Vol.2016-NL-228 No.11
2016/9/30

# Encoder-Decoder モデルを用いた 日本語崩れ表記の正規化

池田 大志[1,a)]　進藤 裕之[1]　松本 裕治[1]

**概要**：近年，ソーシャルメディアの普及に伴い，ウェブ上には人々が投稿した膨大な量のテキストデータが存在するようになった．それに伴い，評価極性分析や話題抽出などの需要が高まり，ウェブ上のテキストデータは重要な言語資源であると考えられている．しかし，新聞記事データを用いて訓練した形態素解析器では，口語表現や表記揺れなどを多く含むウェブ上のテキストに対して，形態素解析誤りを起こすという問題がある．そのため，崩れ表記を含む文を正規化することで，形態素解析誤りを削減することができると考えられる．本研究では，ニューラルネットワークに基づく Encoder-decoder モデルを用いた日本語崩れ表記の正規化手法を提案する．また，擬似崩れ表記データを用いて正規化に適用した実験結果についても報告する．

## 1. はじめに

近年，マイクロブログや SNS が広く利用されるようになり，ウェブ上には人々が投稿した膨大な量のテキストデータが存在するようになった．それに伴い，評価極性分析や話題抽出などの需要が高まり，ウェブ上のテキストデータは重要な言語資源であると考えられている．ウェブ上のテキストは，表記揺れ（ひらがな化，カタカナ化）や口語表現・方言などの崩れ表記が多く含まれており [2]，新聞記事データを用いて訓練した形態素解析器では，崩れ表記を含むウェブ上のテキストに対して，形態素解析誤りを起こす問題がある．

例えば，次のような崩れ表記文を MeCab [3] で形態素解析を行うと，以下のような解析結果となる．

**入力文:　博多最高で〜す！らーめんおいしっ**
**解析結果:　博多 (名詞) 最高 (名詞) で (助詞) 〜(記号) す (名詞) ！ (記号) ら (名詞) ー (名詞) めん (名詞) おい (感動詞) しっ (動詞)**

この例文では，「です」という正規表記が「で〜す」と長音記号が挿入され，単語分割と品詞タグ付けに失敗している．次に「ラーメン」というカタカナ表記がひらがな表記の「らーめん」と書かれており，単語分割に失敗している．これはひらがな表記の「らーめん」が形態素解析器の辞書に存在していないため，このような解析誤りを起こしたと考えられる．例文の崩れ表記を全て取り除いた場合，正規表記文は「博多最高です！ラーメンおいしい」であると考えられる．この正規表記文を MeCab で形態素解析を行うと，以下のような解析結果となる．

**入力文:　博多最高です！ラーメンおいしい**
**解析結果:　博多 (名詞) 最高 (名詞) です (助動詞) ！ (記号) ラーメン (名詞) おいしい (形容詞)**

このように，崩れ表記文を正規化することで形態素解析しやすい文に変換し，結果として正しく単語分割と品詞タグ付けを行うことが可能である．そこで，崩れ表記文を既存の形態素解析器が想定している表記に変換することで，形態素解析誤りを削減することができると考えられる．

本研究では，崩れ表記文を正規化するため，ニューラルネットワークに基づく Encoder-decoder モデルを用いた日本語崩れ表記の正規化手法を提案する．Encoder-decoder モデルは，機械翻訳において提案されて以来，屈折語の生成や誤り訂正などの様々なタスクで高い精度を上げている [1], [4], [5], [6], [7], [14]．しかし，言語資源が少ない言語対では，翻訳精度が低下するとの報告されており [7], Encoder-decoder モデルを学習するためには，大規模なデータが必要となる．また日本語の崩れ表記に対して正規表記が付与されている大規模なコーパスはほとんど存在しない．そこで人手で記述したルールに基づき擬似的な崩れ表記文を生成することで，この問題に対処する．また Encoder-decoder モデルに Attention 機構を適用した手法

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] taishi.ikeda.io7@is.naist.jp

を日本語崩れ表記の正規化に適用した実験結果についても報告する．

## 2. 関連研究

本節では，主に日本語を対象とした正規化の研究について説明する．また機械翻訳を中心とした Encoder-decoder モデルの関連研究についても本節で述べる．

### 2.1 正規化手法

本稿では，新聞記事や形態素解析器の辞書に存在する表記を**正規表記**，口語表現や表記揺れなど正規表記の異表記として書かれた表現を**崩れ表記**，崩れ表記を含む文を**崩れ表記文**，崩れ表記を含まない文を**正規表記文**，崩れ表記を正規表記に変換する処理を**正規化**とする．

正規化と形態素解析を同時に行う手法として Saito ら [9], Kaji ら [10], Sasano ら [11] の研究が存在する．Saito らは，人手でブログや Twitter から崩れ表記を抽出し，抽出した崩れ表記に対して人手で正規表記を付与している．これらのペアから文字列アライメントの推定に基づき文字列レベルの崩れ表記パターンを統計的に抽出し，得られた崩れパターンを形態素ラティスに組み込むことで，正規化と形態素解析を同時に行う手法を提案している．Kaji らも同様に, 正規化と形態素解析を同時に行う手法を提案している．Kaji らは既存の辞書エントリー中の表出形を人手で記述したルールにより崩れ表記に自動変換することによって，正規化辞書を作成している．彼らは，Sasano ら [11] と同様のルールに基づき，長母音から長音記号への変換や任意個の母音または長音記号を挿入することで，正規化辞書のエントリーを自動的に生成している．この正規化辞書を用いて，正規化と形態素解析を同時に行う手法を提案している．

正規化のみに着目した研究は，Ikeda ら [13]， 佐々木ら [12] の研究が存在する．Ikeda らは，ブログと新聞記事を用いて，形態素解析を行い，未知語となった箇所に注目し，自動的に正規化ルールを構築する手法を提案している．少量の汎用な正規化ルールをもとに，正規化ルールを適用する前後の文の形態素解析の結果を比較することで，具体的な正規化ルールをコーパスから学習し，自動的に多数のルールを生成している．佐々木ら [12] は，人手で抽出した崩れ表記に正規表記を付与し，ペアデータから一対一の文字アライメントを抽出している．そして「残す」「削除」「（任意の文字に）置換」をラベルとし，条件付き確率場（Conditional Random Fields, CRF）を用いた系列ラベリングの問題として，文字単位の正規化を行っている．また，Chrupaa ら [22] も同様に CRF を用いて，英語の正規化を行っている．Chrupaa らは大規模な Twitter コーパスから再帰的ニューラルネットワーク（Recurrnet Neural Network, RNN）を用いて，文字単位の分散表現を学習し，素性として CRF に組み込む手法を提案している．

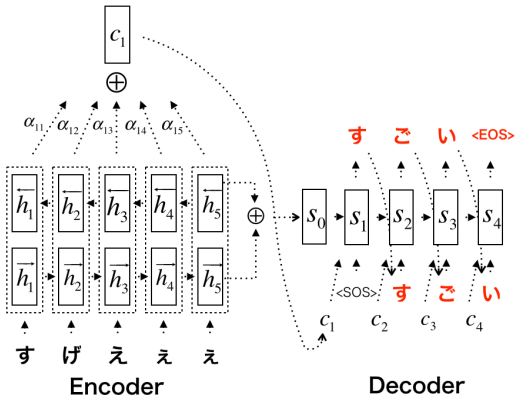


**図 1** 本研究でのモデル

### 2.2 Encoder-decoder モデル

本研究では，ニューラルネットワークに基づく Encoder-decoder モデルを用いた正規化の手法を提案する．そこで機械翻訳を中心とした Encoder-decoder モデルに関連する研究についても述べる．現在，機械翻訳の分野では Encoder-decoder モデルが非常に高い翻訳精度を実現している [1], [4], [5], [6]．具体的には，ニューラルネットワークを用いて入力文を特徴ベクトルに変換し，このベクトルをもとに出力側のニューラルネットワークで出力文を生成している．この 2 つのニューラルネットワークを組み合わせたものを Encoder-decoder モデルと呼ぶ．しかし，通常の Encoder-decoder モデルでは，長い文における翻訳の性能が低下することが報告されており，この問題を解決するために，Attention 機構に基づくモデルが提案されている [1], [4]．このモデルでは，出力文を生成する際に，入力文のどの単語に注目して単語を生成するかを動的に決定することで，文が長い場合における問題を解決している．また単語単位でなく，文字を単位とした Enocder-decoder モデルを設計することで，屈折語の生成や誤り訂正に応用した研究も存在する [7], [14].

Encoder-decoder モデルは，様々なタスクで高い精度を上げているが，モデルを学習するためには大規模なデータセットが必要である．言語資源が少ない言語対では，翻訳精度が低下するとの報告されており，モデルの学習には，ある程度データ量が存在する言語対で事前学習を行うなどの工夫が必要となる [7].

## 3. 提案手法

本節では，本研究で用いる Encoder-decoder モデルのアーキテクチャについて説明する．

### 3.1 Encoder-decoder モデル

2 節で述べた通り，機械翻訳の分野では，Attention 機構を用いた Encoder-decoder モデルが高い精度を実現してい

る [1], [4], [14]．そこで本研究では，Bahdanau ら [1] が提案した Attention 機構を利用して，日本語の崩れ表記の正規化を行う．図 1 に本研究モデルの模式図を示す．

提案手法では，文字系列を入出力とする．まず，入力文（崩れ表記文）を $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$ とする．ここで，$V$ 次元のベクトル $x_i \in \mathbb{R}^V$ は入力文字に対応する One-hot ベクトルである．$N$ は入力文の文字数，$V$ は学習データの総文字数を表す．また，出力文（正規表記文）を $\boldsymbol{y} = (y_1, y_2, \ldots, y_M)$ とする．ここで，$V$ 次元のベクトル $y_i \in \mathbb{R}^V$ は出力文字に対応する One-hot ベクトルである．$M$ は出力文の文字数である．

ここで，正規化のタスクは，崩れ表記文 $\boldsymbol{x}$ が与えられたとき，正規表記文 $\boldsymbol{y}$ を求める問題と定式化できる．また，式 1 の条件付き確率 $P(\boldsymbol{y}|\boldsymbol{x};\theta)$ を最大化する最適なパラメータ $\theta$ を崩れ表記文と正規表記文のペアデータから学習する．

$$P(\boldsymbol{y}|\boldsymbol{x};\theta) = \prod_{t=1}^{M} p(y_t|y_1, y_2, \ldots, y_{t-1}, \boldsymbol{x}) \tag{1}$$

$$y_t = \text{Decoder}(y_1, y_2, \ldots, y_{t-1}, C) \tag{2}$$

$$C = \text{Encoder}(\boldsymbol{x}) \tag{3}$$

ここで，Enocder は入力文 $\boldsymbol{x}$ を特徴ベクトルに変換する．次に，入力文を変換した特徴ベクトルと過去の出力をもとに，Decoder で出力の文字 $y_t$ を生成する．以下でそれぞれの詳細について述べる．

### 3.2 Encoder

Encoder は，RNN を用いて入力文を実数値の特徴ベクトル $h_i \in \mathbb{R}^l$ に変換する役割を果たす．本研究では，Bahdanau ら [1] が Encoder として採用している両方向 RNN（Bidirectional RNN）を用いる．両方向 RNN は，両方向から RNN を学習する方法である．まず入力文を順方向（文頭から文末）から文字列を読み取っていき $\overrightarrow{h}_i$ を計算する．次に逆方向（文末から文頭）から文字列を読み取っていき $\overleftarrow{h}_i$ を計算する．両方向各時刻のベクトル $[\overrightarrow{h}_1, \overrightarrow{h}_2, \ldots, \overrightarrow{h}_N]$, $[\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_N]$ を以下のように計算する．

$$\overrightarrow{h}_t = f(x_t, \overrightarrow{h}_{t-1}) \tag{4}$$

$$\overleftarrow{h}_t = f(x_t, \overleftarrow{h}_{t+1}) \tag{5}$$

RNN の計算は，各時刻 $t$ において，入力系列 $\boldsymbol{x}$ から $x_t$ と $t-1$ 時刻の $h_{t-1}$ を受け取り，$h_t$ を返すような計算を行う．ここで $f$ は任意の非線形関数である．本研究では，$f$ として Gated Recurrent Unit(GRU)[5] を用いる．GRU は，Long short-term memory(LSTM)[8] と同様に，系列の長期的な記憶の保持を目的とした RNN ユニットの 1 種である．GRU には，更新ゲート（Update gate）とリセットゲート（Reset gate）の 2 種類のゲートが存在し，それぞれのゲートが過去の履歴を制御する．$z$ は更新ゲートで，現時点での隠れ層 $h_t$ を更新するか否かを調整する．$r$ はリセットゲートで，1 つ前の隠れ層 $h_{t-1}$ を忘却するか否かを調整する．具体的には，GRU は以下のように計算される．

$$\overrightarrow{h}_t = \begin{cases} (1 - \overrightarrow{z}_t) \circ \overrightarrow{h}_{t-1} + \overrightarrow{z}_t \circ \overrightarrow{h}_{t'} & (\text{if } t > 0) \\ 0 & (\text{if } t = 0) \end{cases} \tag{6}$$

$$\overrightarrow{h}_{t'} = \tanh(\overrightarrow{W} E x_t + \overrightarrow{U}[\overrightarrow{r}_t \circ \overrightarrow{h}_{t-1}] + \overrightarrow{b}) \tag{7}$$

$$\overrightarrow{z}_t = \sigma(\overrightarrow{W}_z E x_t + \overrightarrow{U}_z \overrightarrow{h}_{t-1} + \overrightarrow{b}_z) \tag{8}$$

$$\overrightarrow{r}_t = \sigma(\overrightarrow{W}_r E x_t + \overrightarrow{U}_r \overrightarrow{h}_{t-1} + \overrightarrow{b}_r) \tag{9}$$

$\circ$ は要素ごとの積，$\sigma$ はシグモイド関数，$e$ は文字埋め込みベクトルの次元数，$l$ は隠れ層の次元数を表す．文字埋め込みベクトルは，$E \in \mathbb{R}^{e \times V}$ とする．ここで，$\overrightarrow{W}, \overrightarrow{W}_z, \overrightarrow{W}_r \in \mathbb{R}^{l \times e}$，$\overrightarrow{U}, \overrightarrow{U}_z, \overrightarrow{U}_r \in \mathbb{R}^{l \times l}$ は順方向 RNN の重み行列，$\overrightarrow{b}, \overrightarrow{b}_z, \overrightarrow{b}_r \in \mathbb{R}^l$ はバイアスベクトルである．また，逆方向も同様に重み行列，バイアスベクトルを定義する．さらに，各時刻 $t$ で順方向と逆方向の隠れ層のベクトルを連結（Concat）し，$h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$ とする．最終的に，両方向 RNN でエンコードした入力文は $C = [h_1, h_2, \ldots, h_N]$ となる．

### 3.3 Decoder

Decoder では，Encoder でエンコードした実数値の特徴ベクトルをもとに出力文を文字単位で生成する．Decoder の非線形関数 f として，Encoder と同様に GRU を用いる．さらに Bahdanau ら [1] にならって，Attention 機構を Decoder に組み込む．彼らは，各時刻 $t$ ごとに両方向 RNN でエンコードした $C$ と Decoder の時刻 $t-1$ の隠れ層 $s_{t-1}$ を用いて，文脈ベクトル $c_t$ を計算し，この $c_t$ を入力文の情報として Decoder に組み込む手法を提案している．文脈ベクトル $c_t$ は以下のように計算される．

$$c_t = \sum_{j=1}^{N} \alpha_{ij} h_j \tag{10}$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{N} \exp(e_{ik})} \tag{11}$$

$$e_{ij} = v_a^t \tanh(W_a s_{i-1} + U_a h_j + b_a) \tag{12}$$

ここで，$W_a \in \mathbb{R}^{l \times l}$，$U_a \in \mathbb{R}^{l \times 2l}$ は重み行列，$v_a \in \mathbb{R}^l$ は重みベクトル，$b \in \mathbb{R}^l$ はバイアスベクトルである．次に，Decoder の隠れ層 $s_t$ を以下のように計算する．

$$s_t = \begin{cases} (1 - z_t) \circ s_{t-1} + z_t \circ s'_t & (\text{if } t > 0) \\ (\overrightarrow{h}_N + \overleftarrow{h}_N)/2 & (\text{if } t = 0) \end{cases} \tag{13}$$

$$s'_t = \tanh(W E y_t + U[r_t \circ s_{t-1}] + C c_t + b) \tag{14}$$

$$z_t = \sigma(W_z E y_t + U_z s_{t-1} + C_z c_t + b_z) \tag{15}$$

表 1　崩れ表記化の例

| ルール | 品詞 | 正規表記 | 崩れ表記 |
|---|---|---|---|
| (3), (4) | 終助詞 | な | なぁ，な〜 |
| (2), (3), (4) | 助動詞 | たい | たーい，ええ |
| (1), (2), (3), (4), (5) | 形容詞 | かわいい | かわいいい, かわいぃ |
| (1), (2), (3), (4), (5) | 感動詞 | ありがとう | ありがとー，ありがとぅ |

表 2　正規化結果の精度比較

| 手法 | CER(%) |
|---|---|
| No-op | 17.15 |
| Rule-based | 15.07 |
| CRF (マイクロブログ) | 14.13 |
| EncDec (マイクロブログ) | 123.1 |
| EncDec (擬似崩れ表記データ) | **13.97** |

$$r_t = \sigma(W_r E y_t + U_r s_{t-1} + C_r c_t + b_r) \quad (16)$$

ここで，$E$ は Encoder と同様の文字埋め込みベクトルを用いる．また，$W, W_z, W_r \in \mathbb{R}^{l\times e}$，$U, U_z, U_r \in \mathbb{R}^{l\times l}$，$C, C_z, C_r \in \mathbb{R}^{l\times l}$ は Decoder の重み行列，$b, b_z, b_r \in \mathbb{R}^l$ はバイアスベクトルである．Decoder では，Encoder とは異なり，式 14, 15, 16 に文脈ベクトル $c_t$ が導入されている．この文脈ベクトル $c_t$ が，出力文を生成する際に，入力文のどの単語に注目して単語を生成するか動的に決定している．Decoder の最終的な出力として，$V$ 次元のベクトル $\tilde{t_i}$ を求める．

$$\tilde{t_t} = \mathrm{softmax}(W_s s_t + b_s) \quad (17)$$

ここで，$W_s \in \mathbb{R}^{V\times l}$ は重み行列，$b_s \in \mathbb{R}^V$ はバイアスベクトルである．この $\tilde{t_i}$ の確率が最大となった文字を時刻 $t$ の出力とする．そして，ビーム探索を用いて，$P(\boldsymbol{y}|\boldsymbol{x};\theta)$ を最大とする出力系列 $\boldsymbol{y}$ を求める．

### 3.4　モデルの学習

式 1 の $\theta$ を最適化するような誤差関数を設定する．この誤差関数を最小化することによって，最適化なパラメータ集合 $\theta$ を求める．

$$E(\theta) = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \log P(\boldsymbol{y}|\boldsymbol{x};\theta) \quad (18)$$

ここで，$D$ は学習データ，$\boldsymbol{x}$ は入力の文字列，$\boldsymbol{y}$ は出力の文字列を表す．最適化を行うパラメータ $\theta$ は，$\theta = [E, \overrightarrow{W}, \overrightarrow{W}_z, \overrightarrow{W}_r, \overrightarrow{U}, \overrightarrow{U}_z, \overrightarrow{U}_r, \overrightarrow{b}, \overrightarrow{b}_z, \overrightarrow{b}_r, W, W_z, W_r, U, U_z, U_r, C, C_z, C_r, b, b_z, b_r, W_a, U_a, b_a, v_a, W_s, b_s]$ となる．ただし，逆方向 RNN のパラメータは省いている．学習の詳細については 4.1.3 節で述べる．

## 4.　実験

本研究では，マイクロブログ文を対象とした正規化の実験を行うことで，提案手法の効果を検証した．

### 4.1　比較手法

提案手法と 2 つのベースラインの手法について説明する．

#### 4.1.1　Rule-based

ルールベースの手法として，Sasano ら [11] が形態素ラティスを編集するルールを用いて，ルールベースの正規化を行う．具体的には，長音記号・小書き文字による置換，長音記号・小書き文字の挿入を正規化ルールとして用いる．

#### 4.1.2　CRF

機械学習の手法として，CRF を用いた正規化を行う．佐々木ら [12] は，文字単位の系列ラベリングの問題として正規化を行った．彼らの研究で最も精度の良い結果の素性を採用した．素性として，入力文字とその周囲 3 文字，入力文字が母音であるか否かを用いた．ラベルは「残す」「削除」「(任意の文字に) 置換」の三種類を設定し，入力文字に対してラベルを推定後，その文字に対してラベルの操作を実行した．CRF の実装は CRFsuite [19] を利用した．

#### 4.1.3　EncDec

ここでは, 提案手法の詳細について述べる．パラメータの最適化は，Adam[20] を用いて自動調整した．ミニバッチサイズは 32，文字埋め込みベクトルの次元数は 150，GRU の隠れ層の次元数は 300 とした．モデルの実装は theano[21] を利用した．ビーム幅は 100 として，ビーム探索を行った．正規化を行う際に，入力とする文字列は「ひらがな」「カタカナ」「漢字」とする．正規表現を用いて，「ひらがな」「カタカナ」「漢字」を抽出し，正規化を行い，それ以外の文字列は正規化を行わずそのまま出力とした．

### 4.2　データセット

本研究では，データセットとして，マイクロブログコーパスと擬似崩れ表記データを利用した．マイクロブログコーパスは，日本語の Twitter データである．しかし，少量のマイクロブログコーパスでは，Encoder-decoder モデルを学習することが難しいと考えらるため，Kaji ら [10] にならって，擬似的に生成した大量の崩れ表記データを作り，Encoder-decoder モデルにおける正規化の実験を行った．彼らは，辞書エントリー中の表出形を人手で記述したルールにより崩れ表記に変換し，この正規化辞書を用いることで，正規化と形態素解析の精度向上を実現している．

#### 4.2.1　マイクロブログコーパス

マイクロブログコーパスとして，Kaji ら [10] が提供しているデータセットを利用する．彼らは Twitter から 171,386 の日本語ツイートを収集し，その中から無作為に選んだ 1000 ツイートに対して形態素情報と正規化情報のアノテーションを行い，マイクロブログコーパスを作成した．本研究では，マイクロブログコーパスから崩れ表記を少なくとも 1 つを含む 506 文を学習・テストデータとして利用する．また前処理として，事前に連続する文字は，「ー」「〜」「っ」は 1 文字，それ以外の文字は 3 文字に削減している．

表 3　Encoder-decoder を用いた正規化結果の比較

| 手法 | 正規化結果 |
|---|---|
| 入力文 | ラルクって海外でも人気すげーな☆ |
| 正解文 | ラルクって海外でも人気すごいな☆ |
| EncDec (マイクロブログ) | 一近と一一中ちゃんとうらみたとう☆ |
| EncDec (擬似崩れ表記データ) | ラルクって海外でも人気すごいな☆ |

表 4　各手法の正規化結果の比較

| 手法 | 正規化結果 |
|---|---|
| 入力文 1 | ラルクって海外でも人気すげーな☆ |
| 正解文 1 | ラルクって海外でも人気すごいな☆ |
| Rule | ラルクって海外でも人気すげな☆ |
| CRF | ラルクって海外でも人気すげーな☆ |
| EncDec | ラルクって海外でも人気すごいな☆ |
| 入力文 2 | 久々にＴｗｉｔｔｅｒ浮上したなぁ。 |
| 正解文 2 | 久々にＴｗｉｔｔｅｒ浮上したな。 |
| Rule | 久々にＴｗｉｔｔｅｒ浮上したな。 |
| CRF | 久々にＴｗｉｔｔｅｒ浮上したな。 |
| EncDec | 久久々にＴｗｉｔｔｅｒ浮上したな。 |
| 入力文 3 | 車離れが捗りますなーw |
| 正解文 3 | 車離れが捗りますなw |
| Rule | 車離れが捗りますなw |
| CRF | 車離れが捗りますなw |
| EncDec | 車離れが暇りますなw |

#### 4.2.2　擬似崩れ表記データ

擬似崩れ表記データを作成するため，現代日本語書き言葉均衡コーパス（BCCWJ） [23] の新聞データ（PN）の形態素を正規表記として，人手で記述したルールに基づき正規表記を自動的に擬似的な崩れ表記に変換する．この方法によって正規表記と崩れ表記のペアデータを作成する．ルールの作成には，Kaji ら [10] の研究を参考に行った．具体的には，以下のルールを用いる．

(1) 長母音から長音記号への変換（例：どうする → どーする）
(2) 頻出する語尾への変換 (例：すごい → すげえ，すげっ)
(3) 母音・促音の挿入（例：美味しい → 美味しいいい，美味しいっ）
(4) 長音記号の挿入（例：です → ですー，で〜す）
(5) 小書き文字化（例： い → ぃ）

このルールを新聞データの形態素に対して適用し，崩れ表記への変換を行った．その結果，正規表記文と崩れ表記文のペアデータ 213,618 文を作成することができた（ルールが適用されていない文も含まれている）．表 1 に崩れ表記化の例を示す．

### 4.3　評価手法

評価尺度として Character Error Rate(CER) を用いる．CER は，削除，挿入，置換，それぞれの操作コストを 1 として，正規化後の文と正解文（正規表記文）の編集距離を計算し，正解文の長さで割ったものである．この評価尺度は，正規化後の文が正解文にどの程度近づくかを示す指標である．また CER が小さい方が良い結果となる．

$$CER = \sum_{(x,y)\in T} \frac{edit\_distance(x,y)}{length(y)} \times 100 \qquad (19)$$

$x$ は正規化後の文, $y$ は正解文を表す．$T$ はテストデータの集合である．

ここで，手法ごとのデータセットの利用方法について説明する．Rule-based の場合，学習データは一切使用していない．CRF と EncDec（マイクロブログ）では，学習にマイクロブログコーパスを用いた．これらを用いた実験では，5 分割交差検定を行い 1 つをテストデータ，1 つを開発データ，残りを学習データとして利用した．EncDec（擬似崩れ表記データ）では，擬似崩れ表記データのみを学習に使用した．擬似崩れ表記データからランダムに 150 文を抽出し，開発データとして利用した．エポック数は 50 に設定し，開発データの CER が最も低いエポックで評価した結果について報告する．

## 5.　実験結果

本節では，実験結果について述べる．表 2 に各手法による CER を示す．No-op は，正規化前の入力文と正解文の CER を示す．これは正規化前の 1 文の中に 17.15 %（約 2 文字）の崩れ表記が含まれていることを示す．まず，Rule-based では，No-op と比較して，2.08 %のエラーを削減できている．次に，CRF では，No-op と比較して，3.02 %のエラー削減できており，単純なルールよりも良い結果を得ている．ここで，CRF と同様にマイクロブログコーパスのみを用いて，Encoder-decoder モデルを学習した結果では，CER が 123.1 %とひどく悪化していることがわかる．表 3 の正規表結果を見てわかる通り，入力文とは全く異なる文字列を生成している．マイクロブログコーパスのみでは崩れ表記文と正規表記文のペアが非常に少なく，提案手法の学習が全くできていないことが考えられる．次に，擬似崩れ表記データを利用して Encoder-decoder モデルを学習した結果では，特に入力文と異なる文字を生成することもなく，崩れ表記を正規化することもできている．また，No-op と比較して，CER は 3.18 %のエラー削減と，マイクロブログコーパスを利用することなく CRF と同等の結果を得ることができた．

最後に，提案手法での失敗例について述べる．表 4 の入力文 2 では，文頭の「久」が連続して二文字生成されている．これは Encoder-decoder モデルを用いた機械翻訳で観測されている Over-translation（同じ単語が繰り返し翻訳される現象）が原因であると考えられる．また入力文 3 では，「捗」の文字が「暇」と違う文字として出力されている．これは BCCWJ の新聞データの中に「捗」が一度しか出現おらず，文字の出現頻度が低いことが原因であると考

えられる．結果として，Encoder-decoder モデルを学習するためには，大量のデータが必要であると考えられる．

## 6. おわりに

本研究では，ニューラルネットワークに基づく Encoder-decoder モデルを用いた日本語崩れ表記の正規化手法を提案した．また，大量の擬似崩れ表記データを用いることで，Encoder-decoder モデルの学習の精度を向上できることがわかった．今後の課題としては，入力文を出力文にコピーする機能を持つ Copy-attention モデル [24], [25] を用いることで Over-translation や出現頻度の問題を解決したいと考えている．また，マイクロブログコーパスと擬似崩れ表記データを組み合わせた追加実験を行う予定である．

### 参考文献

[1] Bahdanau, D., Cho, K. and Bengio, Y.: Neural machine translation by jointly learning to align and translate, *Proceedings of the 3rd International Conference on Learning Representations* (2015).
[2] 鍛治伸裕, 森信介, 高橋文彦, 笹田鉄朗, 斉藤いつみ, 服部圭悟, 村脇有吾, 内海慶: 形態素解析のエラー分析, ProjectNext エラー分析ワークショップ (2015).
[3] Kudo, T.: MeCab: Yet Another Part-of-Speech and Morphological Analyzer, http://taku910.github.io/mecab/ .
[4] Luong, T., Pham, H. and Manning, C. D.: Effective approaches to attention-based neural machine translation, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1412-1421 (2015).
[5] Cho, K., and van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724-1734 (2014).
[6] Sutskever, I., Vinyals, O. and Le, Q. V.: Sequence to sequence learning with neural networks, *Advances in neural information processing systems (NIPS)*, pp. 3104-3112 (2014).
[7] Zoph, B., Yuret, D., May, J. and Knight, K.: Transfer Learning for Low-Resource Neural Machine Translation *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP) (to appear)*, (2016).
[8] Hochreiter, S. and Schmidhuber, J.: Long short-term memory, *Neural Computation, Vol. 9, No. 8*, pp. 1735-1780 (1997).
[9] Saito, I., Sadamitsu, K., Asano, H. and Matsuo, Y.: Morphological analysis for Japanese noisy text based on character-level and word-level normalization, *Proceedings of COLING*, pp. 1773-1782 (2014).
[10] Kaji, N. and Kitsuregawa, M.: Accurate Word Segmentation and POS Tagging for Japanese Microblogs: Corpus Annotation and Joint Modeling with Lexical Normalization, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 99-109 (2014).
[11] Sasano, R., Kurohashi, S. and Okumura, M.: A simple approach to unknown word processing in japanese morphological analysis, *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pp.162-170 (2013).
[12] 佐々木彬，水野淳太，岡崎直観，乾健太郎: 機械学習に基づくマイクロブログ上のテキストの正規化, 人工知能学会第 27 回全国大会 (2013).
[13] Ikeda, K., Yanagihara, T., Matsumoto, K. and Takishima, Y.: Unsupervised text normalization approach for morphological analysis of blog documents, *Proceedings of Australasian Joint Conference on Advances in Artificial Intelligence*, pp. 401-411 (2009).
[14] Faruqui, M., Tsvetkov, Y., Neubig, G. and Dyer, C.: Morphological inflection generation using character sequence to sequence learning., *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* (2016).
[15] Xie, Z., Avati A., Arivazhagan N., Jurafsky D. and Ng, A. Y.: Neural language correction with character-based attention, *arXiv preprint arXiv:1603.09727* http://arxiv.org/pdf/1603.09727v1 (2015).
[16] Kudo, T., Yamamoto, K. and Matsumoto, Y.: Applying conditional random fields to Japanese morphological analysis, *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 230-237 (2004).
[17] Kurohashi, S., Nakamura, T., Matsumoto, Y. and Nagao, M.: Improvements of japanese morphological analyzer juman, *Proc. of The International Workshop on Sharable Natural Language Resources*, pp. 22-38 (1994).
[18] Neubig, G., Nakata, Y. and Mori, S.: Pointwise prediction for robust adaptable Japanese morphological analysis, *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short (ACL)*, pp. 529-533 (2011).
[19] Okazaki, N.: CRFsuite: a fast implementation of conditional random fields (CRFs), http://www.chokkan.org/software/crfsuite/ (2007).
[20] Kingma, D. and Ba, J.: Adam: A method for stochastic optimization, *Proceedings of ICR15* (2015).
[21] Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions, *arXiv preprint arXiv:1605.02688* http://arxiv.org/abs/1605.02688 (2016).
[22] Chrupała, G.: Normalizing tweets with edit scripts and recurrent neural embeddings, *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 680-686 (2014).
[23] Maekawa, K., Yamazaki, M., Ogiso, T., Maruyama, T., Ogura, H., Kashino, W., Koiso, H., Yamaguchi, M., Tanaka, M. and Yasuharu, D.: Balanced corpus of contemporary written Japanese, *Language Resources and Evaluation*, pp. 345-371 (2014).
[24] Gu, J., Lu, Z., Li, H and Li, V. O. K.: Incorporating copying mechanism in sequence-to-sequence learning, *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics (ACL) (to appear)* (2016).
[25] Miltiadis, A., Hao P. and Charles, S.: A Convolutional Attention Network for Extreme Summarization of Source Code, *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pp. 2091-2100 (2016).