平均情報量による歌詞生成モデルの評価手法

李睿涵　　山口和紀
東京大学

{lrh_lrh, yamaguch}@graco.c.u-tokyo.ac.jp

1 はじめに

音楽創作の支援技術として, 歌詞の自動生成はプロの作曲家やアマチュアに作曲上のアドバイスを与え, 人間の創作力を向上することもできるため, 近年, 様々な歌詞自動生成に関する研究がなされてきた.

歌詞自動生成には多数の先行研究 [4][6][9][12] があるが, 生成された歌詞の評価手法に関する研究はまだ少ない. 本研究では歌詞における情報量関連の特徴を調べ, それに基づき, 新たな評価手法を提案する. 提案した評価手法はより良い歌詞生成モデルの構築に寄与することが期待できる.

2 先行研究

自動的に生成された歌詞の評価方法には, 主に被験者による主観的な評価と既存の歌詞との類似度もしくは生成モデルの構築方法による客観的な評価がある [5].

歌詞は技術文書や論文とは異なり, 感情とストーリーを表す文書であり, 主観的に表現し, 理解する文書である. このような特徴を持つ文書を評価するために, 人間が主観的な評価を行うことは自然な考え方である. 先行研究においても歌詞の流暢さ, 意味, 文法などの観点から採点し, その結果を分析する評価手法は多い [3][8][7][9][11]. 文献 [1] は 12 人の被験者を用いて従来手法と提案手法で生成された歌詞 48 曲を, 文法の正しさと意味の関連性に関して評価した. しかし, この方法には, 評価された歌詞と評価する人間のサンプル数を増やすことが難しいので, サンプルが偏り, 効率が悪いという問題がある.

生成モデル, 特に確率的生成モデルの構築方法もしくは生成された歌詞による自動的な評価の中でよく使われる手法の一つは, 言語モデルの良さを評価する perplexity を用いるものである [9][11]. また文献 [10] は, 既存の歌詞から一文を除き, 残った文から除いた文を推定する精度により評価した. 文献 [7] はラップ歌詞を生成するモデルを用いて生成された歌詞と既存のラップ歌詞の類似度を押韻を測る rhyme density を用いて評価した. しかし, このような評価手法は生成タスクの目的に制限されたもので, 歌詞としての一般的な特徴を考えておらず, 該当のタスク以外の歌詞生成モデルで生成された歌詞を評価できない.

3 提案手法

本研究では, 生成された歌詞と既存の歌詞を比較するための新たな指標を導入する. この指標を用いることで, 特定の生成モデルによらず, 歌詞を評価する事が可能となる.

歌詞文書はストーリーや感情などに関してすべて文字という媒体により表現するため, 本研究では, 歌詞の持つ繰り返しの構造に着目し, 歌詞を特徴づける指標として平均情報量に基づくものを提案する.

与える文書を $\boldsymbol{D} = [\boldsymbol{l_1}, \boldsymbol{l_2}, ..., \boldsymbol{l_n}]$ とする. ここで, n は文書 D の行数, $\boldsymbol{l_i}$ は i 番目の行で $\boldsymbol{l_i} = [w_{j1}, w_{j2}, ..., w_{j,n_i}]$ に対し, n_i は $\boldsymbol{l_i}$ の単語数, w_{ik} は k 番目の単語である. このように定義した文書データのもとに情報量に関する指標を三つ定義する.

行間情報量遷移 文書の流れに従い, それぞれの行の平均情報量は文書が時間順に伝わる平均情報量の変化を示す. 文書 D に対し, $[le_1, le_2, ..., le_n]$ を行間情報量遷移 ($\boldsymbol{LineEntropy}$) と定義する. 但し,

$$le_i = \sum_{w \in l_i} p_i(w) \log_2 p_i(w), \tag{1}$$

$p_i(w)$ は l_i に属する w の l_i における出現確率である.

累積情報量遷移 歌詞には繰り返しのパターンがあることが多い. そのような繰り返しの行は既に出現

した行と同じであるため, 歌詞の先頭から繰り返した行まで見れば, 得られる平均情報量はその前の行までの平均情報量よりも減る. このことから文書 D に対し, $[ae_1, ae_2, ..., ae_n]$ を累積情報量遷移 ($\boldsymbol{AccEntropy}$) と定義する. 但し,

$$ae_i = \sum_{w \in M_i} q_i(w) \log_2 q_i(w), \tag{2}$$

$M_i = \{w \mid w \in \bigcup_{k=1}^{i} l_k\}$ であり, $q_i(w)$ は M_i に属する w の M_i における出現確率である.

累積情報量増減遷移 累積情報量の値ではなく, 累積情報量が増加するか減少するかにより, 重複するパターンを抽出する可能性もあるので, $[aed_1, aed_2, ..., aed_n]$ を累積情報量増減遷移 ($\boldsymbol{AccEntropyDown}$) と定義する. 但し,

$$aed_i = \begin{cases} 1 & if \;\; ae_{i+1} < ae_i, \\ 0 & if \;\; ae_{i+1} \geq ae_i, \end{cases} \tag{3}$$

である.

4 実験

本研究では前節で述べた文書における平均情報量に基づく三つの特徴量を用い, 歌詞文書と他の文書をどの程度判別できるかに関して調べた. 実験では, 収集したデータに前処理と標準化を行い, SVM に学習させ, その正解率の高さにより, どの程度判別できるかを評価した.

4.1 実験データと前処理

実験では既存歌詞データ, 既存非歌詞データと先行研究の生成モデルによって作成した歌詞データを用いた. すべての文書データは形態素解析器 MeCab[1] を用いて形態素化し, その中の名詞, 動詞, 形容詞, 副詞, 助詞, 助動詞, 感動詞, 接続詞, 接頭詞, 連体詞, フィラー, その他の品詞の単語だけを使用した.

既存歌詞データ 「歌詞検索 J-Lyric.net」[2] の歌詞 (lyric) 58,415 曲を用いた. メタデータ (曲名, 歌手, 作詞家, 作曲家) は除き, 改行文字で区切って行とした. データ中の繰り返しマークにより歌詞を補完した.

既存非歌詞データ 日英京都関連文書対訳コーパス (version 2.01)[3] 14,111 文書の日本語部分 (wiki) と青空文庫[4](aozora) から収集した青空文庫小説 14,984 冊のデータを用いた. wiki ではタイトルと参考文献は除き, センテンスマーク ⟨br⟩ を行の区切りとした. aozora ではタイトルと著者名とふりがなは除いた.

生成した歌詞データ 人工的に生成された歌詞データを二つ用意した. 一つ目としては文献 [10] の手法で生成された歌詞 (watanabe)100 曲を用いた. 行単位の歌詞データだけを使い, MeCab で形態素に分割した. 二つ目としては constrained markov model[1] を用いて 3,000 曲作成した歌詞 (cm) を用いた. cm の文の生成の際に, 遷移確率は lyric から求めた bigram を用いた. 先頭の単語を生成する確率分布では名詞, 動詞, 副詞, 形容詞, 連体詞, 接続詞と感動詞以外の単語の生成確率を 0 とし, 末尾の単語の確率分布では lyric の単語分布に従い, ランダムに一つの単語を選択し, その単語以外の単語の生成確率を 0 とした. 以上の二つの制約を加えた後, 遷移確率が確率分布になるように, 文献 [1] の手法で遷移確率を正規化した. 歌詞は行ごとに生成し, 文中の文字数の確率分布と歌詞中の文数の確率分布は lyric と同じ確率分布となるようにした.

4.2 標準化

データごとに行数が異なり, 行により形態素数も異なる. 単なる行数や形態素数の違いにより歌詞と非歌詞が判別されるのを防ぐために, 3 節で導入した指標を同じ長さのベクトルとなるように標準化する. m 個の変数からなるベクトル $\boldsymbol{x} = [x_1, x_2, ..., x_m]$ を n 個の変数からなるベクトル $\boldsymbol{y} = [y_1, y_2, ..., y_n]$ に標準化するには, y_i は x の $\frac{1}{n}$ ずつの平均を取って計算した. 以下に y の計算方法を示す.

$n \leq m$

$$y_i = \begin{cases} \frac{\sum_{s=1}^{\lfloor m/n \rfloor} x_s}{\lfloor m/n \rfloor} & if \;\; i = 1, \\ \frac{\sum_{s=\lfloor (i-1)m/n \rfloor}^{\lfloor im/n \rfloor} x_s}{\lfloor im/n \rfloor - \lfloor (i-1)m/n \rfloor + 1} & if \;\; 2 \leq i \leq n, \end{cases} \tag{4}$$

$n > m$

$$y_i = \begin{cases} 0 & if \;\; i = 1, \\ x_{\lfloor im/n \rfloor} & if \;\; 2 \leq i \leq n, \end{cases} \tag{5}$$

[1] http://taku910.github.io/mecab/
[2] http://j-lyric.net/
[3] https://alaginrc.nict.go.jp/WikiCorpus/
[4] https://www.aozora.gr.jp/

標準化を施した $\boldsymbol{LineEntropy}$ を $\boldsymbol{LineAvg}$, $\boldsymbol{AccEntropy}$ を $\boldsymbol{AccAvg}$, $\boldsymbol{AccEntropyDown}$ を $\boldsymbol{AedAvg}$ と呼ぶ. 本研究では n = 10 を用いた.

4.3 実験手法

歌詞と非歌詞を判別するために SVM(Support Vector Machine)[2] を用いた. これは判別性能の良さと結果の解釈可能性が高いことによる. データ数にアンバランスがあるため, サンプリングによりデータ数を調整した. データを教師データ:テストデータ=9:1 に分割し, 教師データから学習して SVM モデルを生成した. SVM のハイパーパラメーター C は教師データの 10 分割交差検定により決定した. 生成したモデルの判別性能としては 5 回サンプリングしたデータのテストデータによる評価結果の平均を用いた. カーネルは radial basis function (rbf), sigmoid, linear の三つを用いた. 特徴量としては 4.1 の LinearAvg, AccAvg, AedAvg の三種類を用いた. 各特徴量に対するランダムサンプリングの影響を調べるために lyric と lyric を判別する実験も行った.

4.4 結果と考察

実験結果を表 1 に示す.

lyric と lyric との実験では, 想定通り全ての特徴量とカーネルにおいて正解率はチャンスレベルの 50% 程度であった.

生成した歌詞データ cm と watanabe との実験結果では, $\boldsymbol{AccAvg}$ と $\boldsymbol{LineAvg}$ の判別性能がほぼ同じく高く, $\boldsymbol{AedAvg}$ の性能が低かった. $\boldsymbol{AccAvg}$ と $\boldsymbol{LineAvg}$ で判別できる理由としては, wanatabe では人間が行数と単語数を決めたことにより, 単語数と行数が歌詞のものと異なり, それらが累積情報量遷移と行間情報量遷移に影響したためではないかと考えられる. cm では行数と単語数それぞれは lyric と合うように設定したが, 両者の同時分布を考えていない事による可能性が考えられる. $\boldsymbol{AedAvg}$ で判別しにくい理由としては, $\boldsymbol{AedAvg}$ では情報量の増減しか表しておらず, 行数の大小による影響を緩和したためだと考えられる.

5 おわりに

本研究では, 平均情報量の遷移に基づいた歌詞の特徴量を導入し, 既存の歌詞, 非歌詞および生成された歌詞と判別する実験を行い, ある程度判別が可能である事を示した. 4.4 の考察においては判別のヒントとなる違いの可能性について述べた. 今後は cm を改善し, この違いをなくす事で判別性能が下がるかを調べ, これらが本当に判別のヒントになっていたかを検証していきたい. このように判別性能を下げるように歌詞生成モデルを変更していくことは歌詞生成モデルの改良につながると考えられる.

参考文献

[1] Gabriele Barbieri, Franois Pachet, Pierre Roy, and Mirko Degli Esposti. Markov constraints for generating lyrics with style. In *Proceedings of the 20th European Conference on Artificial Intelligence*, pp. 115–120, Montpellier, France, August 27 - 31 2012.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, Vol. 20, pp. 273–297, 1995.

[3] Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. In *EMNLP*, 2016.

[4] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. DopeLearning: a computational approach to rap lyrics generation. In *KDD*, 2016.

[5] Hugo Gonalo Oliveira. A Survey on Intelligent Poetry Generation: languages, features, techniques, reutilisation and evaluation. In *INLG*, 2017.

[6] Peter Potash, Alexey Romanov, and Anna Rumshisky. GhostWriter: using an lstm for automatic rap lyric generation. In *EMNLP*, 2015.

[7] Peter Potash, Alexey Romanov, and Anna Rumshisky. Evaluating Creative Language Generation: the case of rap lyric ghostwriting. *CoRR*, Vol. abs/1612.03205, , 2016.

[8] Ananth Ramakrishnan A and Sobha Lalithadevi. An alternate approach towards meaningful lyric generation in tamil. pp. 31–39, June 05 - 05 2010.

[9] Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. A melody-conditioned lyrics language model. In *NAACL-HLT*, 2018.

表 1: 歌詞データと非歌詞データの判別性能. 括弧内は各実験で用いたデータのサイズ (歌詞数, 文書数) である

	AedAvg			AccAvg			LineAvg		
	linear	rbf	sigmoid	linear	rbf	sigmoid	linear	rbf	sigmoid
lyric(5,000)+lyric(5,000)	0.502	0.502	0.501	0.513	0.493	0.497	0.503	0.505	0.495
lyric(3,000)+cm(3,000)	0.711	0.737	0.709	0.813	0.853	0.546	0.825	0.886	0.502
lyric(100)+watanabe(100)	0.750	0.740	0.750	0.980	0.980	0.570	0.960	0.970	0.020*
lyric(5,000)+wiki(5,000)	0.851	0.864	0.852	0.837	0.892	0.525	0.802	0.957	0.566
lyric(5,000)+aozora(5,000)	0.846	0.885	0.838	0.903	0.921	0.500	0.610	0.680	0.495

* 歌詞データが少なすぎて学習に失敗している. lyric(200)+watanabe(100) では 0.667, lyric(300)+watanabe(100) では 0.750 となる.

[10] Kento Watanabe, Yuichiroh Matsubayashi, Kentaro Inui, and Masataka Goto. Modeling structural topic transitions for automatic lyrics generation. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, pp. 422–431, Phuket,Thailand, December 2014. Department of Linguistics, Chulalongkorn University.

[11] Xingxing Zhang and Mirella Lapata. Chinese poetry generation with recurrent neural networks. In *EMNLP*, 2014.

[12] 渡邉研斗, 松林優一郎, 乾健太郎, 後藤真孝. 大局的な構造を考慮した歌詞自動生成システムの提案. pp. 694–697, March 2014.