擬似データの事前学習に基づくencoder-decoder型日本語崩れ表記正規化

斉藤 いつみ* 鈴木潤** 貞光 九月* 西田京介* 齋藤邦子* 松尾 義博*
NTT メディアインテリジェンス研究所*
NTT コミュニケーション科学基礎研究所**
{saito.itsumi, suzuki.jun, sadamitsu.kugatsu,
nishida.kyosuke, saito.kuniko, matsuo.yoshihiro}@lab.ntt.co.jp

1 はじめに

近年 Twitter 等を代表とするマイクロブログが普及し，個人によって書かれたテキストを対象とした評判分析や要望抽出，興味推定に基づく情報提供など個人単位のマーケティングのニーズが高まっている．一方このようなマイクロブログ上のテキストでは口語調や小文字化，長音化，ひらがな化，カタカナ化など新聞等で用いられる標準的な表記から逸脱した崩れた表記 (以下崩れ表記語と呼ぶ) が多く出現し，新聞等の標準的な日本語に比べ形態素解析誤りが増加する．

これらの崩れ表記語に対し，崩れた表記を辞書に存在する語（以下正規語と呼ぶ）にマッピングして解析を行うという表記正規化の概念に基づく解析が複数提案され，有効性が確認されている [1, 2, 6]．日本語における表記正規化と形態素解析手法としては，大きく (1) ルールにもとづいて入力文字列の正規化候補を列挙しながら辞書引きを行う方法 [9, 11, 12]，(2) 多数の崩れ表記語を列挙し，列挙した形態素のコストをモデルによって学習する方法 [3, 8, 13] が存在する．(1) では，事前に人手で定めた文字列レベルのルールに基づき，崩れた文字列に対し正規文字列を展開しながら解析するシンプルな方法が提案されている (例えば,「ぉ→う」という文字列正規化ルールを作成し，“ぉ”という文字列が入力文に含まれる場合 “ぉ”を “う”に変換した文字列に対しても辞書引きを行い形態素ラティスを拡張する)．(2) では，鍛治ら [3] は形態素正解データ，斉藤ら [8] は形態素正解と対応する正規語の正解データを用いて識別モデルを学習し，崩れ表記語を精度よく解析する方法を提案した．

一方，形態素解析を同時に行わず，文字列の正規化のみに着目した研究も存在する [15, 14]．これらの研究では，文字列レベルでの正規化を CRF を用いて推定する手法 [15] や，encoder-decoder 型ニューラルネットワークモデルを用いて正規化を行う手法が提案されている [14]．文字列正規化のみに着目するメリットとして，形態素レベルの正解データが必要なく形態素境界の定義に依存せずに正規化を行えることがあげられる．さらに，encoder-decoder に基づく手法では，その他の手法のように文字列レベルでのハードなマッチングを行わないため，表記の揺れのバリエーションに対して頑健に働くことが期待できる．しかし，encoder-decoder モデルで成功を収めている機械翻訳や要約の分野では，多量の入出力ペアの学習データが存在することが前提となっている．日本語の正規化においては人手の正解ペアが数千文程度しか存在しないため [14] ではあらかじめ定めたルールに基づいて生成した擬似正解データを用いて学習を行い，その結果擬似データを用いた場合に正規化の精度が向上することを報告している．しかし，現実の分布と異なる擬似データを用いることの影響や，より効果的な擬似データの利用方法については明らかになっていない．

本研究では，文献 [14] と同様に Encoder-Decoder 型ニューラルネットワークを用いた日本語正規化の検討を行う．この際，自動生成した擬似正解を用いて学習を行うが，擬似データを直接正解データとして使用するのではなく，少量の人手正解データで学習する前の事前学習として用いる方法を提案する．実験により，擬似正解データを用いた学習結果を事前パラメータとして，再度少量の人手正解で学習を行うことで正規化精度が向上することを確認した．

2 手法

2.1 encoder-decoder型ニューラルネットワークモデルを用いた文字列正規化

本研究では，文献 [14, 7] と同様に，文字列正規化の問題を attention を考慮した文字列レベルの encoder-decoder モデルで定義する．ここで，崩れた文字列を $s = (s_1, s_2, ..., s_n)$, 正規の文字列を $t = (t_1, t_2, ..., t_m)$,

とすると，崩れた文字列を正規の文字列に変換する問題は，s, θ が与えられた時，次の確率を最大化する系列 t を選択する問題と考えることができる．

$$p(t|\boldsymbol{s}, \theta) = \prod_{j=1}^{m} p(t_j|t_{<j}, \boldsymbol{s}) \quad (1)$$

ただし，$\boldsymbol{s}$ は入力文字列 s を encoder を用いて変換した固定長符号ベクトルである．decoder では，$\boldsymbol{s}$ とそれまでの出力 $(t_1, t_2, ..., t_{j-1})$ から次の出力 t_j を予測する．パラメータ θ は学習データに基づいて推定する．本研究では，encoder, decoder ともに一層の LSTM で実装し，encoder は bi-directional LSTM とした．また，実験時は文字列の embedding, 隠れ層の次元数をそれぞれ 300, 256 に設定した．

2.2 擬似正解データの作成

文字列正規化の正解データは多くは存在しないため，文字列正規化モデルを効果的に学習するために，擬似データを作成した．擬似データの作成は，1) あらかじめ定めた文字列変換パタン, 2) 学習データに出現した変換パタンの 2 つを用いて，下記のように行った．

1. web 上の平文を mecab[5] を用いて解析する．この際，形態素解析辞書は unidic 辞書を用いた．
2. 解析済文に対して，文字列パタンと品詞条件にマッチした文字列の変換を行う．この際の文字列・品詞照合パタンは上記で述べた 2 パタンである．

自動生成された擬似正解データの例を 1 に示す．これらの中には解析誤りも含まれるため，誤った変換を行ってしまったデータも含まれるが，そのまま学習データとして用いる．1) のあらかじめ定めたパタンとしては，母音の発音化（ああ→あー，いい→いー等），小文字化，末尾への「ー」「っ」「母音」挿入，形容詞の口語化（いい→ええ等）など [3, 14] でも行われているものとほぼ同様のパタンを使った．また，2) の学習データ中のパタンとして，学習データには形態素正解も付与したため形態素単位の変換パタンを抽出した．抽出したパタンは 1538 パタン存在し，頻出するパタンとして「だ：助動詞→や：助動詞」「ほんとう：名詞→ほんま：名詞」などの方言や「はやく：形容詞→はよ：形容詞」などの口語表現が存在した．表 2 に抽出された崩れパタンの例を示した．2) を用いることにより，1) のシンプルなパタンでは得られない多様なパタンを生成することができる．また，1) と 2) の組み合わせに関しても生成を行った．例えば，2) に「やべえ」という崩れ表記があり，1) のルールで語尾の母音を連続させる，というルールがあった場合には「やべえええ」といった表記も生成した．今回は，web 上からランダムサンプリングした Twitter データに対し変換を行い，約 11 万文の擬似正解データを作成した．

表 1: 擬似正解データの例

	解析結果例
元文字列	お家帰るぞ！
解析結果	お/家/帰る/ぞ!
変換パタン	ルール：末尾「ー」挿入
生成文字列	お家帰るぞー!
元文字列	きつねかわいい
解析結果	きつね/かわいい
変換パタン	ルール：形容詞末尾音変換「いい→ええ」
生成文字列	きつねかわええ
入力例 3	ほんま綺麗だな
解析結果	ほんま/綺麗/だ/な
変換パタン	学習データパタン:助動詞「だ」→「や」
生成文字列	ほんま綺麗やな

表 2: 学習データから得た崩れパタンの例

正規文字列	崩れ文字列	品詞
だ	や	助動詞
だろう	やろ	助動詞
ていうか	てか	助詞/動詞/助詞
ほんとう	ほんま	動詞
てる	とる	助動詞
はやく	はよ	形容詞
ぞ	ぞー	終助詞
ちゃん	たん	接尾辞
たい	たす	助動詞

2.3 学習

本研究では，まず作成した擬似データと人手正解データを統合し，事前学習を行った．その後，事前学習によって得られたパラメータを初期値として少量の人手正解データで再度学習を行った．パラメータ推定には chainer[10] を使用した．

3 実験

3.1 実験データ

本研究では，人手アノテーションを行った Twitter データを用いた．ランダムサンプリングしたツイートに対し人手で正規表記と形態素正解を付与したデータを，Train 6361 文, Test 987 文, Develop 493 文にランダムに分割して使用した．また，今回の検討では学習時間の短縮のため 50 文字以内のデータを用いた．Twitter データにおける崩れ表記の割合は表 3 に示した「変換なし」の精度から確認することができる．今回のデータでは，変換なしの場合の精度が 0.966 となっており，全体の約 3.4%の文字列が変化しているデータであり，

全体のごく一部を書き換えるタスクであることがわかる．

3.2 評価方法と比較手法

評価指標として，出力文字列と正解文字列の文字列アライメントを行い，Precision, Recall, F 値を算出した．ここで，出力文字列の文字数を N_s, 正解文字列の文字数を N_r，共通部分の文字数を N_c とすると，precision=N_c/N_s, recall=N_c/N_r として計算した．また，今回のタスクでは変化しない文字が多数を占めることから，変化しない文字（正規文字）と変化する文字（崩れ文字）に分割してそれぞれの再現率も計算した．ここで，正規文字と崩れ文字については，変換前の元文字列と正解文字列のアライメントを計算し，共通部分を正規文字，不一致部分を正規文字と定義した．比較手法として，代表的な統計機械翻訳ツールである Moses[4] を用いた場合，encoder-decoder モデルを用いた場合を比較する．また，学習データとして人手アノテーション学習データ（以下 train と呼ぶ）のみから学習した場合，擬似データと train データをそのまま学習させた場合，擬似データと train データを事前学習として用い，train で学習した場合（提案法）を比較した．Moses の設定に関しては，distortion limit の設定を 0 とした．encoder-decoder モデルを用いてデコードする場合は，beam サーチを行い beam 幅は 10 とした．

3.3 実験結果

3.3.1 文字列正規化の精度

実験の結果を表 3 に示した．Moses を用いた場合，train データのみでも変換なしに比べ約 1.7%の精度向上が見られた．内訳を確認したところ，正規の文字列はあまり変えずに崩れ文字列を約 65%再現できていることがわかった．一方，擬似正解データを追加すると約 0.3%精度が低下した．これは，擬似正解データにノイズが含まれているため悪影響を与えたことが原因と考えられる．Moses を用いた場合，数千文程度の人手正解データに単純に擬似正解データを追加しても精度が向上しないことがわかった．一方，encoder-decoder を用いた実験では，train データのみを用いた場合は何も変換しない場合よりわずかに文字列一致率が向上する程度の精度向上にとどまっている．正規文字列と崩れ文字列の再現率内訳をみると，正規文字列の再現率が 0.982 と低下していることから encoder-decoder モデルを用いて train データのみから学習を行った場合には，変換すべきでない文字への悪影響が発生し精度の向上が低くとどまってしまったと考えられる．擬似データを追加した結果では，正規文字列の再現率が改善しており，全体の F 値も train のみに比べ約 1.3%向上している．このことから，多少のノイズがあるデータであっても，多量のデータを学習させることにより過剰な変換を抑制できるため精度が向上するということがわかった．

擬似データを追加した学習結果を初期値として，再度少量の人手正解データを学習させた場合では，単に人手正解と擬似正解データを結合して学習させた場合に比べさらに精度が向上した．特に，再現率の内訳を確認すると崩れ文字列の再現率が向上している．これにより，ほぼ Moses と同等の精度となっている．擬似データは実際のデータ分布とは異なるため，ベースモデルとして用いて最終的に少量の人手正解で学習することでより対象のコーパスに適応したモデルが得られたものと考えられる．

3.3.2 解析結果と考察

解析結果の例を表 4 に示した．入力例 1 は，擬似データ+train 以外の手法で解析できた例である．「ほんつま」→「ほんとう」という変換例は学習データには存在しなかったが，「ほんま」→「ほんとう」という学習データが存在し，「っ」→「NULL」となるパタンも存在したため組み合わせで解析できた例と考えられる．擬似データを追加した場合には，正規化できなくなってしまっているが，再学習によりただしく解析されるようになった．入力例 2 は，Moses が失敗し encoder-decoder が正解した例である．ひらがなの「なん」という文字列は「何」「なの」と複数の変換候補があり，文脈によって正規化先が異なる．この例に関しては encoder-decoder が適切な候補を選択できた．encoder-decoder モデルの場合，広範囲の文脈を見ることができるため学習データを増やせばこのような曖昧性のある候補についても頑健なモデルとなることが期待できる．例 3 は，再学習した encoder-decoder が誤った例である．擬似学習データに「おもろ」→「おもろい」というパタンが存在したため「おもろい」という変換の確率が高くなり，再学習しても誤ってしまったものと考えられる．このような誤りに関しては，より現実の分布に近い擬似データを作ることによっても解消できる可能性があると考えられる．

4 おわりに

本研究では，崩れ表記の正規化を文字レベルの encoder-decoder モデルを用いて行った．この結果，encoder-decoder モデルを用いた文字列正規化を行う場合，擬似正解データを用いて学習を行った後に再度少量の人手正解データで学習を行うことが有効であることがわかった．今後は，より効果的な学習データの生成方法や文字列正規化により適したモデルの検討を

,

表 3: 解析結果の精度比較

	文字一致率			再現率内訳	
	Precision	recall	F 値	正規文字	崩れ文字
変換なし	0.968	0.964	0.966	1.0	0
Moses (train のみ)	0.985	0.984	**0.984**	0.996	0.655
Moses (train + 擬似データ)	0.980	0.981	0.981	0.995	0.527
Encoder-decoder (train のみ)	0.968	0.968	0.968	0.982	0.528
Encoder-decoder (train+擬似データ)	0.981	0.979	0.980	0.996	0.516
Encoder-decoder (train+擬似データ事前学習 + train 学習)	0.984	0.984	**0.984**	0.996	0.621

表 4: 各手法による正規化例

	解析結果例
入力例 1	ほんっまに帰って寝たい
正解	ほんとうに帰って寝たい
Moses	ほんとうに帰って寝たい
encdec (train)	ほんとうに帰って寝たい
encdec (擬似+train)	ほんっまに帰って寝たい
encdec (提案)	ほんとうに帰って寝たい
入力例 2	ハートに生きる戦士なんで！
正解	ハートに生きる戦士なので！
Moses	ハートに生きる戦士何で！
encdec (train)	ハートに生きる戦士なので！
encdec (擬似+train)	ハートに生きる戦士なので！
encdec (提案)	ハートに生きる戦士なので！
入力例 3	めっちゃおもろなってきたな
正解	めっちゃおもろくなってきたな
Moses	めっちゃおもろくなってきたな
encdec (train)	めっちゃおもろくなってきたな
encdec (擬似+train)	めっちゃおもろいなってきたな
encdec (提案)	めっちゃおもろなってきたな

行っていく予定である.

参考文献

[1] Bo Han and Timothy Baldwin. Lexical normalisation of short text messages: Makn sens a #twitter. *Proceedings of the 49th ACL*, pp. 368–378, 2011.

[2] Bo Han, Paul Cook, and Timothy Baldwin. Automatically constructing a normalisation dictionary for microblogs. *Proceedings of the 2012 Joint Conference on EMNLP ans CoNLL*, pp. 421–432, 2012.

[3] Nobuhiro Kaji and Masaru Kitsuregawa. Accurate word segmentation and pos tagging for japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 Conference on EMNLP*, 2014.

[4] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. 2007.

[5] T. KUDO. Mecab : Yet another part-of-speech and morphological analyzer. *http://mecab.sourceforge.net/*, 2005.

[6] Chen Li and Yang Liu. Improving text normalization using character-blocks based models and system combination. *Proceedings of COLING 2012*, pp. 1587–1602, 2012.

[7] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on EMNLP*, pp. 1412–1421, September 2015.

[8] Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, and Yoshihiro Matsuo. Morphological analysis for japanese noisy text based on character-level and word-level normalization. In *Proceedings of COLING 2014*, pp. 1773–1782.

[9] Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. A simple approach to unknown word processing in japanese morphological analysis. *Proceedings of the Sixth IJCNLP*, pp. 162–170, 2013.

[10] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on NIPS*, 2015.

[11] 岡照晃, 小町守, 小木曽智信, 松本裕治. 表記のバリエーションを考慮した近代日本語の形態素解析. 人工知能学会全国大会講演集, 2013.

[12] 勝木健太, 笹野遼平, 河原大輔, 黒橋禎夫. Web 上の多彩な言語表現バリエーションに対応した頑健な形態素解析. 自然言語処理学会年次大会講演集, pp. 1003–1006, 2011.

[13] 工藤拓, 市川宙, David Talbot, 賀沢秀人. web 上のひらがな交じり文に頑健な形態素解析. 自然言語処理学会年次大会講演集, 2012.

[14] 池田大志, 進藤裕之, 松本裕治. Encoder-decoder モデルを用いた日本語崩れ表記の正規化. 研究報告自然言語処理（NL）, 2016.

[15] 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. 第 27 回人工知能学会全国大会講演集, 2013.