Japanese Text Normalization with Encoder-Decoder Model

Taishi Ikeda, Hiroyuki Shindo and Yuji Matsumoto
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, 630-0192, Japan
{ikeda.taishi.io7, shindo, matsu}@is.naist.jp

Abstract

Text normalization is the task of transforming lexical variants to their canonical forms. We model the problem of text normalization as a character-level sequence to sequence learning problem and present a neural encoder-decoder model for solving it. To train the encoder-decoder model, many sentences pairs are generally required. However, Japanese non-standard canonical pairs are scarce in the form of parallel corpora. To address this issue, we propose a method of data augmentation to increase data size by converting existing resources into synthesized non-standard forms using handcrafted rules. We conducted an experiment to demonstrate that the synthesized corpus contributes to stably train an encoder-decoder model and improve the performance of Japanese text normalization.

1 Introduction

With the rapid spread of the social media, many texts have been uploaded to the internet. As such, social media texts are considered important language resources owing to an increasing demand for information extraction and text mining (Lau et al., 2012; Aramaki et al., 2011). However, these texts include lexical variants such as insertions, phonetic substitutions and internet slangs. These non-standard forms adversely affect language analysis tools that are trained on a clean corpus. Since Japanese has no explicit boundaries between words, word segmentation and part-of-speech (POS) tagging are extremely important in Japanese language processing. For example, the output obtained using the Japanese morphological analyzer: MeCab [1] for a non-standard sentence: "このあぷりすげえええ！" is as follows:

Input:このあぷりすげえええ！ (*kono apuri sugeee*; This app is greeeat!)

Output:この (This, Prenominal adjective) / あ (a, Filler) / ぷりすげえええ (UNK) / ！(Symbol)

where the slashes indicate word boundaries. Although "アプリ" (*apuri*; app) is originally written in Katakana form, it is written in Hiragana form "あぷり" (*apuri*; app) in this example. Japanese has several writing scripts: Kanji, Hiragana and Katakana. Such interchanging between scripts in social media texts often occurs. Furthermore, "すげえええ" (*sugeeee*; greeeat) is derived from "すごい" (*sugoi*; great) by changing some vowels from */oi/* to */eee/*. These non-standard forms such as "あぷり" and "すげえええ" are not registered in the morphological analysis dictionary. For this reason, if a word does not exist in the morphological analysis dictionary, the traditional system cannot predict the correct word segmentation or POS tags. If we could normalize all non-standard forms, then the canonical sentence is considerd as "このアプリすごい！". As a result, we can obtain the correct result of morphological analysis for this normalized sentence as follows:

Input:このアプリすごい！(*kono apuri sugoi*; This app is great!)

Output:この (This, Prenominal adjective) / アプリ (app, Noun) / すごい (great, Adjective) / !(Symbol)

As the above example shows, we would expect improvement in the result of the morphological analysis by transforming non-standard forms into their standard ones.

In this work, we adopt the character-level encoder-decoder model as a method of Japanese text normalization. Since the encoder-decoder model was proposed in the field of machine translation, it has

[1] http://taku910.github.io/mecab/

Proceedings of the 2nd Workshop on Noisy User-generated Text,
pages 129–137, Osaka, Japan, December 11 2016.

achieved good results in morphological inflection generation and error correction (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2015; Cho et al., 2014; Manaal et al., 2016; Barret et al., 2016). As these models are capable of capturing sequential patterns, it is possible to apply the encoder-decoder model to Japanese text normalization. As mentioned previously, Japanese has no explicit boundaries between words. Thus, Japanese text normalization is naturally posed to a character-level sequence to sequence learning.

Although the encoder-decoder model has been shown its effectiveness in large datasets, it is much less effective for small datasets such as low-resource languages (Barret et al., 2016). Unfortunately, contrary to machine translation, Japanese non-standard canonical pairs are scarce in the form of parallel corpora. To address this issue, we propose a method of data augmentation to increase the data size by converting existing resources into synthesized non-standard forms using handcrafted rules. We conducted an experiment to demonstrate that the synthesized corpus provides stable training of the encoder-decoder model and improves the performance of Japanese text normalization.

The contributions of this research can be summarized by citing two points. First, the proposed data augmentation methods can provide stable training of the encoder-decoder model. Second, it can improve the performance of Japanese text normalization by adding the synthesized corpus. The rest of this paper is organized as follows. Section 2 describes the related work to our research including Japanese text normalization and the neural encoder-decoder model. Section 3 introduces the model architecture in this research. Section 4 describes how to construct a synthesized corpus. Section 5 discusses experiments that we have performed and our corresponding analyses of the experimental results. Section 6 concludes the paper with a brief summary and a mention of future work.

2 Related Work

In this section, we mainly describe text normalization for Japanese. Furthermore, we explain some related works with regard to the encoder-decoder model in the field of machine translation.

2.1 Text Normalization

Several studies for joint modeling of morphological analysis and text normalization have been conducted (Saito et al., 2014; Kaji and Kitsuregawa, 2014; Sasano et al., 2013). Saito et al. (2014) extracted non-standard tokens from Twitter and blog texts, and manually annotated their standard forms. They automatically generated derivational patterns based on the character-level alignment between non-standard tokens and their standard forms, then included these patterns to a morphological lattice. Kaji et al. (2014) similarly proposed a joint modeling for morphological analysis and text normalization via their construction of a normalization dictionary using handcrafted rules that were similar to those used in (Sasano et al., 2013) to derive non-standard forms from the entries already registered in a dictionary. Sasaki et al. (2013) solved Japanese text normalization as a character-level sequential labeling problem. Specifically, they extracted one-to-one character alignment from non-standard canonical pairs. Then, they performed character-level Conditional Random Fields (CRF) to normalize a non-standard sentence by assigning a tag to each character. In their work, the sequence labeler specifies the class of rules such as *delete*, *replace* and *no edit*. While these methods need feature engineering, our approach uses character vectors as features instead of discrete features of a surface character sequence. In our approach, a canonical text is generated as a sequence of characters from all character vocabularies, which is different from the approaches that assign tags from a fixed tag set.

2.2 The Encoder-Decoder Model

In this work, we normalize non-standard sentences using the neural encoder-decoder model. The encoder-decoder model has been shown effective in the field of machine translation (Bahdanau et al., 2015; Luong et al., 2015; Sutskever et al., 2015; Cho et al., 2014). It consists of two neural networks: the encoder network and the decoder network. The encoder maps the input sentence to a fixed-dimensional vector and the decoder generates the output sentence. Although the encoder-decoder model has been shown effective with large datasets, it is much less effective in small datasets of low-resource lan-

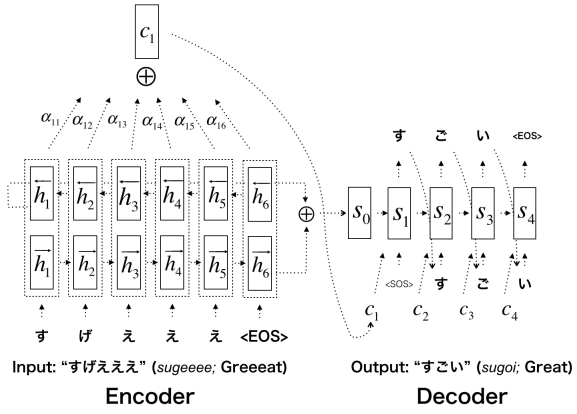

Figure 1: The encode-decoder architecture for Japanese text normalization

guages (Barret et al., 2016). As a solution to this issue, we propose a method of data augmentation to increase data size. Our approach converts existing resources into synthesized non-standard forms with handcrafted rules.

Our model is also inspired by character-level models of error correction and morphological inflection (Manaal et al., 2016; Ziang et al., 2016). However, the character set used in Japanese is numerous much larger than that in English, which makes the encoder-decoder model challenging. For example, there are about 3,500 Kanji character types used in a Japanese newspaper corpus. On the contrary, the character set used in English includes at most 100 characters; lower alphabet, upper alphabet, numerals and special symbols.

3 The Model Architecture

In this section, we briefly describe the architecture of our model. Our neural network model consists of the encoder and decoder networks. The encoder maps the input sentence to a fixed-dimensional vector with bidirectional recurrent neural network. The decoder is also a recurrent neural network (RNN) that uses a content-based attention mechanism (Bahdanau et al., 2015) to attend to the encoded representation, and generate the output sentence one character at a time. Figure 1 shows our text normalization architecture. Our model performs at a character-level process in the encoder as well as the decoder. The encoder reads the input $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, where $x_i \in \mathbb{R}^V$ is an input vector of length V. The length V is equal to the vocabulary size of all characters in the training data. x_i is represented as a one-hot vector that corresponds to a character. N is the length of the input sentence. The decoder generates the output $\boldsymbol{y} = (y_1, y_2, \ldots, y_M)$, where $y_i \in \mathbb{R}^V$ is an output vector of length V. y_i is represented as a one-hot vector that corresponds to a character. M is the length of the output sentence. Then, the task of text normalization is formularized as follows:

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

To summarize, we map a non-standard sentence $\boldsymbol{x}$ to a canonical sentence $\boldsymbol{y}$ with the encoder-decoder model.

3.1 The Encoder Network

The encoder maps the input sentence to a fixed-dimensional vector h with a bidirectional RNN. The bidirectional RNN has two parallel layers in two directions, and these two layers memorize the information of the sentences from both forward and backward direction (Schusteret al., 1997). More specifically, given the input vector $\boldsymbol{x}$, the forward and backward hidden vectors are computed by using an RNN as follows:

$$\overrightarrow{h}_t = \mathrm{GRU}(x_t, \overrightarrow{h}_{t-1}) \tag{2}$$

$$\overleftarrow{h}_t = \mathrm{GRU}(x_t, \overleftarrow{h}_{t+1}) \tag{3}$$

where $\overrightarrow{h}_t \in \mathbb{R}^l$ is a forward hidden vector at time t and GRU denotes the gated recurrent unit function that is similar to long short-term memory units (LSTMs), which have shown to improve the performance of RNN (Cho et al., 2014). GRU allows the retention of information from time steps in the distant past. GRU has two gates: the update gate and the reset gate. These gates control how much each hidden unit remembers or forgets the information while reading a sequence. Finally, in the encoder we get $h_t = [\overrightarrow{h}_t, \overleftarrow{h}_t]$ by concatenating the forward hidden state $\overrightarrow{h}_t$ and the backward one $\overleftarrow{h}_t$ for each time.

3.2 The Decoder Network

The decoder network is also GRU based on a recurrent neural network. The decoder is trained to predict the next output y_t, given the encoded $h = [h_1, h_2, \ldots, h_N]$ and all the previously predicted outputs $y_1, y_2, \ldots, y_{t-1}$.

$$y_t = \mathrm{Decoder}(y_1, y_2, \ldots, y_{t-1}, h) \tag{4}$$

Specifically, the decoder is conditioned on the encoded representation c_t which is the weighted sum of the encoded hidden states h_t by an attention mechanism. The attention mechanism is computed as:

$$c_t = \sum_{j=1}^{N} \alpha_{tj} h_j \tag{5}$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{N} \exp(e_{tk})} \tag{6}$$

$$e_{tj} = v_a^t \tanh(W_a s_{t-1} + U_a h_j + b_a) \tag{7}$$

The attention mechanism decides which parts of the input characters to be paid attention. Then, a hidden state of the decoder s_t is computed as:

$$s_t = \mathrm{GRU}(s_{t-1}, y_{t-1}, c_t) \tag{8}$$

In the decoder, we compute the hidden layer s_t at time t and feed this into a softmax layer to obtain the conditional probability of the output character y_t as:

$$o_t = \mathrm{softmax}(W_s s_t + b_s) \tag{9}$$

Finally, we obtain the max probability of o_t as the output character y_t at time t.

3.3 Training

We use the cross-entropy per time step summed over the output sequence $\boldsymbol{y}$ as a loss function. Given the training data of the non-standard sentence $\boldsymbol{x}$ and the canonical sentence $\boldsymbol{y}$, the goal is to optimize the parameters set θ by minimizing the loss function $E(\theta)$.

$$E(\theta) = -\sum_{(\boldsymbol{x},\boldsymbol{y})\in D} \log P(\boldsymbol{y}|\boldsymbol{x};\theta) \tag{10}$$

We minimize the loss function $E(\theta)$ using stochastic gradient-based optimization techniques.

4 Construction of Synthesized Corpus

To address the problem of a small dataset, a method of data augmentation to increase data size is proposed. To achieve the increase in data size, our approach focuses on converting existing resources into synthesized non-standard forms using handcrafted rules. Specifically, we convert a morpheme in the canonical form from the newswire portion of Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa et al., 2014) into a non-standard form with handcrafted rules. We use some rules from the work of (Sasano et al., 2013). Our proposed synthesized rules are explained below in detail.

rules	part-of-speech	vowel	original canonical form	transformed non-standard form
(1)	adjective	i	かわいい (*kawaii*; cute)	かわいー (*kawaii*)
(2)	auxiliary verb	i	たい (*tai*; want)	てええ (*teee*)
(3)	adjective	i	美味しい (*oishii*; delicious)	美味しぃ(*oishii*)
(4)	sentence-end particle	a	な (*na*)	なぁ(*naa*)
(5)	auxiliary verb	a	です (*desu*)	で〜す (*deesu*)
(6)	interjection	u	ありがとう (*arigatou*; Thank you)	ありがと (*arigato*)

Table 1: Examples of transforming canonical forms into their non-standard forms

Rule 1: Substitution of vowel characters with long sound symbols

Japanese vowel characters "あ"(a), "い"(i), "う"(u), "え"(e) and "お"(o) are substituted by long sound symbols "ー" or "〜". For example, a vowel character "う" in the morpheme "どう" (*dou*; how) is substituted by "ー" and this morpheme is transformed as "どー" (*doo*).

Rule 2: Substitution of an end of the word with a common non-standard form

For example, "すげえええ" (*sugeeee*; greeeat) is derived from "すごい" (*sugoi*; great) by changing the vowels from /oi/ to /eee/, where changing vowels are observed in Japanese social media texts. Thus, we substitute the end of the word of canonical forms with common non-standard form such as changing vowels.

Rule 3: Substitution with lowercases

Some hiragana characters, such as "あ"(a), "い"(i), "う"(u), "え"(e), "お"(o) and "わ"(wa) are substituted by their lowercases: "ぁ", "ぃ", "ぅ", "ぇ", "ぉ" and "ゎ".

Rule 4: Insertion of vowel characters or double consonants

Japanese vowel characters (their lowercases) or double consonants "っ" are inserted into a morpheme.

Rule 5: Insertion of long sound symbols

Long sound symbols "ー" or "〜" are inserted into a morpheme.

Rule 6: Abbreviation

Abbreviate some characters from a morpheme. For example, we drop "い" from this canonical form"している"(*shiteiru*; to be doing), which results in "してる"(*shiteru*) as the non-standard form.

Table 1 shows examples of the application of our rules to the newswire portion of BCCWJ. To apply our rules, we would first check the POS, surface form and vowel of a morpheme from a sentence in BCCWJ; Then, we would employ the appropriate rules to the morpheme. Note that the rules are not applicable to some sentences in the corpus.

5 Experiments

To evaluate the effectiveness of our normalization method, we conducted experiments on a microblog data to confirm the effectiveness of our model. For the experiments, we used the Microblog corpus and the Synthesized corpus for training. Both corpora consist of canonical and non-standard sentence pairs. The Synthesized corpus comprised 213,618 non-standard and canonical sentences pairs. The Microblog corpus was constructed from Japanese Twitter data by Kaji and Kitsuregawa (2014). They collected 17,386 Japanese tweets using the Twitter Api. Among these, 1000 tweets were randomly selected, and they annotated the 1831 sentences with canonical forms and morphological information. In our work, we extracted 506 sentences that included at least one non-standard form from the Microblog corpus. These extracted sentences comprised 697 non-standard canonical pairs. In the preprocessing, repetitions of more than one character of long sound symbols, namely "ー", "〜" and the double consonant "っ" are reduced back to one character, whereas the others are reduced to three characters. We used these sentences as the Microblog corpus.

method	test CER(%)
No-operation	17.15
Rule-based	15.07
CRF (Microblog corpus)	13.71
CRF (Synthesized corpus)	13.63
CRF (Combined corpus)	**13.43**
EncDec (Microblog corpus)	87.07
EncDec (Synthesized corpus)	13.54
EncDec (Combined corpus)	**13.43**

Table 2: Results for test sets on CER

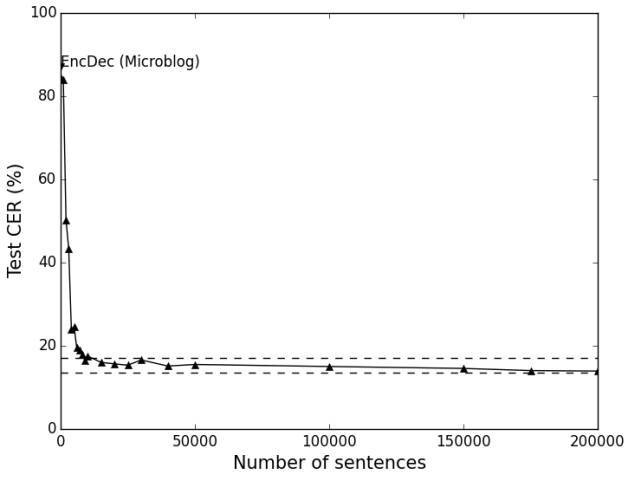

Figure 2: Scatter plot of the relationship between Test CER and the training data size

5.1 Compared Methods

We compared our method with two baseline methods listed in Table 2. Rule-based is a method of applying simple rules to a non-standard sentence in order to normalize it. In particular, we use the following rules: "Substitution of long sound symbols with vowel characters" and "Deletion of long sound symbols or lowercases". CRF is a method of tagging each character with normalization rules proposed by (Sasaki et al., 2013). Specifically, we extracted one-to-many character alignments from non-standard canonical pairs of the training data. Then, we assigned each character the class of rules such as "delete", "replace with arbitrary characters" and "no edit". The features we used in CRF were "the current characters including window size of 2" and "the binary value indicating whether the current character is a vowel character or not".

EncDec is our proposed method. Across all the encoder-decoder models described in this paper, we used the following hyperparameters. We used the Adam optimizer with a learning rate of 0.0002 (Diederik Kingma and Jimmy Ba, 2014). In both the encoder and decoder models, we used a single layer GRU with the hidden vector of size 500. The size of the character embedding vector was 250. The minibatch size was 32. While models are trained for 50 epochs, the parameters set was selected with the lowest character error rate on the development set. All results were obtained by using a beam search with a beam width of 10. Our models were implemented using Theano (Theano Development Team, 2016). In decoding, characters with the exception of Hiragana, Katakana, Kanji and long sound symbols are directly to the output.

We explain how to use the training datasets on each method in Table 2. In the case of Rule-based, the training data is not used at all. In the case of CRF (Microblog corpus) and EncDec (Microblog corpus), we used only the Microblog corpus as the training data. In the case of CRF (Synthesized corpus)

	sentences
input sentence	ラルクって海外でも人気すげーな☆ (L'Arc is greeatly popular in the world ☆)
target sentence	ラルクって海外でも人気すごいな☆ (L'Arc is greatly popular in the world ☆)
EncDec (Microblog)	一近と一ー中ちゃんとうらみたとう☆ (*This sentence doesn't make sense.*)
EncDec (Synthesized)	ラルクって海外でも人気すごいな☆ (L'Arc is greatly popular in the world ☆)

Table 3: System output examples: The training of EncDec on the different training data sets

	sentences
input sentence (1)	おごりっすか？ (isit your treat?)
target sentence (1)	おごりですか？ (Is it your treat?)
Rule-based	おごりっすか？ (isit your treat?)
CRF (Microblog)	おごりっすか？ (isit your treat?)
EncDec (Combined)	おごりですか？ (Is it your treat?)
input sentence (2)	久々にＴｗｉｔｔｅｒ浮上したなぁ。(I've not been on twitter for a long time.)
target sentence (2)	久々にＴｗｉｔｔｅｒ浮上したな。(I've not been on twitter for a long time.)
Rule-based	久々にＴｗｉｔｔｅｒ浮上したな。(I've not been on twitter for a long time.)
CRF (Microblog)	久々にＴｗｉｔｔｅｒ浮上したな。(I've not been on twitter for a long time)
EncDec (Combined)	久久々にＴｗｉｔｔｅｒ浮上したな。(I've not been on twitter for a long long time.)
input sentence (3)	車離れが捗りますなーw (It makes people turning away from driving.)
target sentence (3)	車離れが捗りますなw (It makes people turning away from driving.)
Rule-based	車離れが捗りますなw (It makes people turning away from driving.)
CRF (Microblog)	車離れが捗りますなw (It makes people turning away from driving.)
EncDec (Combined)	車離れが暇りますなw (It makes free turning away from driving.)

Table 4: System output examples: Comparison between our method and the baselines

and EncDec (Synthesized corpus), we used only the Synthesized corpus as the training data, while the Microblog corpus is only used as the development data. In the case of EncDec (Combined corpus) and CRF (Combined corpus), we used the Microblog corpus and the Synthesized corpus together as the training data and then termed both as the Combined corpus.

5.2 Evaluation

As our evaluation metric, we used a character error rate (CER), which is defined as the Levenshtein edit distance between the predicted character sequence $\boldsymbol{x}$ and the target character sequence $\boldsymbol{y}$, normalized by the total number of characters in the target. CER indicates the closeness of the normalized sentence toward the target sentence.

In the experiments, we performed 5-fold cross validation. By dividing the Microblog corpus into the test data, development data and training data, we tested on only the Microblog corpus to evaluate the effectiveness of each method on all the microblog data.

5.3 Results and Discussion

Table 2 shows the results of our experiments on test CER. No-operation indicates a baseline that leaves the input sentence unchanged. This demonstrates that the input sentence has about two non-standard characters in itself. As shown in Table 2, the Rule-based method reduces errors by 2.08 %, as compared with No-operation. CRF trained on only the Microblog corpus: CRF (Microblog corpus) reduces errors by 3.44 %, as compared with No-operation. With these results, when the encoder-decoder model trained on only the Microblog corpus, we found that the EncDec (Microblog corpus) is worse than No-operation, then the results in CER from 17.15 % to 87.07 %. As Table 3 shows, it generated entirely different strings of the input sentence. This indicates that the Microblog corpus is too small for training the encoder-decoder model; however, the Synthesized corpus contributes to the stable training of the encoder-decoder model. Additionally, EncDec (Synthesized corpus) is able to normalize the non-standard form "すげー" (*sugee*; greeat) into the canonical form "すごい" (*sugoi*; great) without generating different strings of the input sentence. Figure 2 shows the relation between test CER and the training data size for the encoder-decoder model. The first dotted line indicates that it needs more than 10,000 sentences to reduce the error rate than No-operation. The second dotted line represents that about 200,000 sentences reach CER on CRF (Microblog corpus). As we have combined the Microblog

corpus and the Synthesized corpus, the Combined corpus contributes to improving the performance of Japanese text normalization in CRF and EncDec. Both methods are able to reduce errors by 3.72 %, as compared with No-operation. Other examples of our normalization system output are illustrated in Table 4. Example (1) shows that our method is able to normalize the spoken form "っす" (*ssu*; is) into its canonical form "です" (*desu*; is). Examples (2) and (3) demonstrate that our method is inability to generate correct normalization sentence. Example (2) shows an incorrect example of over-translation, which generates the same character repeatedly: The Kanji character "久" was generated twice. Example (3) shows an incorrect example of the low frequency of a character: The Kanji character "捗" appears once in the training data. Thus, the wrong character "暇" was generated instead of "捗" in this example.

6 Conclusions and Future Work

In this work, we modeled the problem of text normalization as a character-level sequence to sequence learning problem. To address the problem of a small dataset, we proposed a method of data augmentation to increase data size, which converts existing resources into synthesized non-standard forms using hand-crafted rules. We found that the proposed data augmentation methods could provide a stable training of the encoder-decoder model, and it improved the performance of Japanese text normalization by adding the synthesized corpus. For our future work, we plan to extend the encoder-decoder model by incorporating a copy mechanism (Gu et al., 2016; Miltiadis et al., 2016) to address the issues of over-translation and low frequency of a character.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This work was partially supported by JST CREST and JSPS KAKENHI Grant Number 15K16053, 26240035.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Minh-Thang Luong, Pham Hieu, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412-1421.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le 2014. Sequence to sequence learning with neural networks. In *Proceedings of Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3104-3112.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724-1734.

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 (EMNLP)*, pages 99-109.

Itsumi Saito, Sadamitsu Kugatsu, 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 (COLING)*, pages 1773-1782.

Ryohei Sasano, Sadao Kurohashi, and Manabu Okumura. 2013. A Simple Approach to Unknown Word Processing in Japanese Morphological Analysis. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing (IJCNLP)*, pages 162-170.

Kazushi Ikeda, Tadashi Yanagiharaand, Kazunori Matsumoto, and Yasuhiro Takishima. 2009. Unsupervised text normalization approach for morphological analysis of blog documents. In *Proceedings of Australasian Joint Conference on Advances in Artificial Intelligence*, pages 401-411.

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

Faruqui Manaal, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Zoph Barret, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer Learning for Low-Resource Neural Machine Translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xie Ziang, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*, http://arxiv.org/pdf/1603.09727v1.

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), pages 345-371.

Sanae Fujita, Hirotoshi Taira, Tessei Kobayashi, and Takaaki Tanaka. 2014. Japanese Morphological Analysis of Picture Books. *The Association for Natural Language Processing*, volume 21 number 3, pages 515-539.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of Conditional Random Fields (CRFs). http://www.chokkan.org/software/crfsuite/.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, http://arxiv.org/abs/1605.02688.

Grzegorz Chrupaa. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 680-686.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 368-378.

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 (EMNLP)*, pages 230-237.

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

Graham Neubig, Yousuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL)*, pages 529-533.

Mike Schuster and Paliwal, Kuldip K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* volume 45 number 11, pages 2673-2681.

Jey Han Lau and Nigel Collier, and Timothy Baldwin. 2012. On-line Trend Analysis with Topic Models: # twitter Trends Detection Topic Model Online. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1519-1534.

Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. 2011. Twitter catches the flu: detecting influenza epidemics using Twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1568-1576.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631-1640.

Allamanis Miltiadis, Hao Peng, and Charles Sutton. 2016. Convolutional Attention Network for Extreme Summarization of Source Code. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, pages 2091-2100.