IMPROVING NEURAL TEXT NORMALIZATION WITH PARTIAL PARAMETER GENERATOR AND POINTER-GENERATOR NETWORK

Weiwei Jiang[1,2], *Junjie Li*[1], *Minchuan Chen*[1], *Jun Ma*[1], *Shaojun Wang*[1], *Jing Xiao*[1]

[1]Ping An Technology
[2]Columbia University
{jiangweiwei562, lijunjie815}@pingan.com.cn

ABSTRACT

Text Normalization (TN) is an essential part in conversational systems like text-to-speech synthesis (TTS) and automatic speech recognition (ASR). It is a process of transforming non-standard words (NSW) into a representation of how the words are to be spoken. Existing approaches to TN are mainly rule-based or hybrid systems, which require abundant hand-crafted rules. In this paper, we treat TN as a neural machine translation problem and present a pure data-driven TN system using Transformer framework. Partial Parameter Generator (PPG) and Pointer-Generator Network (PGN) are combined in our model to improve accuracy of normalization and act as auxiliary modules to reduce the number of simple errors. The experiments demonstrate that our proposed model reaches remarkable performance on various semiotic classes.

***Index Terms*—** Text Normalization, Parameter Generator, Pointer-Generator Network, Text-To-Speech

1. INTRODUCTION

Non-standard words (NSW) such as number, currency amounts, abbreviations and dates are common in real text forms [1]. Typically, they are not in the representation of fully verbalized forms and their pronunciations cannot simply be identified by applying 'letter-to-sound' rules. For example, one would probably use '$ 1.99' instead of "one dollars and ninety-nine cents" in written form and "4:10" will be read differently in context of time or the score of a game. Thus, a text normalization component which transforms NSW into their corresponding standard words to determine their pronunciations plays an essential role in systems like text-to-speech synthesis (TTS) and automatic speech recognition (ASR).

In general, the difficulties of Text Normalization mainly lie in two aspects: the variety of semiotic classes [2] and their high degree of ambiguity. According to van Esch and Sproat's work [3], the taxonomy of NSW can be divided roughly into twelve major categories: Wordlike Tokens, Numbers, Measures etc., and fine-grained classes within these major categories often has a quite different verbalization. For NSW in a sentence, a change in the surrounding context will lead to a different verbalization, which makes Text Normalization more complicated and difficult.

Traditionally, a feasible approach to handle text normalization problems is constructing a rule-based system using Weighted Finite State Transducers(WFSTs) with hand-written grammars for every fined-grained NSW class in a specific language [4] [5] [6]. However, this approach is time-consuming and requires the knowledge of linguistics. More recently, machine learning methods have achieved remarkable progress in NLP studies and bring more and more data driven methods in Text Normalization. Sproat and Jaitly [7] proposed a hybrid method which combined a recurrent neural network (RNN) architecture with an FST-based grammar filter. In [8], the author used a seq2seq model with byte pair encoding (BPE) as subword units without grammar. And the use of convolutional networks(CNN) and differentiable neural computer (DNC) for text normalization was separately examined by [9] [10]. Furthermore, aiming to make a Neural Text-to-speech fronted, Conkie and Finch designed a sequence-to-sequence model to handle both Text Normalization and pronunciations on a sentence level [11].

Although these seq2seq models have already achieved very good overall scores, there still exist several downsides [12]. As mentioned in [7], the performance of seq2seq model in cases such as dates and digits are less compelling. The machine learning models are prone to make silly errors, like substituting "hours" for "gigabytes" and changing words which should be kept the same.

To tackle the issues above, we present a pure data-driven text normalization system which directly transforms non-standard words to their standard spoken counterparts. First, an NSW extractor is used to filter naive semiotic classes from the input sentence. Then, we treat normalization as a neural machine translation problem. Bearing in mind that Transformer [13] obtains excellent performance on translation, we implement our normalization generator essentially based on it. Additionally, aiming to reduce simple errors generated by our system, we integrate Partial Parameter Generator and Pointer-Generator Network in Transformer, which we will elaborate on in section 3.

The rest of the paper is organized as follows: Section 2 and Section 3 introduce the detailed structure of our baseline method and proposed model. Section 4 presents the experiment details and evaluation results. And the conclusion is drawn in Section 5.

2. BASELINE MODEL

The baseline approach follows a Bi-GRU model with attention mechanism as in Bahdanau's work [14] on window-based data. A typical window-based training data is like "lives at ⟨norm⟩ 123 ⟨\norm⟩ King Avenue" and the grouping results are provided by the dataset [7]. ⟨norm⟩ and ⟨\norm⟩ indicate the center of the window. The window size N denotes N previous and following tokens of the center word are included. The output tokens are corresponded to the verbalization form of the central words of the window. For the sake of limiting the output vocabulary size, the output of standard words is a special token ⟨self⟩. Subword units are used both in encoder and decoder, since [8] has shown that model with subword units performs better in rare words and out of vocabulary situation than character or word level counterpart.

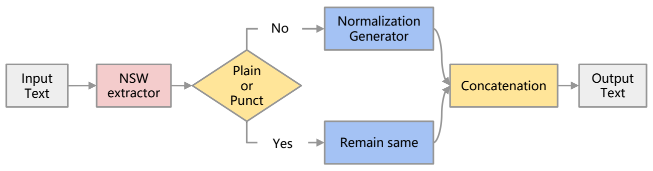

Fig. 1. Flowchart of Proposed Model

3. PROPOSED MODEL

We propose our text normalization system as in Fig. 1. For a given input text, Extreme Gradient Boosting (XGBoost) [15] is used to classify tokens into one of the semiotic classes. Then those non-plain classes will be processed in our normalization generator part. In our approach, we regard normalization generator equivalent to a one-to-many translation model, which is mainly based on the Transformer Framework [13], since one NSW sequence would have a different normalization when local context changes. Motivated by [16][17], which generate parameters for the encoder and decoder respectively for multilingual neural machine translation architecture, we adopt Partial Parameter Generator in our transformer decoder. Furthermore, taking the sparsity of each semiotic class into consideration, we implement Pointer-Generator network in our decoder [18]. Finally, we get the output sentence by concatenating the output of our normalization generator with standard words.

3.1. NSW Extraction

Following the convention mentioned at section 2, a sliding window technique is used to represent the local context information. For the ith word w_i in the input text, $[w_{i-k}, w_{i-k+1}, ..., w_i, ..., w_{i+k-1}, w_{i+k}]$ are taken as the input of the XGBoost model, where k is the size of sliding window. For each word in the window, a character level embedding is adopted. The model is trained to classify word w_i into one of the semiotic classes such as DATE, DIGIT et al.

3.2. Partial Parameter Generator

Since all cases share the same source language, we only apply parameter generation to the decoder part. Each semiotic class is regarded as a unique target label l_t. $\mathbf{l_t}$ is the label

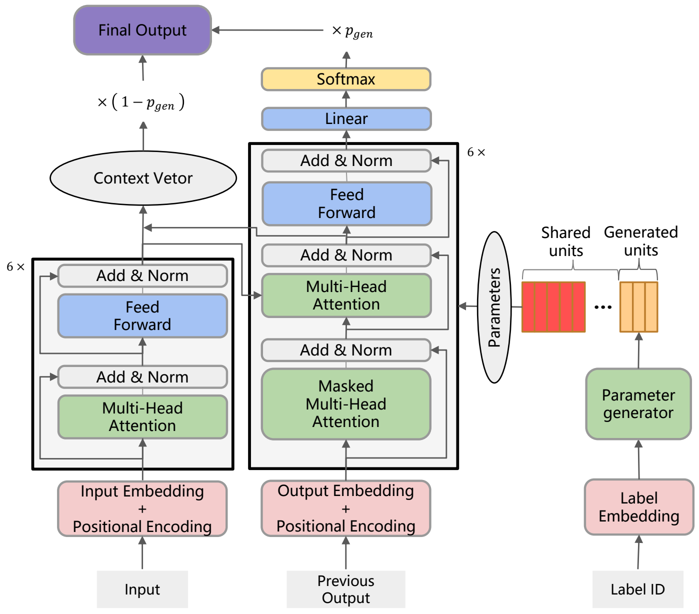

Fig. 2. Overview of our proposed normalization generator

embedding for the target label l_t, which is learned in our experiment. Following [16], we define our parameter generation as a linear transform process.

$$\boldsymbol{\theta}_t^{(gen)} = \boldsymbol{W}^{(dec)}\mathbf{l_t} \tag{1}$$

where $\boldsymbol{W}^{(dec)} \in \mathbb{R}^{P_n \times M}$, $\mathbf{l_t} \in \mathbb{R}^M$, and $\boldsymbol{\theta}_t^{(gen)}$ are the generated parameters, P_n is the number of parameters and M is the size of the label embedding.

To enable cross-label knowledge sharing, partial parameters are materialized in decoder. For each layer of decoder, learnable parameters are divided into two parts: shared units and generated units. The portion of generated units is controlled by a hyper-parameter λ.

3.3. Pointer-Generator Network

It is noticed in the early experiment, our model tends to make simple errors like normalizing "dvd" as "d v b", changing 'd' to ‘b’, or failing to copy word which should be kept the same, like transforming “ultratop.be” to “sixtop dot b e”. The reason for this kind of errors is the imbalance of the output words distribution. Inspired by [18], we introduce a Pointer-Generator network into our transformer model to enhance performance.

For the k_{th} output word of a given input text, a generation probability $p_{gen} \in [0, 1]$ is calculated from the context vector h_k^*, the decoder output O_k and the decoder input x_k.

$$p_{gen} = \sigma(W_{h^*}^T h_k^* + W_o^T O_k + W_x^T x_k + b_{ptr}) \qquad (2)$$

where W_{h^*}, W_o,W_x and b_{ptr} are learnable parameters. For context vector h_k^*, it is derived from the last layer decoder attention score $\boldsymbol{a^k}$ and the final layer encoder output h_k

$$h_k^* = \sum_i a_i^k h_k \qquad (3)$$

Then the final output word probability distribution is

$$P(w) = p_{gen} P_{decoder}(w) + (1 - p_{gen}) \sum_{i:w_i=w} a_i^{k*} \qquad (4)$$

where $P_{decoder}(w)$ is the origin decoder word probability distribution and a_i^{k*} is the attention score scaled by the ratio of input text length d_{text} and vocabulary size $d_{vocabulary}$:

$$a_i^{k*} = a_i^k \times \frac{d_{text}}{d_{vocabulary}} \qquad (5)$$

4. EXPERIMENT

4.1. Dataset

The dataset we used is generated from the publicly available dataset, released by Sproat and Jaity [7]. The original dataset, which contains 1.1 billion parallel corpus and their corresponding semiotic labels, is obtained from Google TTS’s Kestrel text normalization system. Aiming to develop a more general system which can be trained on limited parallel corpus for other languages, we resampled 100k sentences from the original dataset. These sentences are used for training, validation, and testing with the ratio of 8:1:1.

Among the 16 semiotic labels, naive labels including PLAIN and PUNCT account for a large portion of the dataset. The overabundance of ⟨self⟩ and punctuation would disturb our translation model to approximate the reasonable vocabulary distribution which were observed in our early experiment. Therefore, after the NSW extraction processing, these two naive semiotic classes are excluded from our normalization generator. To tackle imbalance dataset problem, oversampling technique is used. Fig 3 is a pie chart of the label distribution.

4.2. System Configuration

Baseline model configuration settings are as follows. Following Zhang [12], we set the window size to be 5 and used a 2-layer bi-direction Gate Recurrent Unit network (Bi-GRU) as encoder and a 2-layer GRU as decoder. Both encoder and decoder have 512 hidden states.

We adopted the OpenNMT [19] toolkit for training and evaluating our Transformer-based model which contains a 6-layer encoder and a 6-layer decoder with 2048-dimentional hidden representation. Wordpiece method [20] is used both in encoder and decoder part and word embeddings are learned from scratch during training. The generated portion λ is 0.5. We used Adam optimizer [21] with learning rate of 8e-6 and gradient clipping with a maximum norm of 0.25. A dropout of 0.2 was used across all modules.

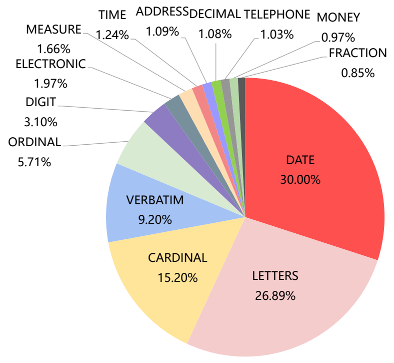

Fig. 3. Semiotic classes distribution for dataset

Table 1. Comparison of different models on test set

Model	SER (%)	WER (%)	BLEU
Identity	92.83	90.45	17.54
Bi-GRU	8.27	5.03	95.45
Model 1 (vanilla)	7.94	3.22	94.94
Model 2 (+ PPG)	3.50	1.45	98.74
Model 3 (+ PGN)	2.91	1.01	99.03
Model 4 (+ PPG & PGN)	**1.90**	**0.71**	**99.52**

4.3. Evaluation and analysis

4.3.1. Overall Performance

To evaluate the performance of the baseline and the proposed system, we use sentence-error-rate (SER), word-error-rate(WER) and BLEU score [22] on the test set.

The results obtained after evaluation are shown in Table 1. For the initial identity model, we simply keep the predicted text exactly the same as the input text. As we can see from the result, the NSW words are remarkably different from standardized spellings. We observe that our model achieves excellent results and significantly outperforms the baseline model in all evaluation indicators.

For better evaluating the performance of our model, we examine the accuracy on semiotic class level in Table 2. It would be considered as an error case if any character in predicted text differs from the corresponding one in reference text. This table illustrates that our model does better than Sproat and Jaity's for most categories and yields very competitive results for the rest class.

Table 2. Model performance on test set

Semiotic class	S&J	Ours
DATE	**0.999**	0.983
LETTERS	0.971	**0.985**
CARDINAL	**0.989**	0.978
VERBATIM	0.980	**1.00**
ORDINAL	0.971	**1.00**
DIGIT	0.865	**0.983**
ELECTRONIC	**1.00**	0.734
MEASURE	**0.986**	0.972
TIME	0.750	**0.995**
ADDRESS	**1.00**	**1.00**
DECIMAL	**1.00**	0.946
TELEPHONE	-	**0.634**
MONEY	**0.972**	0.916
FRACTION	0.923	**0.962**

4.3.2. *Ablation study*

We carry out several experiments to verify the effectiveness of our two auxiliary modules: Partial Parameter Generator and Pointer-Generator Network in Table 1. Model 1 is the vanilla Transformer framework. Model 2-4's configurations are modified based on Model 1's: ② adding Partial Parameter Generator module, ③ adding Pointer-Generator Network module, ④ adding Partial Parameter Generator, and Pointer-Generator network modules. It can be seen from Table 1 that our proposed model—model 4 produces the lowest SER and WER. In addition, the results of the first three models indicate that each auxiliary module makes a positive contribution to the model. For further understanding improvements brought by these two modules, some normalization examples are shown in Table 3.

As shown in Table 3, wrongly normalized words are denoted in red, while correct ones are highlighted in blue. Results in the second column suggest that Pointer-Generator Network module elevate model's performance on generating infrequent words like 'up', which appears far less than 'i' and 'l' in training corpus. In the last column of Table 3, we find that label unique decoder units generated by Partial Parameter Generator prevent our model from interpreting a CARDINAL input '89' as 'eight nine', which is a DIGIT normalization manner.

Table 3. Examples of text normalization

Input Text	1up.com	1089
Expected	one **up** dot com	one thousand **eighty** nine
Model 1 (vanilla)	one **i** dot com	one thousand **eight** nine
Model 2 (+ PPG)	one **l** dot com	one thousand **eighty** nine
Model 3 (+ PGN)	one **up** dot com	one thousand **eight** nine
Model 4 (+ PPG & PGN)	one **up** dot com	one thousand **eighty** nine

4.3.3. *Comparison of generating portion*

Partial Parameter Generator is an essential part in our proposed model. However, to which extent decoder units should be generated as label unique units is an open issue. We conduct an experiment by selecting different generation portion λ and the results are shown in Figure 4. We can observe that the best performance is achieved when we generate half decoder units. It is also noticed that the performance of our model starts to deteriorate when we continue to increase λ from 0.5 to 0.75. One possible and reasonable explanation is that trainable parameters expand along with the increased λ, bringing more difficulties in training and inference when training corpus is limited and sparse.

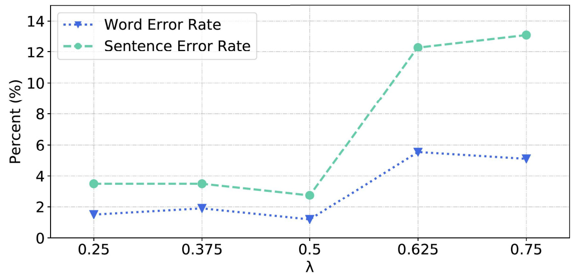

Fig. 4. Comparison of different generating portion

5. CONCLUSION

In this paper, we propose a data-driven based text normalization system using Transformer for English without grammar, which reaches remarkable accuracy on various semiotic categories. We also introduce Partial Parameter Generator and Pointer-Generator Network modules to reduce error rate of our model. The empirical results demonstrate that both modules make a contribution to the model, and combining both modules results in the best performance. The potential future work includes extending our system on multilingual scenarios. Moreover, effective and dynamic parameter generation strategies will be explored.

6. REFERENCES

[1] Richard Sproat, Alan W Black, Stanley Chen, Shankar Kumar, Mari Ostendorf, and Christopher Richards, "Normalization of non-standard words," *Computer speech & language*, vol. 15, no. 3, pp. 287–333, 2001.

[2] Paul Taylor, *Text-to-Speech Synthesis*, Cambridge University Press, 2009.

[3] D. V. Esch and R. Sproat, "An expanded taxonomy of semiotic classes for text normalization," in *INTERSPEECH*, 2017.

[4] Richard Sproat, "Multilingual text analysis for text-to-speech synthesis," in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96*. IEEE, 1996, vol. 3, pp. 1365–1368.

[5] Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai, "The opengrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*, 2012, pp. 61–66.

[6] Peter Ebden and Richard Sproat, "The kestrel tts text normalization system," *Natural Language Engineering*, vol. 21, no. 3, pp. 333, 2015.

[7] Richard Sproat and Navdeep Jaitly, "An rnn model of text normalization.," in *INTERSPEECH*. Stockholm, 2017, pp. 754–758.

[8] Courtney Mansfield, Ming Sun, Yuzong Liu, Ankur Gandhe, and Björn Hoffmeister, "Neural text normalization with subword units," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, 2019, pp. 190–196.

[9] Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth, "Text normalization with convolutional neural networks," *International Journal of Speech Technology*, vol. 21, no. 3, pp. 589–600, 2018.

[10] Subhojeet Pramanik and Aman Hussain, "Text normalization using memory augmented neural networks," *Speech Communication*, vol. 109, pp. 15–23, 2019.

[11] Alistair Conkie and Andrew Finch, "Scalable multilingual frontend for tts," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6684–6688.

[12] Hao Zhang, Richard Sproat, Axel H Ng, Felix Stahlberg, Xiaochang Peng, Kyle Gorman, and Brian Roark, "Neural models of text normalization for speech applications," *Computational Linguistics*, vol. 45, no. 2, pp. 293–337, 2019.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[15] Tianqi Chen and Carlos Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[16] Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell, "Contextual parameter generation for universal neural machine translation," *arXiv preprint arXiv:1808.08493*, 2018.

[17] Yining Wang, Jiajun Zhang, Feifei Zhai, Jingfang Xu, and Chengqing Zong, "Three strategies to improve one-to-many multilingual translation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2955–2960.

[18] Abigail See, Peter J Liu, and Christopher D Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1073–1083.

[19] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush, "Opennmt: Open-source toolkit for neural machine translation," in *Proceedings of ACL 2017, System Demonstrations*, 2017, pp. 67–72.

[20] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[21] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[22] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.