# A Text Editing Approach to Joint Japanese Word Segmentation, POS Tagging, and Lexical Normalization

**Shohei Higashiyama**[1,2], **Masao Utiyama**[1], **Taro Watanabe**[2], **Eiichiro Sumita**[1]
[1]National Institute of Information and Communications Technology, Kyoto, Japan
[2]Nara Institute of Science and Technology, Nara, Japan
{shohei.higashiyama, mutiyama, eiichiro.sumita}@nict.go.jp,
taro@is.naist.jp

## Abstract

Lexical normalization, in addition to word segmentation and part-of-speech tagging, is a fundamental task for Japanese user-generated text processing. In this paper, we propose a text editing model to solve the three task jointly and methods of pseudo-labeled data generation to overcome the problem of data deficiency. Our experiments showed that the proposed model achieved better normalization performance when trained on more diverse pseudo-labeled data.

## 1 Introduction

User-generated text (UGT), such as social media and blog posts, is a valuable source of knowledge and opinions from diverse users. A notable characteristic of UGT is that it contains non-canonical sentences, and this degrades the performance of natural language processing (NLP) systems trained on canonical sentences. To reduce the gap between the performance on general text and on UGT, lexical normalization techniques—which convert nonstandard word forms to standard forms—have been explored, particularly for English (Aw et al., 2006; Baldwin et al., 2015). In addition, Japanese UGT requires a further step: to identify nonstandard words in unsegmented sentences written without word delimiters. For this reason, the problem of Japanese lexical normalization has been solved by predicting word boundaries, part-of-speech (POS) tags, and normalized word forms simultaneously (Sasano et al., 2013; Saito et al., 2014). Similarly to previous work, we tackle the joint task comprising Japanese word **S**egmentation, **P**OS tagging, and lexical **N**ormalization (SPN).

A critical problem in lexical normalization is the lack of labeled data. Manual annotation of normalized forms is a time-consuming task; therefore, the size of the available annotated corpora is quite small (Kaji and Kitsuregawa, 2014; Higashiyama et al., 2021). A prospective solution to this problem is the use of pseudo-labeled data. In this paper, we propose methods of generating pseudo-labeled data using (auto-) segmented sentences and standard and nonstandard word variant pairs. To generate high quality labels, we acquire reliable variant pairs based on lexical knowledge, namely, a dictionary with lemma definition and hand-crafted rules.

For efficient learning from a limited amount of data, we adopt a text editing approach. Our neural tagging model predicts edit operations to normalize input characters, while predicting segmentation and POS tags at the same time. The editing process is similar to that proposed in previous work on English lexical normalization (Chrupała, 2014; Min and Mott, 2015), but we design a specific tag set for the Japanese SPN task, which requires the management of a large number of character types.

Our extensive experiments on the SPN task demonstrated that our model achieved better normalization performance when the model used more additional features, it was trained on more types of pseudo-labeled data, and it was trained on training instances with more diverse context.

## 2 Task Definition

As shown in Table 2, a training instance for the SPN task is defined as a pair, comprising a sentence $\boldsymbol{x} = (x_1, \ldots, x_n)$ and its label sequence $\boldsymbol{t} = \{(f_j, l_j, p_j, S_j)\}_{j=1}^{m}$, where $n$ and $m$ ($\leq n$) are the numbers of characters and words in $\boldsymbol{x}$, $f_j$ and $l_j$ are the indexes of the first and last character in $j$-th word $w_j$, and $p_j$ is the POS tag of $w_j$. The set of standard forms $S_j$ is equal to the empty set $\emptyset$ when $w_j$ is a standard form, whereas $S_j$ consists of one or more standard forms when $w_j$ is a nonstandard form.

A system is required to predict the word boundaries of an input sentence and the POS tag of each word, detect nonstandard words, and generate one of the standard forms of each nonstandard word.



*Proceedings of the 2021 EMNLP Workshop W-NUT: The Seventh Workshop on Noisy User-generated Text*, pages 67–80

| | Meaning | Nonstandard word $w$ | Standard form $s$ | SEdit tags $\boldsymbol{t}^e$ | CConv tags $\boldsymbol{t}^c$ |
|---|---|---|---|---|---|
| (a) | really | まぢ | まじ | K, REP(じ) | K, K |
| (b) | difficult | ムズカシー | むずかしい | K, K, K, K, REP(い) | HR, HR, HR, HR, K |
| (c) | terrific | すごーいー | すごい | K, K, D, K, K, D | K, K, K, K, K |
| (d) | high/expensive | たっけぇ | たかい | K, REP(か), REP(い), D | K, K, K, K |
| (e) | awesome | さいこー | 最高 | K, K, K, REP(う) | KJ, KJ, KJ, KJ |

Table 1: Examples of labels for nonstandard and standard word pairs. K, D, HR, and KJ represent KEEP, DEL, TO_HIRAGANA, and TO_KANJI, respectively.

| $j$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $w_j$ | 日本 (Japan) | 語 (language) | まぢ (really) | ムズカシー (difficult) |
| $f_j, l_j$ | 1, 2 | 3, 3 | 4, 5 | 6, 10 |
| $p_j$ | Noun | Noun | Adverb | Adjective |
| $S_j$ | $\emptyset$ | $\emptyset$ | {まじ, マジ} | {難しい, むずかしい} |

Table 2: Words in and labels of a sentence $\boldsymbol{x}$ =“日本語まぢムズカシー” (*nihon go maji muzukashī*), which means “Japanese language is really difficult.”

## 3 Joint SPN Method

### 3.1 Multiple Sequence Labeling Formulation

In this work, we formulate the SPN task as multiple character-level sequence labeling problems. We convert the label sequence $\boldsymbol{t}$ to four tag sequences: a segmentation tag sequence $\boldsymbol{t}^s$, a character-level POS tag sequence $\boldsymbol{t}^p$, a string edit operation (SEdit) tag sequence $\boldsymbol{t}^e$, and a character type conversion (CConv) tag sequence $\boldsymbol{t}^c$.

We employ a tag set $\mathcal{T}_{\text{seg}} = \{\texttt{B}, \texttt{I}, \texttt{E}, \texttt{S}\}$ for segmentation, where B, I, and E represent the beginning, inside, and end of a multi-character word, and S represents a single-character word. We set $t_i^p = p_j \in \mathcal{T}_{\text{pos}}$ for the POS tag of a character $x_i$ in a word $w_j$ ($f_j \leq i \leq l_j$), where $\mathcal{T}_{\text{pos}}$ denotes a POS tag set. We use two types of tags for the normalization task. For $x_i$ in a standard word $w_j$, we set $t_i^e = t_i^c = \texttt{KEEP}$, which means that no edit operation or conversion is required for $x_i$. For $x_i$ in a nonstandard word $w_j$, two types of tags $t_i^e \in \mathcal{T}_{\text{sedit}}$ and $t_i^c \in \mathcal{T}_{\text{cconv}}$ are generated based on the closest standard form $s_j^\star \in S_j$, where $\mathcal{T}_{\text{sedit}}$ and $\mathcal{T}_{\text{cconv}}$ represent the tag sets of SEdit and CConv, which we define in §3.2. The procedure for selecting the closest standard form is as follows[1]: a character alignment between $w_j$ and $s \in S_j$ is calculated, and then the standard form with the most characters aligned to $w_j$ is selected.

[1]We describe the procedure in detail in Appendix §A.

### 3.2 Tag Definition

The Japanese writing system comprises three major scripts: two syllabographic *kana* (i.e., *hiragana* and *katakana*), and the morphographic *kanji*. The numbers of character types in them are different: approximately 80 in hiragana, 80 in katakana,[2] and more than 4,000 in kanji. To decrease the tag space size, we allow insertion and replacement operations only for kana characters. Specifically, we define the SEdit tags as $\mathcal{T}_{\text{sedit}} = \{\texttt{KEEP}, \texttt{DEL}, \texttt{INSL}(c), \texttt{INSR}(c), \texttt{REP}(c)\}$ for a kana character $c$. DEL indicates deletion of the current character, INSL($c$) and INSR($c$) indicate insertion of $c$ immediately to the left and right of the current character, respectively, and REP($c$) indicates replacement of the current character by $c$. In addition, we define the CConv tags as $\mathcal{T}_{\text{cconv}} = \{\texttt{KEEP}, \texttt{TO\_HIRA}, \texttt{TO\_KATA}, \texttt{TO\_KANJI}\}$, where the last three tags indicate conversion of the current character to hiragana, katakana, and kanji, respectively.[3]

For the example sentence $\boldsymbol{x}$ in Table 2, the tags for $w_3 = \boldsymbol{x}_{4:5}$ =まぢ and $w_4 = \boldsymbol{x}_{6:10}$ =ムズカシー are shown as (a) and (b) in Table 1, and the tags for the other characters are $t_i^e = t_i^c = \texttt{KEEP}$ ($1 \leq i \leq 3$). Both types of tags are automatically generated, according to the character alignments between original and standard tokens. Table 1 lists examples (c)–(e), which have other types of tags.

A remaining problem is the ambiguity of characters assigned with the TO_KANJI tag; for example, あき *aki* can be converted to 秋 ‘autumn’, 空き ‘vacancy’, or 飽き ‘bored’ depending on its surrounding context. We use an external kana-to-kanji converter to select the most likely candidates.

[2]We distinguish kana characters with and without a voicing mark (e.g., “か” *ka* and “が” *ga*).

[3]Tag definition different from above could be used. We investigated two alternative settings, but our preliminary experiments showed no gains over our proposed setting: a case where SEdit and CConv tags were merged into a single tag set and a case where additional SEdit tags similar to the special operators used in the pronunciation feature (§3.4) were introduced.

There still exist cases in which a nonstandard word with many deleted or replaced characters cannot be restored to its standard form (e.g., よろ *yoro* to よろしく *yoroshiku* 'thank you') by the defined tags when the required number of insertion and replacement operations exceeds the number of characters in the original token. This can be solved by introducing multi-character operations (e.g., `INSR(しく)`), but we assume that most instances can be expressed by single-character operations, and leave those cases for future work.

### 3.3 Model Architecture

We use a long short-term memory (LSTM)-based architecture (Hochreiter and Schmidhuber, 1997) for the sequence labeling tasks. Our model consists of shared bidirectional LSTM (BiLSTM) layers and task-specific inference layers.

An input character sequence $\boldsymbol{x}$ is transformed to embedding vectors $\boldsymbol{e}_{1:n} = (\boldsymbol{e}_1, \cdots, \boldsymbol{e}_n)$ and fed into a multi-layer BiLSTM. Hidden vectors from forward and backward LSTMs are concatenated, to form a single hidden vector $\boldsymbol{h}_i$ for each character. $\boldsymbol{h}_i$ is then mapped to a score distribution vector $\boldsymbol{y}_i^u$ for each task $u \in \mathcal{U} = \{\text{seg, pos, sedit, cconv}\}$, via a softmax layer.

Given training data $\mathcal{D}$, the model parameters are learned by minimizing a loss function $L$ during training. The loss $L$ is defined as the sum of the cross-entropy between the gold and predicted tag distributions for all tasks:

$$L = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \sum_{u\in\mathcal{U}} \lambda_u \sum_{1\le i\le|\boldsymbol{x}|} \boldsymbol{t}_i^u \log\, \boldsymbol{y}_i^u\,, \quad (1)$$

where $0 \le \lambda_u \le 1$ is a coefficient to control the contribution of each task $u$ and $\boldsymbol{t}_i^u$ is the one-hot vector of the gold label $t_i^u$, which is assigned to $x_i$.

### 3.4 Features

We use three input features, based on character, pronunciation, and lexicon entries. Feature vectors from the three sources for each character are concatenated, to form a single vector $\boldsymbol{e}_i$ in §3.3.

**Character Feature.** A character embedding vector $\boldsymbol{e}_i^c$ for each character $x_i$ is retrieved from a character embedding matrix.

**Pronunciation Feature.** We introduce a pronunciation element that corresponds to a vowel, a consonant, the long sound symbol, or a special operator (voicing *V*, semi-voicing *P*, or lowercasing *S*) in a kana character sequence.[4] These elements are similar to *romaji* (Roman letter transcription) but differ mainly with respect to the special operators. For example, "グ" *gu*, "ァ" *a*, and "パ" *pa* are decomposed into {*k*, *u*, *V*}, {*a*, *S*}, and {*h*, *a*, *P*}, respectively. Each character $x_i$ is decomposed into one or more pronunciation elements. A pronunciation vector $\boldsymbol{e}_i^p$ for $x_i$ is the average of its pronunciation element embeddings retrieved from an embedding matrix.

**Lexicon Feature.** We define two types of binary features based on a nonstandard word lexicon.[5] A lexicon word feature for a character $x_i$ is defined as a $(|P|\times|K|)$-dimensional vector $\boldsymbol{e}_i^{d,w}$, each element of which indicates whether $x_i$ corresponds to a particular position $p \in P = \{$immediate left, immediate right, beginning, middle, end$\}$ of any nonstandard word of length $k \in K$ in the lexicon. Similarly, a lexicon POS feature for $x_i$ is defined as a $|\mathcal{T}_{\text{POS}}|$-dimensional vector $\boldsymbol{e}_i^{d,p}$, each element of which indicates whether the $x_i$ corresponds to an inside position of any nonstandard word with a particular POS.

## 4 Pseudo-labeled Data Generation

To overcome the lack of training data for the normalization task, we construct a set of standard and nonstandard word variant pairs $\mathcal{V}$ and then generate different types of pseudo-labeled data by two approaches: distant supervision on formal target-side ($\text{DS}_{\text{tgt}}$) and informal source-side text ($\text{DS}_{\text{src}}$).

$\text{DS}_{\text{tgt}}$ generate a sentence where the original tokens are retained but nonstandard tokens among them are annotated with pseudo standard tokens; specifically, given a segmented sentence, a token matching with a nonstandard form $v_{\text{nst}}$ in $\mathcal{V}$ is annotated with SEdit and CConv tags to convert to its standard form, while other tokens are annotated with `KEEP` tags. On the other hand, $\text{DS}_{\text{src}}$ generate a sentence where one or more of the original tokens are replaced by pseudo nonstandard forms; given a standard and nonstandard variant pair $(v_{\text{st}}, v_{\text{nst}}) \in \mathcal{V}$, $\text{DS}_{\text{src}}$ extracts a segmented sentence containing a token with the same lemma as that of the pair, replaces the token by $v_{\text{nst}}$, and generates SEdit and CConv tags to convert $v_{\text{nst}}$ to

[4] We generate pronunciation features only for kana characters and use zero vectors for other types of characters.

[5] We use nonstandard words in dictionary-derived ($\mathcal{V}_d$) and rule-derived variant pairs ($\mathcal{V}_r$) (described in §4) for the models trained on dictionary-derived ($\mathcal{A}_d$) and rule-derived data ($\mathcal{A}_r$) (described in §5.1) in our experiments, respectively.

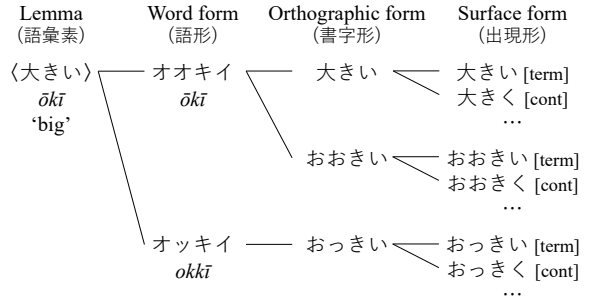



Figure 1: Hierarchical lemma definition in UniDic. Termination forms (term) and continuative forms (cont) are illustrated as examples of surface forms.

$v_{\mathrm{st}}$. We present examples of $\mathrm{DS}_{\mathrm{tgt}}$ and $\mathrm{DS}_{\mathrm{src}}$ in Appendix §B.

As the main difference between two approaches, $\mathrm{DS}_{\mathrm{tgt}}$ does not change original sentences but $\mathrm{DS}_{\mathrm{src}}$ does. Although we can use actual sentences with $\mathrm{DS}_{\mathrm{tgt}}$, we can easily obtain any number of synthetic sentences containing nonstandard words of interest with $\mathrm{DS}_{\mathrm{tgt}}$. However, both approaches require reliable variant pairs to generate useful data. For this purpose, we use two strategies for variant pair acquisition: dictionary-based and rule-based.

### 4.1 Dictionary-derived Variant Pairs

As the first approach to variant pair acquisition, we use UniDic[6] (unidic-cwj-2.3.0) (Den et al., 2008), but any dictionary with lemma and conjugation information can be used. UniDic is an electronic dictionary for Japanese morphological analysis (MA) and employs hierarchical definition of word indexes. As shown in Figure 1, a lemma in UniDic aggregates word forms with different pronunciation and word forms with different conjugation types, a word form aggregates orthographic forms, and an orthographic form aggregates surface forms (which mostly correspond to different conjugation forms). Thus, surface forms with the same lemma and conjugation form compose a variant set; for example, the continuative surface forms of a lemma 〈大きい〉 include 大きく *ōkiku*, おおきく *ōkiku*, and おっきく *okkiku*.

We extract valid standard and nonstandard word pairs from variant sets by the following steps. (1) Words whose POS is symbol, space, person name, or number are excluded. (2) Each variant in a variant set is automatically classified as a standard form or valid nonstandard form by predefined rules,[7] which are based on pronunciation and frequency of occurrence among the variant forms of the lemma in a corpus. (3) Finally, each nonstandard form is paired with the closest standard form.

### 4.2 Rule-derived Variant Pairs

As an alternative approach, we use hand-crafted rules to transform standard forms to nonstandard forms. We classify lexical variations that have been reported in previous work (Kaji et al., 2015; Miyazaki and Sato, 2019) or observed by us into dozens of patterns. We then choose patterns that are easy-to-implement or widely adaptable to many words, and implement them as variant generation rules. Specifically, we use four rules: change of character type (e.g., 疲労 *hirō* 'fatigue' → ひろう), and substitution by a character with the same pronunciation (e.g., マジ *maji* 'really' → マヂ), mora consonant (e.g., 行こう *ikō* 'go' → 行こっ), and uppercase kana (e.g., ちょっと *chotto* 'bit' → ちよつと), as well as six rules similar to those used by Sasano et al. (2013) and Ikeda et al. (2016). We describe the rules in detail in Appendix §C

To obtain plausible variant pairs, we follow these steps: (1) apply the rules to standard forms, which are obtained by the dictionary-based approach, and generate nonstandard form candidates, (2) count frequencies of character $n$-grams that match original standard forms or generated nonstandard forms in a corpus, and (3) accept variant pairs of which both the standard and nonstandard forms have frequencies higher than a threshold value of 10.

## 5 Experimental Settings

### 5.1 Language Resources

As training data $\mathcal{D}_t$ and development data $\mathcal{D}_v$ for the segmentation and POS tagging tasks, we used 57K and 3K sentences, respectively, with the short unit word (SUW) annotation from the core data of the Balanced Corpus of Contemporary Written Japanese (BCCWJ)[8] 1.1 (Maekawa et al., 2014).

For variant pair acquisition, we counted the frequencies of UniDic entries (§4.1) using 3.5M sentences in parts of registers of the BCCWJ non-core data and character $n$-gram frequencies (§4.2) using 8.8M sentences in Yahoo! Chiebukuro data.[9]

[6] https://ccd.ninjal.ac.jp/unidic

[7] The classification procedure is as follows: a variant with different pronunciation from its lemma is regarded as a nonstandard form, a variant with a frequency of occurrence of 5% or less as a nonstandard form, and a variant with a frequency of 10% or more as a standard form.

[8] https://ccd.ninjal.ac.jp/bccwj/en/

[9] https://www.nii.ac.jp/dsc/idr/yahoo/chiebkr3/Y_chiebukuro.html

We constructed three pseudo-labeled datasets using the dictionary-derived and rule-derived variant pairs, which we denote by $\mathcal{V}_d$ and $\mathcal{V}_r$. The first dataset $\mathcal{A}_t$ was generated by applying $\mathrm{DS}_{\mathrm{tgt}}$ to $\mathcal{D}_t$ using $\mathcal{V}_d$.[10] The second dataset $\mathcal{A}_d$ and third dataset $\mathcal{A}_r$ are 173K and 170K synthetic sentences generated by $\mathrm{DS}_{\mathrm{src}}$ from the 3.5M BCCWJ non-core sentences, using the top $n_p = 20$K frequent pairs[11] in $\mathcal{V}_d$ or $\mathcal{V}_r$. Notably, for each pair in $\mathcal{V}_d$ or $\mathcal{V}_r$, we extracted at most $n_s = 10$ original sentences that contained their lemma.[12] Similarly, we constructed an additional 3K sentences from the top 3K frequent pairs in $\mathcal{V}_d$ and 3K from 3K pairs in $\mathcal{V}_r$, and used them as development data, together with $\mathcal{D}_v$.

For evaluation, we used Yahoo! blog and Yahoo! Chiebukuro sentences annotated with SUW segmentation and POS tags, and with normalized forms (Higashiyama et al., 2021), as the test data.

### 5.2 Training Setting

During each training epoch, we randomly constructed a mini-batch consisting only of real training sentences ($\mathcal{A}_t$) or synthetic training sentences ($\mathcal{A}_d$ or $\mathcal{A}_r$) for each iteration, and trained the model for up to 20 epochs. We randomly initialized all parameters, applied mini-batch stochastic gradient descent to optimize parameters, and reduced the learning rate by a fixed decay ratio every epoch after the first five epochs. We set the loss coefficient values in Eq. (1) as $\lambda_{\mathrm{seg}} = \lambda_{\mathrm{pos}} = 1$ for real sentences, and set them as $\lambda_{\mathrm{seg}} = \lambda_{\mathrm{pos}} = \lambda_0$ for synthetic sentences with automatic segmentation and POS tags, where $\lambda_0$ is a hyperparameter. We set the other coefficient values for any sentences as $\lambda_{\mathrm{cconv}} = \lambda_{\mathrm{sedit}} = 1$.

We searched for the best values, within given ranges, for some hyperparameters of the model and used predetermined values for the others: character embedding size 200 from {100, 200, 300}, pronunciation embedding size 30 from {10, 20, 30, 40, 50}, BiLSTM hidden unit size 1,000 from {200, 400, 600, 800, 1,000}, BiLSTM dropout (Zaremba et al., 2014) rate 0.1 from {0.1, 0.2, 0.3, 0.4}, loss coefficient $\lambda_0 = 0.4$ from {0.2, 0.4, 0.6, 0.8, 1.0}, number of BiLSTM layers 2, mini-batch size 100, initial learning rate 1.0, learning rate decay ratio 0.9, and gradient clipping threshold 5.0.

### 5.3 Kana-to-Kanji Conversion Model

For kanji conversion (KC), we trained an n-gram language model (LM) of kana-kanji mixed sentences using SRILM[13] (Stolcke, 2002) on 1.2M sentences in the BCCWJ core and non-core data, and performed Viterbi decoding with negative log probability of the LM using the `decode_ngram.py`[14] script. Specifically, if the `TO_KANJI` tag was predicted for one or more characters in a predicted word span by the normalization model, the word was given to the KC model, together with six preceding and six succeeding characters, and the best hypothesis found was output as a normalized form.

### 5.4 Post-processing

We defined post-processing rules to apply to the predicted segmentation or normalization results. The first rule Seg-PP changes segmentation tags $t^s_{i+1:i+k}$ to `I`$\in \mathcal{T}_{\mathrm{seg}}$ when $k$ consecutive characters $x_{i:i+k}$ are the same vowel kana, long sound symbol, mora nasal, or mora consonant characters, according to our finding that such cases were rare from our preliminary experiment. The second rule Norm-PP changes a predicted normalized word to its original string if the predicted form does not exist in a standard form lexicon. We used standard forms in $\mathcal{V}_d$ as the lexicon.

### 5.5 Baseline Methods

We evaluated the following two methods for comparison. The first method was MeCab[15] (0.996) (Kudo et al., 2004) with UniDic (unidic-cwj-2.3.0), which is a popular Japanese MA toolkit based on conditional random fields (CRFs). The second method, which we call MeCab+ER, was Sasano et al. (2013)'s joint MA and normalization method, implemented by Higashiyama et al. (2021). This enhances MeCab's word lattices by their expansion rules to recognize specific types of nonstandard words shown in Appendix §C.

### 5.6 Evaluation Metrics

We used word-level precision, recall, and $\mathrm{F}_1$ score to evaluate systems on each task. As shown in

[10] We did not construct $\mathrm{DS}_{\mathrm{tgt}}$-based data using $\mathcal{V}_r$.

[11] We obtained 404K pairs from 873K UniDic entries and 47K pairs from 868K rule-generated nonstandard form candidates by the processes described in §4.1 and §4.2.

[12] Fewer than 200K sentences were generated because 10 sentences were not necessarily included for every variant pair in the original corpus.

[13] http://www.speech.sri.com/projects/srilm/

[14] https://github.com/yohokuno/neural_ime

[15] https://taku910.github.io/mecab/

| Method | Use of data | | | Seg | | | POS | | | Norm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathcal{A}_t$ | $\mathcal{A}_r$ | $\mathcal{A}_d$ | P | R | F | P | R | F | P | R | F |
| MeCab | | | | 89.2 | 95.1 | 92.1 | 87.5 | 93.3 | 90.3 | – | – | – |
| MeCab+ER | | | | **93.5** | **96.5** | **95.0** | **91.4** | **94.3** | **92.8** | 55.9 | 25.8 | 35.3 |
| Ours | ✓ | | | 91.3 | 93.8 | 92.6 | 87.6 | 90.0 | 88.8 | 50.9 | 19.4 | 28.1 |
| | ✓ | ✓ | | 91.0 | 93.7 | 92.3 | 88.1 | 90.8 | 89.4 | 42.4 | 28.0 | 33.8 |
| | ✓ | | ✓ | 91.9 | 94.2 | 93.1 | 89.0 | 91.2 | 90.1 | 52.9 | 32.1 | 39.9 |
| | ✓ | ✓ | ✓ | 91.1 | 93.8 | 92.5 | 88.3 | 90.9 | 89.6 | 49.7 | 37.0 | 42.4 |
| Ours +Seg-PP | ✓ | ✓ | ✓ | 92.9 | 94.1 | 93.5 | 89.9 | 91.1 | 90.5 | 50.8 | **37.8** | 43.4 |
| Ours +Seg-PP +Norm-PP | ✓ | ✓ | ✓ | 92.9 | 94.1 | 93.5 | 89.9 | 91.1 | 90.5 | **65.8** | 36.6 | **47.1** |

Table 3: Precision (P), recall (R), and $F_1$ score (F) of the proposed and compared methods for word segmentation, POS tagging, and lexical normalization.

Table 2, a test word has labels corresponding to an index pair of the first and last character, i.e., span, a POS tag, and a standard form set. A predicted word $w^*$ is counted as a true positive (TP) when the span of $w^*$ equals to that of a test word for the segmentation task and when the span and POS tag of $w^*$ equal to those of a test word for the POS tagging task. For the normalization task, a predicted word $w^*$ is counted as a TP when the span of $w^*$ equals to that of a test word and the normalized form of $w^*$ is included in the standard form set of the test word, whereas $w^*$ is counted as a false positive (FP) when either of the span or normalized form of $w^*$ does not match with a test nonstandard word. A test word $w$ is counted as a false negative when the span and normalized form of any predicted word do not match with $w$.

## 6 Results and Analysis

### 6.1 Main Results

Table 3 shows the performance of the proposed model (with the full features, unless otherwise specified) on the three tasks.[16] Although the proposed model trained only on $\mathcal{A}_t$ achieved low normalization recall, the model with additional data $\mathcal{A}_d$ or $\mathcal{A}_r$ achieved a higher score, and the model with the three datasets achieved the highest score. These results are roughly consistent with the observation that adding different types of pseudo-labeled data reduced the number of out-of-vocabulary (OOV) tokens in the test data, as shown in Appendix §D. Our model with post-processing achieved further improvements; Seg-PP improved $F_1$ for each task by approximately 1 point and Norm-PP improved normalization precision by 15 points. However, the latter results indicate that avoiding the predictions of meaningless or unusual forms has the potential to improve our model.

Compared with MeCab+ER, our model achieved better normalization performance when trained on sufficient training data. Conversely, MeCab+ER achieved the best segmentation and POS tagging performance. The superiority of MeCab+ER over MeCab indicates the advantage of the explicit prediction of normalized word spans by the method on the two tasks, which contrasts with the independent prediction of word spans, POS tags, and normalized forms performed by our model.

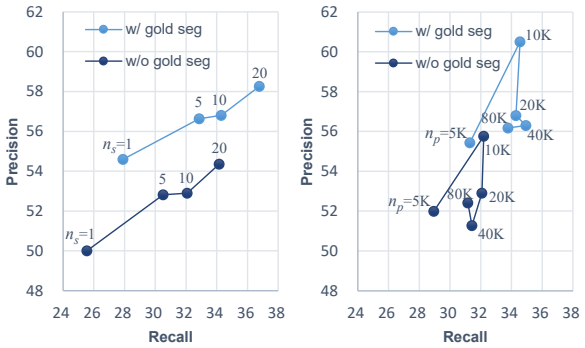
### 6.2 Effect of Dataset Size

To investigate the effect of the amount of pseudo-labeled data, we generated dictionary-derived data $\mathcal{A}'_d$ with different settings of $n_s$ and $n_p$, where $n_s$ is the number of extracted sentences per variant pair and $n_p$ is the number of variant pairs, as described in §5.1. We then evaluated the normalization performance of the proposed model trained on $\mathcal{A}_t \cup \mathcal{A}'_d$.[17]

Figure 2 (a) shows the performance of the model with varying $n_s$ and fixed $n_p = 20$K. A larger $n_s$ led to both better precision and better recall, indicating that training with the same variant pairs but with more diverse context sentences contributed to more robust prediction. Figure 2 (b) shows the performance of the model with fixed $n_s = 10$ and varying $n_p$. Increasing $n_p$ from 5K to larger values contributed to better performance but increasing $n_p$ above 10K did not improve recall, and degraded precision in most cases, probably because of the infrequent and ineffective variant pairs. Although the frequencies of the 5K-th and 10K-th nonstandard words in the constructed variant pairs were six and two, respectively, entries ranked below about

[16] We used a majority-vote to transform character-level POS tags predicted by the proposed model to a word-level tag.

[17] We also evaluated the model's performance given gold segmentation to remove the influence of segmentation errors, but the model showed a similar tendency regardless of whether gold segmentation was given, as shown in Figure 2.




(a) Varying $n_s$ ($n_p$ = 20K) (b) Varying $n_p$ ($n_s$ = 10)

Figure 2: Normalization performance of the proposed model trained on $\mathcal{A}_t \cup \mathcal{A}'_d$ with different size settings.

16K-th had a frequency of zero.[18]

These two results suggest that the gain discussed in §6.1 was caused by both the additional variant pairs and the additional contexts of existing variant pairs in the combined data of $\mathcal{A}_d$ and $\mathcal{A}_r$.

### 6.3 Detailed Results of Normalization

We semi-automatically annotated the test sentences with SEdit and CConv tags. Of 767 nonstandard tokens, six and three words required multi-character edit operations and replacement by Roman letters, respectively. Therefore, the upper bound of normalization recall was 99% by our tag set.

We then evaluated the proposed model with different features trained on the full $\mathcal{A}_t \cup \mathcal{A}_d \cup \mathcal{A}_r$ dataset, with respect to the character-level SEdit/CConv tag prediction accuracy for the KEEP tag and other tags, and the word-level text editing accuracy of negative and positive normalization instances, which is calculated according to gold segmentation. A negative instance indicates a gold word annotated with no standard forms, and a prediction is regarded as correct when only KEEP tags are predicted for the word. Notably, in the test data, KEEP tags account for 95.8% and 96.8% of all SEdit and CConv tags, respectively, and negative instances account for 93.9% of all word tokens.

As shown in Table 4, for the KEEP tag and negative normalization instances, the three models with different features achieved a high recall (close to, or better than, 99%) and a somewhat lower precision (around 97%–98%), indicating that over-normalized words and undetected nonstandard words accounted for the remaining 1% and 2%–3%, respectively. For other tags and positive normalization instances, both character-level and word-level performance were much lower because of the small number of training instances. However, the models with more additional features achieved better performance, indicating the effectiveness of the lexicon and pronunciation features. In particular, each additional feature substantially improved the performance of CConv tag prediction.

[18] Different entries with the same frequency were sorted in Japanese alphabetical order.

| Feature | SEdit (Keep) | | CConv (Keep) | | Norm-neg | |
|---|---|---|---|---|---|---|
| | P | R | P | R | P | R |
| C | 97.4 | 99.5 | 98.2 | 99.0 | 97.2 | 98.7 |
| C+L | 97.2 | 99.6 | 98.4 | 99.0 | 97.0 | 98.8 |
| C+L+P | 97.4 | 99.6 | 98.5 | 99.2 | 97.3 | 99.0 |

| Feature | SEdit (other) | | CConv (other) | | Norm | |
|---|---|---|---|---|---|---|
| | P | R | P | R | P | R |
| C | 77.5 | 39.5 | 56.7 | 41.5 | 48.6 | 36.9 |
| C+L | 77.7 | 34.0 | 60.6 | 48.0 | 50.4 | 35.6 |
| C+L+P | 80.5 | 39.8 | 67.9 | 52.2 | 54.6 | 40.3 |

Table 4: Precision/Recall of the proposed models with character (C), lexicon (L), and pronunciation (P) features for character-level tag prediction (SEdit/CConv) and word-level text editing (Norm-neg/Norm).

| Type | Gold | Det | ValTag | CorSeg | CorKC | R |
|---|---|---|---|---|---|---|
| SCV | 419 | 255 | 164 | 156 | 156 | 37.2 |
| CTV | 248 | 149 | 105 | 94 | 92 | 37.1 |
| AR | 132 | 78 | 55 | 51 | 51 | 38.6 |
| Typo | 23 | 3 | 1 | 1 | 1 | 4.4 |
| FP | – | 121 | – | – | – | – |

Table 5: Evaluation of the proposed model for each category. Sound change variants (SCV), character type variants (CTV), and alternative representations (AR) are categories defined in Higashiyama et al. (2021). FP indicates the number of words detected incorrectly by the model.

### 6.4 Error Analysis

We conducted step-by-step evaluation of the proposed model trained on the full dataset. Table 5 shows the number of gold normalization instances (Gold), predictions correctly detected (Det) out of Gold, predictions with valid SEdit/CConv tags (ValTag) out of Det, predictions correctly segmented (CorSeg) out of ValTag, and predictions matched with correct standard forms after text editing and KC (CorKC) out of CorSeg, for each category. For each of the top three categories (SCV, CTV, and AR), two major errors were detection failure, which accounted for 39%–41% ($\frac{\text{Gold}-\text{Det}}{\text{Gold}}$), and tag prediction error, which accounted for 17%–22% ($\frac{\text{Det}-\text{ValTag}}{\text{Gold}}$), whereas the true positives accounted for 37%–39% (R= $\frac{\text{CorKC}}{\text{Gold}}$). This tendency was

| Type | Gold | DetKC | ValTag | CorSeg | CorKC |
|---|---|---|---|---|---|
| Req. | 116 | 58 | 52 | 49 | 48 |
| Opt. | 170 | 22 | 21 | 21 | 20 |
| FP | – | 47 | – | – | – |

Table 6: Evaluation of the proposed model for kanji conversion (KC). Required (Req.) indicates that any standard forms of a nonstandard word require KC. Optional (Opt.) indicates that some standard forms require KC but other standard forms can be generated without KC. FP indicates the number of words detected incorrectly by the model.

common to all three categories.

Table 6 shows similar evaluation results for KC, where DetKC indicates the number of gold words assigned `TO_KANJI` tag(s) by the model. We found that most errors for the "required" instances were detection failures, and the KC model correctly converted 97% ($\frac{\text{CorKC}}{\text{CorSeg}}$) of the "required" and "optional" instances.

With respect to precision degradation, the model over-normalized 121 negative instances. We found that 61 cases of these instances were interjections or onomatopoeic words, and their characteristics were similar to general nonstandard words; this made it difficult to distinguish them from words to be normalized. We present further analysis and actual examples in Appendix §F.

## 7 Related Work

**Word Segmentation and Lexical Normalization.** For word segmentation and lexical normalization of Japanese UGT, most previous work applied a lattice-based MA framework, which jointly predicts word sequence and POS tag sequences over a word lattice of an input sentence. To incorporate informal word nodes into a word lattice, Sasano et al. (2013) and Kaji and Kitsuregawa (2014) used hand-crafted rules for recognizing variant forms of known words, and Saito et al. (2014) and Saito et al. (2017) acquired formal-informal word pairs from manually-annotated or unlabeled text. In contrast to those methods, Ikeda et al. (2016) applied a sequence-to-sequence model trained on synthetic formal-informal sentence pairs to sentence-level Japanese text normalization.

For Chinese, also an unsegmented language, nonstandard word detection and normalization methods have been proposed. Li and Yarowsky (2008) extracted formal-informal word pairs using web-searched sentences defining informal words and a conditional log-linear ranking model. Wang and Ng (2013) proposed beam-search decoding methods for lexical normalization as well as punctuation correction and recovery of missing words, for Chinese and English UGT, as preprocessing steps for Chinese↔English machine translation (MT). Qian et al. (2015) proposed a transition method based on a perceptron framework and a normalization dictionary for the joint SPN task. Zhang et al. (2017) proposed a transition method using character-level and word-level LSTMs for word segmentation and detection of informal words.

**English Lexical Normalization.** Early work on lexical normalization of English SMS and microblog text employed a noisy channel formulation; to restore plausible standard forms from observed nonstandard words, Aw et al. (2006) trained a statistical MT model and Choudhury et al. (2007) trained a hidden Markov model on parallel sentences of standard and nonstandard English. Liu et al. (2011) automatically collected training word pairs using carefully-designed web search queries and trained CRFs to calculate the conditional probability of a nonstandard character given a standard character. Recently, Muller et al. (2019) adapted BERT (Devlin et al., 2019) for lexical normalization by introducing a subword alignment algorithm between standard and nonstandard words and a task-specific fine-tuning strategy.

Some other work has adopted unsupervised methods: a log-linear model to score standard and nonstandard word sequences (Yang and Eisenstein, 2013), a graph-based method to model contextual and lexical similarity (Sönmez and Özgür, 2014), and a finite-state transducer using word embedding and string similarity (Rangarajan Sridhar, 2015).

**Text Editing.** Text editing methods have also been applied to English lexical normalization. Chrupała (2014) used character embeddings based on a recurrent neural network LM and trained CRFs to predict character-level edit operations. Min and Mott (2015) proposed an LSTM-based model to perform word-level edit operations that aggregate character-level edit operations.

Recently, text editing models based on Transformer and BERT (Malmi et al., 2019; Mallinson et al., 2020; Stahlberg and Kumar, 2020) have been proposed for monolingual sequence transduction tasks, such as grammatical error correction and text normalization for speech synthesis, because of their sample-efficient and fast inference characteristics

compared to sequence-to-sequence models.

**Data Synthesis.** Data synthesis and augmentation methods have been explored for various NLP tasks, to increase the diversity of training examples (Feng et al., 2021) and for lexical normalization to address the deficiency of training data. Ikeda et al. (2016) synthesized Japanese formal-informal sentence pairs by hand-crafted rules to convert standard forms to nonstandard forms. Zhang et al. (2017) synthesized training data for Chinese informal word detection by random substitution of formal words in segmented sentences by informal words in a dictionary of formal-informal word pairs. To train statistical and neural MT models for Turkish text normalization, Çolakoğlu et al. (2019) generated a pseudo-parallel corpus where nonstandard words in original tweet text were aligned with plausible standard words using their weighted edit distance algorithm. Dekker and van der Goot (2020) explored data synthesis methods for English lexical normalization using the clean-to-noisy policy (mainly based on manually-designed rules) and the noisy-to-clean policy (based on predicted standard forms).

## 8 Conclusion

This paper presents our text editing model and methods of pseudo-labeled data generation for the joint segmentation, POS tagging, and normalization task. The experiments demonstrated that the proposed model was successfully trained on generated pseudo-labeled data, but more exhaustive detection and accurate normalization of nonstandard words have the potential to improve the model.

Future work includes (1) explicit consideration of nonstandard word spans for accurate segmentation and normalization, (2) the use of a large-scale pretrained LM for better normalization coverage, and (3) evaluation on broader UGT domains.

## Acknowledgements

We thank our anonymous reviewers for their helpful comments. We used the Balanced Corpus of Contemporary Written Japanese provided by the National Institute for Japanese Language and Linguistics and Yahoo! Chiebukuro data (3rd edition) provided by Yahoo Japan Corporation via IDR Dataset Service of the National Institute of Informatics.

## References

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

Timothy Baldwin, Marie Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China. Association for Computational Linguistics.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition*, 10(3):157–174.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 680–686, Baltimore, Maryland. Association for Computational Linguistics.

Talha Çolakoğlu, Umut Sulubacak, and Ahmet Cüneyd Tantuğ. 2019. Normalizing non-canonical Turkish texts using machine translation approaches. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 267–272, Florence, Italy. Association for Computational Linguistics.

Kelly Dekker and Rob van der Goot. 2020. Synthetic data for English lexical normalization: How close can we get to manually annotated data? In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6300–6309, Marseille, France. European Language Resources Association.

Yasuharu Den, Junpei Nakamura, Toshinobu Ogiso, and Hideki Ogura. 2008. A proper approach to Japanese morphological analysis: Dictionary, model, and evaluation. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, Marrakech, Morocco. European Language Resources Association.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

Shohei Higashiyama, Masao Utiyama, Taro Watanabe, and Eiichiro Sumita. 2021. User-generated text corpus for evaluating Japanese morphological analysis and lexical normalization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5532–5541, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Taishi Ikeda, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Japanese text normalization with encoder-decoder model. In *Proceedings of the 2nd Workshop on Noisy User-generated Text*, pages 129–137, Osaka, Japan. The COLING 2016 Organizing Committee.

Nobuhiro Kaji and Masaru Kitsuregawa. 2014. Accurate word segmentation and pos tagging for Japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 99–109, Doha, Qatar. Association for Computational Linguistics.

Nobuhiro Kaji, Shinsuke Mori, Fumihiko Takahashi, Tetsuro Sasada, Itsumi Saito, Keigo Hattori, Yugo Murawaki, and Kei Utsumi. 2015. Kētaiso kaiseki no error bunseki (Error analysis of morphological analysis) [in Japanese]. In *Proceedings of the 21th Annual Meeting of the Association for Natural Language Processing Workshop "Error Analysis on Natural Language Processing"*, Kyoto, Japan.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

Zhifei Li and David Yarowsky. 2008. Mining and modeling relations between formal and informal Chinese phrases from web corpora. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1031–1040, Honolulu, Hawaii. Association for Computational Linguistics.

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

Kikuo Maekawa, Makoto Yamazaki, Toshinobu Ogiso, Takehiko Maruyama, Hideki Ogura, Wakako Kashino, Hanae Koiso, Masaya Yamaguchi, Makiro Tanaka, and Yasuharu Den. 2014. Balanced corpus of contemporary written Japanese. *Language Resources and Evaluation*, 48(2):345–371.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. FELIX: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Wookhee Min and Bradford Mott. 2015. NCSU_SAS_WOOKHEE: A deep contextual long-short term memory model for text normalization. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 111–119, Beijing, China. Association for Computational Linguistics.

Chiaki Miyazaki and Satoshi Sato. 2019. Classification of phonological changes reflected in text: Toward a characterization of written utterances [in Japanese]. *Journal of Natural Language Processing*, 26(2):407–440.

Benjamin Muller, Benoit Sagot, and Djamé Seddah. 2019. Enhancing BERT for lexical normalization. In *Proceedings of the 5th Workshop on Noisy User-generated Text*, pages 297–306, Hong Kong, China. Association for Computational Linguistics.

Tao Qian, Yue Zhang, Meishan Zhang, Yafeng Ren, and Donghong Ji. 2015. A transition-based model for joint segmentation, POS-tagging and normalization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1837–1846, Lisbon, Portugal. Association for Computational Linguistics.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised text normalization using distributed representations of words and phrases. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 8–16, Denver, Colorado. Association for Computational Linguistics.

Itsumi Saito, Kyosuke Nishida, Kugatsu Sadamitsu, Kuniko Saito, and Junji Tomita. 2017. Automatically extracting variant-normalization pairs for Japanese text normalization. In *Proceedings of the*

*8th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 937–946, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Itsumi Saito, Kugatsu Sadamitsu, Hisako Asano, and Yoshihiro Matsuo. 2014. Morphological analysis for Japanese noisy text based on character-level and word-level normalization. In *Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1773–1782, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. 2013. A simple approach to unknown word processing in Japanese morphological analysis. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*, pages 162–170, Nagoya, Japan. Asian Federation of Natural Language Processing.

Cagil Sönmez and Arzucan Özgür. 2014. A graph-based approach for contextual text normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 313–324, Doha, Qatar. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5147–5159, Online. Association for Computational Linguistics.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 901–904, Denver, Colorado, USA.

Pidong Wang and Hwee Tou Ng. 2013. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 471–481, Atlanta, Georgia. Association for Computational Linguistics.

Yi Yang and Jacob Eisenstein. 2013. A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72, Seattle, Washington, USA. Association for Computational Linguistics.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *Computing Research Repository*, arXiv:1409.2329.

Meishan Zhang, Guohong Fu, and Nan Yu. 2017. Segmenting Chinese microtext: Joint informal-word detection and segmentation with neural networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4228–4234, Melbourne, Australia.

| Original sentence | | | | Updated information | | | |
|---|---|---|---|---|---|---|---|
| Word | Lemma ID | POS | Conjugation form | Word (Substituted) | SEdit tags | CConv tags | Standard form |
| スゴく | 19163 | Adjective | Continuative | – | K, K, K | HIRA, HIRA, K | すごく |
| 気 | 8263 | Noun | – | – | K | K | – |
| に | 28178 | Particle | – | – | K | K | – |
| なる | 28061 | Verb | Termination | – | K, K | K, K | – |

Table 7: An example of $\text{DS}_{\text{tgt}}$. Assume that we have a standard and nonstandard word variant pair $p_a$ = (スゴく, すごく) 'very' with lemma ID of 19163 and continuative form. A segmented sentence $\boldsymbol{x}_a$ ="スゴく|気|に|なる" (*sugoku ki ni naru*), which means "(I'm) very curious.", is annotated with SEdit and CConv tags by $\text{DS}_{\text{tgt}}$, according to the matched token スゴく with the same lemma ID and conjugation form as those of $p_a$.

| Original sentence | | | | Updated information | | | |
|---|---|---|---|---|---|---|---|
| Word | Lemma ID | POS | Conjugation form | Word (Substituted) | SEdit tags | CConv tags | Standard form |
| ほんとう | 34947 | Noun | – | ほんっと | K, K, D, INSR(う) | K, K, K, K | ほんとう |
| に | 28198 | Particle | – | – | K | K | – |
| 心配 | 19516 | Noun | – | – | K, K | K, K | – |
| です | 25653 | Copula | Termination | – | K, K | K, K | – |

Table 8: An example of $\text{DS}_{\text{src}}$. For a standard and nonstandard word variant pair $p_b$ =(ほんとう, ほんっと) 'truth' with lemma ID of 34947 and no conjugation form, a segmented sentence containing a token with the same lemma ID and conjugation form as those of $p_b$ is extracted: $\boldsymbol{x}_b$ ="ほんとう|に|心配|です" (*hontō ni shimpai desu*), which means "(I'm) really worried." Then, a synthetic sentence "ほんっと|に|心配|です" annotated with SEdit and CConv tags is generated by $\text{DS}_{\text{src}}$.

## A Selection of Closest Standard Form

Let $w_j$ be a word and $S_j$ be the set of standard forms of $w_j$. We define four character types of a word: "hiragana-only", "katakana-only", "kanji-kana-mixed", and "other". The process of selecting the closest standard form, which is mentioned in §3.1, comprises the following steps:

1. If $S_j$ contains standard forms with the same character type as $w_j$, those standard forms are prioritized; standard forms with different character types are removed from $S_j$.

2. If $S_j$ contains only standard forms with different character types from $w_j$, the standard forms with the same character type as the standard form occurring most frequently in a corpus are retained, and others are removed from $S_j$.

3. The standard form with the most characters that are aligned to $w_j$ is selected as the closest standard form $s_j^\star$. Character alignment between $w_j$ and $s \in S_j$ is calculated, to find the longest matching substrings recursively, until any substrings in $w_j$ and $s$ are not matched.

## B Examples of Pseudo Label Annotation

An example of $\text{DS}_{\text{tgt}}$ to a sentence $\boldsymbol{x}_a$ and a variant pair $p_a$ is shown in Table 7. Also, an example of $\text{DS}_{\text{src}}$ to a sentence $\boldsymbol{x}_b$ and a variant pair $p_b$ is shown in Table 8. The notation of SEdit and CConv tags in Table 7 and 8 is the same as that in Table 1.

## C Variant Generation Rules

We define 10 rules in Table 9 to generate nonstandard form candidates from standard forms. Rule 2 interchanges お↔を, じ↔ぢ, ず↔づ, ぶ↔う, オ↔ヲ, ジ↔ヂ, ズ↔ヅ, or ブ↔ヴ as characters with the same pronunciation. We generate multiple variants from an original word using any combination of applicable rules in $\{0, 1\} \times \{0, 2, 7\} \times \{0, 3, 8\} \times \{0, 4, 5, 6\} \times \{0, 9, 10\}$, where 0 indicates that no rule is applied.

## D Out-of-vocabulary Tokens in Test Data

Table 10 shows the number and percentage of OOV tokens for each training dataset. Adding either $\mathcal{A}_d$ or $\mathcal{A}_r$ (separately) to $\mathcal{A}_t$ reduced the number of OOV nonstandard tokens by 102 or 110, but adding both datasets decreased the number of OOV tokens by 161. This indicates that the remaining 51 tokens were contained in both datasets.

| ID | Rule | Sub-rule | Original | Variant |
|---|---|---|---|---|
| 1 | Change of character type | (a) Hiragana to katakana | たいへん *taihen* 'hard' | タイヘン |
| | | (b) Katakana to hiragana | スーパー *sūpā* 'super' | すーぱー |
| | | (c) Kana-kanji mixed to hiragana | 疲労 *hirō* 'fatigue' | ひろう |
| | | (d) Kana-kanji mixed to katakana | 苦手 *nigate* 'weak' | ニガテ |
| 2 | Sub. by character with the same pronunciation | – | マジ *maji* 'really' | マヂ |
| 3 | Sub. by mora consonant | (a) "です" | です *desu* (copula) | っす |
| | | (b) Adjective ends with "い" | 広い *hiroi* 'wide' | 広っ |
| | | (c) Verb ends with "う" | 行こう *ikō* 'go' | 行こっ |
| 4 | Sub. by uppercase kana | – | ちょっと *chotto* 'bit' | ちよつと |
| $5^{S,I}$ | Sub. by lowercase kana | – | いや *iya* 'unpleasant' | ぃや |
| $6^{S,I}$ | Sub. of vowel by long sound | – | 楽しい *tanoshī* 'fun' | 楽しー |
| $7^{I}$ | Sub. of vowel sequence | (a) Adjective ends with *-ai* to *ē* | うるさい *uruasi* 'loud' | うるせえ *urusē* |
| | | (b) Adjective ends with *-ui* to *ī* | わるい *warui* 'bad' | わりい *warī* |
| | | (c) Adjective ends with *-oi* to *ē* | おそい *osoi* 'late' | おせえ *osē* |
| | | (d) Word ends with *-o*う to *o*お | そう *sō* 'so' | そお |
| | | (e) "言う/いう" to ゆう | 言い *ī* 'say' | ゆい *yui* |
| $8^{I}$ | Deletion of tail vowel | (a) Adjective ends with "い" | ひどい *hidoi* 'terrible' | ひど *hido* |
| | | (b) Word ends with "う" | だろう *darō* (copula) | だろ *daro* |
| $9^{I}$ | Insertion of mora consonant | (a) Into the middle | きつい *kitsui* 'tough' | きっつい *kittsui* |
| | | (b) Into the end | けど *kedo* 'but' | けどっ |
| $10^{S,I}$ | Insertion of long sound | (a) Long sound sym. into the middle | 大きい *ōkī* 'big' | 大きーい *ōkīi* |
| | | (b) Long sound sym. into the end | 正解 *seikai* 'answer' | 正解ー *seikaī* |
| | | (d) Uppercase vowel into the middle | かなり *kanari* 'quite' | かなあり *kanāri* |
| | | (c) Uppercase vowel into the end | 強い *tsuyoi* 'strong' | 強いい *tsuyoī* |
| | | (e) Lowercase vowel into the middle | ずっと *zutto* 'always' | ずぅっと *zūtto* |
| | | (f) Lowercase vowel into the end | ます *masu* (copula) | ますぅ *masū* |

Table 9: Variant generation rules and examples of generated variants. "Sub." indicates substitution. The IDs with "S" and "I" indicate that similar rules were used in Sasano et al. (2013) and Ikeda et al. (2016), respectively.

| Training data | All | | Nonstandard | |
|---|---|---|---|---|
| | Test OOV | | Test OOV | |
| | # | % | # | % |
| $\emptyset$ | 12,600 | 100.0 | 767 | 100.0 |
| $\mathcal{A}_t$ ($\mathcal{D}_t$) | 1,055 | 8.4 | 445 | 58.0 |
| $\mathcal{A}_t \cup \mathcal{A}_r$ | 734 | 5.8 | 335 | 43.7 |
| $\mathcal{A}_t \cup \mathcal{A}_d$ | 764 | 6.1 | 343 | 44.7 |
| $\mathcal{A}_t \cup \mathcal{A}_d \cup \mathcal{A}_r$ | 636 | 5.1 | 284 | 37.0 |

Table 10: Number and percentage of (all and nonstandard) test OOV tokens for each training dataset.

| Feature | Known | Unknown |
|---|---|---|
| C | 54.3 | 11.8 |
| C+L | 55.8 | 7.3 |
| C+L+P | 62.1 | 11.8 |

Table 11: Recall of the proposed models with character (C), lexicon (L), and pronunciation (P) features for known and unknown normalization instances

## E Performance for Known and Unknown Normalization Instances

Letting a token be known if the token and its gold standard form are included in the full training data $\mathcal{A}_d \cup \mathcal{A}_r \cup \mathcal{A}_t$, normalization instances in the test data consist of 385 known and 382 unknown instances. Recall of the proposed models trained on the full dataset with different features for both type of instances is shown in Table 11. Unsurprisingly, all model variations recognized known instances much better than unknown instances. Although the model with full features achieved the highest recall of 62.1%, this is lower than the model's recall of 86.0% on the pseudo development data that only included known normalization instances. The performance difference for known test instances and development instances is likely because of more distant context distribution of test instances from training instances.

## F Examples of Over-normalization

As mentioned in §6.4, the proposed model over-normalized 121 negative instances, including 61 cases that were interjections or onomatopoeic words.[19] Examples that were over-normalized by the model trained on the full dataset are shown in Table 12. Both interjections and onomatopoeic

[19] Interjections except idiomatic greetings and most onomatopoeic words were not annotated with any standard forms mainly because defining standard forms is difficult for these types of words.

| | Type | Original word | Edited word | KC result | Assess |
|---|---|---|---|---|---|
| (a) | Onomatopoeia | ガコンッ *gakon* (thud) | ガコン | – | ✓ |
| (b) | Onomatopoeia | ジンジン *jinjin* (tingling) | じんじん | – | ✓ |
| (c) | Onomatopoeia | ゴホゴホ *gohogoho* (coughing sound) | ごほごホ | – | ? |
| (d) | Interjection | はぁぁ *haā* (sighing sound) | はああ | – | ✓ |
| (e) | Interjection | ぅん *un* (yeah) | うん | – | ✓ |
| (f) | Interjection | ひーひっひー *hīhihhī* (evil laugh sound) | ひいひっひい | – | ✓ |
| (g) | Interjection | あらっ *ara* (oh) | あらい *arai* | 洗い (wash) | × |
| (h) | Interjection | おお〜〜 *ō* (wow) | おう | 王 (king) | × |
| (i) | Informal | ケータイ *kētai* (mobile phone) | ケイタイ | – | ✓ |
| (j) | Informal | ダルい *darui* (dull/tired) | だるい | – | ✓ |
| (k) | Informal | キズ *kizu* (wound) | キズ | 傷 | ✓ |
| (l) | Informal | かっこ[E] *kakko*[*ī*] (cool) | かっこ | カッコ | ✓ |
| (m) | Informal | ムレる *mureru* (stuffy) | むレる | – | ? |
| (n) | Informal | ヤバイ *yabai* (crazy) | やバイ | – | ? |

Table 12: Examples of over-normalization, i.e., FP, by the proposed model. Words in "[]" indicate surrounding context. The "Edited word" column shows the output of the model after editing, according to predicted tags, and the "KC result" column shows the result after performing kanji conversion (KC). The "Assess" column shows our assessments: "✓", "?", and "×" indicate that the final output is acceptable (the meaning is mostly preserved), questionable (the meaning is understandable but the spelling is peculiar), and obviously incorrect, respectively.

words have characteristics similar to those of general nonstandard forms, such as insertion of Japanese special mora characters (a and e–h in Table 12), use of lowercased kana characters (d–e), and repetition of the same characters (d and h). These characteristics made it difficult to distinguish negative instances from words to be normalized.

Another 29 cases were somewhat informal forms written in katakana or hiragana (i–n in Table 12) and approximately 60% of the predicted normalized forms were acceptable, according to our assessment. This is because of the difficulty of annotating all words in a test sentence with all possible lexical variations.

Incorrect normalization results included cases with peculiar spellings where some hiragana or katakana characters were converted to another type of kana (c and m–n).