}{\Gamma(R_a + \alpha K)}. \quad (1)
\end{aligned}
$$

N_{a0}，N_{a1} はそれぞれ，アーティスト a の全ての歌詞の中で $x = 0$，$x = 1$ が割り当てられた単語数を表す（$N_a = N_{a0} + N_{a1}$）．N_{1v} は単語 v に $x = 1$ が割り当てられた回数である（$N_1 = \sum_{v\in V} N_{1v}$）．$V$ は D 内の語彙集合を表す．N_{kv} は単語 v が $x = 0$ の下でトピック k から生

成された回数である（$N_k = \sum_{v \in V} N_{kv}$）．$R_{ak}$ はアーティスト a の歌詞の中でトピック k が割り当てられた歌詞の数を表す（$R_a = \sum_{k=1}^{K} R_{ak}$）．式（1）に基づき，$z_{ar}$ および x_{arj} をサンプリングする．十分な回数のイテレーションの後，サンプリングされた値を用いてパラメータの値を求める．紙面の都合上，詳細は割愛する．

4.2 Lyric Jumper の機能の実装

4.2.1 データ

Lyric Jumper を実装するにあたり，商用的に歌詞の配信を行っている企業から提供された歌詞データを使用した．データセットには，歌詞ごとの曲名とアーティスト名も含まれる．2016 年 12 月末時点で利用可能な全ての歌詞を使用し，形態素解析エンジン MeCab [39] を用いて各歌詞から日本語の名詞を抽出した．提案モデルは言語に非依存であるが，Lyric Jumper のユーザにとって，推定されたトピックが理解しやすいよう，日本語のみを使用した．トピックの質を保障するため，登録されている楽曲が 10 曲未満のアーティストおよび，10 件未満の歌詞にしか出現しない名詞は除いた．最終的に，3,722 アーティストの 147,990 曲の歌詞データを使用した．

トピック数に関しては，トピック数が少なすぎると，ユーザは Lyric Jumper の使用にすぐに飽きると考えられ，多すぎると，類似したトピックが多数出現し，ユーザがトピック間の差異を把握するのが難しくなると考えられる．このような理由から，様々なトピック数の結果を比較したうえで，Lyric Jumper のトピック数は 20 とした．提案モデルで 20 個のトピックを求めた後，ユーザが各トピックの特徴を容易に理解できるよう，人手で各トピックの名前をラベル付けした．トピック名には「センチメンタル」「青春」「自分探し」などが含まれる．５つのトピックは「恋愛」に関するものであったが，Lyric Jumper ではそれらを「永遠の愛」「一途な愛」「ラブソング」「大人の恋愛（女性編）」「大人の恋愛（男性編）」と表現している．

4.2.2 機能

3 章で述べた Lyric Jumper の各機能は，4.1.4 項で推定したパラメータを用いて以下のように実現している．

[アーティストランキング機能] トピック k が選択されたとき，トピック k とより強く関連したアーティストほど上位に表示されるように，アーティスト a のトピック分布 θ_a におけるトピック k の順位を第一キー（小さいほど良い），トピック k に割り当てられた楽曲数 R_{ak} を第二キー（大きいほど良い）として全アーティストをランキングし，上位 100 アーティストをユーザに表示する．

[類似アーティスト推薦機能] ユーザが選択したアーティスト a が与えられると，a と $a' \in A \setminus \{a\}$ の類似度を θ_a と $\theta_{a'}$ の JS ダイバージェンスの逆数により求める．アーティスト a と各アーティストの類似度を求めた後，データセット内の楽曲数が 100 曲以上のアーティストの中から類似度の上位 8 組をポピュラーなアーティスト，100 曲未満の中から類似度の上位 2 組をマイナーなアーティストとして選択し，計 10 アーティストをユーザに推薦する．

[トピック傾向表示機能] ユーザがアーティスト a を選択すると，θ_a において生起確率の上位 5 トピックの名前を表示する．トピックの生起確率が高いほどトピック名は大きく表示される．トピック名の下に表示される単語は，各トピックの単語生起確率分布 ϕ からトピックに特徴的な 4 単語を人手で選択した．アーティスト a のドーナツチャートは θ_a の上位 5 トピックの割合に応じて円周を分割する．

[歌詞フレーズの強調表示機能] 歌詞の各行とトピック k の関連度を求めるために，まず ϕ_k を用いて，各名詞のスコアを計算する．$rank(k, v)$ を ϕ_k 内での名詞 v の生起確率の高さの順位とする．$rank(k, v) \leq 100$ であれば $w\_rel(k, v) = 101 - rank(k, v)$，それ以外であれば $w\_rel(k, v) = 0$ とする．行 l は $n(\geq 0)$ 個の名詞を含み，$l = (v_1, \cdots, v_n)$ と表せ，行 l とトピック k の関連度を $l\_rel(k, l) = \sum_{i=1}^{n} w\_rel(k, v_i)$ により求める．歌詞内の全ての行のスコアを計算したら，各行のスコアが $[0, 1]$ に収まるよう min-max 正規化を行う．文字サイズと文字の色はスコアが 0 の行の 16pt・#FFFFFF からスコアが 1 の行の 36pt・トピックの色まで線形に変化させる．

[アーティストの楽曲ランキング機能] 楽曲 s とトピック k の関連度は $s\_rel(k, s) = \frac{1}{|L_s|} \sum_{l \in L_s} l\_rel(k, l)$ により求める．L_s は楽曲 s の歌詞内の全ての行のリストである．つまり，s と k の関連度は s の歌詞の各行と k の関連度の平均値により求められる．

[フレーズ推薦機能] アーティスト a とトピック k が与えられると，以下の方法で推薦フレーズを選択する．i 番目のループ（$i = 1, 2, \cdots$）では，アーティスト a の楽曲で $s\_rel(k, s)$ の値が高い順に，各楽曲の歌詞の i 番目に $l\_rel(k, l)$ の値が高い行を抽出する．抽出された行数が 100 件になるまで，このループを繰り返す．ユーザが Lyric Jumper を使用するたびに異なるフレーズに触れられるようにするため，抽出した 100 件のフレーズをランダムな順番で推薦する．

5. 提案モデルの評価実験

4.1 節では，LDA の問題点を指摘し，Lyric Jumper の実現に適した新しいトピックモデルを提案した．提案モデルは Lyric Jumper の実現のためだけに特化したモデルではなく，歌詞の生成過程を LDA よりも適切に反映したモデルであると言えるだろうか．もしそう言えれば，提案モデルは Lyric Jumper だけでなく，歌詞のトピックを扱う様々な研究にも応用できる可能性が広がる．本章では，この疑問に答えるため，提案モデルの定量的評価を行う．

[データセット] 4.2.1 項で述べた企業から提供された歌詞

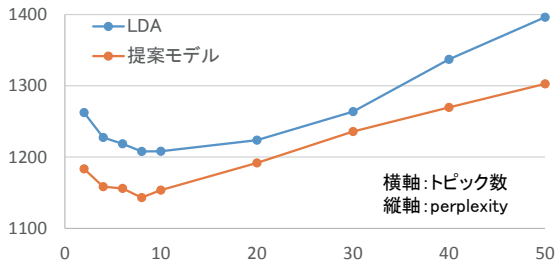

図 5 LDA と提案モデルの perplexity の比較結果.

データを使用した．評価には，2016 年 12 月末時点における楽曲数の上位 1,000 アーティストの歌詞，計 93,716 件を使用した．出現する歌詞の数が 10 以上の名詞を対象とし，各歌詞から日本語の名詞を抽出した．各歌詞から 80%の名詞を学習データとしてランダムに選択してパラメータを推定し，残りの 20%の名詞をテストデータに使用した．

[実験設定] LDA および提案モデルでは，既存研究に倣い，ハイパーパラメータの値を $\alpha = \frac{1}{K}$，$\beta = \frac{50}{|V|}$ とした．これに加えて，提案モデルでは $\gamma = \frac{50}{|V|}$，$\rho = 0.5$ とした．LDA と提案モデルの精度を比較するため，トピックモデルの精度を評価する際の指標として広く使用されている perplexity を使用した [6]．モデルの精度が高いほど perplexity の値は小さくなる．トピック数は $K = 2$，4，6，8，10，20，30，40，50 とし，各値に対する perplexity を比較した．

[結果] 結果を図 5 に示す．トピック数に関わらず，提案モデルは LDA を上回った．いずれのモデルでも，トピック数が 8 のときに perplexity の値は最小となった [*4]．トピック数が 30 を超えて大きくなるほど，2 モデル間の perplexity の差も大きくなる傾向が見られた．これらの結果から，提案モデルは LDA よりも優れたモデルであり，アーティストのトピックに対する傾向を考慮することは歌詞のモデル化を行う際に有用であると言える．

6. Lyric Jumper の利用状況分析

我々は Lyric Jumper を無料で利用可能な web サービスとして 2017 年 2 月 21 日に公開した．本章では，Lyric Jumper の利用状況を，サービス上のログと Twitter[*5] 上の書き込みから分析する．

6.1 ログ分析

Lyric Jumper 上でのユーザの行動を分析するため，2 月 21 日から 9 月 30 日までの全ユーザの操作ログを収集した．PC とスマートフォンのユニークユーザ数はそれぞれ 6,040 と 86,922 であった．図 6 に累積ページビュー数の月ごとの推移を示す．サービスの公開から半年以上経過しても累積ページビュー数はほぼ線形に増加しており，一定のニーズをもってユーザに利用され続けていると言える．ま

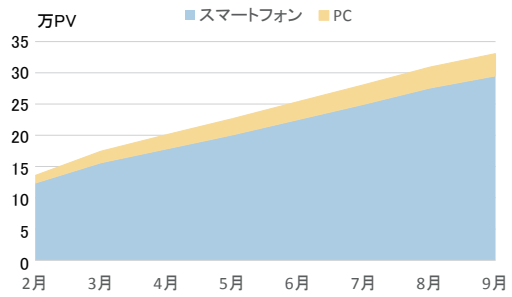

図 6 累積ページビュー数（PV）の月ごとの推移.

表 1 各機能の使用回数.

機能	PC	スマートフォン
アーティストランキング機能	5,574	41,169
類似アーティスト推薦機能	4,206	10,557
アーティストの楽曲ランキング機能	12,375	24,717
フレーズ推薦機能	12,333	456,429

た，デバイスごとのページビュー数に着目すると，スマートフォンからのアクセスによるページビュー数が，常に累積ページビュー数の 90%前後を占めていた．

6.1.1 機能別・トピック別の利用状況

各機能の使用頻度を表 1 に示す．表の上 2 つの機能はアーティストを探すための機能であり，下 2 つは楽曲を探すための機能である．スマートフォンのアーティストランキング機能を除き，楽曲を探すための機能の方がアーティストを探すための機能よりも使用回数が多かった．これは，Lyric Jumper で使用しているアーティスト数が 3,722 であるのに対して楽曲数が 147,990 と多いことから，ユーザがアーティストを発見した後に様々なインタラクションを経て楽曲を発見しているためだと言える．スマートフォンで Lyric Jumper にアクセスすると PC とは異なるインタフェースが表示されるようになっており，アーティストランキング機能への導線が PC より明確であることから，スマートフォンのアーティストランキング機能の使用回数が多かったのではないかと推測される．

フレーズ推薦機能は，スマートフォンでは 456,429 回使用され，PC の 12,333 回とは大きな差が見られた．PC のフレーズ推薦機能では，推薦用のボタンをクリックすると 7 つのフレーズがまとめて推薦されるが，スマートフォンではボタンを押すたびに 1 つずつフレーズが推薦される（図 7）．7 倍の差を正規化した場合でも，スマートフォンでは約 65,204 回使用されており，アーティストの楽曲ランキング機能との比率では PC よりも依然として高い．これは，フレーズが 1 つずつ推薦されることで，フレーズに対するユーザの興味をより強く喚起でき，ユーザが思わず次々とボタンを押した結果ではないかと推測される．

ここからは，ページビュー数の多いスマートフォンを対象として，トピック別の利用状況について分析を行う．図 8 に，アーティストランキング機能において各トピックがクリックされた回数を示す．図中のトピックの並び順は，

[*4] 4.2.1 項で述べたように，Lyric Jumper ではユーザの利便性を考慮して，トピック数を 20 とした．
[*5] https://twitter.com/

図 7　スマートフォンのフレーズ推薦用インタフェース．

Lyric Jumper で表示されるトピックの並び順と同じである．一般的に，リスト形式でアイテムが表示されると，アイテムの順位と各アイテムをユーザが選択する頻度の間には高い相関があることが知られているが [40]，Lyric Jumper では，トピックを表示する順位とトピックがクリックされた回数の順位に関するスピアマンの順位相関係数は 0.46 と中程度であった．恋愛に関する 5 トピックは，ユーザの興味を惹くことが予想されたためリストの上位に表示したが，それらのトピックを除いた 15 トピックに限ると，スピアマンの順位相関係数は −0.025 と相関はみられなかった．したがって，ユーザは単純に上位に表示されたトピックを選択したのではなく，ある程度は自分が興味のあるトピックを選択していると言える．これらのことを踏まえて図 8 を見ると，我々の予想通り，恋愛に関するトピックは平均してクリック回数が多く，恋愛に関する歌詞を探したいというユーザのニーズの高さを表している．また，同じ「大人の恋愛」に関するトピックであっても，「大人の恋愛（女性編）」がクリックされた割合は全体の 7.29%であるのに対して「大人の恋愛（男性編）」は 4.26%であり，女性視点での恋愛ソングにより強い興味を持っているユーザが多いことがわかった．恋愛以外のトピックでは，「コトバ遊び」，「自分探し」，「浮世離れ」，「混沌（カオス）」，「光と影」の順にクリック回数が多かった．

6.1.2　ユーザの行動遷移

スマートフォンで Lyric Jumper のトップページにアクセスし，「はじめる」をクリックすると，アーティスト名を直接入力して検索する方法とトピックからアーティストを選択する方法のいずれかによって特定のアーティストの結果を表示することができる．それぞれの方法が利用された

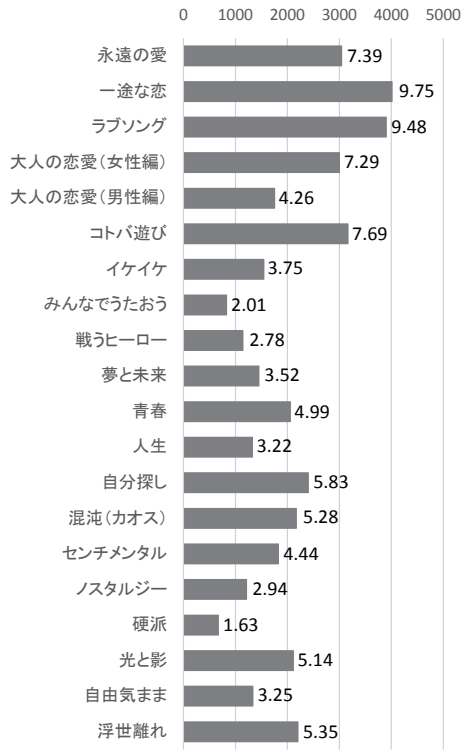

図 8　トピックごとのクリック回数（横軸）．棒グラフの横の数値は全トピックのクリック回数に占める割合（%）を表す．

割合を調べたところ，前者が 34.8%，後者が 65.2%であった [*6]．前者を利用するユーザは，自分が興味のあるアーティストがどのようなトピックを持っているかを知りたくて，直接アーティスト名を入力したと考えられる．最初から特定のアーティストについて調べたいユーザが一定数存在する一方で，多数のユーザはトピックを起点としていずれかのアーティストにたどり着いていた．この結果から，トピックを起点としてアーティストを見つけるというアプローチは新規性が高いだけでなく，ユーザにも積極的に利用されていることが明らかになった．その他，フレーズ推薦機能において「PUSH!」ボタンが押されると，20.5%の割合で推薦されたフレーズのいずれかをユーザはクリックし歌詞を表示していた．このことから，トピックに関連する様々なフレーズを推薦することは，ユーザが興味を持つ歌詞を探し出すうえで有効に働いていると考えられる．

6.2　Twitter の書き込み分析

2017 年 2 月 21 日から 10 月 10 日の間に Twitter に投稿された，Lyiric Jumper に関するツイート 429 件を収集した [*7]．収集したツイートのうち 256 件は，GIZMODO[*8] やマイナビニュース [*9] といったニュースサイトに掲載され

[*6] ユーザの行動遷移データを取得する際，クリック回数の下位 4 トピックについてはデータが取得できなかったため，ここでの割合は残りの 16 トピックのクリック回数の合計値を基に求めた．
[*7] 「“lyric jumper” OR “リリックジャンパー”」をクエリとし，検索結果から Lyric Jumper について言及しているツイートを人手で選択した．429 件のツイートにはリツイートは含まれない．
[*8] https://www.gizmodo.jp/
[*9] http://news.mynavi.jp/

た Lyric Jumper の紹介記事上に設置されたツイートボタンから投稿されたものであった．

98 件のツイートでは，ツイートを投稿したユーザが興味のあるアーティストのトピックについて述べており，6.1.2 項でも述べたように，自分が興味のあるアーティストがどのようなトピックを持っているかを知りたいユーザが一定数存在することがここでも示された．また，プロの歌手が自らのバンドのトピックについて投稿したツイートも 1 件，「便利」という言葉と共に投稿されており，プロの立場からも Lyric Jumper が受け入れられていることが確認できた．より詳細にユーザのツイートを分析すると，Lyric Jumper が表示するアーティストのトピックが妥当であり，納得感を表すツイートが多数を占めていた．他にも，自分の好きなアーティストは「ラブソング」トピックの割合が高いと思っていたが「自分探し」トピックの割合が高く，意外な発見ができた，という投稿をしているユーザも見られた．

収集した各ツイートにポジティブな表現またはネガティブな表現が含まれているかを人手で判定した結果，ポジティブな表現を含むツイートが 95 件，ネガティブな表現を含むツイートが 10 件であった．ポジティブな表現としては「面白い」,「楽しい」,「便利」といったものが多く見られた．その中には，フレーズ推薦機能や類似アーティスト推薦機能などの個別の機能について言及したものも見られた．その他，音楽の楽しみ方が増えた，歌詞を探すのが楽しい，作詞家の素晴らしさを改めて知ることができた，といったより具体的なコメントも投稿されていた．自分がどのようなタイプの歌詞が好きなのかが分かる点が興味深い，と述べているユーザもおり，Lyric Jumper を利用することが自分でも明確には認識していなかった歌詞の好みを把握するためにも役立つことを確認できた．

ネガティブな内容としては，個別の楽曲について，割り当てられたトピックが妥当でない点に言及したものが見られた．提案手法ではトピックの割り当てを確率的に推定している以上，完璧な精度でトピックを割り合てることは難しい．この問題への対処方法としては，楽曲に割り当てられたトピックが不適切であるとユーザが感じたら，ユーザ自らがより適切なトピックを容易に訂正して割り当てられるインタフェースを提供することなどが考えられる．また，Lyric Jumper に洋楽が含まれていないことを残念がる投稿もあった．これに関しては，提案モデルは言語に非依存であるため，提案モデルを英語歌詞に適用し，洋楽版の Lyric Jumper を今後公開する予定である．

7. まとめ

本稿ではアーティストの歌詞トピックに対する傾向を考慮したモデルを提案し，3,722 アーティストの 147,990 件の歌詞に提案モデルを適用することで，ユーザが様々な歌詞に出会える歌詞探索サービス Lyric Jumper を提案した．Lyric Jumper では，アーティストランキング機能，類似アーティスト推薦機能，トピック傾向表示機能，アーティストの楽曲ランキング機能，フレーズ推薦機能，歌詞フレーズの強調表示機能の 6 つの機能を実現した．こうした機能を利用することで，ユーザが歌詞に対する理解を深めながら，インタラクティブに歌詞を探索することを可能にした．また，提案モデルの評価を行い，トピックの数に関わらず提案モデルが従来モデルの LDA を上回る精度であることを perplexity を用いて定量的に示した．さらに，Lyric Jumper の 9 万人以上のユーザの操作ログおよび 400 件以上のツイートの分析の結果，トピックを起点とした歌詞の探索はユーザから積極的に利用されており，Lyric Jumper の楽しさや有用性が評価されていることを明らかにした．

現在の Lyric Jumper で扱っていない情報のひとつとして，時間，つまり楽曲が発売された日にちがあげられる．時間情報を考慮し，例えば各年に発売された楽曲集合ごとに提案モデルを適用することで，トピックの流行り廃りや，各アーティストが歌ってきた歌詞トピックの時間変化などを明らかにできることが期待される．このような情報を提示することで，ユーザの歌詞に対する理解をより深めることができるようになると考えている．また，提案モデルによって得られたトピックを，音響特徴量やタグなどと組み合わせることで，より柔軟にユーザの検索意図を反映できる楽曲の探索システムの提案も行っていきたい．

謝辞 Lyric Jumper の運営を共同で行い，歌詞データを提供していただいた株式会社シンクパワーに感謝する．本研究の一部は JST ACCEL (JPMJAC1602) の支援を受けた．

参考文献

[1] D. Bainbridge *et al.*: “How people describe their music information needs: A grounded theory analysis of music queries”, Proc. of the 4th International Conference on Music Information Retrieval, pp. 221–222 (2003).
[2] J. H. Lee and J. S. Downie: “Survey of music information needs, uses, and seeking behaviours: Preliminary findings”, Proc. of the 5th International Conference on Music Information Retrieval, pp. 989–992 (2004).
[3] E. Brochu and N. de Freitas: “”Name that song!” a probabilistic approach to querying on music and text”, Proc. of the 15th International Conference on Neural Information Processing Systems, pp. 1505–1512 (2002).
[4] M. Müller *et al.*: “Lyrics-based audio retrieval and multimodal navigation in music collections”, Proc. of the 11th European Conference on Digital Libraries, pp. 112–123 (2007).
[5] H. Fujihara *et al.*: “Hyperlinking lyrics: A method for creating hyperlinks between phrases in song lyrics”, Proc. of the 9th International Conference on Music Information Retrieval, pp. 281–286 (2008).
[6] E. M. Airoldi *et al.*: “Mixed membership stochastic blockmodels”, Journal of Machine Learning Research, **9**, pp. 1981–2014 (2008).
[7] M. Baxter: “Voices of resistance, voices of transcen-

dence: Musicians as models of the poetic - political imagination", International Journal of Education & the Arts, **11**, pp. 1–24 (2010).

[8] J. M. Toivanen *et al.*: "Automatical composition of lyrical songs", Proc. of the 4th International Conference on Computational Creativity, pp. 87–91 (2013).

[9] G. Dzhambazov *et al.*: "On the use of note onsets for improved lyrics-to-audio alignment in Turkish makam music", Proc. of the 17th International Conference on Music Information Retrieval, pp. 716–722 (2016).

[10] K. Lee and M. Cremer: "Segmentation-based lyrics-audio alignment using dynamic programming", Proc. of the 9th International Conference on Music Information Retrieval, pp. 395–400 (2008).

[11] V. Thomas *et al.*: "Slave: A score-lyrics-audio-video-explorer", Proc. of the 10th International Conference on Music Information Retrieval, pp. 717–722 (2009).

[12] R. J. Ellis *et al.*: "Quantifying lexical novelty in song lyrics", Proc. of the 16th International Conference on Music Information Retrieval, pp. 694–700 (2015).

[13] H. Hirjee and D. G. Brown: "Automatic detection of internal and imperfect rhymes in rap lyrics", Proc. of the 10th International Conference on Music Information Retrieval, pp. 711–716 (2009).

[14] X. Hu and B. Yu: "Exploring the relationship between mood and creativity in rock lyrics", Proc. of the 12th International Conference on Music Information Retrieval, pp. 789–794 (2011).

[15] E. Nichols *et al.*: "Relationships between lyrics and melody in popular music", Proc. of the 10th International Conference on Music Information Retrieval, pp. 471–476 (2009).

[16] A. Singhi and D. G. Brown: "Are poetry and lyrics all that different?", Proc. of the 15th International Conference on Music Information Retrieval, pp. 471–476 (2014).

[17] P. Knees *et al.*: "Multiple lyrics alignment: Automatic retrieval of song lyrics", Proc. of the 6th International Conference on Music Information Retrieval, pp. 564–569 (2005).

[18] G. Geleijnse and J. H. M. Korst: "Efficient lyrics extraction from the web", Proc. of the 7th International Conference on Music Information Retrieval, pp. 371–372 (2006).

[19] R. Macrae and S. Dixon: "Ranking lyrics for online search", Proc. of the 13th International Conference on Music Information Retrieval, pp. 361–366 (2012).

[20] X. Hu and J. S. Downie: "When lyrics outperform audio for music mood classification: A feature analysis", Proc. of the 11th International Conference on Music Information Retrieval, pp. 619–624 (2010).

[21] R. Mayer *et al.*: "Rhyme and style features for musical genre classification by song lyrics", Proc. of the 9th International Conference on Music Information Retrieval, pp. 337–342 (2008).

[22] R. Mayer and A. Rauber: "Music genre classification by ensembles of audio and lyrics features", Proc. of the 12th International Conference on Music Information Retrieval, pp. 675–680 (2011).

[23] X. Wang *et al.*: "Music emotion classification of chinese songs based on lyrics using TF*IDF and rhyme", Proc. of the 12th International Conference on Music Information Retrieval, pp. 765–770 (2011).

[24] B. Wei *et al.*: "Keyword generation for lyrics", Proc. of the 8th International Conference on Music Information Retrieval, pp. 121–122 (2007).

[25] M. Zaanen and P. Kanters: "Automatic mood classification using TF*IDF based on lyrics", Proc. of the 11th International Conference on Music Information Retrieval, pp. 75–80 (2010).

[26] T. O'Hara *et al.*: "Inferring chord sequence meanings via lyrics: Process and evaluation", Proc. of the 13th International Conference on Music Information Retrieval, pp. 463–468 (2012).

[27] S. Funasawa *et al.*: "Automated music slideshow generation using web images based on lyrics", Proc. of the 11th International Conference on Music Information Retrieval, pp. 63–68 (2010).

[28] T. Hosoya *et al.*: "Lyrics recognition from a singing voice based on finite state automaton for music information retrieval", Proc. of the 6th International Conference on Music Information Retrieval, pp. 532–535 (2005).

[29] C. Wang *et al.*: "An improved query by singing/humming system using melody and lyrics information", Proc. of the 11th International Conference on Music Information Retrieval, pp. 45–50 (2010).

[30] D. Baur *et al.*: "SongWords: Exploring music collections through lyrics", Proc. of the 11th International Conference on Music Information Retrieval, pp. 531–536 (2010).

[31] W. Machida and T. Itoh: "Lyricon: A visual music selection interface featuring multiple icons", Proc. of the 15th International Conference on Information Visualisation, pp. 145–150 (2011).

[32] C. Johnson-Roberson and M. Johnson-Roberson: "Temporal and regional variation in rap lyrics", NIPS Workshop on Topic Models: Computation, Application and Evaluation (2013).

[33] J. Ren *et al.*: "What makes a music track popular in online social networks?", Proc. of the 25th International Conference Companion on World Wide Web, pp. 95–96 (2016).

[34] G. Sharma and M. N. Murty: "Mining sentiments from songs using latent Dirichlet allocation", Proc. of the 10th International Conference on Advances in Intelligent Data Analysis X, pp. 328–339 (2011).

[35] S. Sasaki *et al.*: "LyricsRadar: A lyrics retrieval system based on latent topics of lyrics", Proc. of the 15th International Conference on Music Information Retrieval, pp. 585–590 (2014).

[36] T. Nakano and M. Goto: "LyricListPlayer: A consecutive-query-by-playback interface for retrieving similar word sequences from different song lyrics", Proc. of the Sound and Music Computing Conference 2016, pp. 344–349 (2016).

[37] F. Kleedorfer *et al.*: "Oh oh oh whoah! towards automatic topic detection in song lyrics", Proc. of the 9th International Conference on Music Information Retrieval, pp. 287–292 (2008).

[38] T. L. Griffiths and M. Steyvers: "Finding scientific topics", Proc. of the National Academy of Sciences, **101**, Suppl. 1, pp. 5228–5235 (2004).

[39] T. Kudo *et al.*: "Applying conditional random fields to Japanese morphological analysis", Proc. of the Conference on Empirical Methods in Natural Language Processing, pp. 230–237 (2004).

[40] T. Joachims *et al.*: "Accurately interpreting clickthrough data as implicit feedback", Proc. of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 154–161 (2005).