Building a Japanese Typo Dataset from Wikipedia's Revision History

Yu Tanaka **Yugo Murawaki** **Daisuke Kawahara**[*] **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, Japan
{ytanaka, murawaki, dk, kuro}@nlp.ist.i.kyoto-u.ac.jp

Abstract

User generated texts contain many typos for which correction is necessary for NLP systems to work. Although a large number of typo–correction pairs are needed to develop a data-driven typo correction system, no such dataset is available for Japanese. In this paper, we extract over half a million Japanese typo–correction pairs from Wikipedia's revision history. Unlike other languages, Japanese poses unique challenges: (1) Japanese texts are unsegmented so that we cannot simply apply a spelling checker, and (2) the way people inputting kanji logographs results in typos with drastically different surface forms from correct ones. We address them by combining character-based extraction rules, morphological analyzers to guess readings, and various filtering methods. We evaluate the dataset using crowdsourcing and run a baseline seq2seq model for typo correction.

1 Introduction

For over a decade, user generated content (UGC) has been an important target of NLP technology. It is characterized by phenomena not found in standard texts, such as word lengthening (Brody and Diakopoulos, 2011), dialectal variations (Saito et al., 2017; Blodgett et al., 2016), unknown onomatopoeias (Sasano et al., 2013), grammatical errors (Mizumoto et al., 2011; Lee et al., 2018), and mother tongue interference in non-native writing (Goldin et al., 2018). Typographical errors (typos) also occur often in UGC.[1] Typos prevent machines from analyzing texts properly (Belinkov and Bisk, 2018). Typo correction systems are important because applying them before analysis would reduce analysis errors and lead to improved accuracy in various NLP tasks.

Neural networks are a promising choice for building a typo correction system because they have demonstrated their success in a closely related task, spelling correction (Sakaguchi et al., 2017). Since neural networks are known to be data-hungry, the first step to develop a neural typo correction system is to prepare a large number of typos and their corrections. However, to our best knowledge, no such dataset is available for Japanese.[2] This motivated us to build a large Japanese typo dataset.

Typos are usually collected using data mining techniques because thorough corpus annotation is inefficient for infrequently occurring typos. Previous studies on building typo datasets have exploited Wikipedia because it is large, and more importantly, keeps track of all changes made to an article (Max and Wisniewski, 2010; Zesch, 2012). In these studies, to collect typo–correction pairs, the first step is to identify words changed in revisions and the second step is to apply a spell checker to them or calculate the edit distance between them.

While these methods work well on the target languages of the previous studies, namely French and English, they cannot be applied directly to languages such as Japanese and Chinese, where words are not delimited by white space. This is because a typo may cause a word segmentation error and can be misinterpreted as a multiple-word change, making it difficult to identify the word affected. Although state-of-the-art word segmenters provide reasonable accuracy for clean texts, word segmentation on texts with typos remains a challenging problem. In addition, languages with complex writing systems such as Japanese and Chinese have typos not found in French and English. These languages use logographs, *kanji* in Japanese, and they are

[*]Current affiliation is Waseda University

[1]In the present study, typos may cover some grammatical errors in addition to spelling errors.

[2]Although the publicly available multilingual GitHub Typo Corpus (Hagiwara and Mita, 2020) covers Japanese, it contains only about 1,000 instances and ignores erroneous kanji-conversion, an important class of typos in Japanese.

typically entered using input methods, with which people enter phonetic symbols, *kana* in the case of Japanese, and then select a correct logograph from a list of logographs matching the reading. Typos occurring during this process can be drastically different from the correct words.

In this paper, we build a Japanese Wikipedia typo dataset (JWTD) from Japanese Wikipedia's revision history. To address problems mentioned above, we treat adjacent changed words as one block and obtain the readings of kanji using a morphological analyzer. This dataset contains over half a million typo–correction sentence pairs. We evaluate JWTD by using crowdsourcing and use it to train a baseline seq2seq model for typo correction. JWTD is publicly available at `http://nlp.ist.i.kyoto-u.ac.jp/EN/edit.php?JWTD`. To the best of our knowledge, this is the first freely available large Japanese typo dataset.

2 Japanese Typos

We classify Japanese typos into four categories: erroneous substitution (hereafter **substitution**), erroneous deletion (hereafter **deletion**), erroneous insertion (hereafter **insertion**), and erroneous kanji-conversion (hereafter **kanji-conversion**).[3] An example and its correction for each category are shown in Table 1. Substitution is the replacement of a character with an erroneous one, deletion is the omission of a necessary character, insertion is the addition of an unnecessary character, and kanji-conversion is misconverting kanji, which needs some explanation.

To enter kanji, you first enter hiragana syllabary, either by converting roman-letter inputs or directly using a hiragana keyboard. The hiragana sequence indicates the reading, and accordingly, the input method shows a list of candidate kanji that match the reading, allowing you to choose the correct one. Errors in kanji-conversion typically occur at the last step. A typo of this category shares the reading with the correct one but in most cases, does not contain the same characters at all. For example, the typo–correction pair, "貼り付け (*harituke*) → 磔 (*harituke*)", which mean paste and crucifixion respectively, shares no character at all so that a simple edit distance-based method does not work. This is why kanji-conversion requires a special treatment.

[3]We do not collect erroneous transposition (Baba and Suzuki, 2012) because we observe that it occurs only infrequently in Japanese.

3 Data Construction

We construct JWTD in two steps. We first extract candidates of typo–correction sentence pairs from Wikipedia's revision history and then filter out pairs that do not seem to be typo corrections.

3.1 Mining Typos from Wikipedia

We extract candidates of typo–correction sentence pairs from Wikipedia's revision history according to the following procedure.[4]

1. For each revision of each article page, we extract a plain text from an XML dump[5] and split it into a list of sentences.[6]
2. For each article, we compare each revision with the revision immediately preceding it in a sentence-by-sentence manner using the Python3 difflib library.[7] We extract only sentence pairs that have differences. We remove pairs that have a sentence with 10 or fewer characters, or 200 or more characters. Too short sentences lack the context for us to determine whether the changes are typo corrections while too long sentences may arise from preprocessing errors. We also remove pairs with the edit distance of 6 or more because we assume that a revision having a large edit distance is not typo correction.
3. For each sentence pair, we split each sentence into a word sequence using MeCab (Kudo et al., 2004),[8] compare them using difflib, and identify *diff* blocks. Note that difflib outputs the replacement of multiple words as a single block. Therefore, typos causing a change of multiple words are also obtained.
4. We extract sentence pairs with a diff block that falls into one of the following categories:

   Substitution

   - the edit distance is 1,[9]

[4]Due to space limitations, we do not explain in detail additional measures to clean up the data: removing reverted revisions, removing looping revisions (for example, A→B and B→A), and replacing transitive revisions (for example, replace A→B and B→C to A→C).

[5]To strip wiki markup, we use WikiExtractor (`https://github.com/attardi/wikiextractor`)

[6]We use an in-house sentence splitter: `https://github.com/ku-nlp/textformatting`.

[7]`https://docs.python.org/3/library/difflib.html`

[8]`https://taku910.github.io/mecab/`

[9]We limit the edit distance to one because our preliminary investigation suggests that changes with the edit distance of two or more are increasingly likely to be content revisions rather than typos. The coverage problem needs to be addressed in the future.

Substitution	兄の部隊【の (*no*) → に (*ni*)】所属していた兵士でもあり、...
	(He was also a soldier belonging【of → to】his brother's unit, and ...)
Deletion	...民間レスキュー組織をもっていること【+で (*de*)】知られる。
	(... is known【+ for】having a civilian rescue organization.)
Insertion	特に免疫力の差などがそう【-う (*u*)】である。
	(In particular, differences in immunity is like that【- t】.)
Kanji-conversion	まだ、全学全てが大学院に【以降 (*ikou*) → 移行 (*ikou*)】していないため、...
	(Because all of the universities have not yet made a【after → transition】to graduate school, ...)

Table 1: Examples of Japanese typos and their corrections. +, -, and → indicate insertion, deletion, and substitution from the left hand side to the right hand side, respectively.

- the two sentences are the same in length, and
- both of the characters changed before and after the revision are hiragana, katakana, or alphabet.[10]

Deletion

- the edit distance is 1,
- the change in sentence length is $+1$, and
- the added character is hiragana, katakana, or alphabet.

Insertion

- the edit distance is 1,
- the change in sentence length is -1, and
- the deleted character is hiragana, katakana, or alphabet.

Kanji-conversion

- the two sentences have the same reading,[11] and
- both of the diff blocks before and after the revision contain kanji.

Mining typos separately for each category is a reasonable decision because each category has its own characteristics. However, this mining strategy prevents us from obtaining a balanced dataset. We leave it for future work.

3.2 Filtering

Sentence pairs obtained according to the above procedure contain pairs that do not seem to be typo corrections. We use the following three methods to remove them.

[10]We use the Python3 regex library (`https://pypi.org/project/regex/`) to determine character types, hiragana, katakana, kanji, or alphabet.

[11]We use the morphological analyzers Juman++ (Tolmachev et al., 2018) (`http://nlp.ist.i.kyoto-u.ac.jp/index.php?JUMAN++`) and MeCab to get readings of kanji. If at least one of the analyses of reading matches, we regard the pairs as having the same reading.

Part of speech and morphological analysis This filters out sentence pairs in which the changes concern acceptable variants, rather than typos. Based on the morphological analysis by Juman++, we remove sentence pairs that have edits related to name, number, tense, etc.

Redirect data This filters out sentence pairs that have a different spelling but the same meaning such as "ケニヤ (*keniya*)" and "ケニア (*kenia*)", both of which mean Kenya. For such close variants, Wikipedia provides special *redirect* pages that automatically send visitors to article pages. A page and its redirect can be treated as a pair of acceptable spelling variants. We obtain a list of redirects from an XML dump and remove a sentence pair if the diff block is found in the list.

Language model By using a character-level LSTM language model, we filter out sentence pairs in two ways. We trained the model by using all the latest pages of Japanese Wikipedia generated in September 2019, which contains 19.6M sentences.

The first filter measures how much the loss (negative log-likelihood) decreases by a revision. This filters out sentence pairs that seem to be spam or both sentences seem to be natural. Let $loss_{\text{pre}}$ and $loss_{\text{post}}$ be the total language model loss of the pre-revision sentence and that of the post-revision sentence, respectively. We filter out sentence pairs that satisfy the following:

$$\frac{loss_{\text{post}} - loss_{\text{pre}}}{\text{the number of characters changed in the pairs}} > \alpha,$$

where α is determined heuristically. It is set to -4 for substitution, -5 for deletion, and -6 for insertion. We do not apply this filter to kanji-conversion. We found that a change from high-frequency kanji to low-frequency kanji often yielded a large value even if the change was correct.

The second filter focused on the loss of the post-revision sentence. This filters out sentence pairs in which the post-revision sentence seems to be unnatural. We filter out sentence pairs that satisfy the following:

$$\frac{loss_{\text{post}}}{\text{the number of characters of the post-revision sentence}} > \beta,$$

where β is set to 5 heuristically.

4 Dataset Analysis

We built JWTD by applying the method explained in Section 3 to Japanese Wikipedia's revision history generated in June 2019. The number of typo–correction pairs obtained for each category is shown in Table 2. Because the first of the two language model filters was not applied to kanji-conversion, the effect of filtering was less drastic for kanji-conversion.

The top 10 most frequent typo–correction pairs are listed in Table 3. The unit of corrections is the morpheme-level edits obtained by using Juman++ and difflib. For deletion, there were typo–correction pairs associated with colloquialism, such as "る (*ru*) $\rightarrow$ いる (*iru*)" and "た (*ta*) $\rightarrow$ いた (*ita*)" (*i*-omission), and "れる (*reru*) $\rightarrow$ られる (*rareru*)" (*ra*-omission). Both *i*- and *ra*-omissions are considered inappropriate for formal writing. For kanji-conversion, there were typo–correction pairs with similar meanings. For example, the media and entertainment industry distinguishes the homonymous pair "製作 (*seisaku*) $\rightarrow$ 制作 (*seisaku*)". The latter refers to production in a narrow sense while the former covers production management. However, people outside of the industry usually are not aware of the distinction. Perhaps for difficulty in differentiating them, we noticed that some typo–correction pairs did not seem to be genuine typo corrections but were acceptable variants. Further work is needed to remove them.

The top 5 most frequent typo–correction pairs in terms of parts-of-speech are shown in Table 4. There were many typo–correction pairs related to particles in substitution, deletion, and insertion while there were many typo–correction pairs related to nouns in kanji-conversion. The special part-of-speech "Undefined" in the substitution category were mostly related to katakana proper nouns.

5 Evaluation through Crowdsourcing

By using crowdsourcing, we evaluated the dataset. We randomly sampled 2,996 article pages and evaluated sentence pairs extracted from them. They consisted of 1,861 substitutions, 2,147 deletions, 2,395 insertions, and 4,388 kanji-conversions. To prevent data bias, we sampled only 3 sentence pairs and discarded the others if an article page contains the same corrections more than 3 times.

Typo	Before filtering	After filtering
Substitution	680,097	86,742
Deletion	490,673	89,428
Insertion	407,413	110,305
Kanji-conversion	296,757	240,219
Total	1,874,940	526,694

Table 2: Number of typo–correction pairs before and after filtering.

5.1 Task

For each sentence pair, we presented the pre- and post-revision sentences and asked whether these are natural or unnatural in written-style texts. Note that we added the phrase "in written-style text" because, as we have seen above, Wikipedia tends to remove colloquialism. We randomly swapped pre- and post-revision sentences and presented them simply as sentences A and B so that crowdworkers were unable to figure out which is the pre-revision sentence. We asked crowdworkers to choose one from the five choices: "A is natural in written-style text, but B is unnatural", "B is natural in written-style text, but A is unnatural", "Both are natural in written-style texts", "Both are unnatural in written-style text", and "Not sure". Each sentence pair was shown to 10 crowdworkers.

5.2 Results

After aggregating crowdworkers' answers, we classified pairs with single majority votes as follows: "Correct revision" if the pre-revision sentence was unnatural and the post-revision sentence was natural, "Bad revision" if the pre-revision sentence was natural and the post-revision sentence was unnatural, "Both natural" if the choice was "Both are natural in written-style text", "Both unnatural" if the choice was "Both are unnatural in written-style text", and "Not sure" if the choice was "Not sure". We classified pairs into "Other" if no choice got more than 5 votes.

The classification results are listed in Table 5. 83.0% of substitution, 77.6% of deletion, 88.8% of insertion, and 69.8% of kanji-conversion were classified as "Correct revision". "Both natural" was more frequent in the deletion category than in the other categories, and 41% of this typo group

Substitution		Deletion		Insertion		Kanji-conversion	
の (*no*) → を (*wo*)	4.0%	る (*ru*) → いる (*iru*)	13.3%	-の (*no*)	14.2%	製作 → 制作 (*seisaku*)	2.8%
の (*no*) → に (*ni*)	3.5%	+に (*ni*)	10.9%	-を (*wo*)	11.3%	始めて → 初めて (*hazimete*)	1.4%
づつ (*dutu*) → ずつ (*zutu*)	3.1%	+を (wo)	10.4%	-に (*ni*)	10.5%	制作 → 製作 (*seisaku*)	1.0%
を (*wo*) → の (*no*)	2.7%	+と (*to*)	5.1%	-は (*ha*)	6.6%	運行 → 運航 (*unkou*)	1.0%
の (*no*) → が (*ga*)	1.9%	た (*ta*) → いた (*ita*)	4.5%	-が (*ga*)	6.5%	後 → 跡 (*ato*)	0.9%
に (*ni*)→ の (*no*)	1.8%	+が (*ga*)	4.3%	-と (*to*)	4.7%	作詩 → 作詞 (*sakusi*)	0.9%
を (*wo*)→ が (*ga*)	1.8%	+の (*no*)	3.8%	-で (*de*)	4.4%	勤めた → 務めた (*tutometa*)	0.8%
が (*ga*)→ を (*wo*)	1.3%	れ (*re*)→ れて (*rete*)	2.6%	-い (*i*)	1.7%	勤める → 務める (*tutomeru*)	0.7%
か (*ka*)→ が (*ga*)	1.1%	+い (*i*)	2.2%	-た (*ta*)	1.5%	開放 → 解放 (*kaihou*)	0.5%
と (*to*)→ を (*wo*)	1.0%	れる (*reru*)→ られる (*rareru*)	1.3%	-し (*si*)	1.5%	付属 → 附属 (*fuzoku*)	0.5%

Table 3: Top 10 most frequent typo–correction pairs in JWTD. In kanji-conversion, the both hand sides are the same reading.

Substitution		Deletion		Insertion		Kanji-conversion	
Particle → Particle	28.5%	+Particle	36.8%	-Particle	57.5%	Noun → Noun	57.3%
Undefined → Noun	6.9%	Suffix → Suffix	29.6%	-Suffix	6.5%	Verb → Verb	17.0%
Noun → Noun	6.5%	Verb → Verb	5.5%	-Noun	4.4%	Noun/Noun → Noun	1.7%
Verb → Verb	5.9%	Noun → Noun	2.9%	Verb/Suffix → Verb	2.5%	Suffix → Suffix	1.6%
Noun → Suffix	3.4%	+Suffix	2.5%	-Verb	2.3%	Noun → Suffix	1.4%

Table 4: Top 5 most frequent typo–correction pairs in terms of parts-of-speech in JWTD.

Typo	Correct revision	Bad revision	Both natural	Both unnatural	Not sure	Other
Subst.	83.0	0.3	6.8	0.1	0.2	9.6
Deletion	77.6	0.1	13.1	0.0	0.0	9.3
Insertion	88.8	0.4	7.1	0.0	0.0	3.6
Kanji-conv.	69.8	7.5	3.1	0.0	0.1	19.5

Table 5: Results of crowdsourcing.

concerned *i*-omission. In our view, they should have been classified as "Correct revision" because *i*-omission is considered inappropriate in formal writing, but crowdworkers turned out to be tolerant of colloquialism. "Other" of kanji-conversion was more frequent than those of other categories. This means that the answers of crowdworkers were diverse. We conjecture that judging whether kanji is correct or not needs higher-level knowledge of kanji. Some pairs that should have been classified as "Correct revision" were classified as "Other" or "Bad revision". These imply that the quality of deletion and kanji-conversion was better than the scores indicate.

6 A Typo Correction System using JWTD

We built a baseline typo correction system using JWTD.

6.1 Settings

We used OpenNMT (Klein et al., 2017)[12], a Python toolkit of encoder-decoder-based machine translation, as a typo correction system. We trained the model separately for each category of typos. For training and validation, we used sentence pairs not used in the crowdsourced evaluation. The training set contained 79,714 substitutions, 82,227 deletions, 102,897 insertions, and 230,490 kanji-conversions and the validation set contained 5,000 sentence pairs of each category. The test set contained 1,689 substitutions[13], 1,665 deletion, 2,127 insertion, and 3,061 kanji-conversion sentence pairs classified as "Correct revision" as the result of crowdsourcing. The training and validation sets were constrained to be distinct from the test set at the level of article pages because of the sampling method used in the crowdsourced evaluation. We compared morpheme-level and character-level representations of inputs and outputs. For the OpenNMT settings, we set the train step as 200,000, and learning rate as 0.5. We used the default settings for the others: the encoder and decoder were 2-layer RNNs and the embedding size

[12]https://github.com/OpenNMT/OpenNMT-py

[13]In the article pages sampled for the crowdsourcing, 144 sentence pairs of substitution were almost indistinguishable to the human eye, such as revisions related to "へ (*he*)" (hiragana) and "ヘ (*he*)" (katakana). We removed these sentence pairs from crowdsourcing-based evaluation, but not from the test set for the typo correction evaluation. We manually evaluated these and confirmed that all of them are correct.

Model	Typo	P	R	$F_{0.5}$	Match	SARI
Morph	Subst.	11.9	39.8	13.8	31.6	61.2
	Deletion	23.2	69.9	26.7	60.9	80.2
	Insertion	16.8	79.7	19.9	69.3	83.8
	Kanji-conv.	30.8	57.0	33.9	48.7	71.0
Char	Subst	4.5	37.2	5.5	25.2	54.2
	Deletion	6.5	59.4	8.0	44.7	69.6
	Insertion	6.2	76.0	7.6	51.8	72.5
	Kanji-conv	10.0	43.5	11.8	33.7	60.9

Table 6: Results of the typo correction experiment.

and the hidden size were 500.

6.2 Evaluation metrics

Our evaluation metrics were precision, recall and $F_{0.5}$ score in typo correction, the percentage of exact matches between system outputs and references, and SARI (Xu et al., 2016). We defined precision and recall as follows. For each sentence, we calculate the character-level minimum edits from the input to the gold G, the character-level minimum edits from the input to the system output O, and $G \cap O$. Let N_G, N_O, and $N_{G \cap O}$ be the sums of $|G|$, $|O|$, and $|G \cap O|$ in all sentences, respectively. We calculated Precision = $N_{G \cap O}/N_O$ and Recall = $N_{G \cap O}/N_G$. We used the Python3 python-Levenshtein library[14] for calculating minimum edits. SARI is a metric for text editing. This calculates the averaged F1 scores of the added, kept, and deleted n-grams. We used character-level 4-gram.[15]

6.3 Results

The results of the experiment are presented in Table 6. The morpheme-level model outperformed the character-level model in all categories with large margins. It is interesting that for morpheme-level insertion, the precision scores were low while the exact match scores were high. In some sentences, text generation did not go well and the same token was generated repeatedly, greatly lowering precision. The SARI scores indicate improvement for all categories, given that the output identical to the input yields a score of about 30.

[14]https://pypi.org/project/python-Levenshtein/

[15]We used the implementation available at https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/sari_hook.py, setting β-deletion = 1 (Geva et al., 2019).

7 Conclusion

In this paper, we built a Japanese Wikipedia typo dataset (JWTD) which contains over half a million typo–correction sentence pairs obtained from Wikipedia's revision history. We classified Japanese typos into four categories and presented mining procedures for each of them. We evaluated JWTD using crowdsourcing and built a baseline typo correction system on top of it. To the best of our knowledge, JWTD is the first freely available large Japanese typo dataset.

While the focus of this paper was on data construction, developing a higher-quality typo correction system is the future direction to pursue. It is also interesting to apply our methods to other languages requiring word segmentation, most notably, Chinese.

References

Yukino Baba and Hisami Suzuki. 2012. How are spelling errors generated and corrected?: a study of corrected and uncorrected spelling errors using keystroke logs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 373–377. Association for Computational Linguistics.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Su Lin Blodgett, Lisa Green, and Brendan O'Connor. 2016. Demographic dialectal variation in social media: A case study of african-american english. *arXiv preprint arXiv:1608.08868*.

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!!: using word lengthening to detect sentiment in microblogs. In *Proceedings of the conference on empirical methods in natural language processing*, pages 562–570. Association for Computational Linguistics.

Mor Geva, Eric Malmi, Idan Szpektor, and Jonathan Berant. 2019. Discofuse: A large-scale dataset for discourse-based sentence fusion. 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 3443–3455.

Gili Goldin, Ella Rabinovich, and Shuly Wintner. 2018. Native language identification with user generated content. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3591–3601.

Masato Hagiwara and Masato Mita. 2020. Github typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC 2020)*. To appear.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

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.

Lung-Hao Lee, Yuen-Hsien Tseng, and Li-Ping Chang. 2018. Building a TOCFL learner corpus for Chinese grammatical error diagnosis. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Aurélien Max and Guillaume Wisniewski. 2010. Mining naturally-occurring corrections and paraphrases from Wikipedia' s revision history. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC' 10)*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Itsumi Saito, Jun Suzuki, Kyosuke Nishida, Kugatsu Sadamitsu, Satoshi Kobashikawa, Ryo Masumura, Yuji Matsumoto, and Junji Tomita. 2017. Improving neural text normalization with data augmentation at character-and morphological levels. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 257–262.

Keisuke Sakaguchi, Kevin Duh, Matt Post, and Benjamin Van Durme. 2017. Robsut wrod reocginiton via semi-character recurrent neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*.

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*, pages 162–170.

Arseny Tolmachev, Daisuke Kawahara, and Sadao Kurohashi. 2018. Juman++: A morphological analysis toolkit for scriptio continua. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 54–59, Brussels, Belgium. Association for Computational Linguistics.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Torsten Zesch. 2012. Measuring contextual fitness using error contexts extracted from the Wikipedia revision history. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 529–538. Association for Computational Linguistics.