古典単語と年代情報を活用した俳句の言い換え

宇野　健太 † 上垣外 英剛 ‡ 高村 大也 ‡§ 奥村 学 ‡
† 東京工業大学工学院 ‡ 東京工業大学科学技術創成研究院 § 産業技術総合研究所
{uno@lr., kamigaito@lr., takamura@, oku@}pi.titech.ac.jp

1 はじめに

近年，言語処理の分野において人間の芸術的創作活動を機械により再現する取り組みの 1 つとして詩の生成が注目されている．現在，詩の生成を対象とした研究において，韻律・リズム・トーン・詩の一貫性・詩らしさなどを考慮することで，人間が作成したような詩を生成することに成功している．

一方，俳句は「季語を含む」「字数が五七五である」「切れが存在する」などを基本技法とする詩であるが，現代俳句においては現代語だけでなく古典単語による表現がなされる作品が見受けられるなど，多くの詩とは異なる特徴も持っている．俳句を生成対象とする研究には LSTM[1] を用いた生成や SeqGAN[2] を用いた生成などが先行研究として報告されている．しかしながら，これらの研究には以下の 2 つの問題点が存在する．

1 つ目の問題点は LSTM を用いて生成される俳句は学習データに類似している句が多いという点である．学習データとして既存の現代・古典俳句を用いて言語モデルを学習していることに起因する．2 つ目の問題点は入力を改行文字として一文字目をランダムにサンプリングしているため生成俳句が制御できていないという点である．これは学習した言語モデルのみに基づき俳句を出力しているためにユーザーが出力される俳句を制御することができないという点に起因する．

これらの問題に対処するために，本研究では入力された俳句に対して，対象とする年代，古典単語が出現する比率を条件として，制御可能な言い換えを通じて，指定された年代に応じた俳句を生成可能な手法を提案する．提案手法では，入力された俳句を元に言い換えを行い，事前に用意した俳句の基本技法に忠実なテンプレートを用いて生成を行うため，学習データと類似した俳句は出力されない．また，テンプレートは各年代の特徴を考慮した上で作成され，言い換える対象となる古典単語の比率についても入力により決定されるため，出力される俳句を制御することが可能である．

2 関連研究

俳句の生成に関しては国内で様々な研究が行われており，国外においても中国詩を始めとしたポエム生成の研究が盛んに行われている．

米田ら [3] は俳句データで学習した LSTM を用いて出力俳句の候補を生成し，入力画像との適合度の高い俳句を出力する方法を提案している．廣田ら [4] は SeqGAN を用いた俳句生成を行っている．彼らはされに SeqGAN の手法を元に生成器・識別器に異なるデータセットを用いることで，データセットによって出力俳句の性質が変化することを検証した．Wang ら [5] は言語モデルによるキーワード入力からのポエム生成を行った．Yang ら [6] は教師なし翻訳タスクとして文章入力からのポエム生成を行った．

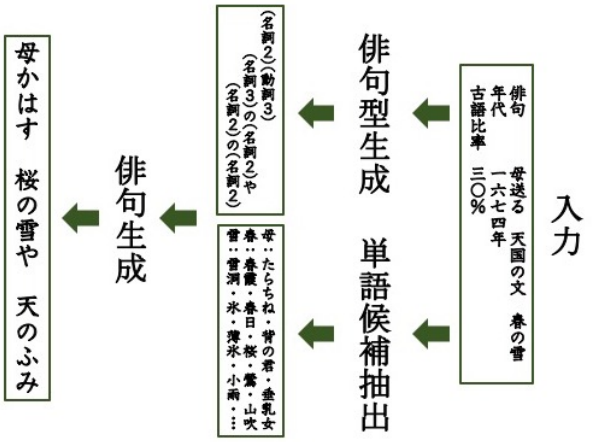

図 1: モデル概要図

しかしながら，米田らの研究では LSTM により生成される俳句は改行文字を入力としたランダムな出力から選択されており，また，入力画像との適合率の精度もキーワードに依存するためにユーザーは出力を制御できない．廣田らの研究でも敵対的学習済の生成器によりランダムに俳句を生成するため，ユーザーは出力を制御することができない．Wang,Yang らの研究ではキーワードや文章入力により詩のトピックは制御できるものの，年代や古典単語の比率といった詩の特性や構成は制御できない．

本研究では入力として俳句・年代・古典単語の比率を設けることで，ユーザーの制御に基づいて年代の特徴を捉え，言い換えの比率を調整した俳句を出力することができる．

3 提案手法

本研究では俳句言い換えのために現代・古語マッピング，年代別の俳句型生成，俳句生成の 3 つのステッ

プを組み合わせた手法を提案する．

図 1 にモデルの概要図を示す．入力には俳句・年代・古語単語の比率を設ける．入力年代を元に俳句の方を生成し，入力俳句を元に言い換える古語単語の候補をリストアップする．最後に型と単語候補を用いて俳句を生成する．

3.1 現代・古語マッピング

現代・古語マッピングでは，現代語ベクトル空間と古語ベクトル空間を独立に学習し，現代単語・古典単語ペアデータを元に線形写像を行うことによって 2 つのベクトル空間を同一空間上にプロットする．マッピング手法は複数言語間における単語埋め込みにおいて最高精度を達成している MUSE[7] を用いる．MUSE により学習した現代語ベクトル・古語ベクトルを用いて，入力俳句から抽出された現代単語のベクトルと古語のベクトルの類似度を計算し，類似度が高い上位 30 位までの古典単語を俳句生成の単語候補としてリストアップする．

3.2 年代別俳句型生成

年代別俳句型生成では，入力値である年代情報を用いて年代の特徴を反映させた型を生成する．本研究では古典俳句型と入力俳句型の 2 つの手法を提案する．

3.2.1 古典俳句型

古典俳句型は収集した年代情報付き古典俳句データを用いて型を生成する．

年代情報付きの古典俳句データの各俳句に対して形態素解析器 mecab と古典文学用システム辞書中和古文 Unidic を用いて形態素解析を行う．これにより得られた形態素から内容語（名詞・動詞・形容詞）を抜き出し「品詞+単語長」の形に置き換え，年代タグ付き古典俳句型を生成する．また，型の選定は入力値である年代（本研究では俳句が大成した 1644 年〜1902 年までを入力値として設けることとする）を用いて行う．入力年の前後 30 年のタグが付いたものを選定対象の型候補とする．

3.2.2 入力俳句型

入力俳句型生成は入力値である現代俳句を用いて型を生成する．

入力俳句に対して古典俳句型と同様に形態素解析により得られた形態素から内容語を抜き出し「品詞+単語長」の形に置き換え，入力俳句型を生成する．

これにより生成された型に入力値の年代を元に切れ字の挿入を行う．挿入する切れ字は連歌・俳諧で秘伝とされた 18 字（かな・もがな・し・じ・や・らん・か・けり・よ・ぞ・つ・せ・ず・れ・ぬ・へ・け・に）を対象とする．年代のセグメントはそれぞれ代表的な俳人（松尾芭蕉・小林一茶・正岡子規）と特徴的な排風（蕉風・一茶調・近代俳句）が存在した江戸初期 (1644 年〜1783 年)，江戸後期 (1783 年〜1867 年)，明治初期 (1867 年〜1902 年) とする．古典俳句データからそれぞれの年代の切れ字の出現確率を算出し，入力年に該当する年代の出現確率より挿入する切れ字を決定する．この操作により得られた型を入力俳句型とする．

表 1: 切れ字出現比率

	江戸前期		江戸後期		明治初期	
切れ字	出現数	出現率	出現数	出現率	出現数	出現率
じ	42	3%	76	2%	46	1%
らむ	355	**23%**	43	1%	65	2%
けり	342	22%	1862	**60%**	2169	55%
つ	45	3%	51	2%	92	2%
ず	221	14%	173	6%	508	13%
ぬ	552	**35%**	884	29%	1091	27%
や	1720	67%	6454	**74%**	7837	73%
かな	227	9%	461	5%	2169	**20%**
よ	75	3%	556	**6%**	142	1%
ぞ	240	9%	754	9%	185	2%
もがな	8	0%	2	0%	0	0%
か	294	**11%**	458	5%	472	4%
し	683	52%	1531	51%	1755	**64%**
いかに	37	**3%**	16	1%	6	0%
せ	197	**15%**	350	12%	199	7%
れ	210	16%	445	15%	366	13%
へ	163	12%	492	**17%**	390	14%
け	24	2%	103	3%	47	2%

3.3 俳句生成

俳句生成では，マッピングで得られた単語候補と年代別の俳句型生成で得られた型を用いて俳句の生成を行う．本研究ではランダム生成と言語モデルベース生成の 2 つの生成手法を提案する．

3.3.1 ランダム生成

ランダム生成では型の空欄箇所にマッピングにより得られた単語候補から品詞と長さが一致するものをランダムに選定し俳句を生成する．生成された俳句に対して動詞の活用を前後の単語に応じて修正し，俳句候補として出力する．

3.3.2 言語モデルベース生成

言語モデルベース生成では古典俳句データで学習した言語モデルを用いて生成を行う．言語モデルで次の単語の出現確率を予測し，マッピングにより得られた単語候補から次の単語をサンプリングする．その際，ランダム生成同様に型の空欄箇所の品詞と長さが一致するよう単語を選定する．

3.4 スコアリング

peplexity とコサイン類似度の平均，2 つの計算式を用いてスコアリングを行う．俳句らしさを測る指標として俳句データで学習した言語モデルで peplexity を計算する．出力俳句の単語系列を $o = (o_1, o_2, \cdots, o_N)$，$o_i$ より前に出現した単語を $o_{\setminus i}$ とすると，perplexity

表 2: データセット

	データ量	ジャンル
現代俳句	107,403 句	俳句
古典俳句	64,239 句	俳句
現代文	800 万文	wikipedia
古典文	135,900 文	物語・歌集
現代・古典ペア (単語)	30,148 単語	古語辞書
現代・古典ペア (文)	15,740 文	訳付き物語

は次のように定義できる：

$$pp(o) = -\frac{1}{N}\sum_{i=1}^{N}\log_2 P(o_i|o_{\setminus i}).$$

また，出力俳句と入力俳句の類似度を測る指標としてコサイン類似度の平均を計算する．入力俳句の中で古典単語に言い換えた単語の数を n 個とする．入力俳句の単語のベクトル系列を $M = (\vec{m_1}, \vec{m_2}, \cdots, \vec{m_n})$，出力俳句の単語のベクトル系列を $O = (\vec{o_1}, \vec{o_2}, \cdots, \vec{o_n})$ とした時，n 個の単語対の余弦類似度の平均は次のように計算できる：

$$ave_{cos}(M, O) = \frac{1}{n}\sum_{i=1}^{n} cos(\vec{m_i}, \vec{o_i}).$$

本研究では以上 2 つの計算式を用いて出力俳句のスコアを計算する：

$$score(o) = \frac{a}{pp(o)} + (1-a)ave_{cos}(M, O).$$

4 データセット

本研究では実験データとして俳句データ，古典文学データ・現代文データ，古典単語・現代単語ペアデータの収集を行った．収集したデータの概要を表 2 に示す．

4.1 俳句データ

俳句データでは web 上から現代俳句と古典俳句データのスクレイピングを行った．現代俳句 107,430 句，古典俳句 64,239 句　（年代情報付き古典俳句データ 35,862 句を含む）をそれぞれ収集した．

4.2 古典文学データ・現代文データ

古典文学データは web 上に公開されている古典文学（物語・歌集）をスクレイピングすることで収集した．源氏物語や竹取物語など 30 作品から計 135,900 文（現代語訳付きデータ 15,740 文含む）を収集した．現代文データには wikipedia データ 800 万文を用いた．

4.3 古典単語・現代単語ペアデータ

古典単語・現代単語ペアデータは web 上に公開されている古典単語辞書と現代語訳付きの古典文学データを用いて収集した．現代語訳付き古典文学データからは古典文学と現代語訳に同じ単語が含まれている場合，昔と現在で表現が同じ単語とみなし，ペアとして収集した．これにより合計 30,148 単語ペアを収集した．

表 3: ベクトル空間の学習パラメータ

	現代語	古語
データ名	日本語 wikipedia	古典文学+古典俳句
データサイズ	8,000K	200K
次元数	300	300
window size	2	2
学習率	0.1	0.1
negative sampling	5	5

5 実験

5.1 ベクトル空間学習

ベクトル空間学習では，現代語ベクトル・古語ベクトルの 2 つのベクトル空間の学習を行った．学習時のパラメータを表 3 に示す．学習には機械学習ライブラリ fasttext を用いた．

5.2 評価方法

評価は言い換えた俳句について，出力俳句について，入力情報について，人の俳句と提案手法の俳句についての 4 つの設問の評価を人手により行った．評価項目は型 (切れ・季語・字数) に忠実であるか．文法は正しいか，入力に類似しているか，意味が通じるかの各項目に対して 5 段階評価を設けた．評価者は国語教師や文学部文学科の学生を始めとした成人日本人 20 名で，一人あたり 100 句を評価した．ベースラインとして入力俳句に対してペアデータに存在する現代単語を古典単語に入れ替え生成する手法を設けた．また，出力俳句の評価の際に比較対象としている LSTM 生成では米田らの手法の実験設定で俳句生成を行った．

5.3 評価結果

俳句・古典単語率・年代を入力とし，言い換えた俳句を出力する．提案手法の出力結果を表 4 に示す．また，俳句言い換えの実験結果を表 5 に示す．いずれの評価項目においても入力俳句型の言語モデルベース生成が最も評価が高かった．入力の型を応用した型の生成を行うため類似性の項目が高い評価となっており，言語モデルベースの生成のため自然な単語の並びの句が生成され文法の項目が高い評価になっていると考えられる．ベースラインの句は現代単語と古典単語のシンプルな変換のため，類似性は平均値以上の評価となっているが，型に関して最低の評価になっており，単語の文字数のズレなどが起因していると考えられる．

次に提案手法の出力俳句と LSTM により生成された俳句の評価結果を表 6 に示す．こちらも入力俳句型の言語モデルベースの生成が最高評価であった．

表 4: 出力俳句例

入力			出力
俳句	古典単語率	年代	俳句
母送る　天国の文　春の雪	0.2	江戸前期	母かわす　桜の雪や　天のふみ
荒鋤の　田へ舞ひ降りぬ　山桜	0.4	江戸後期	天龍や　鋤に舞い降る　山桜
松の幹　みな傾きて　九月かな	0.6	明治初期	曼珠沙華　九月の幹を　つなぎけり
昼空に　月あり桃の　節句なり	0.8	明治初期	冴夜や　花曇り増す　虚空なり

表 5: 言い換えた俳句の評価結果

	型	文法	類似性	意味	平均
baseline	2.45	3.17	3.60	3.11	3.08
classic-random	3.54	3.12	1.88	1.98	2.63
classic-LM	3.86	3.82	2.82	2.50	3.25
input-random	4.30	4.08	3.33	3.10	3.71
input-LM	**4.44**	**4.26**	**4.18**	**3.76**	**4.16**

表 6: 出力俳句の評価結果

	型	文法	意味	平均
LSTM 生成	4.02	3.55	2.88	3.48
classic-random	3.80	3.91	3.06	3.59
classic-LM	4.43	4.38	3.87	4.23
input-random	4.06	4.03	3.41	3.84
input-LM	**4.45**	**4.39**	**3.89**	**4.25**

さらに，入力情報である古典単語率と入力年代の質問の正答率は古典単語率が 20%，年代情報が 47%という結果であった．評価者によって古典単語の定義が異なったことが古典単語率の正答率の低さに起因していると考えられる．

最後に，人が作った俳句か提案手法の俳句かどちらかを見分ける質問の回答結果を表 7 に示す．提案手法の俳句を人の俳句と答えた割合が 45%，提案手法の俳句と答えた割合が 40%という結果になった．この結果から提案手法により人の俳句と見分けがつかない俳句を出力することができたと言える．

表 7: 人と提案手法の俳句を見分ける質問の結果

		回答		
		人	提案手法	不明
正解	人	40%	43%	17%
	提案手法	45%	42%	13%

6　おわりに

本研究では，古典情報を用いた俳句の言い換え手法を提案した．実験では，出力俳句の型は忠実か，文法が正しいか，入力俳句に類似しているか，意味が通じるかの項目において提案手法である入力俳句型の言語モデルベース生成が最も高い評価であることを確認した．今後の課題としては,「1 句 1 章」など更に高度な俳句の型の組み込みや「コトとモノの区別」など発展的な俳句の技法を取り入れたモデルの作成などが考えられる．

参考文献

[1] Martin Sundermeyer, Ralf Schluter, and Hermann Ney. LSTM Neural Networks for Language Modeling. In *ISCA*, 2012.

[2] Lantao Yu, Weinan Zhang, Jun Wang, and yong Yu. SeqGAN:Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*, 2017.

[3] 米田航紀, 横山想一郎, 山下倫央, 河村秀憲. LSTM を用いた俳句自動生成器の開発. 人工知能学会, 2018.

[4] 廣田敦士, 岡夏樹, 荒木雅弘, 田中一晶. 学習データセットを分けた seqGAN による俳句生成. 言語処理学会, 2018.

[5] Zhe Wang, Wei He, Hua Wu, Haiyang Wu, Wei Li, Haifeng Wang, and Enhong Chen. Chinese Poetry Generation with Planning based Neural Network. In *COLING*, 2016.

[6] Zhicao Yang, Pengshan Cai, Yansong Feng, Weijiang Feng, Suet-Ying Elena Chine, and Hong Yu. Generating Classical Chinese Poems from Vernacular Chinese. In *EMNLP*, 2019.

[7] Alexis Conneau, Guilaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herve Jegou. Word translation without parallel data. In *ICLR*, 2018.