NORMALIZATION OF TRANSLITERATED WORDS USING SEQ2SEQ MODEL WITH SPELL CHECKER

Byambadorj Zolzaya[1], Ryota Nishimura[1], Ayush Altangerel[2], Norihide Kitaoka[3]

[1]*Department of Advanced Technology and Science, Tokushima University, Tokushima, Japan*
[2]*Department of Information Technology, Mongolian University of Science and Technology, Ulaanbaatar, Mongolia*
[3]*Department of Computer Science and Engineering, Toyohashi University of Technology, Toyohashi, Japan*
C501947001@tokushima-u.ac.jp, nishimura@is.tokushima-u.ac.jp, a.altangerel@must.edu.mn, kitaoka@tut.jp

1. Introduction

There are two written systems in Mongolian language:
-Classic Mongolian (Uyghur Mongolian)
-Cyrillic
Both of them are used in Mongolia. The Mongolian People's Republic, as it was called then, first started using a modified Russian Cyrillic alphabet in 1940 which is still used and the official written system. Mongolian Cyrillic has 35 characters. Even though the official written system is the Cyrillic script before as mentioned, recently many people use Latin alphabets to write text on social media like Facebook and Twitter. While writing transliterated text using the Roman script on social media, there is no rule. Therefore, one word can be written in different forms. The text processing of social media is one of the important subjects in NLP. Therefore, in the last years, there has been a lot of work that focuses on social media. But there is a lack of research in this area for the Mongolian language and this is the first study of text normalization for Mongolian.
Text normalization is a pre-processing stage for speech and language processing applications. At first, text normalization was to convert words in non-standard forms such as numbers, dates, acronyms, and abbreviations to standard forms in the formal text. But later this content was expanded to convert informal text on social media into formal text. Both source and target texts are the same languages in the most research work of noisy text normalizations. In our case, it is a little bit different and our purpose is to convert noisy transliterated text on social media to the formal style. In other words, the scripts of source and target texts are different, Roman and Cyrillic scripts, respectively.

2. Related works

With the increase of noisy texts, noisy text normalization has become one of the hot topics in NLP. Aw et al. (2006) used a phrase-based statistical model for SMS text normalization. Vilari˜no et al. (2012) used machine translation techniques, a statistical bilingual dictionary constructed using the IBM-4 model, to normalize SMS texts. Saloot et al. (2014) implemented an unsupervised normalization system that had two phases for noisy text normalization: candidate generation and candidate selection. Six methods such as one-edit distance lexical generation, phonemically generation, blending the previous methods, two-edit distance lexical generation, dictionary translation, and heuristic rules were used to generate candidates. The language model probability score was used to select the most appropriate candidate. Kaur and Mann et al. (2016) implemented a hybrid approach consisted of SMT and direct mapping to transform a non-standard text into standard text.
Recently, neural methods for machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014) are also used for text normalization. Ikeda et al. (2016) used a character level encoder-decoder model for normalizing of Japanese noisy text. They also built a synthetic database with predefined rules for data augmentation and compared their neural network model with a rule-based method and CRF. Lusetti et al. (2108) normalized Swiss German WhatsApp messages using a neural network model. They integrated the language model into the character level neural model and compared state-of-the-art CSMT with their model. Lourentzou et al. (2019) used a hybrid seq2seq model which consisted of two nested encoder and decoder architectures: word and character level seq2seq models. When an unknown symbol is encountered when using the word level seq2seq model, the second character level seq2seq model is used for OOV. Their hybrid model achieved the best performance so far among neural models in this related works, but the performance of their model was lower than some traditional methods. Mager et al. (2019) proposed an auxiliary task for the sequence-to-sequence neural architecture novel to the text normalization task, which improved the base seq2seq model up to 5%. This increase of performance closed the gap between statistical machine translation approaches and neural ones for low-resource text normalization. Mandal and Nanmaran et al. (2018) normalized noisy transliterated Bengali words in Roman into words in the native script of Bengali. They used a seq2seq model and Levenshtein distance algorithm for normalization of the transliterated words. Tursun and Cakici et al. (2017) normalized Uyghur text using of Latin alphabets into text using CTA /Common Turkic alphabets/ and compared the noisy channel model and neural encoder-decoder model as normalizing methods. They picked the character-based solution in the encoder-decoder model but chose the word-based solution for the noisy channel model. They also used both synthetic and authentic data. The last two researches are similar to our research, because transliterated texts written in Latin alphabets were normalized into canonical form in other scripts.

3. Dataset

In the experiment, two kinds of data sets were prepared. One is real data consisting of 2200 sentences in Roman script collected from social media. These sentences were split by word and canonical forms of the noisy texts were created manually. When writing a transliterated text on social media, one Latin letter can be used for many alternatives of Cyrillic letters. It can happen to especially 'o', 'ө', 'y', 'ү' Cyrillic letters. Table 1 shows standard transcriptions of these 4 letters, but we almost don't use **ü, ö** transcriptions. Therefore, only 2 Latin letters, 'u', 'o', are used for these letters as in Table 2.

Table 1. Standard transcriptions of some Cyrillic letters

Cyrillic letter	Standard transcription	Cyrillic letter	Standard transcription
y	u	ү	**ü**
o	o	ө	**ö**

Table 2. Nonstandard transcriptions of some Cyrillic letters

Latin letter	Possible alternatives
u	‘ө’ ,’y’, ‘ү’
o	‘o’, ‘ө’

We collected 7267 canonical words which begin these 4 letters. Then the words were transliterated correctly in Roman and added into the training dataset. These are not noisy data. Our training dataset is the word pairs dataset and statistics of the training dataset are shown in Table 3. The training dataset was used all different neural and statistical models.

Table 3. Statistics of the dataset

Description	Input	Output
Total words	24073	25023
Total characters	151784	150714
Distinct words	14197	12030
Unique token	30	42
Max length of sequence	21	22
Average length of sequence	7	6

We also prepared test data consisting of 200 sentences collected from social media as shown in Table 4.

Table 4. Statistics of Test data

Description	Test data
Total words	1663
Total characters	9112
Distinct words	1178
Known words in the training data	970 (58%)
Unknown words in the training data	694 (42%)

Two more data corpora were prepared. One is target language monolingual corpus (Table 5) used to train a language model. Other is the transliteration corpus (Table 6) used to train a transliteration model in phrase-based statistical machine translation. It contains word alignments which are standard transliterated words in Roman and their respective canonical words in Cyrillic.

Table 5. Data corpus used to train a Language Model

Description	Monolingual text data
Total sentences	24000
Total words	27960

Table 6. Data corpus used to train a Transliteration model in a SMT model

Description	Standard transliterated words	Canonical Cyrillic words
Total distinct words	7680	7680
Total characters	62882	57959
Unique token	23	35

4. Method

We implemented different character level neural seq2seq models and the best achieved method has 2 stages. The character level seq2seq model was used to normalize noisy transliterated text in the first stage. In the next stage, we used the edit distance and a neural language model to correct the output of the first model.

4.1 Character level sequence to sequence model

We built a character level sequence to sequence model with attention (Bahdanau et al., 2015) to normalize noisy text (Figure 1). The model learns to map user transliterations to their canonical form. The sequence to sequence model has encoder and decoder recurrent neural networks. An **encoder** processes the input sequence and compresses the information into context vectors of a *fixed length*. A **decoder** is initialized with context vectors to generate the transformed output.

Encoder produces hidden states of each element in the input sequence. Bahdanau’s alignment score function (1) is used to calculate alignment scores between the previous decoder hidden state and each of the encoder’s hidden states.

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^T \bar{h}_s & dot \\ h_t^T W_a \bar{h}_s & general \\ \boldsymbol{v_a^T} \tanh(\boldsymbol{W_a}[\boldsymbol{h_t}; \bar{\boldsymbol{h}}_s]) & concat \end{cases} \quad (1)$$

Alignment weights are calculated for each hidden state of encoder (2).

$$\alpha_{ts} = \frac{\exp(score(h_{t-1}, \bar{h}_s))}{\sum_{s'=1}^{S} \exp(score(h_{t-1}, \bar{h}_{s'}))} \quad (2)$$

Each encoder hidden state is multiplied by corresponding alignment weight and they are summed up to produce the *context vector* (3).

$$c_t = \sum_s \alpha_{ts} \bar{h}_s \ , \ (3)$$

for each t. The context vector is concatenated with the previous decoder output and fed into the Decoder RNN. Then it produces a new output.

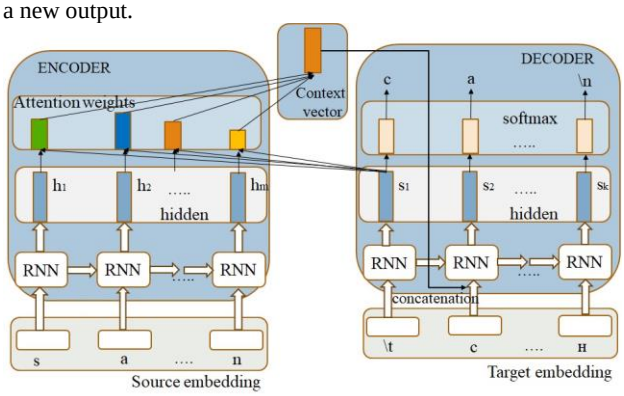

Figure 1. Architecture of Seq2Seq model with attention

The character level neural model used the following hyperparameters. Both encoder and decoder models used a single GRU layer with 512 hidden units and the size of the embedding vector was 250. Adam optimizer with the learning rate 0.0002 was used and the batch size was 32.

4.2 Edit distance and Neural language model

In the next stage, our algorithm was used to improve the output of the first model. The algorithm used 2 methods. One is an edit distance method to generate all possible candidates from incorrectly normalized words. A dictionary was created by using the monolingual corpus shown in Table 5 and was used for candidate generation. We used one and two edits methods to generate candidates.

Another method is a word-level neural language model shown in Figure 2 to select the most appropriate candidate. The neural language model was trained on the monolingual corpus shown

in Table 5. It learned the probability of occurrence of a word based on the previous sequence of words used in the text. In other words, the model computes the probability of occurrence of sequence as output.

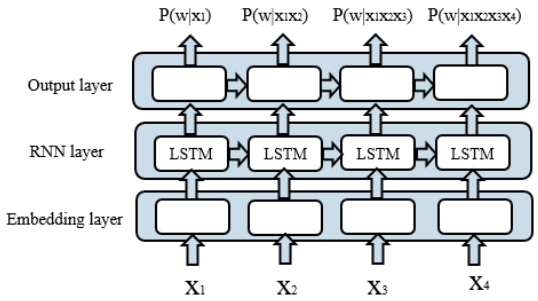

Figure 2. Architecture of Neural language model

Our neural language model has an Embedding layer, a single LSTM layer, and a Dense layer. The neural model used the following hyperparameters. Adam optimizer with a learning rate of 0.001 was used in the model. The model had a single LSTM layer with 100 hidden units. The size of the embedding vector was 50. The batch size was 32.

5. Statistical machine translation

In this section, we give a short description of our baseline model that is based on the phrase-based statistical machine translation (SMT) implemented by using the Moses tool. The SMT generates translations based on statistical models including translation model and language model, whose parameters are derived from the analysis of bilingual text corpora. The phrase-based SMT was used for our noisy text normalization task. The training word pairs dataset and monolingual corpus shown in Table 3 and Table 5 were used to train the translation model and the language model, respectively. The transliteration module has been integrated into Moses and the module is completely unsupervised and language independent. Moses builds the transliteration model from the transliteration corpus. The transliteration corpus shown in Table 6 was used to train the transliteration model. First, we used the transliteration model to normalize all words in the test data. Second, the transliteration model was used to normalize only OOV in the test data when using SMT. These two results were compared with the output of neural models. Figure 3 shows the architecture of SMT.

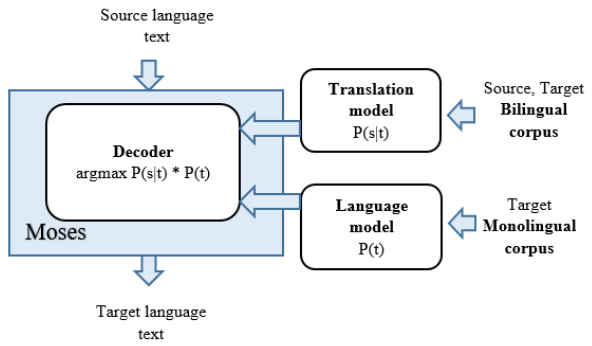

Figure 3. Architecture of SMT

The decoder calculates $p(t|s)$ and t is the translation result of s. After using Bayes theorem, the problem can be expressed as below (4).

$$p(t|s) \propto p(s|t)p(t) \qquad (4)$$

The translation model $p(s|t)$ is the probability of translation, and the language model $p(t)$is the expression of fluency of the sentence. The system outputs the best translation $\tilde{e}$ which is done by picking up the one with the highest probability (5).

$$\tilde{e} = arg \max_{e \in e^*} p(t|s) = arg \max_{e \in e^*} p(s|t)p(t) \qquad (5)$$

6. Experiment

We implemented different character-level neural seq2seq models and compared the results of all the neural models with SMT. Table 7 shows the results. The first two models are our baseline models which are the transliteration model of SMT and the SMT. The next two models are the original seq2seq model with and without attention. After that, there are two seq2seq models with and without attention, but differences from the previous two models are using the beam search and a pre-trained neural language model. The bottom four models are the same and two stages methods. The difference between these models is that statistical and neural language models were used in the second stage. All models were trained on the same training corpus. Two types of accuracies which are word-level (6) and character-level (7) were calculated to evaluate the performance of all models.

$$AccWord = \frac{Number\ of\ correct\ words}{Number\ of\ total\ words} \quad (6)$$

$$AccChar = \frac{Number\ of\ correct\ characters}{Number\ of\ total\ characters} \quad (7)$$

Table 7. WORD and CHARACTER level accuracies of the test data

Model	WORD level accuracy	CHARACTER level accuracy
Baseline model: TM in SMT	*80.51%*	*92.24%*
Baseline model: SMT	*82.08%*	*92.97%*
M1	77.51%	93.02%
M2	77.87%	93.51%
M1 with BS+NLM	81.05%	92.66%
M2 with BS+NLM	82.20%	93.06%
M1 with ED+SLM	83.34%	92.60%
M2 with ED+SLM	84.72%	93.70%
M1 with ED+NLM	85.08%	93.37%
M2 with ED+NLM	**86.04%**	**94.25%**

/ TM – transliteration model, SMT – statistical machine translation, M1 and M2 – seq2seq model without and with attention, respectively, BS – beam search, NLM – neural language model, ED – edit distance, SLM – statistical language model /

All experiments showed that the performance of the seq2seq model with attention was better than the seq2seq model without attention. Also when using only the transliteration model of SMT to normalize noisy text, output was lower than SMT. Original seq2seq models achieved the worst and accuracies were lower than all other models. But when using the seq2seq model with the beam search and language model, results were almost the same with baseline. All results of two stages method (bottom four experiments in Table 7) were higher than the baselines and other neural models. The best word level and

character level accuracies, 86.04% and 94.25%, respectively, were obtained by the seq2seq with attention followed by edit distance and word-level neural language model.
Tables 8 and 9 give the detailed results of the different models. SMT achieved the best performance to normalize IV, however, the performance of our 2 stages method to normalize OOV was the best in all experiments.

Table 8. WORD level accuracy of IV and OOV in the test data

Model	WORD level accuracy	
	Accuracy of IV	Accuracy of OOV
Baseline model: TM in SMT	*93.60%*	*62.19%*
Baseline model: SMT	*96.28%*	*62.19%*
M1	95.05%	52.95%
M2	94.94%	53.96%
M1 with BS+NLM	95.36%	61.03%
M2 with BS+NLM	94.74%	64.64%
M1 with ED+SLM	95.56%	66.23%
M2 with ED+SLM	92.15%	70.12%
M1 with ED+NLM	95.56%	70.41%
M2 with ED+NLM	**95.25%**	**73.16%**

Table 9. CHARACTER level accuracy of IV and OOV in the test data

Model	CHARACTER level accuracy	
	Accuracy of IV	Accuracy of OOV
Baseline model: TM in SMT	*97.45%*	*87.58%*
Baseline model: SMT	*99.00%*	*87.58%*
M1	98.49%	88.14%
M2	98.51%	89.04%
M1 with BS+NLM	98.36%	87.58%
M2 with BS+NLM	98.29%	88.39%
M1 with ED+SLM	98.51%	87.33%
M2 with ED+SLM	98.47%	89.46%
M1 with ED+NLM	98.51%	88.79%
M2 with ED+NLM	**98.54%**	**90.42%**

7. Conclusion

We have shown in this paper that different neural models for normalization of noisy text and compared their performance with traditional statistical machine-translation method. Our two stages approach, the seq2seq model for initial normalization followed by edit distance and neural language model, could increase the accuracy of SMT and got the accuracy of 86.04% on the test data. In future work, we would like to increase the noisy data size and also identify the language of code-mixed English-Cyrillic sentence to improve noisy text normalization.

8. References

(Cho et al., 2014) Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "*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.

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

(Bahdanau et al., 2015) Dzmitry Bahdanau, KyungHyun Cho and Yoshua Bengio, "*Neural machine translation by jointly learning to align and translate*", published as a conference paper at ICLR 2015

(Luong et al., 2015) Minh-Thang Luong, Pham Hieu, and Christopher D. Manning, "*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.

(Mandal and Nanmaran et al., 2018) Soumil Mandal and Karthick Nanmaran , "*Normalization of Transliterated Words in Code-Mixed Data Using Seq2Seq Model & Levenshtein Distance"*, 2018 The 4th Workshop on Noisy User-generated Text (W-NUT), Nov 1, 2018, Brussels, Belgium (at EMNLP 2018)

(Ikeda et al., 2016) Taishi Ikeda, Hiroyuki Shindo and Yuji Matsumoto," Japanese *Text Normalization with Encoder-Decoder Model"*, 2016 The 2nd Workshop on Noisy User-generated Text (W-NUT), December 11, 2016, Osaka, Japan

(Tursun and Cakici et al., 2017) Osman Tursun and Ruket Cakici, "*Noisy Uyghur Text Normalization*", 2017 The 3rd Workshop on Noisy User-generated Text (W-NUT), September 7th, Copenhagen (at EMNLP 2017)

(Lourentzou et al., 2019) Ismini Lourentzou, Kabir Manghnani and ChengXiang Zhai, "*Adapting Sequence to Sequence models for Text Normalization in Social Media*", Association for the Advancement of Artificial Intelligence, 13th International AAAI Conference on Web and Social Media, *12 Apr 2019*

(Lutti et al., 2018) Massimo Lusetti, Tatyana Ruzsics, Anne Göhring, Tanja Samardžić and Elisabeth Stark, "*Encoder-Decoder methods for text normalization*", The fifth workshopon NLP for similar languages, Varieties and Dialects, August 20, 2018

(Mager et al., 2019) Manuel Mager, Monica Jasso Rosales, Ozlem Cetinoglu and Ivan Meza, "*Low-resource neural character-based noisy text normalization*", Journal of Intelligent & Fuzzy Systems 36 (2019) 4921-4929, DOI: 10.3233/JIFS-179039, IOS Press

(Saloot et al., 2014) Mohammad Arshi Saloot, Norisma Idris and AiTi Aw, *"Noisy text normalization using an enhanced language model"*, Proceedings of the International Conference on Artificial Intelligence and Pattern Recognition, Kuala Lumpur, Malaysia, 2014

(Aw et al., 2006) AiTi Aw, Min Zhang, Juan Xiao and Jian Su, "*A Phrase-based Statistical Model for SMS Text Normalization*", Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 33–40, Sydney, July 2006. c 2006 Association for Computational Linguistics

(Vilari˜no, et al., 2012) Darnes Vilari˜no, David Pinto, Beatriz Beltr´an, Saul Le´on, Esteban Castillo, and Mireya Tovar, "*A Machine-Translation Method for Normalization of SMS*", J.A. Carrasco-Ochoa et al. (Eds.): MCPR 2012, LNCS 7329, pp. 293–302, 2012. c Springer-Verlag Berlin Heidelberg 2012

(Kaur and Mann et al., 2016) Harpreet Kaur and Er. Jasdeep Singh Mann, "*Text Normalization using Statistical Machine Approach*", International Research Journal of Engineering and Technology (IRJET), 08 Aug -2016