歌詞の潜在的意味分析に基づく
Billlboard ランキング予測

長倉大輔研究会マーケティング班　経済学部３年　加治佐巧・神原俊久

概要

本研究は、歌詞と楽曲の人気度の関連をテーマにしている。世の中には数々のヒット曲が存在するが、それら楽曲のどのような点が人々に支持されているのだろうか。今回、我々は人気度が反映されたランキング結果として「Billboard Japan」に着目し、ランキングに取り上げられる楽曲の歌詞を潜在的ディリクレ配分法（以降 LDA（Latent Dirichlet Allocation の略）と呼ぶ。）で分析した。LDA とは文章は複数のトピックが割合的に配分されて構成されるという仮定の元、指定したトピック数に応じて文章に対する単語の出現頻度やその意味を学習させた結果、潜在的に近しい意味をもつ単語群分布を推定するモデルである。具体的には 2008 年から 2017 年までの「Billboard Japan」の年度末ランキング TOP100 に属する曲の中で取得可能な歌詞データからトピックと 1 曲当たりのトピック構成比を推定する。その構成比率を元に重回帰モデルとランダムフォレストの２つのパターンで楽曲のランキングを予測した。結果としていずれのモデルにおいてもランキング順位を予測することは難しく、歌詞の潜在トピックがランキング順位に与える影響は小さいのではないかという結論を得た。

目次

1．概要

1.1 研究背景・目的

昨今、スマートフォンの普及によって、スマートフォンにインストールされている音楽再生アプリを使い音楽を再生することが当たり前になった。MMDLabo株式会社が行った「2017年3月スマートフォンでの音楽視聴に関する調査」によると、利用する音楽再生機器として「スマートフォン」が76.8%で最多となった。つまり、音楽の視聴がスマートフォンの普及を経て更に身近で手軽なものとなり、我々の生活における音楽の存在感はかつてと比べて大きくなったといえるだろう。

ところで、アーティストが日々楽曲をリリースしていく中で、人々の人気を多く集める楽曲はほんのひと握りに過ぎない。一般的に、ヒット曲というのは1度聞いただけでも印象に残りやすく、その楽曲がヒットした所以というのは語るまでもないように思える。しかし、実際のところ、我々はヒット曲の人気の秘訣が何であるのか明確に理解しているわけではない。

上記で述べたように、音楽の視聴が人々により身近になっていく一方で、楽曲の人気の要因が分析されていないという現状がある。従って、楽曲の人気の要因を理解することはマーケティング戦略における楽曲の売上拡大という面でとても重要であるといえる。そのため、本研究では、楽曲の人気の要因を歌詞の側面から捉えることを目的とする。

1.2 先行研究とその課題・研究の方向性

ヒット曲の歌詞に着目し、その特徴を研究した先行研究として、戒野・鈴木（2010）の研究がある。これは、ヒット曲の歌詞を、被験者に感性ワード（明るい―暗い、など）で見たときの印象を数値で回答させている。そして、このアンケートを多数の感性ワードで行い、それらのワードを因子分析で 3 因子に分け、因子ごとの平均点を算出し、歌詞の特徴を捉えている。この研究の課題として、歌詞の特徴を人々の主観に頼って捉えていることがあげられる。歌詞の特徴を捉えるうえで、聞く人の主観によって変動してしまうのは好ましくない。できるならば、特徴を客観的にとらえられることが望ましい。

歌詞をトピック解析した先行研究として、佐々木・吉井・中野・後藤・森島（2014）の研究がある。これは、数千曲の歌詞をトピック解析し、全ての楽曲に共通のトピック5種類の比率を各楽曲の特徴としている。また、トピック比率による特徴の評価実験を行ったところ、2つの楽曲の、トピック比率が類似している度合と、被験者の楽曲を聞いた時の印象の近さは相関していることが実証されている。そのため、トピック解析による歌詞の客観的特徴の作成に成功している。しかし、作成された特徴を生かして、歌詞の楽曲の人気度への影響などは調べられていな

い。そこで、楽曲の人気度を歌詞の側面から探る。

今回の論文では 2008 年～2017 年の「Billboard Japan」年度末ランキングに掲載されている楽曲計 900 曲を元データとした。LDA を元に推定された各楽曲を構成するトピックと楽曲ごとのトピック構成比率を説明変数、確定済みのランキング順位を被説明変数にそれぞれおいた重回帰分析を機械学習させることで、一番当てはまりの良いランキング推定モデルを構築する。確定済みのデータである 2018 年のランキング結果でモデルの検証を行い、モデルが十分に構築され関係性が証明された場合のみ、最終的には 2019 年 10 月末時点での上位ランキングに掲載される楽曲が年度末時点でどのように順位変動するのかを推定する。

2. トピック解析

2.1 トピックモデルとは

1．　トピック k＝1,…,K 全てに関して
　単語分布を生成する　$\phi_k \sim Dirichlet(\beta)$
2．　文書 d=1,…,D に関して
（a）トピック分布を生成　$\theta_d \sim Dirichlet(\alpha)$
（b）単語 n=1,…,N_d全てに関して
　i．トピックを生成　$z_{dn} \sim Categorical(\theta_d)$
　ii．単語を生成　$w_{dn} \sim Categorical(\phi_{z_{dn}})$

図 1 ：トピックモデルの生成過程

トピックモデルとは、ある文書は、d 個の単語からなり、また、1 つの単語ごとに 1 つのトピックを持つと仮定する文書生成モデルである。トピックの数は K 個であるとすると、トピックモデルでは文書ごとにトピック分布$\boldsymbol{\theta_d} = (\theta_{d1} \dots \theta_{dK})$がある。ここで、$\theta_{dk} = p(k|\boldsymbol{\theta_d})$は文書 d の単語にトピック k が割り当てられる確率で、$\theta_{dk} \geq 0, \sum_{k=1}^{K} \theta_{dk} = 1,$を満たす。トピック分布$\boldsymbol{\theta_d}$に従って文書 d のそれぞれの単語にトピック$z_{dn}$が割り当てられる。トピックごとの単語分布$\Phi = (\phi_1, \dots \phi_k)$が存在し、$\phi_k = (\phi_{k1}, \dots, \phi_{kV})$はトピック k の単語分布を表し、$\phi_{kv} = p(v|\phi_k)$はトピック k で語彙 v が生成される確率$(\phi_{kv} \geq 0, \sum_{v=1}^{V} \phi_{kv} = 1)$を表している。具体的な文書集合の生成過程を以下の図に示す。文書ごとのトピック分布$\boldsymbol{\theta_d}$およびトピック毎の単語分布$\boldsymbol{\phi_k}$はカテゴリ分布のパラメータだから、その共役事前分布であるディリクレ分布から生成されると仮定する。

トピック分布$\boldsymbol{\theta_d}$と単語分布集合Φが与えられたときの文書$\boldsymbol{w_d}$の確率は以下の式で表される。

$$p(w_d|\theta_d,\Phi)=\prod_{n=1}^{N_d}\sum_{k=1}^{K}p(z_{dn}=k|\theta_d)p(w_{dn}|\phi_k)=\prod_{n=1}^{N_d}\sum_{k=1}^{K}\theta_{dk}\,\phi_{kw_{dn}}$$

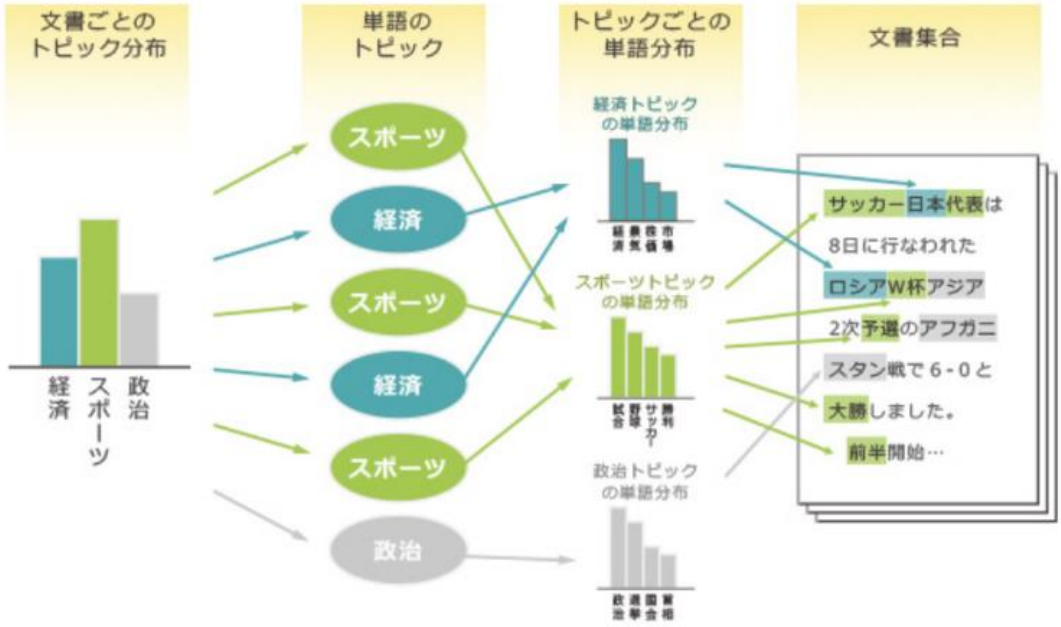


図２：トピックモデルイメージ

（株式会社 Albert 提供の Web ページ：データ分析基礎知識より抜粋）

表１：本章で用いる記号

記号	説明
D	文書数
N_d	文書 d に含まれる単語数（文書長）
V	全文書のなかで現れる単語の種類数（語彙数）
$\boldsymbol{W}$	文書集合
$\boldsymbol{w_d}$	文書 d
w_{dn}	文書 d の n 番目の単語
K	トピック数
N_k	文書集合全体でトピック k が割り当てられた単語数
N_{dk}	文書 d でトピック k が割り当てられた単語数
N_{kv}	文書集合全体で語彙 v にトピック k が割り当てられた単語数

θ_{dk}	文書 d でトピック k が割り当てられる確率
ϕ_{kv}	トピック k のとき語彙 v が生成される確率

2.2 LDA（潜在的ディリクレ配分モデル）および変分ベイズ推定について

トピックモデルの推定法として、最尤推定、変分ベイズ推定、ギブスサンプリングなどが提案されている。トピックモデルを最尤推定する手法を確率的潜在意味解析（PLSA）という。一方、トピック分布$\boldsymbol{\theta}_d$にディリクレ事前分布を仮定し、ベイズ推定（変分ベイズ推定やギブスサンプリング）する手法は、潜在的ディリクレ配分モデル（latent Dirichlet allocation, LDA）と呼ばれている。ちなみにディリクレ分布とは、連続型の確率分布である。ディリクレ分布の確率密度関数は、同時に発生することのない K 個の事象がそれぞれ$\alpha_i - 1$回発生したときに、各事象の起こる確率がx_iである確率を与える。

変分ベイズ推定を用いることで、未知変数の事後分布を推定できる。まず、変分ベイズ推定における周辺尤度の最大化についてみていく。トピックモデルにおける未知変数はトピック集合$\boldsymbol{Z} = (z_{11}, \dots, z_{1N_1}, z_{21}, \dots, z_{DN_D})$、トピック分布集合$\boldsymbol{\Theta} = (\boldsymbol{\theta_1}, \dots, \boldsymbol{\theta_D})$、単語分布集合$\boldsymbol{\Phi}$である。これらの事後分布の近似である変分事後分布を求める。$q(\boldsymbol{Z}, \boldsymbol{\Theta}, \Phi)$を求める。

トピックの対数周辺尤度

$$\log p(\boldsymbol{W}|\alpha, \beta) = \log \int \int \sum_{Z} p(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\Theta}, \boldsymbol{\Phi}|\alpha, \beta)\, d\boldsymbol{\Theta} d\boldsymbol{\Phi}$$

の変分下限 F はイェンゼンの不等式を用いて、

$$F = \int \int \sum_{\boldsymbol{Z}} q(\boldsymbol{Z}) q(\boldsymbol{\Theta}, \boldsymbol{\Phi}) [\log p(\boldsymbol{Z}|\boldsymbol{\Theta}) p(\boldsymbol{\Theta}|\alpha) p(\boldsymbol{W}|\boldsymbol{Z}, \boldsymbol{\Phi}) p(\boldsymbol{\Phi}|\beta) - \log q(\boldsymbol{Z}) q(\boldsymbol{\Theta}, \boldsymbol{\Phi})] d\boldsymbol{\Theta} d\boldsymbol{\Phi}$$

と計算できる。

変分下限 F を最大にするトピック分布の事後分布 q($\boldsymbol{\Theta}$)を求めると、

$$q(\boldsymbol{\Theta}) = \prod_{d=1}^{D} Dirichlet(\boldsymbol{\theta_d}|\alpha_{d1}, \dots, \alpha_{dK})$$

$$(Dirichlet(\boldsymbol{\theta_d}|\alpha_{d1}, \dots, \alpha_{dK}) = \frac{1}{B(\alpha_{d1}, \dots, \alpha_{dK})} \prod_{i=1}^{K} {\theta_{di}}^{\alpha_{di}-1})$$

が得られる。α_{dk}はディリクレ分布である変分事後分布 q($\boldsymbol{\theta_d}$)のパラメータ

$$\alpha_{dK} = \alpha + \sum_{n=1}^{N_d} q_{dnk}$$

である。ここで$q_{dnk} \equiv q(z_{dn} = k)$を文書 d の n 番目の単語トピック$Z_{dn}$が k となる

変分事後確率とした。また、単語分布の変分事後確率$q(\boldsymbol{\Phi})$は

$$q(\boldsymbol{\Phi}) = \prod_{\boldsymbol{k=1}}^{\boldsymbol{K}} Dirichlet(\phi_k|\beta_{k1}, \ldots, \beta_{kV})$$

となる。β_{kv}はディリクレ分布である変分事後分布$q(\boldsymbol{\phi_k})$のパラメータ

$$\beta_{kv} = \beta + \sum_{d=1}^{D} \sum_{n:w_{dn}=v} q_{dnk}$$

である。トピックの変分事後分布$q(Z) \propto \prod_{d=1}^{D} \prod_{n=1}^{N_d} \prod_{k=1}^{K} q_{dnk}^{I(z_{dn}=k)}$は
$q_{dnk} \propto \exp\left(\Psi(\alpha_{dk}) - \Psi\left(\sum_{k'=1}^{K} \alpha_{dK'}\right) + \Psi\left(\beta_{kw_{dn}}\right) - \Psi(\sum_{v=1}^{V} \beta_{kv})\right)$となる。ここで I(･)は、A が真であれば I(A)＝１、そうでなければ I(A)= 0 となる、指示関数を表わす。以下プログラミングでパラメータを推定するさいの流れを書き表したものである。

表２：変分ベイズ推定のプログラミングの流れ
（機械学習モデルプロフェッショナルシリーズトピックモデル p.64 から抜粋）

1: $N_{dk} = 0,\ N_{kv} = 0,\ N_k = 0$　#カウントを初期化　$d = 1, \cdots, D,\ k = 1, \cdots, K,\ v = 1, \cdots, V$
2: $z_{dn} = 0$　#トピックを初期化 $d = 1, \cdots, D,\ n = 1, \cdots, N_d$
3: **repeat**
4:　**for** $d = 1, \cdots D$ **do**
5:　　**for** $n = 1, \cdots, N_d$ **do**
6:　　**if** $z_{dn} > 0$ **then**
7:　　$N_{dz_{dn}} = N_{dz_{dn}} - 1$　#カウントから z_{dn} の割当分を引く
8:　　$N_{z_{dn}w_{dn}} = N_{z_{dn}w_{dn}} - 1$
9:　　$N_{z_{dn}} = N_{z_{dn}} - 1$
10:　　**end if**
11:　　**for** $k = 1, \cdots, K$ **do**
12:　　$p(z_{dn} = k|\mathbf{W}, \mathbf{Z}_{\backslash dn}) \propto (N_{dk} + \alpha)\frac{N_{kw_{dn}} + \beta}{N_k + \beta V}$　#サンプリング確率を計算
13:　　**end for**
14:　　$z_{dn} \sim \text{Categorical}(p(\mathbf{z}_{dn}|\mathbf{W}, \mathbf{Z}_{\backslash dn}))$　#トピックをサンプリング
15:　　$N_{dz_{dn}} = N_{dz_{dn}} + 1$　#カウントに新たに割り当てたトピックの分を加える
16:　　$N_{z_{dn}w_{dn}} = N_{z_{dn}w_{dn}} + 1$
17:　　$N_{z_{dn}} = N_{z_{dn}} + 1$
18:　**end for**
19:　**end for**
20:　$\alpha^{\text{new}} = \alpha \frac{\sum_{d=1}^{D}\sum_{k=1}^{K} \Psi(N_{dk}+\alpha) - DK\Psi(\alpha)}{K\sum_{d=1}^{D} \Psi(N_d+\alpha K) - DK\Psi(\alpha K)}$　#ハイパーパラメータを更新
21:　$\beta^{\text{new}} = \beta \frac{\sum_{k=1}^{K}\sum_{v=1}^{V} \Psi(N_{kv}+\beta) - KV\Psi(\beta)}{V\sum_{k=1}^{K} \Psi(N_k+\beta V) - KV\Psi(\beta V)}$
22: **until** 終了条件を満たす

2.3 確率モデルの評価について

確率モデルの性能を評価する尺度として、テストデータに対するハープレキシティ（perplexity）が使われる。ハープレキシティは負の対数尤度から計算できる値で、低いハープレキシティはテストデータを高い精度で予測できる良い確率モデルあることを表わす。ハープレキシティは

$$perplexity(\boldsymbol{W}^{test}|M) = exp(\frac{\sum_{d=1}^{D^{test}} \log p(\boldsymbol{W}_d^{test}|M)}{\sum_{d=1}^{D^{test}} N_d^{test}})$$

で計算できる。Test はテストデータであること。M は確率モデルを表わす。ハープレキシティは尤度の逆数$1/\mathrm{p}\,(\boldsymbol{W}_d\hat{}test\,|\,M)$の 1 単語当たりの幾何平均であり、平均して 1/ハープレキシティの割合で単語を予測することを表わす。すべての語彙が一様の確率で出現するモデルのハープレキシティは語彙数 V になり、すべての単語を予測できるモデルのハープレキシティは 1 になる。トピックモデルの尤度は

$$p(w_d|M) = \prod_{n=1}^{N_d} \sum_{K=1}^{K} \theta_{dk}\, \phi_{kw_{dn}}$$

となる。

3. 実証分析

3.1 データの選択理由

「Billboard Japan」の年度末（11 月時点集計・12 月ランキング公開）における楽曲ランキングを元にした。

まず、「Billboard Japan」のデータを選定した理由としては、2 点ある。

1 つ目に、世界的に著名なランキングであるからである。「Billboard」は 1894 年に創業した米国の芸能メディアのブランドであり、音楽業界誌『ビルボード』を出版している。世界的にも知名度の高い米国の音楽チャートである「Billlboard Hot100」を発表しており、かつて故坂本九の代表作である「SUKIYAKI（和名：上を向いて歩こう）」が音楽史上歴史的快挙となるランキング 1 位に輝いたエピソードは有名な話だ。今回はそのような著名なランキングの日本版として、日本における楽曲をランキングした「Billboard Japan」をとりあげた。

2 つ目に、評価項目である。一般的に音楽ランキングいえば、CD の売上や配信ダウンロード数等特定の 1 項目における集計結果をランキング化したものであるが、「Billboard Japan」においては「売上」と「オンエア」の 2 側面から評価している。一例には itunes の販売実績やラジオ放送回数、ツイート

数などがこれに当てはまる。

3.2 データの取得過程

データの取得は python を用いた web スクレイピングによっておこなった。なお私的な分析用のみ無断のスクレイピングが許可されているため注意である。

今回必要なデータを取得するために考えた工程は 3 工程であった。そのそれぞれに関して詳しく説明していきたい。なおデータの取得コードに関しては appendix に記載する。そちらも併せて参照されたい。

Billboard ランキングから 2008 年から 2017 年のランキング 100 に掲載されている「順位」「歌手名」「楽曲名」のデータを取得した。

図 1 ：取得コード①

```
def get_info(ele):


        #辞書形式で順位、歌手名、楽曲名を抜き取る。
        info = {
                "rank": (ele.find("span")).string.strip("<span>"),
                "music_title": ele.find("p", class_="musuc_title").string.strip(),
                "singer": ele.find("p", class_="artist_name").string.strip()
                }
        return info


def main():


    for n in range(0,10):
        url = uri + 'charts' + '/' + 'detail?a=hot100_year&year=' + str(2008+n)
        soup = ama(url)
        for n in range(0,101):
            for ele in soup.find_all("tr",class_="rank{}".format(n)):
                info.append(get_info(ele))
```

まず、取得対象である「Billboard Japan」ランキングの URL に着目し、URL の最後の部分を変更しながら繰り返し読み込みさせることにより、10 年分のデータを指定した。次に取得したいデータの HTML タグを指定し、python の web スク

レイピングモジュール「Beautiful soup」を使用して「順位」「歌手名」「楽曲名」を取得している。

なお、2009 年と 2010 年に関してはランキング掲載分がもともと 50 曲であったため計 900 曲分の取得となっている。

3.2.1 にて取得した「順位」「歌手名」「楽曲名」のデータを元に歌詞検索サイトにて該当情報の検索を行った。今回歌詞の検索サイトとしては「J-Lyric.net」を利用した。

歌詞の検索においては「歌手名」「楽曲名」を検索し、検索結果から見たい楽曲の個別ページにアクセスするのが主流ではあるが、今回はその検索結果ページの URL（下記見本）の法則性に着目し、3.2.1 の情報を元に繰り返し、楽曲個別ページのスクレイピングをおこなった。

図 2 ：URL 見本

```
url=uri+'index.php?kt='+（楽曲名）+ '&ct=1&ka='+ （歌手名）+ "&ca=1&kl=&cl=2"
```

3.2.3 で取得した歌詞個別ページへの URL を参照に、URL アクセス後のページから歌詞データを取得した。取得した歌詞データを csv ファイル化し、のちに行う分析過程の読み込みファイルとして取り扱った。

なお取得した歌詞データのうち、複数の年度にまたがってランキング入りをしている楽曲に関しては一番ランキング順位が高かった年度のみを残し、その他年度は除外している。

またスクレイピング可能な楽曲のみを取得した結果、収集できた歌詞データは 662 曲となった。

3.3 データの処理過程

当論文では取得した歌詞のデータの処理に「形態素解析」を用いた。

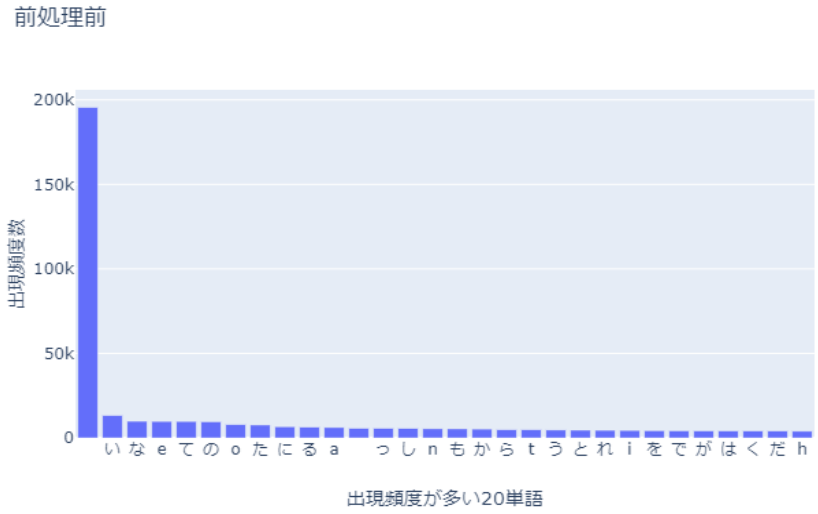
形態素解析とは例えば「8 月 3 日に放送された「中居正広の金曜日のスマイルたちへ」という文字列を下記のような品詞と意味の塊に区切って捉えるということである。

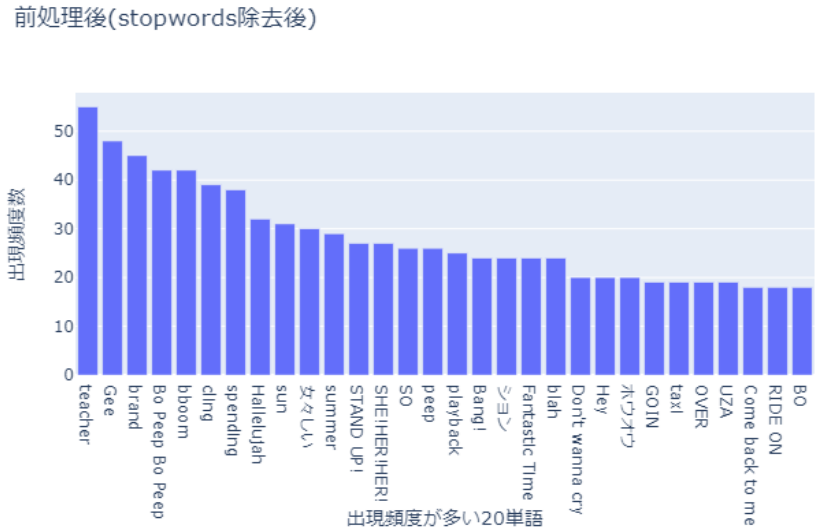
8 月 3 日	ハチガツミッカ	8 月 3 日	名詞-固有名詞-一般		
に	ニ	に	助詞-格助詞-一般		
放送	ホウソウ	放送	名詞-サ変接続		
さ	サ	する	動詞-自立	サ変・スル	未然レル接続
れ	レ	れる	動詞-接尾	一段	連用形

た　　タ　　た　　助動詞　特殊・タ基本形
「　　「　　「　　記号-括弧開
中居正広の金曜日のスマイルたちへ ナカイマサヒロノキンヨウビノスマイルタチヘ　　中居正広の金曜日のスマイルたちへ 名詞-固有名詞-一般
」　　」　　」　　記号-括弧閉

そして、この区切られた単語群の中からストップワードと呼ばれる、通常多用されておりそれ自体では意味をなさない「ある」「する」「ない」などの不要語を取り除く精度を向上させた。

下記の単語出現頻度をまとめた図をみてほしい。





2 つの違いを見てもわかるように、明らかに不要な助詞が削除され基本的に名詞が一番多い出現回数で観測された。また近年の歌詞の特徴として同一文言を繰り返し歌詞内で多用する傾向がわかり、特に「Bo Peep Bo Peep」や Gee 等の K-POP の楽曲でその傾向は顕著に見受けられた。

3.4 分析結果

3.3 で前処理を行った歌詞データに対し 2 章で取り扱ったトピックモデル分析を行った。トピックモデルを分析するにあたり設定しなければならないパラメータとして「トピック数」と「アルゴリズムの学習回数」があるが、当論文では歌詞データ量も多くない状態で高い精度が出るようにしたいという意図の元で交差検証した結果、トピック数は 3、アルゴリズムの学習回数は 25 が適正値であると判断された。このパラメータによる条件の元、導かれたトピックが下記である。

Topic # 0	Topic # 1	Topic # 2
いい: 0.00630756514146924	YOU: 0.01693001762032509	YOU: 0.013954239897429943
心: 0.005849752575159073	キミ: 0.006969604175537825	It: 0.010872512124478817
僕ら: 0.005843274295330048	＝LOVE: 0.006608517840504646	yeah: 0.008046994917094707
生きる: 0.005743515212088823	SO: 0.005141167435795069	AH: 0.007016039453446865
夢: 0.005539447069168091	MY: 0.004850344732403755	baby: 0.0064558046869933605
くれる: 0.005336042959243059	愛: 0.004760330077260733	Me: 0.006082487292587757
好き: 0.005187285132706165	wanna: 0.004632297437638044	ON: 0.005895985756069422
知る: 0.005164839792996645	いい: 0.0042932601645588875	N・O: 0.005878681782633066
いく: 0.0048111239448189735	NA: 0.004258356057107449	TO: 0.005166532471776009
空: 0.004761694930493832	L・A: 0.004058786667883396	MY: 0.0050821262411773205

そしてこのモデルの精度を表す perplexity の数値は

perplexity: 2218.609253571976

であったため比較的精度が高いモデルが組めたのではないだろうか。
また其々のトピックに含まれる語句から各トピックがどのような楽曲であるかをまとめたものが下記になる。

Topic0：青春系
Topic1：ラブソング
Topic2：ノリの良い楽曲

この結果から、Billboard Japan で評価される楽曲の歌詞の特徴に、今回判明したトピック傾向と含まれる語群の割合が高いことが判明した。

3.5 ランキング予測

当論文の目的である歌詞の潜在的トピックとランキング結果との有意な関係性があるのかということに関して論じる。

3.5.1 トピック割合

3.4 で構成したモデルで出力されるデータを説明変数にとり、ランキングを予測する。出力データは下記のような歌詞ごとに設定したトピックの配分割合である。例として 2008 年ランキング 1 位の楽曲「キセキ」における配分割合は下記である。

Topic0：青春系	Topic1：ラブソング	Topic2：ノリの良い曲
0.994428	0.002827	0.002743

このように「キセキ」は青春系のトピック配合割合が極めて高いことがわかる。

3.5.2 重回帰分析

トピック割合を説明変数に、ランキング順位を被説明変数にとった重回帰分析をおこなった。重回帰モデルにおける予測モデルの形と出力結果は下記である。

【予測モデル】

$v = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$

※凡例

α:切片項

$\beta_1, \beta_2, \beta_3$：偏回帰係数

x_1, x_2, x_3：説明変数

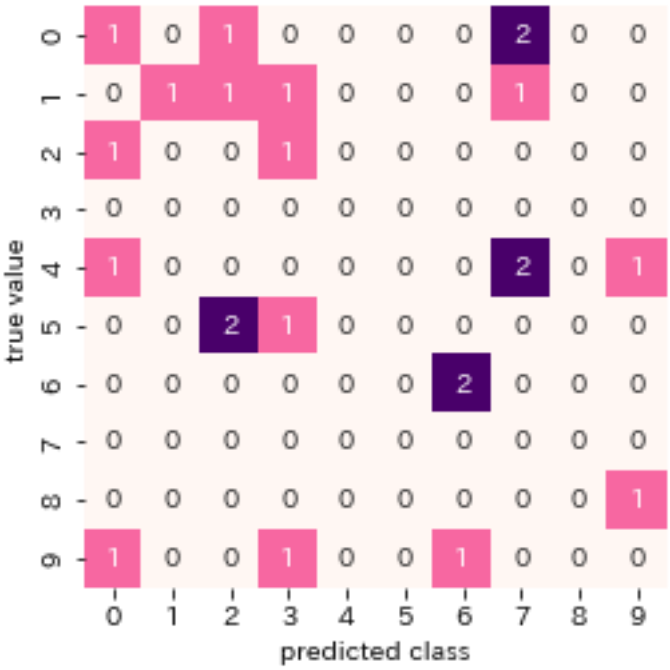
【出力結果】

Name	Coefficients
x_1(Topic1)	-2188.804354
x_2(Topic2)	-2182.696915
x_3(Topic3)	-2181.769001
α	2233.162831436396

この結果を実測値に当てはめ、正規化などの処理を行ったが、純粋なランキング数値を求めることはできなかった。

3.5.3 RandomForestClassifier

次に回帰モデルでうまくいかなかったことから分類問題として取り扱った。
使用した手法「ランダムフォレスト」はある条件を満たすか否かの 2 値問題で条件分岐をおこない（1 つ 1 つの条件分岐を決定木という）、その条件分岐の組み合わせでランキング順位のどこに行きつくのかを分類する手法である。
今回の分類ではランキング 100 位までを分類することは困難と判断したため、ランキング 10 位までが分類できているかを観測することとする。
分類の結果、練習データとテストデータの学習精度に関して、練習データの精度は 100%となったが、テストデータを当てはめた時の精度は 17%と非常に低い結果が出た。
また混合行列による識別結果は下記である。



縦軸が本当の分類を横軸が予測の分類をそれぞれ表すため、対角成分が正しい判定がされた回数を表す。この表を見てわかるように識別率は非常に低いことが分かった。

4　結論

Billboard ランキングに掲載されている楽曲の歌詞データを用いてランキング順位を予測することは難しいことが分かった。歌詞データの潜在的なトピックはあくまで歌詞を文字列として取り扱った際に単語本来の意味から推測されるものである。それゆえ楽曲を評価する人物が楽曲の歌詞に着目して場合でも、歌詞の大

まかな印象はトピックで説明できるが文脈における歌詞に込められた意味などのより訴求力の高い因子により印象が変化してしまうのではないかと考える。当論文の課題と展望は次節に示すが、いずれにせよランキング順位の予測には多くの要素が絡んでいることは必至であり、今回取り上げた歌詞のトピックという因子はその要素の中でも大きな影響を与えない因子であるということが分かった。
今後の展望を段階別に整理をする。
まずデータ分析段階に関して、今回取得した歌詞データは年毎にランキング掲載数が異なっていた。また、ランキングを元にした歌詞データの取得の段階で約 1/3 のデータの欠損が起こった。
この点は後者の方に改善余地があり、主にスクレイピングコードの改良でより取得可能数が増えるのではないかと思われる。
次にデータ処理段階に関して、今回取り上げた LDA 自体の学習精度は高かったが、ランキングとの関連性に移った際に急激に精度が落ちた。その原因としてそもそもトピックとランキングの関連性が低いことが前提にある可能性が高いが、それ以外の要因としてモデル精度が低かったことが考えられる。これに対しては LDA の上位モデルとして sLDA(supervised LDA)の実装を目指すことが有効かと考える。このモデルは被説明変数の情報を加味しながら説明変数であるトピックとその配分割合が設定される。しかしこのモデルはまだあまり普及しておらず実装している例が少ないため、今後さらに注意深く動向を見守る必要がある。
最後にデータ分析段階に関して、ランキング推定においては今回の結果から対象とする説明変数の種類が圧倒的に少ないことが課題であった。そのため今回取り上げたトピックの他に、人がが曲を評価する項目である、曲（メロディー）、歌手に対するデータを組み込むことで、より精度の高いモデルを作ることができるとともに、どの因子が一番ランキングに強い影響を与えているかが判明するであろう。

当論文では当初の目的であるランキングの将来予測まで、モデル構築がうまくいかなかったために至らなかったが、課題から見える今後の方針は定まった。今回の論文で分かったことを元にさらなるモデルの拡張に取り組んでいきたい。

5　付録

データ取得のみ参照コードを記載する

①3.2.1 ソースコード

```
import urllib.request, urllib.error
from bs4 import BeautifulSoup
import time
import datetime
import pandas as pd

uri = 'http://www.billboard-japan.com/'

info = []


def ama(url):
    #各ページを html 形式で抜き取る
    html = urllib.request.urlopen(url)
    soup = BeautifulSoup(html, 'html.parser')
    time.sleep(1)

    return soup


def get_info(ele):

        #辞書形式でランク、タイトル、歌手名を抜き取る。
        info = {
                "rank": (ele.find("span")).string.strip("<span>"),
                "music_title":                                    ele.find("p",
class_="musuc_title").string.strip(),
                "singer": ele.find("p", class_="artist_name").string.strip()
                }
        return info

def main():

    for n in range(0,10):
        url = uri + 'charts' + '/' + 'detail?a=hot100_year&year=' + str(2008+n)
        soup = ama(url)
        for n in range(0,101):
            for ele in soup.find_all("tr",class_="rank{}".format(n)):
                info.append(get_info(ele))

    today = datetime.datetime.now()
    today = '{}'.format(today.strftime('%Y.%m.%d'))

    #pandas を使用して CSV 形式で出力
    df = pd.DataFrame(info)
    df.to_csv('Billboard_top100_analytics_actual_'     +     str(today)     +
'.csv',encoding='utf-8_sig')

if __name__ == '__main__':
```

```
    main()
```

②3.2.2 ソースコード

```
def from_dict_method1(sort=False):
    some_df2 = pd.DataFrame([],columns=("music_title","rank"))
    dict_array = []
    for j in range(0,len(result)):
        dict_array.append(result[j])
        some_df3 = pd.concat([some_df2, pd.DataFrame.from_dict(dict_array)])
    return some_df3
p1=from_dict_method1()
rank1=[]
for i in range(0,len(p)):
    for j in range(0,len(p1)):
        if (p["music_title"][i]==p1["music_title"][j]):
            rank1.append({"music_title": p1["music_title"][j],"rank":
p1["rank"][j],"link":p["link"][i]})
        else:
            pass

import itertools
result = [music_title[0] for music_title in itertools.groupby(rank1)]
#import itertools
#result1 = [music_title[0] for music_title in itertools.groupby(rank1)]

def from_dict_method2():
    some_df4 = pd.DataFrame([],columns=("link","music_title","rank"))
    dict_array = []
    for j in range(0,len(result1)):
        dict_array.append(result1[j])
        some_df5 = pd.concat([some_df4, pd.DataFrame.from_dict(dict_array)])
    return some_df5
p2=from_dict_method2()
p3=p2.drop_duplicates(subset='link')
p4=p3.reset_index()
```

```
p4=p4.drop("index", axis=1)
print(p4)
p4.to_csv("link_music_rank.csv", sep=",")
```

③3.2.3 歌詞取得コード

```
#取得したリンク・曲タイトルからリンクアクセス後の楽曲歌詞をスクレイピング
info2=[]
def ama(url):
    #各ページを html 形式で抜き取る
    html =url
    res = req.get(html, timeout=2000)
    soup = BeautifulSoup(res.content, 'html.parser')
    time.sleep(2)
    return soup
def get_kashi(sing):
        info2={
                "lyrics": (sing.find("p",id="Lyric")).get_text(),
                "author": (sing.find("p",class_="sml")).get_text()
                }
        return info2
def main():

    for n in range(0,len(p4)):
        url = p4["link"][n]
        soup = ama(url)
        for sing in soup.findAll("div",class_="lbdy"):
            info2.append(get_kashi(sing))
            time.sleep(1)
if __name__ == '__main__':
    main()
kashi=[]
for n in range(0,len(info2)):
    kashi.append(info2[n]["lyrics"])
df_kashi=pd.DataFrame(kashi)
```

6　参考文献

岩田具治．“トピックモデル”．機械学習モデルプロフェッショナルシリーズトピックモデル．6 版．講談社．2015 年．

佐々木将人・吉井和佳・中野倫靖・後藤真孝・森島繁生．LYRICS RADAR:歌詞の潜在的意味分析に基づく歌詞検索インターフェース.情報処理学会論文誌.
57(5),1365-1374

Web 上の資料

戒野敏浩・鈴木智博．感性 J-POP ヒット要因分析．2007 年
https://www.jstage.jst.go.jp/article/jasmin/2010f/0/2010f_0_72/_pdf
(2019/1213)

株式会社 Albert.データ分析基礎知識
https://www.albert2005.co.jp/knowledge/machine_learning/topic_model/about_topic_model（2019/12/13）