楽曲の歌詞をもとにした曲名生成

菊地 晏南　高崎 環　中本 圭

東京大学

{gakuseimail33, takasaki-meguru548, nakamoto-kei889}@g.ecc.u-tokyo.ac.jp

1　はじめに

曲名は楽曲の内容や情景と強く結びついており，言わば楽曲のメッセージの要約である。曲名と歌詞の関係は様々で，歌詞中の語から曲名をつける方法がふさわしい場合もあれば，歌詞全体のイメージを別の言葉で端的に表現する方がふさわしい場合もあり，曲名をつける作業は単純ではない。

音楽情報処理の中でも言語処理の分野では，歌詞の自動生成 [1] や歌詞に合うメロディ生成 [2] などの研究は数多くなされてきたが，歌詞から曲名を自動生成する研究はあまり前例が見られない。類似研究としてテキスト自動要約 [3, 4] や論文タイトルの自動生成 [5] などがあるが，それらは論理的に重要な内容をまとめたものであることが多く，歌詞と曲名の関係を考慮すると曲名を生成することはまた性質が異なるのではないかと考える。

本研究の目的は，歌詞から曲名を生成するという特殊な要約に対して有効なモデルの構築方法を探ることである。そのベースとして，本研究では歌詞全体を入力としたエンコーダ・デコーダモデルを採用し，Seq2Seq モデルを構築した。その上でデータの前処理及びモデル構築において異なる手法を用意し，対照実験を行った。手法のパターンとしては，次の 3 つを用意した。

- アテンションの実装の有無
- トークン化手法（MeCab による形態素解析/Sentence Piece によるサブワード分割）
- 日本語の活用語を原形に直す処理の有無（MeCab を用いた場合）

実験の結果，エンコーダ・デコーダモデルが歌詞から曲名を生成する際にも有効な手法であることが分かり，また SentencePiece の導入やアテンションの実装によりその精度が改善することが確認できた。MeCab を用いた際の結果に未知語の出力が多く，ロスによる比較が有効でないと考えられるため，活用語処理の有効性は判定できなかった。

2　関連研究

音楽言語処理に関する研究として，深山らは [2]，旋律を音の遷移経路として音の出現確率などを定義し，動的計画法により歌詞の韻律に基づいた作曲手法を提案した。

テキスト要約の関連研究としては，安藤らは [5]，TF･IDF などを用いて論文概要から重要語及び重要文を抜き出してタイトルを生成した。See らは [4]，ニュース記事の要約を生成するためにエンコーダ・デコーダモデルをベースに，固有名詞の適切な処理や出力の繰り返し回避を実装したモデルを構築した。本研究では重要語を手がかりに曲名を生成する方法も考えられるが，歌詞中に曲名が出ない楽曲も多くあることを考慮し，歌詞全体の情感を反映した曲名を生成するためにニューラルネットワークを用いることとした。

3　提案手法

3.1　ベースモデル

歌詞から曲名を生成するために，ベースモデルとしてエンコーダ・デコーダモデルを用いる。一般的なエンコーダ・デコーダモデルを図 1 に示す。

歌詞をトークン化し，それぞれのトークンに ID を振ってエンコーダに入力する。エンコーダは埋め込み層と双方向 LSTM からなる。LSTM は入力ベクトルを内部状態として RNN の通常の状態ベクトルに加え，メモリーセルの状態ベクトルを保持する。デコーダは埋め込み層と単層 LSTM からなり，その内部状態に

エンコーダの内部状態を反映している。デコーダの入力は，学習時は埋め込みされた正解曲名のトークン一つ一つであり，テスト時は一つ前の出力である。学習では，デコーダの出力と次の入力トークンとのロスが小さくなるように学習していく。

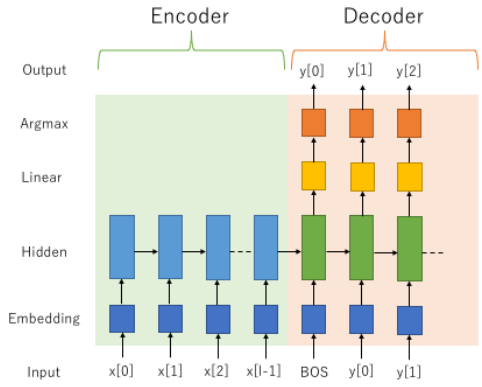

図 1: エンコーダ・デコーダモデル

このベースモデルに次の二つの手法を加え，最適なモデルを検証する。

3.2 検証手法 1：アテンション

上記のエンコーダ・デコーダモデルを，系列長の長さによる圧縮精度の影響を軽減するために，図 2 のように，エンコーダの各単語の状態ベクトルの情報をデコーダで利用する。具体的には，エンコーダの状態ベクトルに重みをつけて足し合わせることでコンテキストベクトルを作り，デコーダの状態ベクトルと合成して出力とする。これによって，デコーダの状態ベクトルとの類似度が高いエンコーダの状態ベクトルが出力に反映されやすくなることが期待される。

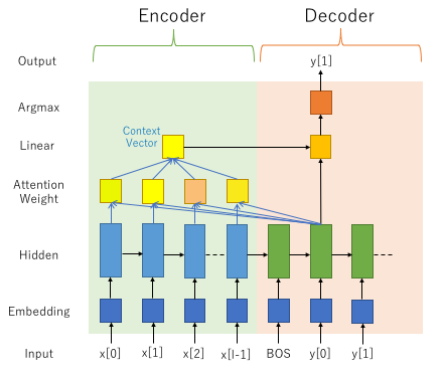

図 2: アテンション付きエンコーダ・デコーダモデル

3.3 検証手法 2：トークン化

前処理において歌詞データ及び曲名データをトークン化する際の手法として，MeCab による形態素解析と SentencePiece によるサブワード分割を用意した。形態素による分割では大量のデータを用意すると語彙数が膨大になり処理時間に大きく影響する一方で，サブワードによる分割では語彙数を指定することができるため，計算量を減らすことが期待できる。

4 実験

4.1 データと前処理

歌詞検索サイト J-Lyric.net から約 24 万曲分の歌詞と曲名のデータを取得し，訓練セット：開発セット：テストセット = 198：1：1 に分割した。開発セットは学習の早期打ち切りのために用意した。その後の処理は以下の選択手法によって異なる。また本研究では，2 曲を 1 バッチとしてバッチ処理を行った。バッチ内でトークン数を揃えるため歌詞の短い曲に対しては最後に<pad>という文字列を不足分だけ補ってからモデルに入力した。

4.1.1 前処理手法 1：MeCab

全データを形態素解析器 MeCab により分かち書きし，訓練セットの出現単語から歌詞と曲名に対してそれぞれ辞書を作成した。その際，活用語（動詞・形容詞・形容動詞・助動詞）を出現した活用形のまま辞書に入れる方法と，原形に直して辞書に入れる方法とを比較することとした。原形に直すことによって語彙数が削減できる。また固有名詞を除くため及び学習時のメモリ量の制限から歌詞中の累積出現回数が 250 回未満の単語と，曲名中の累積出現回数が 5 回未満の単語は辞書から除外した。

4.1.2 前処理手法 2：SentencePiece

訓練セットを SentencePiece により語彙数 4000 でサブワード分割し，歌詞と曲名に対してそれぞれ辞書を作成した。ただし，SentencePiece は低頻度文字を指定された語彙数にカウントしないため，最終的な語彙数は歌詞では約 11100，曲名では約 4800 となった。

4.2 未知語・終端文字・補填文字の扱い

本研究では，データから得られた単語群に加え，<unk>,<eos>,<pad>の 3 つの特殊文字を用意した。学習時及びテスト時に辞書に存在しない単語は全て共通して<unk>という文字列に置換して入力した。また，テストで曲名を出力する際の終端を知るために<eos>という文字列を用意して学習した。<pad>は前述したようにバッチ内で歌詞の長さを揃えるために用意した。

4.3 実験設定

本研究において学習時のロスは次のような手順で定義した。

1. デコーダの出力と正解の曲名の各トークンのベクトルを Softmax 関数により正規化する。
   $x = [x_1, x_2, ..., x_n]$ を
   $x' = [f(x_1), f(x_2), ..., f(x_n)]$ に正規化する Softmax 関数は次のように定義される。

$$f(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{n} e^{x_k}} \tag{1}$$

2. 正規化されたデコーダの出力と正解の曲名との間で 1 トークンずつ Cross Entropy 誤差を求める。
   真のベクトル $x' = [x'_1, ..., x'_n]$ と
   推定のベクトル $y' = [y'_1, ..., y'_n]$ の Cross Entropy 誤差は次のように定義される。

$$H = -\sum_{k=1}^{n} x'_k \log y'_k \tag{2}$$

3. 全トークン分の Cross Entropy 誤差の平均をロスとした。

また 1 エポック学習するたびに開発セットを用いてロスを計算し，このロスが 3 エポック続けて減少が見られなくなったところで学習を打ち切った。

4.4 評価方法

本研究では，歌詞から生成した曲名が歌詞のメッセージを表現できるモデルであることを確認したい。歌詞と出力された曲名を見せて被験者に歌詞にふさわしい曲名かどうかを評価してもらう方法もあるが，人手評価はコストがかかるので今回は簡便のため前述したロスをそのままモデルの性能評価に利用した。

5 結果と考察

5.1 結果

各モデルでテストデータを用いて計算されたロスを表 1 に示す。トークン化に SentencePiece を用いた場合と，MeCab を用いてさらに活用語処理をした場合はアテンションを実装したほうがロスが下がったが，MeCab を用いて活用語非処理の場合はロスが上がった。

表 1: 各モデルにおけるロス

使用モデル			ロス
アテンションなし	MeCab	活用語処理	51104
		活用語非処理	56148
	SentencePiece		128024
アテンションあり	MeCab	活用語処理	51997
		活用語非処理	52852
	SentencePiece		110679

出力された曲名のうち，<unk>が含まれた曲数の割合を表 2 に示す。

表 2: 各モデルにおける<unk>の出現確率

使用モデル			出現確率
アテンションなし	MeCab	活用語処理	96.29%
		活用語非処理	96.63%
	SentencePiece		0%
アテンションあり	MeCab	活用語処理	77.51%
		活用語非処理	77.00%
	SentencePiece		0%

また、具体的に出力された曲名のうち比較的<unk>が少ないものを表 3 に示す。

5.2 考察

本研究でベースモデルとして採用したエンコーダ・デコーダモデルは，歌詞から曲名を生成する特殊な要約でも有用な手法であると考えられる。SentencePiece を用いたモデルでは，アテンションを付けた方がロスが下がったため，アテンションが曲名生成に対して有効な手法であることがわかった。表 3 に示すように正解曲名中に出現する語句を出力できていたが，同じ語句の繰り返しが多くなった。MeCab を用いた場合は表 3 に示すように<unk>が多く出力された。これは学習時のメモリ量制限から曲名の語彙数を減少させたため，正解曲名中に<unk>が出現したからであると

表 3: 各モデルにおける生成曲名の比較

曲名	アテンションあり+SentencePiece	アテンションなし+SentencePiece
ブルー・ライト・ヨコハマ	こころの旅人 su	ブルーライト・ヨコハマ・ヨ
おふくろさん	みちのくい	おふくろさん
哀愁波止場	酒の港こころ	霧波止場霧波
男じゃないか	酒の男	男男
冬のカモメ	こころの旅人 super	冬のカモメ冬のカモメ冬の
Because I love you, good-bye street	good-byestreet	君をのせて君をのせて君をのせて
曲名	アテンションあり+MeCab+活用語処理	アテンションなし+MeCab+活用語処理
ブルー・ライト・ヨコハマ	ブルー・ヨコハマブルー・	ブルーライト・ヨコハマブルー
おふくろさん	おふくろさん	おふくろさん
哀愁波止場	霧の波止場波止場	<unk><unk><unk>
男じゃないか	<unk>おとこえ	夢の<unk>夢
冬のカモメ	冬の冬	冬のかもめ
Because I love you, good-bye street	<unk>BecauseBecauseBecauseBecause	BecauseofloveBecauseofloveBecauseof
曲名	アテンションあり+MeCab+活用語非処理	アテンションなし+MeCab+活用語非処理
ブルー・ライト・ヨコハマ	<unk>通り	<unk>
おふくろさん	涙すうそう	雪の華
哀愁波止場	<unk>川	北の宿
男じゃないか	<unk>じかけの	男の子守唄
冬のカモメ	桜の季節	風の中の
Because I love you, good-bye street	<unk>LOVER	Iloveyou

考えられる。正解曲名にも出力にも<unk>が出現したとき，モデルが正解トークンを出力したことになる。<unk>が出力されてもそれはふさわしい曲名とは言えないため，MeCab を用いた場合のロスはモデル精度の比較に用いることができないとした。

6 まとめ

本研究の目的は，歌詞から曲名を生成するという特殊な要約に対して有効なモデルの構築方法を探ることである。そこで，エンコーダ・デコーダモデルをベースとし，トークン化には MeCab と SentencePiece を用い，活用語を原形に直す処理の有無などの前処理手法を比較し，加えて，アテンション実装の有無も比較した。SentencePiece を用いた場合は同じ語句の繰り返しはあるものの，正解曲名に近い曲名を出力できていた。MeCab を用いた場合は曲名の語彙数を減少させたため有効な結果が得られなかった。

謝辞

本研究の実験及び論文誌筆にあたり，東京大学大学院 情報理工学系研究科電子情報学専攻 鶴岡研究室の鶴岡慶雅教授及び修士 2 年生の李凌寒氏より有益なご意見をいただいた。

参考文献

[1] 渡邉 研斗, 後藤 真孝. ありがちでない歌詞生成に向けた曲調と歌詞の関係に基づくベクトル空間モデル. 言語処理学会 第 25 回年次大会 発表論文集, pp.958–961, 2019.

[2] 深山 覚, 中妻 啓, 米林裕一郎, 酒向慎司, 西本 卓也, 小野 順貴, 嵯峨山 茂樹. Orpheus : 歌詞の韻律に基づいた自動作曲システム. 情報処理学会研究報告, 76, pp.179–184, 2008.

[3] Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In Proc. of NAACL, pp.93-98, 2016.

[4] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with Pointer-Generator Networks. In Proc. of ACL, pp.1073-1083, 2017.

[5] 安藤 一秋, 新居 雅也, 溝渕 昭二. 論文概要からのタイトル自動生成の試み. 言語処理学会 第 10 回年次大会 発表論文集, pp.A10C2-04, 2004.