IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 5, 2020, accepted February 14, 2020, date of publication February 18, 2020, date of current version March 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2974674

A Text Normalization Method for Speech Synthesis Based on Local Attention Mechanism

LAN HUANG[1,2], SHUNAN ZHUANG[1], AND KANGPING WANG[1,2]
[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbol Computation and Knowledge Engineering, Jilin University, Changchun 130012, China

Corresponding author: Kangping Wang (wangkp@jlu.edu.cn)

This work was supported in part by the Jilin Province Science and Technology Development Plan Project under Grant 20180623010TC and Grant 20190201273JC, and in part by the Jilin Provincial Key Laboratory of Big Data Intelligent Computing.

ABSTRACT This paper proposes a deep learning model based on a recurrent neural network (RNN) to solve the problem of text normalization for speech synthesis. Traditional rule-based models cannot take advantage of contextual information and do not handle text outside of rules well, while deep learning-based models can handle these problems better. Based on the seq2seq neural network, we construct a new text-normalized deep learning model that considers the context of words in sentences by using the gated recurrent unit (GRU) and local attention mechanism. In recent years, seq2seq has made many achievements in different fields of natural language processing. Our research proves that well-constructed small network models in specific application fields can also achieve meaningful results. In our small network, the local attention mechanism is used to reduce the computational complexity without decreasing accuracy. In the experiments, we compared our model with the attention-based model proposed by Sproat, the without attention model and the attention model. Experimental results show that our method reduces the network scale while gathering the main context information, overcomes the defect of high complexity of the traditional attention mechanism model, and achieves higher accuracy. It can also be seen from experiments that the most important related words are relatively close to the target word in text normalization.

INDEX TERMS Natural language processing, deep learning, text normalization.

I. INTRODUCTION

Text-to-speech(TTS) is an active research area in modern natural language processing. In TTS, the first step, called text normalization, is standardizing text before generating language modeling data from raw text. The TTS program cannot recognize nonstandard words in text, and a major reason for the low quality of TTS systems is the nonstandardization of text. For this reason, text normalization has become an important part of speech synthesis research in recent years.

With the large increase in content in online social platforms, many studies on text normalization have focused on social media in the past few years. For example, Xia *et al.* [1] proposed a speech-based Chinese chat text normalization method, and Yang and Eisenstein [2] used a unified unsupervised statistical model to standardize the text. In addition, in sentiment analysis, microtext normalization is a necessary step for preprocessing text before performing polarity detection. For example, Satapathy *et al.* [3] used the Soundex algorithm to convert out-of-vocabulary to in-vocabulary and analyzed its impact on the sentiment analysis task. After that, Satapathy *et al.* [4] proposed PhonSenticNet, a concept-based lexicon that exploits phonetic features to normalize the out-of-vocabulary concepts to in-vocabulary concepts. There are many irregular spellings in social media, such as b4 (before), cul8er (see you later), and goooooood (good), which are rarely seen in most TTS applications. Additionally, transferring numbers (year, amount, date, time, etc.) to words is critical to TTS, but text normalization in social media does not need to be processed.

In speech processing, text normalization can be traced back to comprehensive TTS synthesis work [5]. Sproat formally treated text normalization as a primary task of the language modeling problem [6]. They used the weighted finite-state

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

converter (WFST) to classify the input text and then sorted the corresponding elements to generate normalized text [7].

With the development of deep learning, algorithms for natural language processing have achieved better results. In machine translation, which belongs to the field of natural language processing and is most similar to text normalization, end-to-end neural machine translation [8] has developed rapidly since 2013 and has been increasingly superior to traditional statistical machine translation, with an encoder-decoder framework [9] and attention mechanism [10]. The commercial translation system represented by Google Translate integrates the features of neural machine translation and has made many improvements in recent years. For text normalization, researchers from North Carolina State University proposed a deep contextual long-short term memory model [11], after which Sproat and Jaitly [12] proposed using different recurrent neural network (RNN) structures and finite-state filters (FST) to address different categories of nonstandard word normalization.

In the research field of RNN approaches to text normalization: Sproat presented a challenge that used only RNN to construct the text normalization function [12]. In different contexts, a nonstandard word could be in different forms. In this paper, local attention text normalization (LATN) is proposed, which exploits the encoder-decoder framework accompanied by the local attention mechanism. In text normalization processing, neighboring words are more important to the results, and the local attention mechanism focuses on neighboring words, which is very suitable for text normalization tasks.

II. RELATED WORK

For text normalization, Chengduo *et al.* [13] summarized that the text normalization problem can be expressed by a noisy channel model, that is, for a given word sequence s, looking for a target sequence t that maximizes the conditional probability $P(t|s)$. The conditional probability $P(t|s)$ is calculated as:

$$\arg\max_t P(t|s) = \arg\max_t \frac{P(s,t)}{P(s)} = \arg\max_t \frac{P(s|t)P(t)}{P(s)} \quad (1)$$

In formula (1), $P(s)$ is the constant value for a word. Both the prior probability $P(t)$ and the conditional probability $P(s|t)$ can be calculated from the statistical data, so the target is obtained as a sequence t that maximizes $P(t|s)$.

For the noise channel model, there are three typical applications in text normalization, spell checking, sequential labeling and machine translation. Spell checking needs to calculate the similarity of a large number of text data, it is difficult to determine a set of candidate canonical words with sequence labeling, so they are unfit for text normalization in real-time speech synthesis. In these applications, both words and phrases can be used as inputs. For example, Aw [14] proposed a phrase-based machine translation method. Fine-grained characters can also be used as inputs. For example, Pennell and Liu [15] proposed a character-based machine translation method.

Based on these previous studies, particularly on deep learning, the state-of-art sequence to sequence (seq2seq) model was proposed for machine translation. Bahdanau *et al.* [10] proposed an attention mechanism in the alignment model of machine translation, achieving a 36.15 BLEU score on the ACL WMT'14 English-French corpus. Ikeda *et al.* [16] exploited the seq2seq attention model for the standardization of Japanese text. Xu *et al.* [17] proposed soft attention that assigns weights to each feature and block and hard attention that assigns weights only for several features. Researchers from the Manning Research Group at Stanford University proposed a global attention mechanism that focuses on all original words and a local attention mechanism that only focuses on parts of the original words in machine translation [18], which achieved a 25.9 BLEU score in the WMT '15 English to German translation task.

In the text-normalization task for TTS, Sproat and Jaitly [12] exploited the attention-based model and achieved a higher accuracy rate than traditional methods, but the model requires expensive computing costs and is not good at some sentences. Therefore, the LATN model combining the gated recurrent unit (GRU) and local attention mechanism is proposed in this paper, which decreases the computing cost and increases the accuracy rate.

III. TEXT NORMALIZATION MODEL LATN

The text normalization task by a neural network can be regarded as a classification problem with a supervisor. The word quantity of normalization results is unfixed. For example, the normalization text of "20" is "twenty" – one word, and the normalization text of "21" is "twenty-one" – two words. Therefore, the seq2seq model is a good choice for dealing with variable-length text sequences.

The text normalization model LATN with the local attention mechanism proposed in this paper exploits the encoder-decoder framework. The model's architecture is shown in Fig. 1. The encoder contains a word embedding layer and a 2-layer bidirectional gated recurrent unit network layer (BGRU). Compared with the attention-based model proposed by Sproat and Jaitly [12], where the encoder used 4-layer bidirectional LSTM, the complexity of the LATN model is simplified. The decoder contains a word embedding layer and a 2-layer unidirectional gating recurrent unit network layer (GRU), a local attention module named the local block, and a linear transformation network named the linear block.

The encoder uses word embedding to extract the basic features of the source text sequence and then obtains the contextual correlation features of the text sequence through the BGRU. Similarly, on the decoder side, word embedding is used to extract the basic features of the normalized text sequence. The output is passed through the unidirectional GRU, then the GRU output and the initial text correlation information obtained from the encoder are sent to the local attention mechanism module (local block) to calculate the

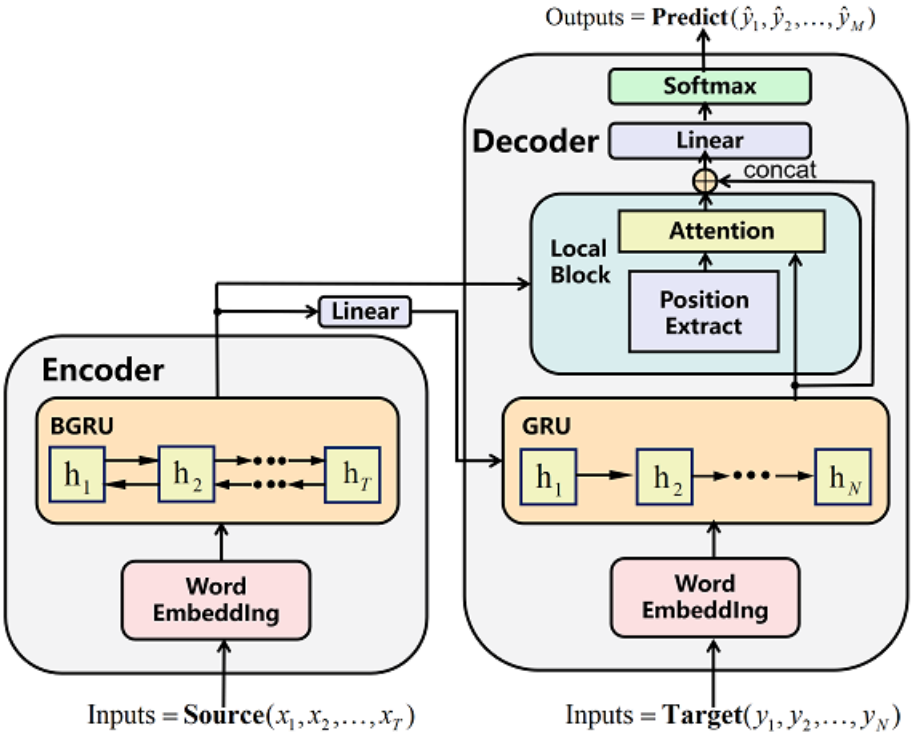


FIGURE 1. Text normalization model LATN.

local attention feature, after which the linear block integrates local block outputs and GRU outputs. Finally, the integrated results are sent to the classifier to output normalized words.

A. INPUT AND OUTPUT

The training dataset is sentences, but targets are words, so it is necessary to introduce a tag to mark target words. For instance, the target word "1994" in sentence "He was born in 1994." is embraced by a pair of *<norm>* tags. In this way, the actual input sentence is:

"*He was born in* <norm>*1994*<norm>."

Different from the attention-based model, which places each unnormalized word in a window of 3 words to the left and 3 to the right as input proposed by Sproat Jaitly [12], the LATN model uses the whole sentence as input. For example, when "*6ft*" is normalized, the input of the LATN model is "*A baby giraffe is* <norm>*6ft*<norm> *tall and weighs 150 lbs*", while the input of the attention-based model is "*baby giraffe is* <norm>*6ft*<norm> *tall and weighs*". With the whole sentence as input, the LATN model makes use of the advantages of the local attention mechanism without losing the information of other words in the sentence.

The actual inputs are index sequences. In our model, there are two types of input sequences: character index sequences and word index sequences. Character indexes are built by scanning the training corpus to form the dictionary char_dictionary of the "*character: index*" mapping pair. Word indexes are built by scanning the training corpus to form the dictionary word_dictionary of the "*word: index*" mapping pair. Since the word quantity is too large to train effectively, a preprocessing step in which words do not change after normalization and are replaced by a specific symbol *<self>* is introduced.

The seq2seq model has two inputs for the encoder and decoder. For the encoder, both character sequences and word sequence inputs are tested in experiments. Inputs for the decoder are word index sequences. Inputs for the decoder are notated as target($y_1, y_2 \ldots, y_N$), each y in the vector is the index of the current word in the word_dictionary, and N is the normalized number of words. Finally, the model outputs the target vector predict ($\hat{y}_1, \hat{y}_2 \ldots, \hat{y}_M$), each $\hat{y}$ in the vector is the index of a word in the word_dictionary, and M is the number of words output by the normalized model. In training processing, each value y in target($y_1, y_2 \ldots, y_N$) is derived from the training sample. In forecasting processing, each value y comes from the predicted output of the previous round of the decoder, following the equation $y_i = \text{decoder}(y_{i-1}) = \hat{y}_{i-1}$.

B. WORD EMBEDDING

Word embedding transforms the word index into a vector for subsequent processing. For a small dataset, the one-hot model is intuitive by converting words into V-dimensional vectors (V is the word quantity in the vocabulary of the corpus), but it is too large for even a middle size corpus. In this paper, we construct word vectors through the embedding layer in PyTorch.

As shown in Fig. 1, both the encoder and the decoder contain a word embedding layer, but the inputs are different. The encoder inputs, source($x_1, x_2 \ldots, x_T$), are indexes of char_dictionary or word_dictionary in different experiments, and the decoder inputs, target($y_1, y_2 \ldots, y_N$), are indexes of word_dictionary. The word embedding layer in the encoder converts the input vector source($x_1, x_2 \ldots, x_T$) into a matrix of size $T \times d$ (d is the dimension of embedding result), notated as embed($x_1, x_2 \ldots, x_T$), $x_t \in R^d$. The embedding layer in the decoder is similar.

C. GATED RECURRENT UNIT

In this paper, GRU is chosen for processing text sequences because it performs better in the long-distance dependence and gradient disappearance problem than traditional RNN. Compared with LSTM, another variant of RNN, the internal unit of GRU has one fewer control door, and GRU has a simplified structure and faster convergence speed. To obtain the semantic information both before and after the input character, the BGRU is used in the encoder. The BGRU processes input sequences embed($x_1, x_2 \ldots, x_T$) from both forward and backward directions. Taking forward processing as an example, in some timesteps t, x_t is the input and $\overrightarrow{h}_{t-1}$ is the hidden layer output generated at the previous timestep, the formula of GRU is as follows:

Update gate:

$$\overrightarrow{z}_t = \sigma(\overrightarrow{W}_z x_t + \overrightarrow{U}_z \overrightarrow{h}_{t-1})$$

Reset gate:

$$\overrightarrow{r}_t = \sigma(\overrightarrow{W}_r x_t + \overrightarrow{U}_r \overrightarrow{h}_{t-1})$$

The state of memory:

$$\underline{\overrightarrow{h}_t} = \tanh(\overrightarrow{W} x_t + \overrightarrow{U}(\overrightarrow{r} \circ \overrightarrow{h}_{t-1}))$$

The state of the hidden layer:

$$\overrightarrow{h}_t = (1 - \overrightarrow{z}_t) \circ \overrightarrow{h}_{t-1} + \overrightarrow{z}_t \circ \underline{\overrightarrow{h}_t}$$

The overhead arrow represents the direction of the GRU calculation. Underlined variables are not final results, which means that they must participate in the next step, where $\overrightarrow{W}_z, \overrightarrow{W}_r, \overrightarrow{W} \in R^{n \times d}$, $\overrightarrow{U}_z, \overrightarrow{U}_r, \overrightarrow{U} \in R^{n \times n}$ are weight parameter matrices. d and n are the word embedding dimensionality and the number of hidden units, respectively. $\sigma(\cdot)$ is the sigmoid function. $\circ$ is the elementwise multiplication operator, representing the Hadamard product. Finally, the $\overrightarrow{h}_T$ and $\overleftarrow{h}_T$ computed from the bidirectional calculation are spliced into a 2n-dimensional vector h_T, and the forward output $\overrightarrow{y}_t$ and the backward output $\overleftarrow{y}_t$ at all times are spliced into a $2n \times T$-dimensional matrix called encoder_output, which is the source of computational attention in the decoder. The encoder outputs the hidden layer state h_T through a linear neural network (linear) with the input node quantity $2n$ and the output node quantity n.

D. IMPROVED LOCAL ATTENTION MECHANISM

The traditional attention mechanism focuses on all hidden layer states in the encoder; that is, at the t-th step of the decoder, the hidden layer state s_i is given an attention weight $w_{ti} (i \in 1, 2, \ldots, T)$, then accumulates the multiplication of the hidden layer state s_i and weight w_{ti}, notated by c_t, which is calculated as follows:

$$w_{ti} = \frac{\exp(score(h_t, s_i))}{\sum_{j=1}^{T} \exp(score(h_t, s_j))} \tag{2}$$

$$c_t = \sum_{j=1}^{T} w_{ij} s_j \tag{3}$$

where score is a function that calculates the correlation between h_t and s_j. In the formula, a weight is calculated for each s_i that results in a long training term. At the same time, the attention weights w_{ti} are associated with all s_i without emphasis that decreases the accuracy.

To figure out this shortage, an improvement is proposed for the local attention mechanism proposed by Luong *et al.* [18]. As shown in Fig. 2, only contiguous units of the encoder output are selected for attention calculation. For the *encoder_output* whose length is T, the context vector c_t is from the window [position $-$ D, position $+$ D], where D is an experimental hyperparameter. The calculation of the position is listed below:

$$position_t = T \cdot sigmoid(v_p^T \tanh(W_p h_t)) \tag{4}$$

where $W_p \in R^{n \times n}$ and $v_p \in R^n$. Since the output of the sigmoid function is (0,1), $position_t \in (0, T)$. Only the [position-D, position $+$ D] subarray of *encoder_output* is considered, and the formula for w_{ti} is changed from equation (2) to (5):

$$w_{ti} = \frac{\exp(score(h_t, encoder\_output_i))}{\sum_{j=position-D}^{position+D} \exp(score(h_t, encoder\_output_j))} \times \exp(-\frac{(i - position)^2}{2\sigma^2}) \tag{5}$$

where $\sigma = D/2$ and the score function is:

$$score(h_t, encoder\_output_i) = h_t^T W_{score} encoder\_output_i \tag{6}$$

where $W_{score} \in R^{n \times n}$, and the formula to calculate the context vector c_t is changed from (3) to (7):

$$c_t = \sum_{j=position-D}^{position+D} w_{tj} \cdot encoder\_output_j \tag{7}$$

where $c_t \in R^{2n}$. Finally, the context vector c_t was combined with feature information that was output by GRU to generate a new vector $attention\_out_t \in R^{3n}$.

Different from the typical local attention model proposed by Luong *et al.* [18], the LSTM is replaced by GRU to reduce the parameter size. In addition, since the inputs considered by Ikeda *et al.* [16] in machine translation are words in the encoder, and the semantic contribution of a word is generally

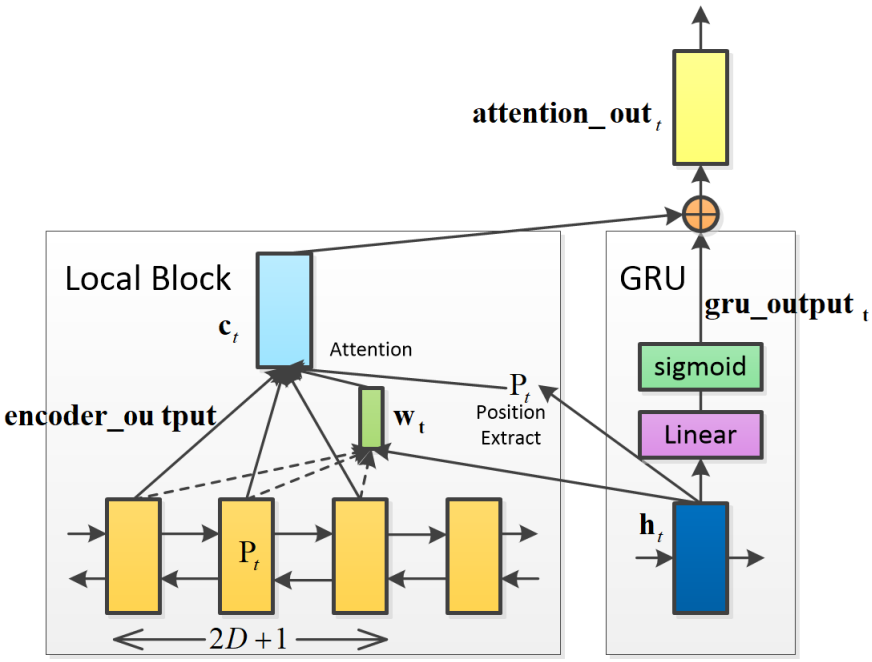


FIGURE 2. Improved local attention mechanism.

greater than the semantic contribution of a single character, the w_t value near $position_t$ is relatively large when assigning weights. Therefore, for character input, we abandon the centralized Gaussian distribution calculation, and function 5 is simplified as follows:

$$w_{ti} = \frac{\exp(score(h_t, encoder\_output_i))}{\sum_{j=position-D}^{position+D} \exp(score(h_t, encoder\_output_j))} \quad (8)$$

E. LINEAR LAYER AND SOFTMAX LAYER

After the attention calculation, $attention\_out_t$ is connected with a linear neural network whose input node quantity is $3n$ and the output node quantity is V; then, after a softmax function, a decoder output is notated as $\hat{y}_t$, where $\hat{y}_t \in R^V$ and V is the number of words contained in the word_dictionary.

IV. EXPERIMENT AND ANALYSIS

A. ENVIRONMENT

In the experiments, the software environments are Ubuntu-18.04, Anaconda 4.5.11, PyTorch 0.4, the hardware environments are a dual Intel CPU (4 cores per CPU), 64 GB memory, and GeForce GTX 1080, whose memory is 8 GB.

B. DATASET

The experimental dataset was downloaded from En Baseline, which is an open source dataset released by the Google Speech Synthesis scientist Sproat team in the Kaggle Community in 2016. The project aims to find a reasonable solution to exploit RNN for solving text normalization problems. In the En Baseline dataset, there are approximately 750,000 sentences and nearly 9.9 million words. In our experiments, the training set includes the first 9,000 sentences, including words that need to be normalized and downsampling sentences without normalization words. The test set includes the next 3,000 sentences through the same processing.

There are only 6.1% of words that need to be normalized in the dataset. We exploit downsampling methods to select the training data following Sproat and Jaitly [12] paper. In the experiments, the downsampling number is 15 for sentences in which no words need to be normalized.

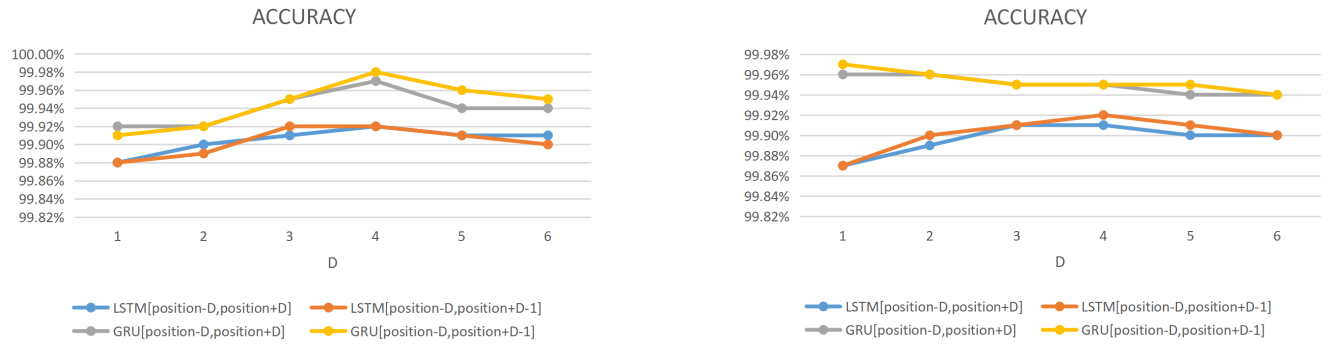
C. EXPERIMENTAL SETTING

To find the optimal window size of local attention, a few different sizes are included in the experiments. At the same time, for the purpose of comparing the advantages of using GRU over LSTM, we also performed the same number of experiments using LSTM. When we obtained the best size–D, comparative experiments among the without attention model, typical attention model and LATN model were conducted.

In the experiments, the number of hidden layer nodes is 256, the cross entropy loss is used as the loss function, and the Adam optimization method is used to update the network model parameters. The initial learning rate is 0.00003, and the batch is 4. In the decoder, the dropout rate is 0.1, and the iterations of the epoch are 60.

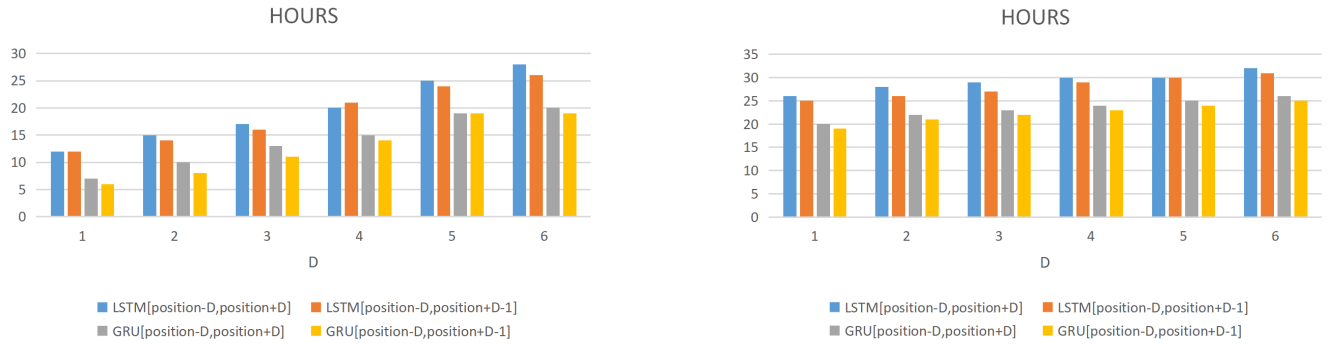
D. RESULTS AND ANALYSIS

The accuracy results and time costs of the LATN model with different window sizes are shown in Fig. 3 and Fig. 4.



(a) character-level (b) word-level

FIGURE 3. Accuracy results of the LATN model with different window sizes.



(a) character-level (b) word-level

FIGURE 4. Time costs of the LATN model with different window sizes.

TABLE 1. Comparison of results between the model without attention, attention model and LATN model.

Model	Input Level	Feature Extractor	Train Set Accuracy	Test Set Accuracy	Training Time
Without Attention	character	GRU	99.22%	97.52%	5 hours
Attention	character	GRU	99.95%	97.86%	35 hours
LATN	**character**	**GRU**	**99.98%**	**97.93%**	**14 hours**
Without Attention	character	LSTM	99.01%	97.39%	8 hours
Attention	character	LSTM	99.87%	97.57%	40 hours
LATN	**character**	**LSTM**	**99.92%**	**97.65%**	**16 hours**
Without Attention	Word	GRU	99.93%	97.48%	15 hours
Attention	Word	GRU	99.95%	97.70%	24 hours
LATN	**word**	**GRU**	**99.97%**	**97.82%**	**19 hours**
Without Attention	Word	LSTM	99.78%	97.45%	21 hours
Attention	Word	LSTM	99.89%	97.54%	33 hours
LATN	**Word**	**LSTM**	**99.92%**	**97.62%**	**29 hours**

The results show that using GRU as a feature extractor has higher accuracy and less training time than LSTM. Furthermore, it can be seen that in the GRU experiments, the best size is 8, and the interval is [position−4,position+3] when inputs are characters, the best size is 2, and the interval is [position−1,position] when inputs are words. Then, the comparison results between the LATN model, attention model and without attention model are shown in Table 1.

From these results, the overall performance of GRU is better than that of LSTM. In addition, the results show that the accuracy of the attention model is better than that of the without attention model, but the training time of the attention model is nearly six times longer than that of the without attention model when inputs are both characters. For most window sizes, the outcomes of the attention model and LATN model are almost the same, but the training time of LATN is much less. For instance, when input data have N characters, the computational complexity for the typical attention model is $O(T \times N)$, and the computational complexity for the LATN model is $O((2D+1) \times N)$. In our experiments, the LATN model can compete with the attention model when $2D+1 \ll T$. Additionally, the character inputs model outperforms the word inputs model.

Typical attention models focus on all the characters or words in input sentences. For the text normalization task, it considers more noise characters/words to reduce accuracy and waste computing power. Some attention results after training are listed in Fig. 5. These results show that the LATN model focuses on adjacent characters/words, and these behaviors are more suitable for text normalization tasks.

In addition to the total accuracy, the comparison results for all 15 categories of words between the LATN model and the attention-based model proposed by Sproat [12] are listed in Table 2.

(a) Attention Model

(b) LATN Model

FIGURE 5. Attention matrix visualization.

TABLE 2. Comparison of the accuracy of various categories of LATN model and attention-based model under the training set.

Category	Attention-Based	LATN
ALL	99.6%	99.98%
PLAIN	99.7%	100.00%
PUNCT	100.0%	100.0%
DATE	97.4%	99.90%
LETTERS	97.4%	99.90%
CARDINAL	99.1%	99.88%
VERBATIM	97.7%	100.00%
MEASURE	95.8%	98.91%
ORDINAL	97.1%	100.00%
DECIMAL	100.0%	99.03%
ELECTRONIC	95.2%	91.95%
DIGIT	70.3%	100.00%
MONEY	97.2%	100.00%
FRACTION	84.6%	96.03%
TIME	62.5%	100.00%
ADDRESS	100.0%	100.00%

As seen in Table 2, the accuracy on most categories of the LATN model is better than that of the attention-based model. For some words that are difficult to normalize, such as "*10.5*" in "*The total area is* <norm>*10.56 km2*<norm>." and "*190 mph*" in "*The speed in this sector could reach* <norm>*190 mph*<norm>.", the LATN model can obtain correctly normalized results "*ten point five six square kilometers*" and "*one hundred ninety miles per hour*", but the attention-based model does not. In ORDINAL category words, the ordinal number such as "*55th*" is normalized to "*five fifth*" in the attention-based model, which only recognizes the suffix and ignores the overall meaning of the word. However, the LATN model can obtain the correct result. For the DATE category words mentioned by Sproat, which is prone to being wrong in the attention model, the LATN model can also address this. For the date "*2009-10-02*", the attention-based model tends to output the result "*the ninth of October twenty thousand two*", while the LATN model can obtain the correct result "*the second of October two thousand nine*".

TABLE 3. Some correct text normalization examples of LATN.

Source	Results
The total area is <norm>10.56 km2<norm> .	ten point five six square kilometers
The village has a population of <norm>200<norm> .	two hundred
Mary Grive Art F <norm>200<norm> X .	two o o
Next Saturday is the <norm>55th<norm> anniversary of the school .	fifty fifth
<norm>2009-10-02<norm>	the ninth of october twenty thousand two

For ELECTRONIC category words, the accuracy of the LATN model is slightly worse than that of the attention-based model. Statistical results show that more than 95% of the ELECTRONIC category words are URLs, such as "*www.mountainproject.com*", and among them, nearly 70% contain more than 10 characters, and 10% contain more than 50 characters. Fifty characters are too many for the LATN model because of its short window. For example, the correct normalization of "*http://medical-dictionary.thefreedictionary.com/mesaticephalic*" is "*h t t p colon slash m e d i c a l dash d i c t i o n a r y dot t h e f r e e d i c t i o n a r y dot com slash m e s a t i c e p h a l i c*", while the LATN model outputs "*h t t p colon slash slash m e d i c a l dash d i c t i o n a r y dot com slash m e s a t i c e p h a l i c*"; that is, the LATN model loses the part of "*thefreedictionary*".

In addition, the LATN model can obtain the context for the same words in different sentences. For example,

word "*200*" in "*The village has a population of* <norm>*200*<norm>." should be normalized to the corresponding number "*two hundred*". However, in "*Mary Grive Art F* <norm>*200*<norm> *X.*" should be "*two o o*". Word "*II*" in "*He served in the Army Air Forces during World War* <norm>*II*<norm>." should be "*two*", in "*Francis* <norm>*II*<norm>*, Rakoczi later used the castle as the center of a war of independence led by himself.*" should be "*the second*". The LATN model obtains correct results in all these situations.

V. CONCLUSION

For the TTS text normalization task, this paper proposes a model LATN using the encoder-decoder framework and GRU recurrent network and introduces a local attention mechanism to obtain the most important context, which improves the accuracy of text normalization. This model inspires us to construct a specific model according to the particularity of the task to achieve better results and reduce computing costs.

ACKNOWLEDGMENT

The authors would like to thank all members of Key Laboratory of Symbol Computation and Knowledge Engineering. They are also like to thank Kaggle for hosting the Text Normalization Challenge. They are very grateful to Google Brain for opening up the valuable dataset to the community.

REFERENCES

[1] Y. Xia, K. F. Wong, and W. Li, "A phonetic-based approach to Chinese chat text normalization," in *Proc. ACL*, Sydney, Australia, Jul. 2006, pp. 993–1000.

[2] Y. Yang and J. Eisenstein, "A log-linear model for unsupervised text normalization," in *Proc. EMNLP*, Seattle, WA, USA, Oct. 2013, pp. 61–72.

[3] R. Satapathy, C. Guerreiro, I. Chaturvedi, and E. Cambria, "Phonetic-based microtext normalization for twitter sentiment analysis," in *Proc. ICDMW*, New Orleans, LA, USA, Nov. 2017, pp. 407–413.

[4] R. Satapathy, A. Singh, and E. Cambria, "PhonSenticNet: A cognitive approach to microtext normalization for concept-level sentiment analysis," in *Proc. CSoNet*, Ho Chi Minh City, Vietnam, Nov. 2019, pp. 177–188.

[5] J. Allen, M. Hunnicutt, D. H. Klatt, R. C. Armstrong, and D. B. Pisoni, *From Text to Speech: The MITalk System*. New York, NY, USA: Cambridge Univ. Press, 1987.

[6] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards, "Normalization of non-standard words," *Comput. Speech Lang.*, vol. 15, no. 3, pp. 287–333, Jul. 2001.

[7] P. Ebden and R. Sproat, "The Kestrel TTS text normalization system," *Natural Lang. Eng.*, vol. 21, no. 3, pp. 333–353, May 2015.

[8] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Washington, DC, USA, Oct. 2013, pp. 1700–1709.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. ICONIP*, Montreal, QC, Canada, Dec. 2014, pp. 3104–3112.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473

[11] W. Min and B. Mott, "NCSU SAS WOOKHEE: A deep contextual long-short term memory model for text normalization," in *Proc. ACL Workshop Noisy User-Generated Text*, Beijing, China, Jul. 2015, pp. 111–119.

[12] R. Sproat and N. Jaitly, "RNN approaches to text normalization: A challenge," 2016, *arXiv:1611.00068*. [Online]. Available: http://arxiv.org/abs/1611.00068

[13] L. Chengduo, W. U. Xiaorui, X. Kai, Y. Fei, and W. Baolu, "An overview of social text normalization," *J. Netw. New Media*, vol. 6, no. 5, pp. 10–14, 2017.

[14] A. Aw, M. Zhang, J. Xiao, and J. Su, "A phrase-based statistical model for SMS text normalization," in *Proc. COLING/ACL Main Conf. Poster Sessions (COLING-ACL)*, Sydney, NSW, Australia, Jul. 2006, pp. 33–40.

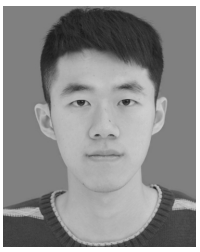
[15] D. Pennell and Y. Liu, "A character-level machine translation approach for normalization of SMS abbreviations," in *Proc. IJCNLP*, Chiang Mai, Thailand, Nov. 2011, pp. 974–982.

[16] T. Ikeda, H. Shindo, and Y. Matsumoto, "Japanese text normalization with encoder-decoder model," in *Proc. WNUT*, Osaka, Japan, Dec. 2016, pp. 129–137.

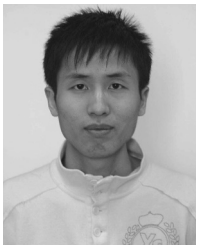
[17] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell Neural image caption generation with visual attention," in *Proc. ICML*, Lille, France, 2015, pp. 2048–2057.

[18] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*. [Online]. Available: http://arxiv.org/abs/1508.04025

LAN HUANG received the Ph.D. degree. She is currently a Professor and a Supervisor of Ph.D. candidates. She is mainly engaged in business intelligence theory and application research. She was invited to Italy Trento University as a Senior Visitor, in 2010. As PI and Co-PI, she has undertaken or accomplished more than ten teaching and scientific research projects, granted by the National 863 Hi-Tech Research and Development Program, National Science Foundation China, provincial/ministerial foundations and other sources. The software results researched and developed by her team have brought good economic benefits for cooperative enterprises and application enterprises. She has published 64 academic articles and obtained seven software copyrights. In recent years, her research interests have focused on business intelligence applications and social network mining algorithms. She was one of the outstanding youth project funding winners of Jilin Province, in 2005, and the person in charge of the Young and Middle-aged Leader and Innovation Team of Jilin Province, in 2012. The works that she performed as the Main Investigator were awarded the first prize for the National Commercial Science and Technology Award bestowed by the China General Chamber of Commerce (the first prize winner, in 2010), the second prize for Jilin Province Scientific and Technological Progress Award (first prize winner, in 2011), the second prize for the National Commercial Science and Technology Award (fourth prize winner, in 2007, and sixth prize winner, in 2004), the second prize for Jilin Province Scientific and Technological Progress Award (fourth prize winner, in 2004), and the second prize for Jilin Province Teaching Achievements (second prize winner, in 2005).



SHUNAN ZHUANG was born in 1994. He received the bachelor's degree from Jilin University, in 2017, where he is currently pursuing the master's degree in computer application and technology with the direction of natural language processing and deep learning.



KANGPING WANG received the bachelor's, master's, and Ph.D. degrees from Jilin University, in 2000, 2003, and 2008, respectively. He is currently a Faculty Member with the College of Computer Science and Technology, Jilin University. In the past several years, his research interest has included heterogeneous computing and deep learning.