# Morphological Analysis for Japanese Noisy Text Based on Character-level and Word-level Normalization

**SAITO Itsumi, SADAMITSU Kugatsu, ASANO Hisako and MATSUO Yoshihiro**
NTT Media Intelligence Laboratories
{saito.itsumi, sadamitsu.kugatsu,
asano.hisako, matsuo.yoshihiro}@lab.ntt.co.jp

## Abstract

Social media texts are often written in a non-standard style and include many lexical variants such as insertions, phonetic substitutions, abbreviations that mimic spoken language. The normalization of such a variety of non-standard tokens is one promising solution for handling noisy text. A normalization task is very difficult to conduct in Japanese morphological analysis because there are no explicit boundaries between words. To address this issue, in this paper we propose a novel method for normalizing and morphologically analyzing Japanese noisy text. We generate both character-level and word-level normalization candidates and use discriminative methods to formulate a cost function. Experimental results show that the proposed method achieves acceptable levels in both accuracy and recall for word segmentation, POS tagging, and normalization. These levels exceed those achieved with the conventional rule-based system.

## 1 Introduction

Social media texts attract a lot of attention in the fields of information extraction and text mining. Although texts of this type contain a lot of information, such as one's reputation or emotions, they often contain non-standard tokens (lexical variants) that are considered out-of-Vocabulary (OOV) terms. We define an OOV as a word that does not exist in the dictionary. Texts in micro-blogging services such as Twitter are particularly apt to contain words written in a non-standard style, e.g., by lengthening them ("goooood" for "good") or abbreviating them ("thinkin' " for "thinking"). This is also seen in the Japanese language, which has standard word forms and variants of them that are often used in social media texts. To take one word as an example, the standard form is おいしい (*oishii*, "It is delicious") and its variants include おいしい いいい(*oishiiii*), おいし ぃ(*oishii*), and おいし ー(*oishii*), where the underlined characters are the differences from the standard form. Such non-standard tokens often degrade the accuracy of existing language processing systems, which are trained using a clean corpus.

Almost all text normalization tasks for languages other than Japanese (e.g., English), aim to replace the non-standard tokens that are explicitly segmented using the context-appropriate standard words (Han et al. (2012), Han and Baldwin (2011), Hassan and Menezes (2013), Li and Liu (2012), Liu et al. (2012), Liu et al. (2011), Pennell and Liu (2011), Cook and Stevenson (2009), Aw et al. (2006)). On the other hand, the problem is more complicated in Japanese morphological analysis because Japanese words are not segmented by explicit delimiters. In traditional Japanese morphological analysis, word segmentation and part-of-speech (POS) tagging are simultaneously estimated. Therefore, we have to simultaneously analyze normalization, word segmentation, and POS tagging to estimate the normalized form using the context information. For example, the input パンケーキぉぃしーぃ(*pan-keiki oishiiii*, "This pancake tastes good") written in the standard form is パンケーキおいしい (*pan-keiki oishii*). The result obtained with the conventional Japanese morphological analyzer MeCab (Kudo (2005)) for this input is **パンケーキ (pancake, noun)**/**おいし (unk)**/**ー (unk)**/**ぃ(unk)**/, where slashes indicate the word segmentations and "unk" means an unknown word. As this result shows, Japanese morphological analyzers often fail to





*Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1773–1782, Dublin, Ireland, August 23-29 2014.

correctly estimate the word segmentation if there are unknown words, so the pipeline method (e.g., first estimating the word segmentations and then estimating the normalization forms) is unsuitable.

Moreover, Japanese has several writing scripts, the main ones being Kanji, Hiragana, and Katakana. Each word has its own formal written script (e.g., 教科書 (*kyoukasyo*, "textbook") as formally written in Kanji), but in noisy text, there are many words that are intentionally written in a different script (e.g., きょうかしょ (*kyoukasyo*, "textbook") is the Hiragana form of 教科書). These tokens written in different script also degrade the performance of existing systems because dictionaries basically include only the standard script. Unlike the character-level variation we described above, this type of variation occurs on a word—level one. Therefore, there are both character-level and word-level non-standard tokens in Japanese informal written text. Several normalization approaches have been applied to Japanese text. Sasano et al. (2013) and Oka et al. (2011) introduced simple character level derivational rules for Japanese morphological analysis that are used to normalize specific patterns of non-standard tokens, such as for word lengthening and lower-case substitution. Although these approaches handle Japanese noisy text fairly effectively, they can handle only limited kinds of non-standard tokens.

We propose a novel method of normalization in this study that can handle both character- and word-level lexical variations in one model. Since it automatically extracts character-level transformation patterns in character-level normalization, it can handle many types of character-level transformations. It uses two steps (character- and word-level) to generate normalization candidates, and then formulates a cost function of the word sequences as a discriminative model. The contributions this research makes can be summarized by citing three points. First, the proposed system can analyze a wider variety of non-standard token patterns than the conventional system by using our two-step normalization candidate generation algorithms. Second, it can largely improve the accuracy of Japanese morphological analysis for non-standard written text by simultaneously performing the normalization and morphological analyses. Third, it can automatically extract character alignments and in so doing reduces the cost of manually creating many types of transformation patterns. The rest of this paper is organized as follows. Section 2 describes the background to our research, including Japanese traditional morphological analysis, related work, and data collection methods. Section 3 introduces the proposed approach, which includes lattice generation and formulation, as a discriminative model. Section 4 discusses experiments we performed and our analyses of the experimental results. Section 5 concludes the paper with a brief summary and a mention of future work.

## 2 Background

### 2.1 Japanese Morphological Analysis

Many approaches to joint word segmentation and POS tagging including Japanese Morphological analysis can be interpreted as re-ranking while using a word lattice (Kaji and Kitsuregawa (2013)). There are two points to consider in the analysis procedure: how to generate the word lattice and how to formulate the cost of each path. In Japanese morphological analysis, the dictionary-based approach has been widely used to generate the word lattice (Kudo et al. (2004), Kurohashi et al. (1994)). In a traditional approach, an optimal path is sought by using the sum of the two types of costs for the path: the cost for a candidate word that reflects the word's occurrence probability, and the cost for a pair of adjacent POS that reflects the probability of an adjacent occurrence of the pair (Kudo et al. (2004), Kurohashi et al. (1994)). A greater cost means less probability. The Viterbi algorithm is usually used for finding the optimal path.

### 2.2 Related Work

Several studies have been conducted on Japanese morphological analysis in the normalized form. The approach proposed by Sasano et al. (2013) aims to develop heuristics to flexibly search by using a simple, manually created derivational rule. Their system generates normalized character sequence based on the derivational rule, and adding new nodes that are generated from normalized character sequence when generating the word lattice using dictionary lookup. Figure 1 presents an example of this approach. If the non-standard written sentence すーごく楽しぃ (*suugoku tanoshii*, "It is such fun") is input, the

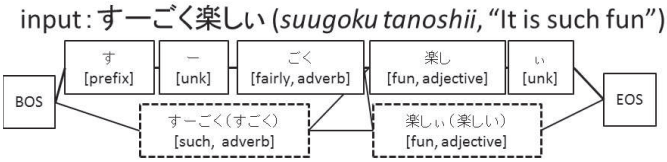


Figure 1: Example of Japanese morphological analysis and normalization

| type | non-standard form | standard form |
|---|---|---|
| (1) Insertion | ありがと ーー う (*arigatoou*) | ありがとう (*arigatou*, "Thank you") |
| (2) Deletion | さむ _ (*samu*) | さむい (*samui*, "cold") |
| (3) Substitution with phonetic variation | かわ ええ (*kawaee*) | かわいい (*kawaii*, "cute") |
| (4) Substitution with lowercases and uppercases | ぁりがと ぅ (*arigatou*) | ありがとう (*arigatou*, "Thank you") |
| (5) Hiragana substitution | あいでぃー (*aidei*) | ID (*aidei*, "identification card") |
| (6) Katakana substitution | アリガトウ (*arigatou*) | ありがとう (*arigatou*, "Thank you") |
| (7) Any combination of (1) to (6) | かぅんたー (*kaunta*) | カウンター (*kaunta*, "counter") |
| | あ っ つ ぃ (*attsui*) | あつい (*atsui*, "hot") |

Table 1: Types of non-standard tokens and examples of annotated data

traditional dictionary-based system generates Nodes that are described using solid lines, as shown in Fig. 1. Since "すーごく" (*suugoku*, "such") and "楽しぃ" (*tanoshii*, "fun") are OOVs, the traditional system cannot generate the correct word segments or POS tags. However, their system generates additional nodes for the OOVs, shown as broken line rectangles in Fig. 1. In this case, derivational rules that substitute "ー" with "null" and "ぃ" (*i*) with "い" (*i*) are used and the system can generate the standard forms "すごく" (*sugoku*, "such") and "楽しい" (*tanoshii*, "fun") and their POS tags. If we can generate sufficiently appropriate rules, these approaches seem to be effective. However, there are many types of derivational patterns in SNS text and it is difficult to cover all of them by hand. Moreover, it becomes a serious problem how to set the path cost for appropriately re-ranking the word lattice when the number of candidates increases. Our approach is also based on the dictionary-based approach, however, our approach is significantly dissimilar from their approach in two ways. First, we automatically generate derivational patterns (we call them transformation tables) based on the character-level alignment between non-standard tokens and their standard forms. Compared to generating the rules by hand, our approach can generate broad coverage rules. Second, we use discriminative methods to formulate a cost function. Jiang et al. (2008), Kaji and Kitsuregawa (2013) introduce several features to appropriately re-rank the added nodes. This enables our system to perform well even when the number of candidates increases.

On the other hand, several studies have applied a statistical approach. For example, Sasaki et al. (2013) proposed a character-level sequential labeling method for normalization. However, it handles only one-to-one character transformations and does not take the word-level context into account. The proposed method can handle many-to-many character transformations and takes word-level context into account, so the scope for handling non-standard tokens is different. Many studies have been done on text normalization for English; for example Han and Baldwin (2011) classifies whether or not OOVs are non-standard tokens and estimates standard forms on the basis of contextual, string, and phonetic similarities. In these studies it was assumed that clear word segmentations existed. However, since Japanese is an unsegmented language the normalization problem needs to be treated as a joint normalization, word segmentation, and POS tagging problem.

### 2.3 Data Collection and Analysis of Non-standard Tokens

In previous studies (Hassan and Menezes (2013), Ling et al. (2013), Liu et al. (2011)), the researchers proposed unsupervised ways to extract non-standard tokens and their standard forms. For Japanese text, however, it is very difficult to extract word pairs in an unsupervised way because there is no clear word segmentation. To address this problem we first extracted non-standard tokens from Twitter text and blog

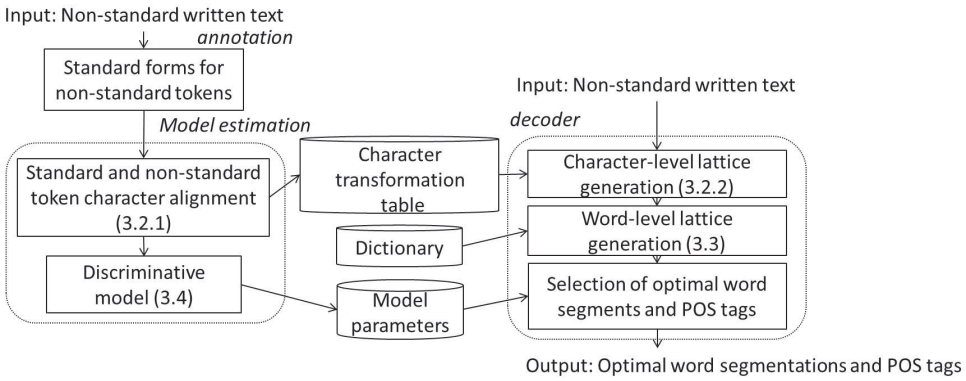


Figure 2: Structure of proposed system

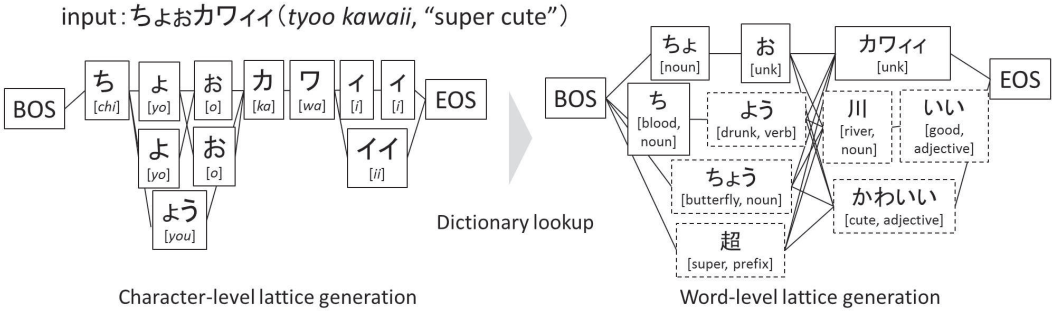


Figure 3: Example of candidate generation

text and manually annotated their standard (dictionary) forms. In total, we annotated 4808 tweets and 8023 blog text sentences. Table 1 lists the types of non-standard tokens that we targeted in this study and examples of the annotated data. Types (1), (2), (3) and (4) are similar to English transform patterns. Types (5) and (6) are distinctive patterns in Japanese. As previously mentioned Japanese has several kinds of scripts, the main ones being Kanji, Hiragana, and Katakana. These scripts can be used to write the same word in several ways. For example, the dictionary entry 先生 (*sensei*, "teacher") can also be written in Hiragana form せんせい (*sensei*) or Katakana form センセイ (*sensei*). Most words are normally written in the standard form, but in informal written text (e.g., Twitter text), these same words are often written in a non-standard form. In examining Twitter data for such non-standard tokens, we found that 55.0% of them were types (1) to (3) in Table 1, 4.5% were type (4), 20.1% were types (5) to (6), 2.7% were type (7), and the rest did not fall under any of these types since they were the result of dialects, typos, and other factors. In other words, a large majority of the non-standard tokens fell under types (1) to (7). We excluded those that did not as targets in this study because our proposed method cannot easily handle them. Types (1) to (4) occur at character-level and so can be learned from character-level alignment, but types (5) to (6) occur at word-level and it is inefficient to learn them on a character—level basis. Accordingly, we considered generating candidates and features on two levels: character-level and word-level.

## 3 Proposed Method

### 3.1 Overview of Proposed System

We showed the structure of the proposed system in Fig. 2. Our approach adds possible normalization candidates to a word lattice and finds the best sequence using a Viterbi decoder based on a discriminative model. We introduced several features that can be used to appropriately evaluate the confidence of the added nodes as normalization candidates. We generate normalization candidates as indicated in Fig. 3.

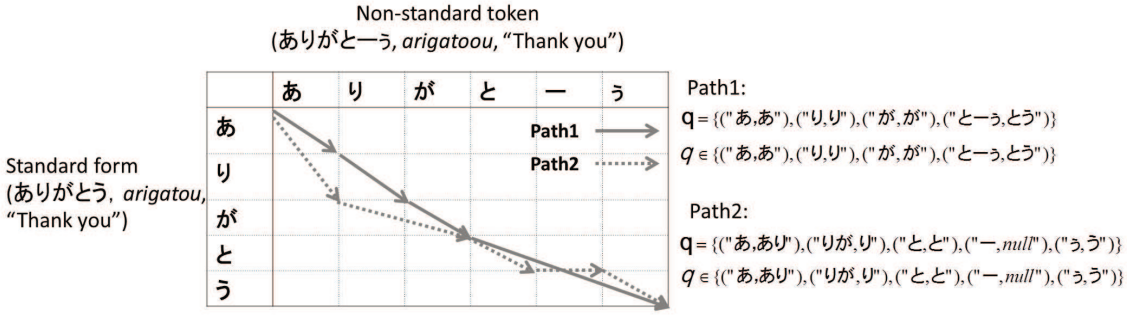


Figure 4: Example of character alignment

We describe the details in the following section.

### 3.2 Character-level Lattice

#### 3.2.1 Character Alignment between Non-standard Tokens and Their Normalized Forms

We have to create a character-level transformation table to generate the character-level lattice. We used the joint multigram model proposed by Sittichai et al. (2007) to create the transformation table because this model can handle many-to-many character alignments between two character sequences. In observing non-standard tokens and their standard forms, we find there are not only one-to-one character transformations but also many-to-many character transformations. Furthermore, unlike in translation, there is no character reordering so the problems that arise are similar to those in transliteration. Accordingly, we adopted a joint multigram model that is widely used for transliteration problems. The optimal alignment can be formulated as $\hat{\mathbf{q}} = \arg\max_{\mathbf{q} \in K_d} \prod_{q \in \mathbf{q}} p(q)$ , where $d$ is a pair of non-standard tokens and its standard form (e.g., $d$ is ありがとーぅ (*arigatoou*), ありがとう (*arigatou*). Here, $q$ is a partial character alignment in $d$ (e.g., $q$ is "とーぅ, とう"), $\mathbf{q}$ is the character alignment $q$ set in $d$ (e.g., $\mathbf{q}$ of path 1 in Fig. 4 is {("あ, あ"), ("り, り"), ("が, が"), ("とーぅ, とう")}. $K_d$ is the possible character alignment sequence candidates generated from $d$. We generate n-best optimal path for $K_d$ in this study. The maximum likelihood training can be performed using the EM algorithm derivated in Bisani and Ney (2008) and Kubo et al. (2011) to estimate $p(q)$. $p(q)$ can be formulated as follow:

$$p(q) = \gamma_q / \sum_{q \in Q} \gamma_q \tag{1}$$

$$\gamma_q = \sum_{d \in D} \sum_{\mathbf{q} \in K_d} p(\mathbf{q}) n_q(\mathbf{q}) = \sum_{d \in D} \sum_{\mathbf{q} \in K_d} \frac{\prod_{q \in \mathbf{q}} \bar{p}(q)}{\sum_{\mathbf{q} \in K_d q \in \mathbf{q}} \prod \bar{p}(q)} n_q(\mathbf{q}),$$

and where $D$ is the number of the $d$ pair, $Q$ is the set of $q$, and $n_q(\mathbf{q})$ is the count of $q$ that occurred in $\mathbf{q}$. In our system, we allow for standard form deletions (i.e., mapping of a non-standard character to a null standard character) but not non-standard token deletions. Since we use this alignment as the transformation table when generating a character-level lattice, the lattice size becomes unnecessarily large if we allow for non-standard form deletions. In the calculation step of the EM algorithm, we calculate the expectation (partial counts) $\gamma_q$ of each alignment in the E-step, calculate the joint probability $p(q)$ that maximizes the likelihood function in the M-step as described before, and repeat these steps until convergence occurs. $\bar{p}(q)$ indicates the result of $p(q)$ calculated in the previous step over the iteration. When generating the character-level lattice, we used alignments that were expected to exceed a predefined threshold. We used $\gamma_q$ ($q = (c_t, c_v)$) and $r(c_t, c_v)$ as thereshold, where $c_t$ and $c_v$ are the partial character sequence of non-standard token and it's standard form respectively. $r(c_t, c_v)$ is calculated by $r(c_t, c_v) = \gamma_q / n_{c_v}$., where $n_{c_v}$ is the number of occurrences of $c_v$ in the training data. We set the threshold $\gamma_{q\_thres} = 0.5$ , and $r(c_t, c_v)_{thres} = 0.0001$ in this study. We also used $r(c_t, c_v)$ as a feature of cost

function in subsection. 3.4.2. When calculating initial value, we set $p(c_t, c_v)$ high if the character $c_t$ and $c_v$ are the same character and the length of each character is 1. We also give the limitation that a Kanji character does not change to a different character and is aligned with same character in the calculation step of the character alignment.

#### 3.2.2 Generation of Character-level Lattice Based on Transformation Table

First, repetitions of more than one letter of "ー", "〜", "-", and "っ" are reduced back to one letter (e.g., ありがと ーーー う (*arigatooooou*, "Thank you") is reduced to ありがと ー う (*arigatoou*)) for the input text. In addition, repetitions of more than three letters other than "ー", "〜", "-", and "っ" are reduced back to three letters (e.g., うれし いいいいい (*uresiiiiii*, "I'm happy") is reduced back to うれし いいい (*uresiii*)). These preprocessing rules are inspired by Han and Baldwin (2011) and determined by taking the Japanese characteristics into consideration. We also used these rules when we estimated the alignments of the non-standard tokens and their standard forms. Next, we generate the character-level normalization candidates if they match the key transformation table in the input text. For example, if the transformation table contains ($q$, log$p(q)$)= ("よお (*yoo*), よう (*you*)", -8.39), ("お (*o*), お (*o*)", -7.56), and the input text includes the character sequence "ちよお" (*tyoo*), we generate a new sequence "ちょう" (*tyou*) and "ちょお" (*tyoo*). In other words, we add new nodes "ょう" (*you*) and "お" (*o*) in the position of "よお" (*yoo*) and "お" (*o*), respectively (see Fig. 3).

### 3.3 Generation of Word-level Lattice

We generate the word lattice based on the generated character-level lattice using dictionary lookup. We exploit dictionary lookup by using the possible character sequence of the character-level lattice while the traditional approach exploits it by using only the input character sequence. For example, we exploit dictionary lookup for character sequences such as "ちよおカワィィ" (*tyoo kawaii*) and "ちょう カワィィ" (*tyou kawaii*) and "ち よおカワィィ" (*chiyou kawaii*) and "ちよおカワ イイ" (*tyoo kawaii*) (see Fig. 3)

Furthermore, we use the phonetic information of the dictionary to generate the normalization candidates for Hiragana and Katakana substitution. For example, assume "超" (*tyou*, "super") and "かわいい" (*kawaii*, "cute") are the dictionary words. Then, if the input text contains the character sequences "ちょう" (*tyo*) (which is written in Hiragana) and "カワイイ" (*kawaii*) (which is written in Katakana), we add "超" (*tyo*, "super") and "かわいい" (*kawaii*, "cute") to the word lattice as the normalization candidates since the two character sequences are pronounced identically. By using this two-step algorithm, we can handle any combinational derivational patterns, such as Katakana substitutions or substitutions of lowercases like "カワィィ" (*kawaii*) → "カワイイ" (*kawaii*) → "かわいい" (*kawaii*, "cute") (see Fig. 3). Note that we filtered candidates on the basis of a predefined threshold to prevent the generation of unnecessary candidates. The threshold was defined on the basis of the character sequence cost of normalization, which is described in subsection 3.4.2. Furthermore, we limited the number of character transformations to two per word.

### 3.4 Decoder

#### 3.4.1 Objective Function

The decoder selects the optimal sequence $\hat{y}$ from $L(s)$ when given the candidate set $L(s)$ for sentence $s$. This is formulated as $\hat{y} = \arg\min_{y \in L(s)} \mathbf{w} \cdot \mathbf{f}(y)$ (Jiang et al. (2008), Kaji and Kitsuregawa (2013)), where $\hat{y}$ is the optimal path, $L(s)$ is the lattice created for sentence $s$, and $\mathbf{w} \cdot \mathbf{f}(y)$ is the dot product between weight vector $\mathbf{w}$ and feature vector $\mathbf{f}(y)$. The optimal path is selected according to the $\mathbf{w} \cdot \mathbf{f}(y)$ value.

#### 3.4.2 Features

The proposed lattice generation algorithm generates a lattice larger than that generated in traditional dictionary-based lattice generation. Therefore, we need to introduce an appropriate normalization cost into the objective function. We listed the features we used in Table 2. Let $w_i$ be the $i$th word candidate and $p_i$ be the POS tag of $w_i$. $p_{i-1}$ and $w_{i-1}$ are adjacent POS tag and word respectively. We also used the word unigram cost $f_{w_i p_i}$, the cost for a pair of adjacent POS $f_{p_{i-1}, p_i}$ that are quoted from MeCab (Kudo,

| Name | Feature |
|---|---|
| Word unigram cost | $f_{w_i p_i}$ |
| POS bi-gram cost | $f_{p_{i-1},p_i}$ |
| Word-POS bi-gram cost | $-\log p_{w_{i-1}p_{i-1},w_i p_i}$ |
| Character sequence cost | $\log(p'_s/p'_{t_i})$ |
| | where, $p'_x = p_x^{1/length(x)}$, $p_x = \prod_{j=1}^{n} p(c_j \mid c_{j-5}^{j-1})$, $x \in \{s, t_i\}$ |
| Character transformation cost | $\phi_{trans_i} \cdot (-\log r(c_t, c_v))$ |
| Hiragana substitution cost | $\phi_{h_i} \cdot f_{w_i p_i}$ |
| Katakana substitution cost | $\phi_{k_i} \cdot f_{w_i p_i}$ |

Table 2: Feature list of the decoder. $\phi_{trans_i}$ is 1 if $w_i$ is generated by character transformation, otherwise 0. $\phi_{h_i}$ is 1 if $w_i$ is generated by Hiragana substitution, otherwise 0. $\phi_{k_i}$ is 1 if $w_i$ is generated by Katakana substitution, otherwise 0.

2005), and five additional types of costs. These are the word-pos bi-gram cost $-\log p_{w_{i-1}p_{i-1},w_i p_i}$ of a blog corpus; the character transformation cost $\phi_{trans_i} \cdot (-\log r(c_t, c_v))$, which is calculated in Section3.2, for nodes generated by character transformation; the Hiragana substitution cost $\phi_{h_i} \cdot f_{w_i p_i}$ for nodes generated by Hiragana substitution; the Katakana substitution cost $\phi_{k_i} \cdot f_{w_i p_i}$ for nodes generated by Katakana substitution; and the character sequence cost $\log(p'_s/p'_{t_i})$ for all the normalized nodes. The character sequence cost reflects the character sequence probability of the normalization candidates. Here, $s$ and $t_i$ are input string and transformed string respectively. (e.g., In Fig. 3, for the normalized node "かわいい" (cute, adjective), $s$ is "ちよおカワイイ" and $t_i$ is "ちよおかわいい"). Then $p_s$ and $p_{t_i}$ are calculated by using the character 5-gram of a blog corpus, which is formulated by $p_s = p(c_1 \cdots c_n) = \prod_{j=1}^{n} p(c_j \mid c_{j-5}^{j-1})$, where $c_j$ is the $j$ th character of character sequence $s$. $p'_{t_i}$ and $p'_s$ are normalized by using the length of each string $s$ and $t_i$ as $p'_{t_i} = p_{t_i}^{1/length(t_i)}$. We set the threshold $(p'_s/p'_{t_i})_{thres} = 1.5$ for generating a Hiragana or Katakana normalization candidate in this study. Since all those features can be factorized, the optimal path is searched for by using the Viterbi algorithm.

#### 3.4.3 Training

We formulated the objective function for tuning weights $\mathbf{w}$ by using Eq. 2. The weights $\mathbf{w}$ are trained by using the minimum error rate training (MERT) Machery et al. (2008). We defined the error function as the differences between the reference word segmentations and the POS tags of the reference sequence $y_{ref}$ and the system output $\arg\min_{y \in L(s)} \mathbf{w} \cdot \mathbf{f}(y)$.

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w} \in \mathbf{W}} \sum_{i=1}^{N} \mathrm{error}(y_{ref}, \arg\min_{y \in L(s)} \mathbf{w} \cdot \mathbf{f}(y)) \quad (2)$$

## 4 Experiments

### 4.1 Dataset and Estimated Transformation Table

We conducted experiments to confirm the effectiveness of the proposed method, in which we annotated corpora of a Japanese blog and Twitter. The Twitter corpus was split into three parts: the training, development, and test sets. The test data comprised 300 tweets, development data comprised 500 sentences and the training data comprised 4208 tweets. We randomly selected the test data which contained at least one non-standard token. The test data comprised 4635 words, 403 words of them are non-standard token and are orthographically transformed into normalized form and POS tags. The blog corpus comprised 8023 sentences and all of them were used as training data. Training data was used for extracting character transformation table and development data was used for estimating parameters of discriminative model. We used the IPA dictionary provided by MeCab to generate the word-level lattice and extracted the dictionary-based features. We itemized the estimated character transformation patterns in Table 3. There were 5228 transformation patterns that were learned from the training data and we used 3268 of them, which meets the predefined condition. The learned patterns cover most of the previously pro-

| non-standard character $c_t$ | standard character $c_v$ | $\log p(q)$ | non-standard character $c_t$ | standard character $c_v$ | $\log p(q)$ |
|---|---|---|---|---|---|
| ー | null | -4.233 | っす (*ssu*) | です (*desu*) | -5.999 |
| まぁ(*maa*) | まあ (*maa*) | -5.059 | どー (*doo*) | どう (*dou*) | -6.210 |
| しょ(*syo*) | しょう (*syou*) | -5.211 | ねー (*nee*) | ない (*nai*) | -6.232 |
| だろ (*daro*) | だろう (*darou*) | -5.570 | りゃ(*rya*) | れは (*reha*) | -6.492 |
| っ(*ttsu*) | null | -5.648 | てん (*ten*) | てる (*teru*) | -6.633 |
| んと (*nto*) | んとう (*ntou*) | -5.769 | ゆう (*yuu*) | いう (*iu*) | -6.660 |
| わ(*wa*) | は (*wa*) | -5.924 | なん (*nan*) | なの (*nano*) | -6.706 |

Table 3: Example of character-level transformation table

posed rules. In addition, our method can learn more of the variational patterns that are difficult to create manually.

### 4.2 Baseline and Evaluation Metrics

We compared the five methods listed in Table 4 in our experiments. Traditional means that which generates no normalization candidates and only uses the word cost and the cost for a pair of adjacent POS, so we can consider it as a traditional Japanese morphological analysis. We compared three baselines, Baseline1, Baseline2 and Baseline3. Baseline1 is the conventional rule-based method (considering insertion of long sound symbols and lowercases, and substitution with long sound symbols and lowercases), which was proposed by Sasano et al. (2013). In Baseline2, 3, and Proposed, we basically use the proposed discriminative model and features, but there are several differences. Baseline2 only generates character-level normalization candidates. Baseline3 uses our two-step normalization candidate generation algorithms, but the character transformation cost of all the normalization candidates that are generated by character normalization is the same. Proposed generates the character-level and Hiragana and Katakana normalization candidates and use all features we proposed.

We evaluated each method on the basis of precision and recall and the F-value for the overall system accuracy. Since Japanese morphological analysis simultaneously estimates the word segmentation and POS tagging, we have to check whether or not our system is negatively affected by anything other than the non-standard tokens. We also evaluated the recall with considering only normalized words. That value directly reflects the performance of our normalization method. We registered emoticons that occurred in the test data in the dictionary so that they would not negatively affect the systems' performance.

### 4.3 Results and Discussion

The results are classified in Table 4. As the table shows, the proposed methods performed statistically significantly better than the baselines and the traditional method in both precision and recall ($p < 0.01$), where the precision was greatly improved. This indicates that our method can not only correctly analyze the non-standard tokens, but can also reduce the number of wrong words generated. Baseline1 also improved the accuracy and recall compared to the traditional method, but the effect was limited. When we compare Proposed with Baseline2, we find the F-value is improved when we take the Hiragana and Katakana substitution into consideration. Baseline3 also improved the F-value but its performance is inferior to proposed method.This proves that even if we can generate sufficient normalization candidates, the results worsen if the weight parameter of each normalization candidate is not appropriately tuned. The column of "recall*" in Table 4 specifies the improvement rates of the non-standard tokens. The proposed methods improve about seven times when using Baseline1 while preventing degradation. These results prove that we have to generate appropriate and sufficient normalization candidates and appropriately tune the cost of each candidate to improve both the precision and recall.

We show examples of the system output in Table 5. In the table, slashes indicate the position of the estimated word segmentations and the words that were correctly analyzed are written in bold font. Examples (1) to (5) are examples improved by using the proposed method. Examples (6) to (7) are examples that were not improved and example (8) is an example that was degraded. Examples (1) to (3) include phonetic variations and example (4) is a Hiragana substitution. Example (5) is a combinational trans-

| method | word segmentation | | | word segmentation and POS tag | | | |
|---|---|---|---|---|---|---|---|
| | precision | recall | F-value | precision | recall | F-value | recall* |
| Traditional | 0.716 | 0.826 | 0.767 | 0.683 | 0.788 | 0.732 | - |
| Rule based (BL1**) | 0.753 | 0.833 | 0.791 | 0.717 | 0.794 | 0.754 | 0.092 |
| Proposed | **0.856** | **0.883** | **0.869** | **0.822** | **0.849** | **0.835** | **0.667** |
| - without Hiragana and Katakana normalization (BL2) | 0.834 | 0.875 | 0.854 | 0.798 | 0.838 | 0.818 | 0.509 |
| - character transformation cost is fixed (BL3) | 0.838 | 0.865 | 0.851 | 0.807 | 0.834 | 0.821 | 0.533 |

* considering only normalized words, ** BL:baseline

Table 4: Results of precision and recall of test data

| input | traditional | proposed | gold standard |
|---|---|---|---|
| (1) あ ぢー(*adii*) | あ (*a*)/ぢ (*di*)/ー | **あつい** (*atsui*) | あつい (*atsui*, “hot”) |
| (2) す げー(*sugee*) | すげ (suge)/ー | **すごい** (*sugoi*) | すごい (*sugoi*, “great”) |
| (3) ご っめ ー ん (*gommeen*) | ご (*go*)/っ/め (*me*)/ー/ん (*n*)/ | **ごめん** (*gomen*) | ごめん (*gomen*, “I’m sorry”) |
| (4)ひつよう(*hitsuyou*) | ひつ (*hitsu*)/よう (*you*) | **必要** (*hitsuyou*) | 必要 (*hitsuyou*, “necessary”) |
| (5)だいちゅき(*daichuki*) | だ (*da*)/いち (*ichi*)/ゅ(*yu*)/き (*ki*)/ | **大好き** (*daisuki*) | 大好き (*daisuki*, “like very much”) |
| (6) お せえええ(*oseee*) | おせ (*ose*)/ええ (*ee*)/え (*e*) | おせ (*ose*) | おそい (*osoi*, “slow”) |
| (7) か ん わいい (*kanwaii*) | かん (*kan*)/わ (*wa*)/いい (*ii*) | 官話 (*kanwa*)/いい (*ii*) | かわいい (*kawaii*, “cute”) |
| (8) いない (*inai*) | **い (*i*)/ない (*nai*)** | 以内 (*inai*) | い/ない (*i*/*nai*, “absent”) |

Table 5: System output examples

formation pattern of a phonetic variation and Hiragana substitution. We can see our system can analyze such variational non-standard tokens for all these examples. Two types of errors were identified. The first occurred as the result of a lack of a character transformation pattern and the second was search errors. Example (6) shows an example of a case in which our system couldn’t generate correct normalization candidate because there was not corresponding character transformation pattern, even though there was a similar phonetic transformation pattern. To ensure there will be no lack of transformation patterns, we should either increase the parallel corpus size to enable the learning of more patterns or derive new transformation patterns from the learned patterns. Example (7) shows an example of a case in which a normalized candidate was generated but a search failed to locate it. Example (8) shows an example of a case in which the result was degraded. Our system can control the degradation well, but there are several degradation caused by normalization. We will need to develop a more complicated model or introduce other features into the current model to reduce the number of search errors.

## 5 Conclusion and Future Work

We introduced a text normalization approach into joint Japanese morphological analysis and showed that our two-step lattice generation algorithm and formulation using discriminative methods outperforms the previous method. In future work, we plan to extend this approach by introducing an unsupervised or semi-supervised parallel corpus extraction for learning character alignments to generate more patterns at a reduced cost. We also plan to improve our model’s structure and features and implement it with a decoding method to reduce the number of search errors. In addition, we should consider adding other types of unknown words (such as named entities) to the morphological analysis system to improve its overall performance.

## References

AiTi Aw, Min Zhang, Juan Xiao, and Jian Su. 2006. A phrase-based statistical model for sms text normalization. *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, pages 33–40.

Maximilian Bisani and Hermann Ney. 2008. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Commun.*, 50(5):434–451, May.

Paul Cook and Suzanne Stevenson. 2009. An unsupervised model for text message normalization. *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 71–78.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 368–378.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432.

Hany Hassan and Arul Menezes. 2013. Social text normalization using contextual graph random walks. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1586, August.

Wenbin Jiang, Haitao Mi, and Qun Liu. 2008. Word lattice reranking for chinese word segmentation and part-of-speech tagging. *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, pages 385–392.

Nobuhiro Kaji and Masaru Kitsuregawa. 2013. Efficient word lattice generation for joint word segmentation and pos tagging in japanese. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 153–161.

Keigo Kubo, Hiromichi Kawanami, Hiroshi Saruwatari, and Kiyohiro Shikano. 2011. Unconstrained many-to-many alignment for automatic pronunciation annotation. *In Proc. of APSIPA ASC*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to japanese morphological analysis. *In Proc. of EMNLP*, pages 230–237.

T. Kudo. 2005. Mecab : Yet another part-of-speech and morphological analyzer. *http://mecab.sourceforge.net/*.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of japanese morphological analyzer juman. *In Proc. of The International Workshop on Sharable Natural Language Resources*, page 22—38.

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

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2013. Paraphrasing 4 microblog normalization. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84, October.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution? normalizing text messages without pre-categorization nor supervision. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 71–76, June.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1035–1044.

W Machery, F J Och, and I Uszkoreit J Thayer. 2008. Lattice-based minimum error rate training for statistical machine translation. *In Proc. of EMNLP*, 1:725–734.

Teruaki Oka, Mamoru Komachi, Toshinobu Ogiso, and Yuji Matsumoto. 2011. Handling orthographic variations in morphological analysis for near-modern japanese (in japanese). *In Proc. of The 27th Annual Conference of the Japanese Society for Articial Intelligence*.

Deana Pennell and Yang Liu. 2011. A character-level machine translation approach for normalization of sms abbreviations. *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 974–982, November.

Akira Sasaki, Junta Mizuno, Naoaki Okazaki, and Kentaro Inui. 2013. Normalization of text in microblogging based on machine learning(in japanese) (in japanese). *In Proc. of The 27th Annual Conference of the Japanese Society for Articial Intelligence*.

Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. 2013. A simple approach to unknown word processing in japanese morphological analysis. *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 162–170.

Jiampojamarn Sittichai, Kondrak Grzegorz, and Sherif Tarek. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. *In Proc. of The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 372–379.