Vol.2013-NL-214 No.5
2013/11/14

# 正規-崩れ表記のアライメントに基づく 表記崩れパタンの抽出と形態素解析への導入

斉藤いつみ　貞光九月　浅野久子　松尾義博

**概要**：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が多く出現し形態素解析誤りを起こす一因となっている．本研究では，ソーシャルメディア上のテキストから抽出した崩れ表記に対し正規表記を付与した正解データを用いて，文字列アライメントを求め文字列レベルの表記の崩れパタンを自動抽出する．また得られたパタンに基づき，入力文の崩れ表記から正規の表記を展開し形態素ラティスを拡張することで，従来法に比べ多様な崩れパタンの解析を可能とした．実験では，対象とした崩れ表記箇所の解析結果に関して，従来法に比べ約 30%の解析誤りを改善することができた．

# Extracting Derivational Patterns based on the Alignment of a Standard Form and its Variant towards the Japanese Morphological Analysis for Noisy Text

ITSUMI SAITO　KUGATSU SADAMITSU　HISAKO ASANO　YOSHIHIRO MATSUO

***Abstract:*** Twitter and other micro-blogging data are written in an informal style, so there are many types of non-standard tokens such as abbreviations, phonetic substitution. When analyzing such noisy text, conventional text analysis tools often perform poorly. In this study, we propose a method for simultaneous morphological analysis and normalization using derivational patterns which was extracted based on the alignment of standard tokens and its variant tokens. The experimental study demonstrates that our approach outperforms conventional Japanese morphological analysis tools in the analysis of non-standard tokens.

## 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．多くの言語処理技術において形態素解析は前処理として用いられ，解析誤りはその後の処理に悪影響を及ぼすため，高い解析精度が求められる．例えば，次の例 (a) の文を考えよう．

**(a) 映画を見ようそぉしよう！**

上記の文に対して，mecab[1] を用いて解析を行うと以下のような解析結果となる．

**映画 (名詞)/を (助詞)/見よ (動詞・命令)/うそ (名詞)/おしよう (名詞)/！ (記号)/**

これは正規表記の「そう」という表記が「そぉ」（例文 7,8 文字目）という表記に崩れたために解析誤りを起こし，周囲の解析結果にも悪影響を及ぼしている例である．上記の例の場合，「ぉ」という崩れた表記によって「そう（副詞）」という正しい形態素の解析に失敗するだけではなく「うそ (名詞)」という誤った単語の湧き出しも生じている．このように，崩れ表記による解析誤りは意味解析において大きく影響を及ぼす可能性がある．上記の例について「ぉ」を

[1] NTT メディアインテリジェンス研究所
Yokosuka, Kanagawa, Japan
saito.itsumi@lab.ntt.co.jp

表 1　崩れ表記の分類と本研究の対象範囲

| 大分類 | 小分類 | 具体例 |
|---|---|---|
| 口語 | 促音の挿入，置換 | かっわいい, いこっか |
| | 長音の挿入，置換 | ねむーーい, とーきょー |
| | 母音の挿入 | まあるく, きたあああああ |
| | 発音の崩れ | すっげぇ, くだしゃい |
| 異表記 | 小文字化 | ぃぃよ, おうち |
| | カタカナ/ひらがな化 | アリガトウ, てすと |
| | 同音異表記 | まぢ, した yo |
| | 形の類似 | ネ申 |
| 誤字脱字 | タイプミス | これをを |
| | 送り仮名誤り | 面倒臭せ |
| | 漢字の誤用 | 待ち通しい |
| ネットスラング | ネット用語 | よかた, ですた |
| | 伏字 | ドラ○モン |
| 略語 | 略語 | おめ, よろ |
| 方言 | 方言 | やらへん, してんねや |

「う」に変換して解析を行った場合の解析結果は以下のようになり正しく解析される．

**映画 (名詞)/を (助詞)/見よ (動詞・未然)/う (助動詞)/そう (副詞)/しよ (動詞)/う (助動詞)/！ (記号)/**

従来はこれらの崩れ表記に対し，崩れた表記を直接辞書に追加する方法 (例えば，「そぉ (副詞)」を辞書に追加する) や，事前に人手で定めたルールに基づき崩れた表記を新聞等の表記に正規化して解析する方法 (例えば，「ぉ→う」というルールを作成し，崩れた文から正規化された文への書き換えを行う，正規文字列を展開して辞書引きを行い形態素ラティスを拡張する) などが用いられてきた．しかし，崩れた表記のパタンは多様であるため，人手による辞書追加やルール作成で現実に存在する多くのパタンを網羅することは高コストであり，再現率の点で限界がある．

本研究では，表記崩れパタン (「う→ぉ」等) を用いて正規文字列を展開し形態素ラティスを拡張するという手法を用いるが，ルールを人手で作成するのではなく正規表記と崩れ表記のアライメントから統計的に求め，再現率の向上を試みる．この際，ルール増加に伴う列挙候補の増加による解析誤りの増加を抑えつつ，再現率を向上させる方法を示す．表 1 には，崩れ表記の分類と本研究で扱う範囲（網掛け部）を示した．対象範囲は，音的な類似という点で特定のパタンが存在すると考えられる口語調の崩れ表記や，異表記（小文字化，同音異表記），またネット上で用いられる特有語（よかた，ですた等）とした．これらを対象とした理由は，崩れ表記全体の中で占める割合が大きく今回の提案手法で統一的に表現できる現象であり，従来の解析範囲を一つのモデルで拡張できる可能性があると考えたためである．本研究の構成を述べる．2 章では関連研究について整理し，3 章では提案手法について詳述する．4 章では実験内容と結果を示し，5 章でまとめと今後の課題を述べる．

入力文:すーごく楽しい

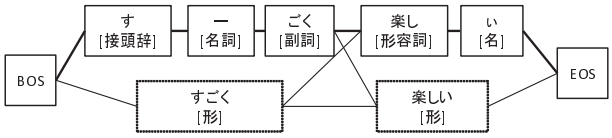


図 1　崩れ表記に対するラティス拡張の例

## 2. 関連研究

日本語の崩れ語正規化や解析に関する研究は近年増加しているが，既存研究におけるアプローチと対象は大きく次の 4 つに分類できる．(1) 正規化ルールの獲得（池田ら [19], [20]），(2) 崩れた文の正規化（佐々木ら [18]），(3) 正規化ルールを用いたラティスの拡張と辞書引きに基づく形態素解析 (勝木ら [13], 笹野ら [15], 岡ら [17])，(4) ひらがな化現象の正規化を考慮した言語モデルに基づく形態素解析（工藤ら [14]）．(1),(2) は主に正規化のみに着目した研究であり，(3),(4) は正規化と形態素解析を同時に行っている手法である．以下，それぞれの手法の特徴について述べる．

(1) では，ブログ上で形態素解析が未知語になった箇所について隣接する形態素をキーとして新聞の表記とマッチングを行い正規-崩れパタンの自動獲得を行っている．ルール自動取得のアイデアは重要であるが，実際に崩れ表記によって解析誤りが生じる箇所では未知語と判断されず既知語の組み合わせによって解析されてしまう箇所も多く，仮に未知語として解析されても区切りが誤っているために周辺の単語素性を用いても適切な候補が検索されるとは限らないため，必ずしも適切な候補が取得できるわけではないという課題がある．さらに Twitter はブログに比べ非常にノイズの多いコーパスであるため不要な候補が多く含まれてしまうことが予想される．

(2) では，入力文字列の「削除」「置換」「挿入」を機械学習によって直接推定する手法を提案している．日本語の崩れ現象は文字列レベルで検討する必要があるため文字列レベルで正規化を行う概念は必要となるが，同時に前後の単語にも依存関係があるため単語列のもっともらしさを同時に考慮することによってより精度を高めることができると考えられる．

(3) は事前に定めた正規化ルールに従って，文字列の辞書マッチを拡張する手法である．具体例を図 1 に示した．入力文「すーごく楽しぃ」を mecab を用いて解析すると図 1 の上段の系列（す/ー/ごく/楽し/ぃ）という結果となってしまうが，事前に正規化ルール（ー→ null，ぃ→い）などを設定しておき，入力文に対しルールを適用して正規化された文字列に対しても辞書引きを行うと図 1 の下段に示すような「すごく，楽しい」といった正規化された正しい

形態素を列挙することができる．そして，このように拡張された候補をすべて統合したラティスから全体が最適となるような単語区切りと品詞を決定する．本研究で扱うアプローチも基本的な概念は手法 (3) と同じであるが，(3) が人手でルールを作成しているのに対し，本研究では人手抽出した崩れ表記と対応する正規表記のペアから統計的に崩れルールを抽出している点，全体を識別モデルとして定式化し複数の素性を用いて展開候補のもっともらしさを評価している点が異なる．

(4) はひらがな化現象に特化して個々のひらがな化確率を推定している点で興味深いが，本研究では正規表記の列挙候補が多数存在するような崩れ現象（音的な崩れや類似する音での表記）を扱うという点で扱う対象が異なる．

英語の崩れ表現解析に関する研究では，Han ら [8] が未知語の正規語推定を周辺情報の分布類似度と単語内の文字列類似度（編集距離等）を用いて行っており，Ling ら [9], [10] は崩れ文に対する正規文を自動的に取得し，対応する語を抽出する手法を提案している．英語の崩れ語解析ではテキスト上に出現した辞書にない単語が崩れ語であるか否かの推定モデルや，周辺文脈から正規語を推定する問題設定が多い．英語の場合は単語と単語の間にスペースが挿入されるため単語の区切りが明示的にわかるためこのような問題設定が主流であると考えられるが，日本語においては単語区切りは分かち書き（形態素解析）を行わないと決定されないため，英語の手法をそのまま適用することは難しい．

日本語の崩れ表現解析においては，分かち書きがされていない文に対して，文字列レベルでの書き換えのもっともらしさと単語レベルでの連接のもっともらしさを同時に考慮する必要がある．日本語崩れ表現解析における従来手法はいずれも，多様な崩れ表現のカバー率向上を課題として上げており，解析可能な範囲をどのように拡張していくかということが今後の崩れ表現解析の大きな課題であるといえる．本研究では，正規-崩れ表記の正解ペアデータから学習した文字列単位の崩れパタンを形態素解析器のデコーダに組み込み解析可能な範囲の拡大を試みる．

## 3. 提案手法

本研究の全体構成を図 2 に示す．まず，アライメントに基づく崩れ表記パタン抽出のための正解データを作成するため，崩れ表記を含むテキスト（ブログや Twitter）から崩れ表記（例：おぃしーぃ）を人手で抽出する．そして，抽出した崩れ表記に対して人手で正規表記を付与する（おぃしーぃ→おいしい）．これらのペアデータ（おぃしーぃ, おいしい）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表記崩れパタンを統計的を抽出する．（例：null →ー, い→ぃ）．そして得られた崩れパタンを解析に組み込みラティスの拡張を行うことで正規化と形態素解析を同時に行う．以下各項目で詳しく述べる．

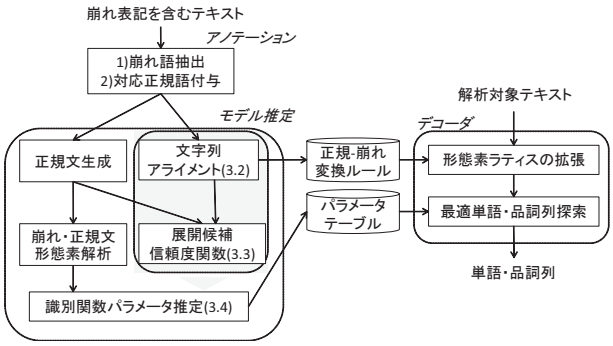


**図 2** 提案手法の構成

### 3.1 識別モデルに基づく定式化

識別モデルは柔軟な素性設計と高精度な解析が可能になることから形態素解析や翻訳などで広く用いられている．本研究の対象である崩れ表記も多様な素性を同時に考慮することが望ましい問題と考えられるため，識別的モデルを用いて表現することにする．今，ある入力文に対して最適な出力 $\hat{\mathbf{w}}$ を計算する問題は以下のような線形モデルの最大化として定義できる．

$$\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} \sum_{k} \lambda_k \phi_k(\mathbf{w}) \tag{1}$$

ここで，$\mathbf{w}$ は単語表層+品詞列，$\lambda_k$ は $k$ 番目の素性に対する重み，$\phi_k(\mathbf{w})$ は $k$ 番目の素性関数を表す．今回は素性として品詞連接コスト $h(t_{i-1}, t_i)$，単語生起コスト $h(w_i, t_i)$，単語表層 n-gram コスト $-\log(p(w_i|w_{i-1}))$，表記崩れコスト $-\log(p(c_{sk}|c_{wk}))$，展開候補信頼度 $g(\theta)$，の 5 つを用いた．このうち，品詞連接コスト $h(t_{i-1}, t_i)$，単語生起コスト $h(w_i, t_i)$ は mecab で用いられている推定値を用いた．ここで，$t_i$ は $i$ 番目の品詞，$w_i$ は $i$ 番目の単語を表す．単語表層 n-gram コスト $-\log(p(w_i|w_{i-1}))$ はブログの形態素正解が付与されていないラベルなしコーパスの自動解析結果より求めた．表記崩れコスト $-\log(p(c_{sk}|c_{wk}))$ と展開候補信頼度 $g(\theta)$ については，それぞれ 3.2, 3.3 節で述べる．ここで，$c_{sk}, c_{wk}$ はそれぞれ崩れ文字列, 正規文字列を表し，$\theta$ は展開候補信頼度を求める際の素性ベクトルを表す．式 1 の重みパラメータ $\lambda_k$ の推定については 3.4 節で述べる．これらを組み込んだデコーダについては 3.5 節で述べる．

### 3.2 正規-崩れ表記ペアからの崩れパタン抽出

本節では，式 1 で定義した表記崩れコストの推定と表記崩れパタンの抽出について述べる．本研究では，正規-崩れ表記の正解ペアデータを用いて推定を行う．正規-崩れ表記ペアの例を表 2 に示した．表 2 に示したように，人手で適当な単位ごとに崩れ表記の抽出と対応する正規表記付与を行い正解ペアデータを作成した．正規-崩れの文字列変換パタンを推定するため，Brill ら [4] で示された以下の式を用いて推定する．Brill らは，1 文字単位の編集距離を拡張し重み付きの複数文字列レベルの変換確率を推定する方

表 2　崩れ表記抽出と正規表記付与の例

| 崩れ表記 | 正規表記 |
|---|---|
| ありがとうございま〜す | ありがとうございます |
| いちお | いちおう |
| うっつらうっつら | うつらうつら |
| お願いしま〜す | お願いします |
| くださ〜い | ください |
| じゃあない | じゃない |
| ずーーーっと | ずっと |
| だぞい | だぞ |
| ちびっと | ちょびっと |
| っつーと | っていうと |
| てか | ていうか |
| てゆうか | ていうか |
| どーしても | どうしても |
| なぁ〜んて | なんて |

法を提案している．

$$p(s|w) = \max \prod_{k=1}^{K} p(c_{sk}|c_{wk}) \quad (2)$$

ここで，$s$ は崩れ文字列，$w$ は正規文字列を表し，$c_{sk}$,$c_{wk}$ はそれぞれ $s$,$w$ 中の部分文字列を表す．$K$ はアライメントにおける部分文字列の個数を表す．例えば，図 3 における例で経路 1 が最も確率の高い経路の場合，$p($ かなぁーり $|$ かなり $)=p($ かな $|$ かな $) \cdot p($ ぁー $|$null$) \cdot p($ り $|$ り $)$ となる．

文字列アライメントの正解は未知なので，初期アライメントを設定し繰り返し計算を行うことでもっともらしい分割と文字列アライメントを求める．予備実験よりアライメント結果は初期値に大きく依存することが明らかになったため，本研究では Brill らの方法に従って初期アライメントを計算する．以下に具体的な推定手順を示す．

**step1:初期確率値設定**

1 文字単位の編集距離を用いてヒューリスティックに初期確率を計算する．具体的には，次のようにして初期確率を求める．

(a)$s, w$ のペアに対して一文字単位の重みなし編集距離を用いて編集距離が最小となるようなアライメントを求める．

例)$w$=かなり，$s$=かなぁーりの場合，

$(c_{s1}, c_{w1})$=(か, か)，$(c_{s2}, c_{w2})$=(な, な)，

$(c_{s3}, c_{w3})$=(ぁ,null)，$(c_{s4}, c_{w4})$=(ー,null)，

$(c_{s5}, c_{w5})$=(り，り) というアライメントが求まる．

(b)(a) で求めたアライメントを用いて，周辺の窓枠 $j$ までの部分アライメントを結合して複数文字列単位の可能なアライメントを生成する．

例えば，$j = 2$ の時，可能なアライメントとして (かな, かな)，(なぁ, な)，(ぁー,null)，(ーり, り) が生成される．このようなルールによって生成されたアライメントから，各アライメントの生成確率を求め初期値とする．日本語の崩れの現象は周辺の数文字単位でまとまっておこることが多いためこのような方法によって現実的な初期値の設定を行った．

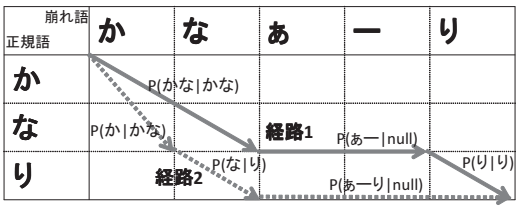


図 3　正規表記と崩れ表記の文字列アライメント計算例

**step2:期待値の計算**

全てのテキストについてアライメントを求めた結果を用いて，文字列変換確率を以下の式に従って求める．ただし，$N(c_{sk}, c_{wk})$ は全テキスト中で部分文字列アライメント $(c_{sk}, c_{wk})$ が出現した回数である．

$$p(c_{sk}|c_{wk}) = \frac{N(c_{sk}, c_{wk})}{\sum_{c_{sk}} N(c_{sk}, c_{wk})} \quad (3)$$

**step3:経路確率とアライメントの計算**

ステップ 2 で求めた文字列変換確率 $p(c_{sk}, c_{wk})$ を用いて，もっとも生起確率が高い経路（アライメント）を新たな最適アライメントとして求める．(図 3 で可能な経路をすべて求めて最適経路を算出する)

ただし，現実的なアライメントを求めるため可能な部分アライメントの文字長の最大値を 3 として計算を行った．

**step4:繰り返し**

アライメントが収束するまでステップ 2, 3 を繰り返す．上記で求めたアライメントは初期値に依存するため，必ずしも最適解とは限らず不自然なアライメントも存在する．また，正解データに特殊な崩れ表記が含まれていれば，非常に起こりにくい崩れパタンも抽出される．信頼度の高い変換ルールを用いるため，デコードの際には上記で求めた変換ルールのうち，確率値が閾値以下のものを足切りして使用した．

### 3.3　展開候補信頼度モデル

3.2 節で求めた正規-崩れの文字列アライメントとその確率は周辺の素性等は使用しておらず単純に部分文字列の崩れ現象のみをモデル化した非常にシンプルなモデルとなっている．これはできるだけ細かい単位の表記崩れパタンを抽出することで解析における崩れ現象のカバー率を向上させるためであるが，ルール数が多くなることから実際に解析に組み込む際には正解以外の候補展開数が非常に多くなり，解析誤りを起こす可能性も高くなる．このため周辺文字列まで考慮して重み付けを行うことが精度の観点からは

元文:めーっちゃかわえー
正規文:めっちゃかわいい

**図 4** 崩れ文と正規文例

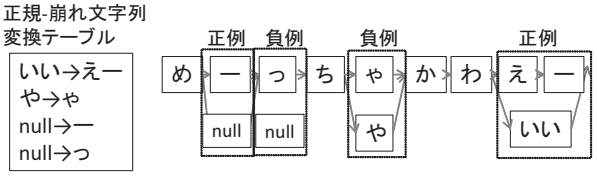


**図 5** 崩れ文に対する候補展開と正解ラベル付け例

望ましい．文字列情報を直接全体の識別モデルの素性としてパラメータ推定を行うことも可能であるが，本稿では計算を簡略化するため事前にすべての展開候補について周辺文字列の情報を用いて正解展開候補と誤り展開候補の識別モデル (線形 SVM) を構築した．このモデルによって得られた識別関数を展開候補の信頼度関数 $g(\theta)$ として全体の識別モデルでの素性として用いることで文字列情報というより細かい情報を間接的にモデルに導入する．

#### 3.3.1 素性

展開候補信頼度推定の周辺素性 $\theta$ として以下の 4 つを用いた．

- $\frac{\text{展開候補先の周辺文字 n-gram 確率}}{\text{展開候補元の周辺文字 n-gram 確率}}$
- 展開候補先の周辺文字 n-gram 確率
- 展開した文字列変換確率（例 : $p(ぁ \mid あ)$，$p(ん \mid れ)$）
- 展開候補元と展開候補先の文字列表層（前の文字 1 文字を追加）

例えば，図 5 の 5 文字目の「ゃ」から「や」を展開する箇所において，展開候補先の周辺文字 bi-gram は $p(や \mid ち)$, 展開候補元の周辺文字 bi-gram は $p(ゃ \mid ち)$ などとなる．このように文字列レベルでの多様な周辺情報を考慮することでアライメント時の確率情報のみに比べ正しい候補と誤った候補の識別精度が高くなると考えられる．

#### 3.3.2 学習データの作成

展開候補の正解・不正解について人手でラベル付けされた学習データは存在しないため学習データを作成する必要がある．本稿では，3 節で述べた人手ラベリングによって正規文のわかっている文に対して 3.2 節で求めた文字列崩れパタンを用いて候補展開を行い，正解と誤りの候補に対し自動的にラベリングした．図 5 に具体例を示す．図 5 の例の場合，2 文字目の「ー」から「null」を展開する箇所について正規文と比較すると一致するため正例ラベルを付与する．3 文字目の「っ」から「null」を展開する箇所については，変換先の文字列が正規文と一致しないた負例ラベルを付与する．このようにして，すべての展開候補について正例と負例のラベルを付与し正解データとする．推定には，SVMlight[2] を用いた．

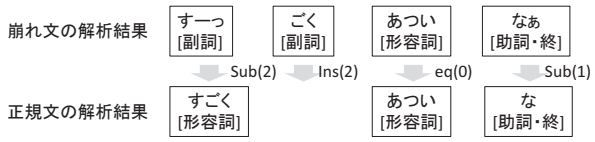


**図 6** 損失関数の例

### 3.4 パラメータ推定

本研究では、式 1 における重みパラメータの推定を MERT（誤り率最小化学習）[7] を用いて行った．正規文と崩れ文の形態素解析結果の異なり数を直接最小化するために MERT を用いた．MERT とは以下で定義される損失関数を最小化することで最適な重みを求める手法である．

$$\hat{\lambda} = \arg\min_{\lambda \in \Lambda} l(\mathbf{w}; \lambda) \tag{4}$$

$$l(\mathbf{w}; \lambda) = \sum_{i=1}^{N} \text{error}(E, \arg\max_{\mathbf{w} \in A} \sum_{k} \lambda_k \phi_k(\mathbf{w})) \tag{5}$$

ここで，$\hat{\lambda}$ は最適な重みパラメータであり，$\arg\max_{\mathbf{w} \in A} \sum_k \lambda_k \phi_k(\mathbf{w})$ はパラメータ $\lambda$ における 1-best 解を表す．$A$ は可能な単語列の候補集合（n-best 解），$E$ は正解コーパスの単語（+品詞）列とする．$N$ は全体の文数である．これより，式 5 における損失関数 $l(\mathbf{w}; \lambda)$ は，$N$ 文における正規文の単語（+品詞）列とシステム出力の単語（+品詞）列の差分を表す． 損失関数は自由に設定することが可能であるが，今回は人手抽出した崩れ表記を正規表記に置き換えた文（正規文）の解析結果と元文のシステム解析結果（1-best 解）の異なり数を損失関数として用いた．異なり数を計測する際には形態素正解，正規語正解の双方がラベル付けされたデータを正解として用いることが望ましいが，人手で全ての正解を付与することは高コストなため，正規文の自動解析結果とシステム解析結果の単語単位の編集距離を求め変換回数をエラー回数として設定した．この際，表層の誤りと品詞の誤りを区別することとし，削除・挿入操作はコスト 2，置換の場合は表層・品詞ともに異なる要素に置き換える場合はコスト 2，表層と品詞いずれかが一致している要素に置き換える場合はコスト 1，表層・品詞ともに一致する場合はコスト 0 として計算を行った．

図 6 に損失関数の具体例を示す．図 6 の例の場合，すーっ（副詞），すごく（形容詞）の置き換えは表層，品詞ともに異なるためコストは 2 になる．なぁ（助詞・終），な（助詞・終）の場合は品詞が一致しているためコストは 1 となる．上記の文の場合，1 文の総コストは 5 となる．このように，すべての学習データの文に対して正解との差分コストを算出し，全体を総和した値を目的関数として最小化する．実験では，200 文の正規文の解析結果を正解デー

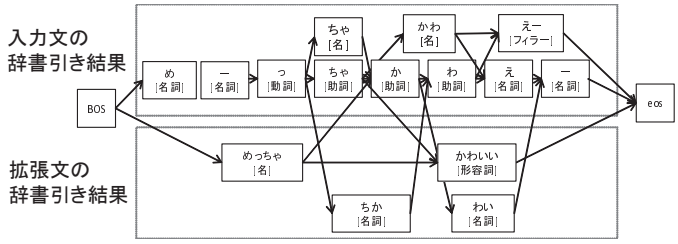


**図 7** ラティス合成の例

タとして利用した．最適化計算は powell 法 [21] を用いて行った．

### 3.5 デコーダ

ここまで推定したモデルを解析に組み込む方法を述べる．まず前処理として，「っ」と「ー」の連続に関しては 1 文字まで縮約させる．例えば，「ぜんっっっぜん」という表記であれば，「ぜんっぜん」まで縮約を行ってから候補展開を行う．また，母音の連続に関しては 3 文字まで縮約させる．これらの縮約ルールは，日本語の単語において縮約しても影響がない範囲を考慮した上で定めた．次に，正規-崩れ文字列アライメントモデルで抽出された変換テーブル（表記崩れパタン）を用いて，解析対象の入力文字列を拡張する．拡張の方法は従来法 [13] と同様で変換テーブルの崩れ文字列を入力文にマッチさせ，マッチした箇所について正規文字列と置き換えて展開した文字列に関しても辞書マッチを行う．これによって得られた，元文に対する辞書引き結果と拡張した文字列に対する辞書引き結果の双方を一つの形態素ラティスに合成して最適な単語・品詞列の組み合わせを求める．図 7 にラティス合成の例を示した．

入力文の拡張を行う際，すべての候補を展開すると形態素ラティスサイズが大きくなり計算コストに影響を及ぼしたため，今回は 3.3 節で示した展開候補信頼度モデルの結果を用いて識別ラベルが負例（誤った置き換え）と推定された候補に対しては足切りを行った．実験の結果，全ての候補を展開した場合の正解候補と誤り候補の展開数割合は約 1:150 であったが足切りによって正解候補の展開再現率約 90 %で誤り候補数を約 1/10 に圧縮した．

## 4. 実験

本研究では，Twitter 文を対象として提案手法を用いた解析結果と従来の解析器を用いた解析結果を比較することで提案手法の効果を評価した．以下にそれぞれの詳細を示す．

### 4.1 実験データ

モデル推定に使用したデータはブログデータから収集した崩れ表記と対応する正規表記のペアデータ 11150 ペアである．アライメント計算の結果，得られた文字列崩れパタン数は 2631 であった．表 3 に獲得された文字列崩れパタンの例を示す．「ない」や「たい」といった正規文字列に対しては多くのパタンが獲得できていることがわかる．また，ヴァ→バといった音の類似や「う→ぅ」（小文字化），「う→ー」（長音化）といった従来のルールベースで用いられている代表的なルールのほかにも，「う→ォ」，「い→っ」，「な→にゃ」といった低頻度な崩れ表記に関しても獲得可能であることが確認できた．

**表 3** 獲得された正規-崩れ文字列パタン例

| $c_{wk}$ | $c_{sk}$ | $p(c_{sk}\|c_{wk})$ | $c_{wk}$ | $c_{sk}$ | $p(c_{sk}\|c_{wk})$ |
|---|---|---|---|---|---|
| ない | ねぇ | 0.02923 | う | ウ | 0.00074 |
| ない | ねー | 0.02216 | う | ぉ | 0.00351 |
| ない | ねえ | 0.01414 | う | ぅ | 0.00388 |
| ない | ね | 0.00330 | う | ォ | 0.00037 |
| ない | ねェ | 0.00189 | う | ー | 0.00480 |
| ない | ね〜 | 0.00141 | う | お | 0.00055 |
| ない | あねぇ | 0.00094 | ヴァ | バ | 0.08333 |
| ない | ァねえ | 0.00047 | い | ぇ | 0.00016 |
| ない | ナィ | 0.00047 | い | っ | 0.00024 |
| ない | な | 0.00047 | い | え | 0.00016 |
| ない | にゃぃ | 0.00047 | い | ー | 0.00056 |
| ない | ナーィ | 0.00047 | い | ぃ | 0.00399 |
| ない | ナイ | 0.00047 | い | イ | 0.00016 |
| たい | てー | 0.00540 | の | ん | 0.00132 |
| たい | てぇ | 0.00360 | な | ナ | 0.00046 |
| たい | てぇ〜 | 0.00180 | な | にゃ | 0.00046 |
| たい | て〜 | 0.00180 | し | すぃ〜 | 0.00024 |
| たい | てえー | 0.00180 | し | ち | 0.00108 |
| たい | てぇー | 0.00180 | っ | つ | 0.00049 |
| たい | てえ | 0.00180 | っ | ッ | 0.00049 |
| いい | ぃぃ | 0.01217 | は | ゎ | 0.01235 |
| いい | いぃ | 0.01565 | は | ば | 0.00041 |
| いい | いー | 0.02957 | は | ハ | 0.00041 |
| しい | し | 0.01047 | null | ー | 0.23077 |
| しい | しー | 0.02618 | null | っ | 0.14919 |
| しい | し〜 | 0.01047 | null | ♪♪♪ | 0.00403 |
| しい | すぃ〜 | 0.00349 | null | ッ | 0.00962 |
| しい | しぃ | 0.02443 | null | い | 0.02016 |
| しい | ち | 0.00349 | null | う | 0.00837 |
| しい | すぃー | 0.00349 | null | 〜 | 0.20596 |

また，文字 n-gram モデルや単語表層 n-gram モデルの構築には，ブログの形態素正解ラベルなしデータを約 824 万を mecab を用いて自動解析した結果を使用した．言語モデルの構築に際し，SRILM[3] を使用した．

評価データは Twitter データから崩れ箇所を 1 箇所以上含む 120tweets を人手で抽出して作成した．120tweets のうち，今回対象とした崩れ箇所の総数は 168 箇所存在した．

### 4.2 評価方法と比較手法

本研究では，評価文の崩れ箇所に対し 1)mecab を用いて解析した結果，2)JUMAN([11], [12], [13], [15]) を用いて解析した結果，3) 提案手法で解析した結果の 3 つの解析結果を比較した．提案手法を用いた解析では，mecab と辞書体系等が同じであるため，mecab と提案手法の差分をみることで提案手法を用いることによる性能向上を直接比較する

**表 4** 解析結果の精度比較

| | 単語区切り+品詞正解 | | 誤り | |
|---|---|---|---|---|
| | 正規化あり | 正規化なし | 品詞のみ | 品詞+区切り |
| mecab | 0（0%） | 79（47.0%） | 26（14.9%） | 63（38.1%） |
| JUMAN | 56（33.3%） | 32（19.0%） | 24（14.3%） | 56（33.3%） |
| 提案法 | **79（47.0%）** | **54(32.1%)** | **13（7.7%）** | **22（13.1%）** |

**表 5** 提案手法による解析結果の改善例

| | 解析結果例 |
|---|---|
| 改善例 1 | 分かるょそれ〜 |
| mecab | 分かる (動詞)/ょそれ (名詞)/〜(名詞)/ |
| JUMAN | 分かる (動詞)/ょ (未定義)/それ (指示詞)/〜(未定義)/ |
| 提案法 | 分かる (動詞)/よ (助詞)/それ (名詞)/〜(名詞)/ |
| 改善例 2 | ぢゃーあたしもクレジットにしてみよっかな |
| mecab | ぢ (名詞)/ゃ (名詞)/ー (名詞)/あたし (名詞)/<br>も (助詞)/クレジット (名詞)/に (助詞)/し (動詞)/<br>て (助詞)/み (動詞)/よっ (動詞)/か (助詞)/な (助詞)/ |
| JUMAN | ぢ (未定義)/ゃ (未定義)/ー (未定義)/あたし (名詞)/<br>も (助詞)/クレジット (名詞)/に (助詞)/して (動詞)/<br>みよ (名詞)/っ (未定義)/か (助詞)/な (助詞)/ |
| 提案法 | じゃ (接続詞)/あたし (名詞)/も (助詞)/<br>クレジット (名詞)/に/し (動詞)/て (助詞)/<br>みよ (動詞)/う (助動詞)/か (助詞)/な (助詞)/ |
| 改善例 3 | 人が良ければたのすぃよ |
| mecab | 人 (名詞)/が (助詞)/良けれ (形容詞)/ば (助詞)/<br>た (助動詞)/の (助詞)/す (接頭辞)/ぃよ (名詞)/ |
| JUMAN | 人 (名詞)/が (助詞)/良ければ (形容詞)/<br>たの (連体詞)/すぃ (形容詞)/よ (助詞)/ |
| 提案法 | 人 (名詞)/が (助詞)/良けれ (形容詞)/ば (助詞)/<br>たのしい (形容詞)/よ (助詞)/ |

ことができる．また，崩れ語解析に対応している既存手法として今回は JUMAN を用いた．JUMAN は mecab とは品詞体系や連接コストの設定方法が異なるが，崩れ語に対してルールによるラティス拡張や辞書自動獲得などの処理を行っており mecab 等の解析器に比べ崩れ表記に頑健であることが知られている．このことから提案手法における崩れ語解析の性能を比較するための代表的な既存手法として JUMAN を用いた．mecab と juman は品詞体系等は異なるものの，新聞等の標準的な日本語ではどちらも同程度の性能を達成しているため，崩れ表記に対する頑健性の傾向を比較するという点では両者の結果を比較しても差支えないと考えられる

それぞれの解析の集計結果を表 4 に示す．ここで，単語区切り+品詞正解（正規化なし）とは，人手で該当箇所を崩れ表記と判定したが崩れた表記が辞書に入っており形態素解析が成功した事例を指す．たとえば，「まぁ/副詞（正規語：まあ/副詞）」，「なー/助詞（正規語：な/助詞）」，「よー/助詞（正規語：よ/助詞）」などが該当する．正規化ありの正解は，これらを正規化して正解した場合（まぁ→まあ）であり，正規化せずにそのままの表記で正解した場合（まぁ→まぁ）を正規化なしの正解として分類した．また，部分的に正解を含んでいる場合も今回は正規化なしの正解に含んだ．たとえば，「あ〜」という表記の正規化正解は「あ〜（フィラー）」であるが，「あ（フィラー）/〜（名詞・一般）」と解析された場合も誤りとは言えないため，正規化なしの正解に分類した．その他，誤った解析として「品詞のみ誤り（単語区切りは正解）」と「不正解（単語区切り，品詞ともに誤り）」に分類した．

また，誤りの「品詞のみ」とは，単語区切りは正解したが品詞が誤ったもの，「品詞+区切り] とは，単語区切りも品詞も両方誤ったものを指す．

### 4.3 実験結果

表 4 の集計結果より，mecab，JUMAN ともに今回対象とした崩れ表記において解析誤りが 50%程度存在することが分かる．正規化ありの正解に関しては，mecab が 0%（約半数は辞書対応している）なのに対し JUMAN は辞書の自動獲得や長音化，小文字化のルールを用いていることから約 33.3%を正規化語と紐づけて解析できている．これに対し提案法では，正規化ありの正解が約 47%と JUMAN に比べても高い値を示している．誤りに関しても，mecab,JUMAN が約 50%誤りが存在していたのに対し，提案手法では約 20%と大きく誤りを減少させている．なお，今回は崩れ表記箇所のみの集計を行っているが，今回の処理によって生じた崩れ表記以外の箇所での解析誤りの増加は 2 箇所のみであり，提案手法によって解析誤りをほとんど増加させずに解析の精度を向上させられることが明らかになった．表 5,6 にはそれぞれ解析結果の改善例と失敗例を示した．改善例では，小文字化や異表記（音の類似），発音の崩れについて正しく正規化して解析していることがわかる．失敗例に関してはパラメータ調整に関する誤り，候補列挙ができないことによる誤り，それ以外の誤りが存在した．パラメータ調整や候補列挙に関しては，素性追加や正解データの拡充によって改善することが可能であると考えられる．それ以外の誤りについて以下に整理する．

- **失敗例 1:中間表現 (おもにひらがな表記) が辞書にない場合**
  失敗例 1 では，「さいこー」という崩れ表記に対し，「さいこう」というひらがな正規表記を列挙することはできたが，「さいこう」というエントリが辞書になかったため解析失敗している例である．この場合は，「最高」という漢字まで正規化を行うか「さいこう」というひらがなエントリを追加して解析を行うことが必要になる．この場合は JUMAN で「さいこう」というひらがなエントリが含まれており解析に成功しているため，「さいこう」というひらがなエントリを追加することで解析が成功する可能性は高くなると考えられる．

ただし，一般的にはひらがなエントリをむやみに辞書に追加すると解析悪化が増加することが考えられるため，適切なコスト設定についての検討が必要になる．ひらがな化現象の正規化については工藤ら [14] の研究で生成的言語モデルによるパラメータ推定の検討がなされているが，提案手法の枠組みに組み込むための手法検討が必要となる．

- **失敗例 2:ひらがな表記＋助詞抜けの場合**
  失敗例 2 では，本来の意味は「頭に入らなくない？」という意味であるが，「頭はい（居）なくない？」という異なった意味に解析されてしまっている（提案手法）．mecab や JUMAN でも，「頭はいら（要ら）なくない？」といった異なった意味に解析されてしまっている．この場合，「はいる」というひらがな表記の動詞は辞書に存在しているため，ひらがな化そのものが解析誤りの原因ではなく，助詞が抜けている点と「はいる」の「は」が助詞にもなりうるという点が複合して生じた誤りである．このような場合，「頭はいらないくない？」といったひらがな正規表記を列挙できたとしても「は」が助詞として解析されるため正しく解析できない．このほかにも，仮にひらがな正規表記が列挙できてひらがな正規表記が辞書に存在していたとしても連接コストの影響で異なる解が選択されてしまう場合がある．このような問題に対しては，(1) 品詞連接コストを調整する，(2)「はいる」を「入る」と漢字まで正規化する，(3) 助詞を挿入する（助詞抜けの場合），などの解決策が考えられるが，いずれも難しい課題であると考えられる．
- **失敗例 3:カタカナ表記の場合**
  失敗例 3 では，「ダケド」という崩れ表記に対し，「だけど」というひらがな文字列を列挙すれば正しく解析が可能になる．本研究の手法では，正解データから学習した個々の文字列単位での変換テーブルを使用しているため一部のカタカナ化現象には対応できているが (例：ます→マス，です→デス，な→ナ)，あらゆるカタカナ化現象には対応できない．この点については正解データを増やしてもカバー率には限界があるため，「カタカナ化」といった文字種レベルの変換ルールをモデルに組み込む必要がある．ただし，「カタカナ化」という現象をモデルに組み込み解析文に出現する全てのカタカナ表記に対してひらがな表記も列挙する，などを行った場合，辞書に存在しない固有名詞のカタカナが不要にひらがな化されて解析誤りが増えるなどの可能性も考えられる．この点について，詳細な検討が必要になる．

## 5. まとめと今後の課題

本研究では，崩れ表記の正規化に基づく形態素解析の基礎的検討を行い，正規表記と崩れ表記の正解ペアデータから抽出した表記崩れパタンを解析器に組み込むことで解析可能な崩れ表記のカバー率を高められることを示した．今後の課題は，(1) パラメータ推定の際のコスト関数の設定（本来正規化候補（正解）は複数存在するが，今回は正解を一つに限定している），(2) 素性関数の追加による精度向上，(3) 対象範囲の拡大によるカバー率の向上，(4) 正規-崩れ表記ペアデータの作成方法である．(1) については，異なる損失関数を設定した場合や，複数の正解を許した場合についての検証を行う予定である．(2) に関しては，文字 n-gram などの文字列情報を直接全体の目的関数の素性として組み込む方法や，読み n-gram，音素などの素性を追加した方法の検討が必要だと考えている．(3) については，ひらがな化やカタカナ化を提案モデルに追加することを検討している．これらの現象に関しては文字種レベルでの変換ルールに関する重みをモデルに導入することで，本稿で述べた枠組みの中で統一的に扱えると考えられる．最後に (4) についての検討も重要である．今回は崩れ表記と正規表記のアライメントから文字列崩れパタンの抽出を行ったが，正解データを大量に作成するのは高コストであり何らかの方法で自動的に正解ペアデータを作成するなどの効率的な方法が必要となるだろう．この点については，池田ら [19], [20] などの手法も参考に，検討を進めていく予定である．

**表 6** 提案手法による解析結果の失敗例

| | 解析結果例 |
|---|---|
| 失敗例 1 | 二代目と DOBER さいこー！ |
| mecab | 二 (名詞)/代目 (名詞)/と (助詞)/DOBER(名詞)/さ (名詞)/いこ (動詞)/ー (名詞)/！ (記号)/ |
| JUMAN (正解) | 二 (数詞)/代 (接尾辞)/目 (接尾辞)/と (助詞)/DOBER(未定義)/さいこう (副詞)/！ (記号)/ |
| 提案法 | 二 (名詞)/代目 (名詞)/と (助詞)/DOBER(名詞)/さ (名詞)/いこ (動詞)/う (助動詞)/！ (記号)/ |
| 失敗例 2 | 頭はいんなくね？ |
| mecab | 頭 (名詞)/は (助詞)/いん (動詞)/なく (助動詞)/ね (助詞)/？ (記号)/ |
| JUMAN | 頭 (名詞)/は (助詞)/い (動詞)/ん (助動詞)/なく (助動詞)/ね (助詞)/？ (記号)/ |
| 提案法 | 頭 (名詞)/は (助詞)/い (動詞)/なく (助動詞)/ね (助詞)/？ (記号)/ |
| 失敗例 3 | 今更ダケド。 |
| mecab | 今更 (副詞)/ダケド (名詞)/。(記号)/ |
| JUMAN | 今更 (副詞)/ダケド (未定義)/。(記号)/ |
| 提案法 | 今更 (副詞)/ダケド (名詞)/。(記号)/ |

### 参考文献

[1] Kudo,T., Japanese Morphological Analyzer, http://mecab.googlecode.com/svn/trunk/mecab/doc/index.html
[2] Joachims,T., http://svmlight.joachims.org/
[3] Stolcke, A. SRILM-an extensible language modeling toolkit. In Proc. INTERSPEECH. 2002.

[4] Brill,E., Moore,R.C., An improved error model for noisy channel spelling. In Proc. ACL, pp. 286—293,2000.
[5] Li.H., Min.Z., Jian,S., A joint source-channel model for machine transliteration. In Proc. of ACL pp.159—166, 2004.
[6] Aw,a., Zhang,M., Xiao,J., Su,J., A phrase-based Stastical Model for SMS Text Normalization, In Proc. of Coling/ACL, pp.33-40,2006
[7] Machery,W., Och,F.J., Thayer,I., Uszkoreit,J., Lattice-based Minimum Error Rate Training for statistical Machine Translation, In Proc. of EMNLP, pp.725-734, 2008
[8] Han,B., Baldwin,T., Lexical Normalization of short Text Messages, Makn Sens a #twitter, In Proc. of ACL, pp.368-378, 2011
[9] Ling,W., Xiang,G., Dyer,C., Black,A., Transcoso,I., Microblogs as Parallel Corpora, In Proc. of ACL, pp.176-186, 2013
[10] Ling,W., Dyer,C., Black,A., Transcoso,I. Paraphrasing 4 Microblog Normalization, In Proc. of EMNLP, pp.73-84, 2013
[11] 笹野遼平, 黒橋禎夫, 形態素解析における連濁および反復形オノマトペの自動認識, 自然言語処理学会年次大会講演集, 2007
[12] 村脇 有吾, 黒橋 禎夫, 形態論的制約を用いた未知語の自動獲得, 自然言語処理学会年次大会講演集, 2008
[13] 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
[14] 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
[15] 笹野遼平, 鍛冶伸裕, 新しい語・崩れた表記の処理, 情報処理, vol.53, No.3, 2012
[16] 萩原正人, 関根聡, 混合ディリクレ分布を用いた潜在クラス翻字生成モデル, 自然言語処理学会年次大会講演集, 2012
[17] 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
[18] 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, 人工知能学会全国大会講演集, 2013
[19] 池田和史, 柳原正, 松本一則, 滝嶋康弘, ブログ的表記を正規化するためのルール自動生成方式の提案と評価, 日本データベース学会論文誌, vol, 8, no.1, 2009
[20] 池田和史, 柳原正, 松本一則, 滝嶋康弘, くだけた表現を高精度に解析するための正規化ルール自動生成手法, 情報処理学会論文誌, データベース, vol.3, no.3, pp.68-77, 2010
[21] William H. Press,Saul A. Numerical Recipes in C, 技術評論社,1993