Neural Inverse Text Normalization with Numerical Recognition for Low Resource Scenarios

Tuan Anh Phan[1,2], Ngoc Dung Nguyen[1], Huong Le Thanh[2], and Khac-Hoai Nam Bui[1](✉)

[1] Viettel Cyberspace Center, Viettel Group, Hanoi, Vietnam
{anhpt161,dungnn7,nambkh}@viettel.com.vn
[2] Hanoi University of Science and Technology, Hanoi, Vietnam
huonglt@soict.hust.edu.vn

Abstract. Neural inverse text normalization (ITN) has recently become an emerging approach for automatic speech recognition in terms of post-processing for readability. In particular, leveraging ITN by using neural network models has achieved remarkable results instead of relying on the accuracy of manual rules. However, ITN is a highly language-dependent task that is especially tricky in ambiguous languages. In this study, we focus on improving the performance of ITN tasks by adopting the combination of neural network models and rule-based systems. Specifically, we first use a seq2seq model to detect numerical segments (e.g., cardinals, ordinals, and date) of input sentences. Then, detected segments are converted into the written form using rule-based systems. Technically, a major difference in our method is that we only use neural network models to detect numerical segments, which is able to deal with the low resource and ambiguous scenarios of target languages. Regarding the experiment, we evaluate different languages in order to indicate the advantages of the proposed method. Accordingly, empirical evaluations provide promising results for our method compared with state-of-the-art models in this research field, especially in the case of low resource scenarios.

Keywords: Inverse text normalization · Automatic speech recognition · Neural network models · Rule based systems

1 Introduction

Inverse text normalization is a natural language processing (NLP) task of converting a token sequence in spoken form (source sentence) to the corresponding written form (target sentence), which is applied to most speech recognition systems. Figure 1 depicts the pipeline of a spoken dialogue system with ITN. ITN is the inverse problem of text normalization (TN), which transforms the written form into spoken form. However, different from the exploitation of promising methods for TN problem in recent years [6], there are not many remarkable

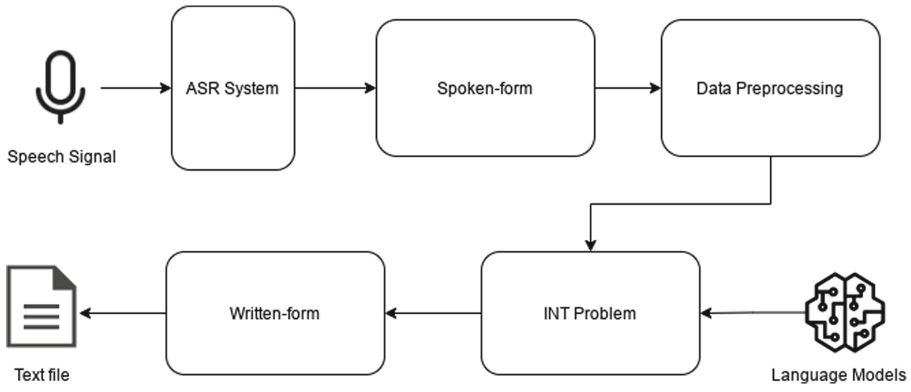

Fig. 1. ITN in spoken dialogue systems.

achievements for the ITN problem, which is regarded as one of the most challenging NLP tasks.

The conventional approach for addressing ITN is rule-based systems, for instance, finite state transducer (FST) based models [2], which have proved the competitive results [15]. However, the major problem with this approach is the scalability problem, which requires complex accurate transformation rules [14]. Recently, Neural ITN has become an emerging issue in this research field, by exploiting the power of neural networks (NN) for ITN tasks. Specifically, NN-based models, typically seq2seq, have achieved high performances and become state-of-the-art models for the ITN problem [3,7,10]. Nevertheless, as we mention above, ITN is a highly language-dependent task and requires linguistic knowledge. In this regard, the data-hungry problem (i.e., low resource scenarios) is an open issue that needs to take into account for improving performance. Furthermore, due to the significant difference between written and spoken forms, handling numbers with minimal error is a central problem in this research field. In particular, to be able to read the numeric values, the models should be worked on both consecutive tasks such as recognizing the parts that belong to numeric values and combining those parts to precise numbers. Specifically, in the shortage-data situation, models might lack information for the training stage in order to recognize and transform numerical segments, which is the cause of the bad performance of ITN. Using pre-trained embedding models (e.g., BERT, GPT3,...) can improve performance. However, for real-world systems, this issue is inefficient due to the cost of memory and time calculation.

In this study, we take an investigation to improve the performance of Neural ITN in terms of low resources and ambiguous scenarios. Particularly, for formatting number problems, conventional seq2seq models might fail to generate sequentially character by character digit, which often appears in long numbers (e.g., phone numbers, big cardinal). For example, the number 'one billion and eight' must be converted to 10 sequential character: '1 0 0 0 0 0 0 0 0 8'. Moreover, the poverty of data in the training process can cause this issue more worse in case the considered languages have lots of ambiguous semantics between numbers and words. For instance, in Vietnamese, the word 'không' (English translate: no) can be a digit '0', but also used to indicate a negative opinion. Table 1 illustrates several examples of the ambiguous semantic problem in the Vietnamese language.

Table 1. Examples of the ambiguous semantic problem in Vietnamese language. The bold parts are ambiguous words.

Spoken form (English translation)	Number	Word
tôi **không** thích cái bánh này (I do not like this cake)		✓
không là số tự nhiên nhỏ nhất -(zero is the smallest natural number)	✓	
năm một nghìn chín trăm chín bảy (nineteen ninety-seven)		✓
năm mươi nghìn (fifty thousand)	✓	
chín qủa táo (nine apples)	✓	
quả táo **chín** (a ripe apple)		✓

In this paper, the proposed framework includes two stages: i) In the first stage, we use a neural network model to detect numerically segments in each sentence; ii) Then, the output of the first phase is converted into the written form by using a set of rules. Accordingly, the main difference compared with previous works is that we use the neural network to detect numerical segments in each sentence as the first phase. Reading number is processed in the second stage by a set of rules, which is able to supply substantial information for the system without requiring much data to learn as end-to-end models. Generally, the main contributions of our method are threefold as follows:

- We propose a novel hybrid approach by combining a neural network with a rule-based system, which is able to deal with ITN problems in terms of low resources and ambiguous scenarios.
- We evaluate the proposed methods in two different languages such as English and Vietnamese with promising results. Specifically, our method can extend easily to other languages without requiring any linguistics knowledge.
- We present a pipeline to build an ITN model for Vietnamese. To the best of our knowledge, this is the first study of the ITN problem for Vietnamese speech systems.

The rest of the paper is organized as: Section 2 presents the literature review and background of our study. Our methodology is proposed in Sect. 3. We report and analyze the evaluated results in Sect. 4. Section 5 concludes our work and discusses the future work regarding this study.

2 Literature Review

The research on ITN is closely related to the TN problem in which recent works can be classified into three approaches, which are sequentially described as follows:

2.1 Rule-Based Systems

Most ASR and Text-to-Speech (TTS) systems are based on Weighted FST grammar for the TN problem. Kestrel, a component of the Google TTS synthesis system [2], has achieved around 99.9% on the Google TN test dataset. For ITN, the set of rules in [7] achieves 99% on internal data from a virtual assistant application. In order to develop products, Zhang et al. introduce an open-source python WFST-based library [15], which includes two-stage normalization pipeline that first detects semiotic tokens (classification) and then converts these to written form (verbalization). Both stages consume a single WFST grammar. The major problem is that this approach requires significant effort and time to scale the system across languages, especially linguists experiences.

2.2 Neural Network Models

Recurrent Neural Network (RNN)-based seq2seq models [11], have been adopted for reducing manual processes. Sproat et al. [9] considers the TN problem as a machine translation task and develop an RNN-based seq2seq model trained on window-based data. Specifically, an input sentence is regarded as a sequence of characters, and the output sentence is a sequence of words. Furthermore, since the length of sequence input problem, they split a sentence into chunks with window size equals three for creating sample training in which normalized tokens are marked by distinctive begin tag <norm> and end tag </norm>. In this regard, this approach is able to limit the number of input and output nodes to something reasonable. Their architecture neural network follows closely that of [1]. Sequentially, Sevinj et al. [13] proposed a novel end-to-end Convolutional Neural Network (CNN) architecture with residual connections for the TN task. Particularly, they consider the TN problem as the classification problem which includes two stages: i) First, the input sentence is segmented into chunks, which is similar to [9] and use a CNN-based model to label each chunk into corresponding class based on scores of soft-max function; ii) After the classification stage, they apply rule-based methods depending on each class.

Recently, Transformer is a new seq2seq structure, which has achieved high performance for most NLP tasks [12]. Accordingly, the work in [10] has shown the advantage of Transformer compared with RNN-based models in the ITN problem. Figure 2 depicts a general architecture of Transformer.

Particularly, the model handles the input into three kinds of vectors (i.e., Key, Value, and Query vectors), which are driven by multiplying the input embedding with three matrices that we trained during the training process. Furthermore,

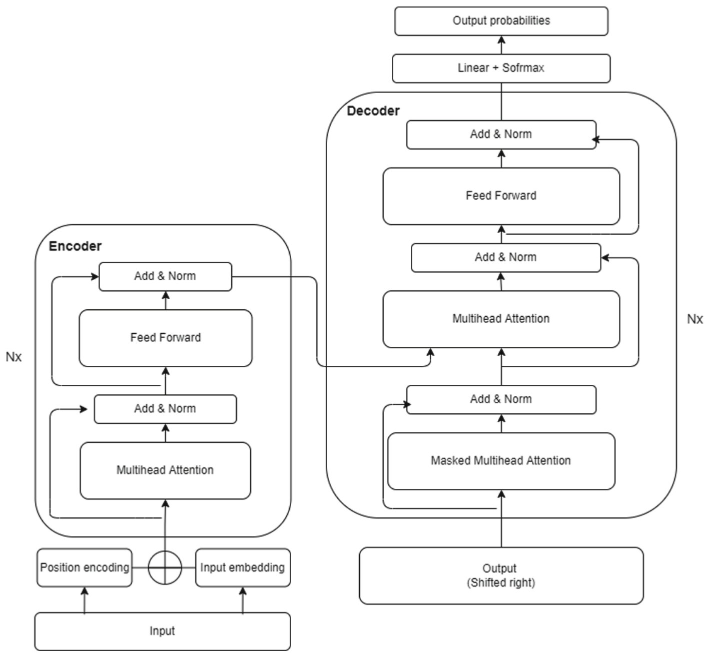

Fig. 2. The general architecture of Transformer. The encoder layer is built by 6 identical components which contain one multi-head attention layer with a fully connected network. These two sub-layers are equipped with residual connection as well as layer normalization. The decoder layer is more complicated and also includes 6 components stacked. Each component includes three connected sub-layers in which two sub-layers of multi-head self-attention and one sub-layer of fully-connected neural network.

Positional Encoding is adopted to the model for the sequence order, and Self-Attention with multi-head is applied for allowing the model to jointly attend to inform different representation sub-spaces at different positions. Sequentially, the log conditional probability can be interpreted as follows:

$$logP(y|x) = \sum_{t=1}^{t=N} logP(y_t|y_{<t}, x) \tag{1}$$

Furthermore, although there is no out-of-vocab (OOV) problem of the input by using the window-based seq2seq model, however, it is able to occur in the decode. In this regard, Courtney et al. [5] propose a new method for directly translating the input written-form to output spoken form without tagger or window-based segment. Accordingly, in order to handle OOV on both sides, the study uses the subword method [8] to decompose words into subparts, which have been proved the capability in open-vocabulary speech recognition and machine translation tasks.

2.3 Hybrid Models

Employing a weak covering grammar to filter and correct the misreading of NN models. Pusateri et al. [7] presents a data-driven approach for ITN problems by a set of simple rules and a few hand- crafted grammars to cast ITN as a labeling problem. Then, a bi-directional LSTM model is adopted for solving the classification problem. [10] propose a neural solution for ITN by combining transformer-based seq2seq models and FST-based text normalization techniques for data preparation. Specifically, similar to [5], they implement a word and subword-based seq2seq neural network models except reversing the order of input sentence and output sentence. Accordingly, integrating Neural ITN with an FST is able to overcome common recoverable errors in production environments.

3 Methodology

3.1 General Framework

In this paper, we propose a novel hybrid model Neural ITN problem using seq2seq neural network and rule base systems by considering the ITN task as the Machine Translation (MT) task. Figure 3 describe general our framework, which includes two main stages.

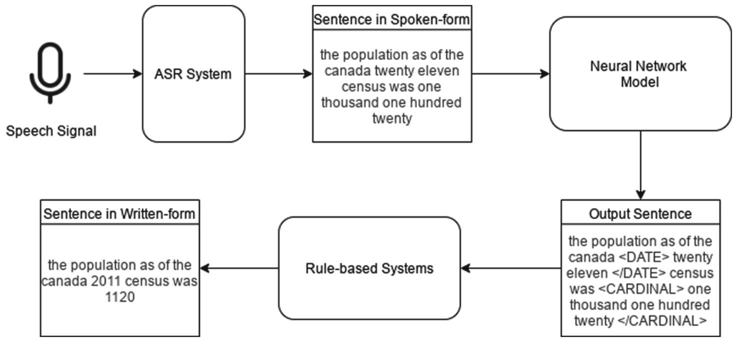
Speech Signal
ASR System
Sentence in Spoken-form
the population as of the canada twenty eleven census was one thousand one hundred twenty
Neural Network Model
Output Sentence
the population as of the canada <DATE> twenty eleven </DATE> census was <CARDINAL> one thousand one hundred twenty </CARDINAL>
Rule-based Systems
Sentence in Written-form
the population as of the canada 2011 census was 1120

Fig. 3. The general framework of the proposed method for the Neural ITN approach

Specifically, in the first stage, each sentence is put into a transformer-based seq2seq model for detecting numerical segments by using tag <n> and </n>, in which n represents a numerical classes (e.g., *DATE*, *CARDINAL*, *ORDINAL*). Then, a set of rules is employed to convert tokens, which be wrapped by tag to number, into the written form. Otherwise, all parts of a sentence, which are not in the tag are conserved. Particularly, instead of using a neural network to directly translate numbers [9], we only use NN to detect numerical segments. Essentially, NN is only utilized for distinguishing which is in the number and which is not. After that, when the model has candidates for numbers, they are transformed to the correct form by the set of rules. Consequentially, the model is able to read accurately numbers in sentences.

For example, if we have input spoken sentence: '*the population as of the canada twenty eleven census was one thousand one hundred twenty*'. After pass through this sequence to NN model, the output will be formalized as: '*the population as of the canada <DATE> twenty eleven </DATE> census was <CARDINAL> one thousand one hundred twenty </CARDINAL>* '. As result, numerical phrases '*twenty eleven*' and '*one thousand one hundred twenty*' are wrapped, and transformed to written form by rules: '27' and '1120' based on two class DATE and CARDINAL. A set of rules, which are utilized can be considered as the replacement for a great deal of knowledge in the training process.

3.2 Data Creation Process

Since there is no publicly available data for training ITN, following the work in [10], we employ a novel data generation pipeline for ITN using the TTS system. Figure 4 shows the main steps of our data creation process, which are sequentially described as follows:

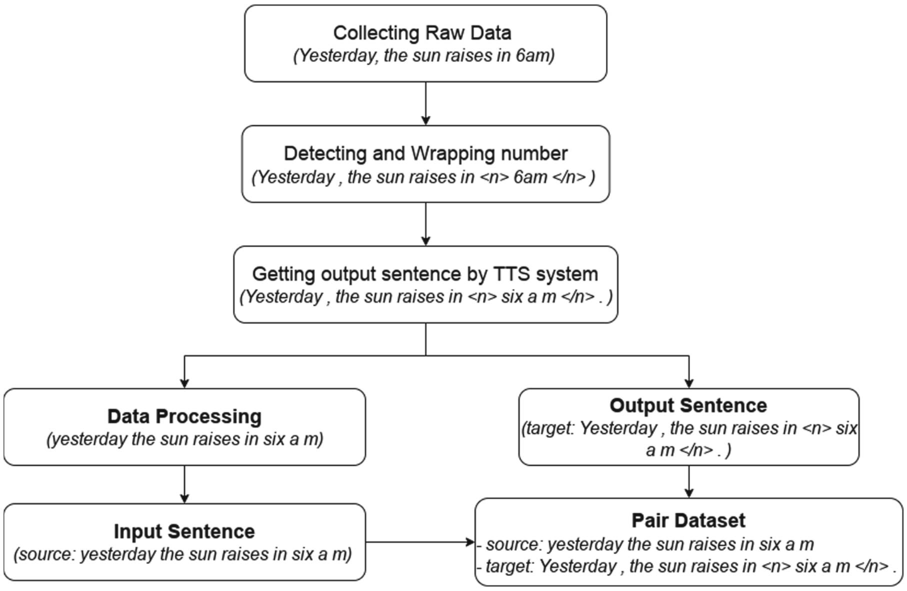
Collecting Raw Data
(Yesterday, the sun raises in 6am)
Detecting and Wrapping number
(Yesterday , the sun raises in <n> 6am </n>)
Getting output sentence by TTS system
(Yesterday , the sun raises in <n> six a m </n> .)
Data Processing
(yesterday the sun raises in six a m)
Output Sentence
(target: Yesterday , the sun raises in <n> six a m </n> .)
Input Sentence
(source: yesterday the sun raises in six a m)
Pair Dataset
- source: yesterday the sun raises in six a m
- target: Yesterday , the sun raises in <n> six a m </n> .

Fig. 4. Data creation process for training seq2seq model

- **Step 1**: Crawling/downloading raw data from published websites.
- **Step 2**: Cleaning raw data (e.g., removing HTML, CSS, and so on), and removing noise documents.

- **Step 3**: Detecting non-standard words, for instance, alphabet (e.g 'David') or number (e.g '2017'), in sentence. For handling alphabet words, we split them into characters (extremely subword) and bound them by using tag <oov> and </oov>. For handling numerical words, we split them into sequence digits and bound them by couple tag <n> and </n>.
- **Step 4**: Passing output sentences through TTS systems (e.g., Google TTS).
- **Step 5**: The output sentences from TTS systems are used as target sentences for NN models. For creating source sentences, we remove punctuation, tag <n>, lower all tokens, and only preserve tag <oov> similar to the spoken form.
- **Step 6**: Saving data to file, simultaneously.

3.3 Training Model

As we mention above, we model the ITN as an MT problem where the source and target are the output of spoken form and the detected segments of text, respectively. For NN models, we implement two training models which are RNN-based and transformer-based seq2seq models. Specifically, for the RNN model, we employ a stacked bi-direction long short term memory network (bi-LSTM) as encoder and a stacked long short term memory network (LSTM) as decoder, respectively [1]. Regarding the non-recurrent model, we implement a transformer model based on the work in [12]. Additionally, the source and target sentences are segmented into subword sequences [5].

3.4 Rule-Based Systems

The output of NN models with detected segments is transformed into the written form using a set of rules. Table 2 demonstrates an example with the input is the output sequence of the first stage and final outputs of this process.

Table 2. An example output in second stage. Tokens labeled by NONE is conserved, while tokens are bounded by numerical tag (eg. CARDINAL) will be rewritten under using rules.

Spoken-form	Label	After label	Written-form
He	NONE	he	he
Collected	NONE	collected	collected
Four Hundred Thousand	CARDINAL start CARDINAL in CARDINAL end	⟨ CARDINAL⟩ four hundred thousand ⟨ /CARDINAL⟩	400000
Records	NONE	records	records

Specifically, the input sentence is segmented and specified by labels using post-processing grammar. Sequentially, with each numerical token, we used the word2number1 python package[1] for converting spoken numbers into written numbers. In particular, since the tool works only for positive numbers and the largest value is limited to 999,999,999,999, we extended the tool in order to handle negative cardinals and larger numbers. Moreover, we also construct the python modules for reading the number, which belongs to other classes such as MEASURE, DATE, PHONE, TIME... and so on based on the aforementioned extended tool.

4 Experiment

4.1 Dataset

Regarding evaluation datasets, we test our method on two different language datasets such as English and Vietnamese, which are extracted from publicly available data sources as follows:

– **English Dataset**: the original version consists of 1.1 billion words of English text from Wikipedia, divided across 100 files. The normalized text is obtained by running through Google TTS system's Kestrel text normalization system [2]. In this study, we use the first file which contains approximately 4 million samples for our experiments. We split randomly the file into two parts in which the first part includes 1 million sentences for training data, and the second part contains 50,000 sentences for testing. For the ITN dataset, we conserved all tokens of sentences and swapped input and output.
– **Vietnamese Dataset**: the raw dataset is extracted from the largely published source[2]. After that, we decode them as UTF-8 and remove all sentences including tags (e.g., Html and CSS). We also extracted 1 million and 50,000 sentences for training and testing data, respectively. Sequentially, we execute the dataset through Viettel TTS System[3]. For formatting into the ITN dataset, we construct the data following the pipeline in Sect. 3.2.

Since our study focuses on low resource scenarios, we divide the training data into various numbers of samples such as 100 k, 200 k, 500 k, and 1000 k, respectively, in order to evaluate the advantage of the proposed method. Specifically, Table 3 illustrates the size of training, validation, and vocab size of the training datasets.

4.2 Baseline Models and Hyperparameter Configuration

Regarding baseline models, we implement two well-known NN models such as RNN and Transformer. Specifically, the Neural ITN is regarded as the MT problem. Furthermore, all models are implemented with the subword approach, which has proven the better performance [5]. Particularly, the baseline models are sequentially configured as follows:

[1] https://pypi.org/project/word2number/.
[2] https://github.com/binhvq/news-corpus.
[3] https://viettelgroup.ai/.

Table 3. Size of training datasets with 20% for validation.

Language	Dataset	Training	Validation	Vocabulary size
English	100 k	80 k	20 k	35903
	200 k	160 k	40 k	47328
	500 k	400 k	100 k	65079
	1000 k	800 k	200 k	80585
Vietnamese	100 k	80 k	20 k	7223
	200 k	160 k	40 k	8225
	500 k	400 k	100 k	10400
	1000 k	800 k	200 k	11657

- **RNN Model**: For the recurrent seq2seq baseline model, we use Encoder-Decoder architecture with RNN-Encoder consisting of two bi-directional long short-term memory (Bi-LSTM) layers and two LSTM layers for the decoder. Both encoder and decoder contain 512 hidden states. Global attention mechanisms [4] is implemented in Decoder.
- **Transformer Model**: For the non-recurrent seq2seq baseline model, we implement the architecture, which is similar to [12]. Specifically, we employ the subword-transformer model with 6 layers for both Encoder and Decoder. Each sub-layer block contains a 512-dimension vector hidden state. Furthermore, the number of multi-head self-attention is set to 8.

Consequentially, we execute our proposed method with two versions by adopting two aforementioned baseline models for the first phase of segment detection, respectively. For the hyperparameter configuration, we use Adam optimizer with learning rate annealing and the initial value is 0.01, and dropout is set to 0.1. All models are trained with 100k timesteps, early stop is used on validation loss.

4.3 Evaluation Metrics

Bi Lingual Evaluation Understudy (BLEU) is the popular way to measure the performance of a machine translation system, which is calculated as:

$$BLEU = BP\,exp\left(\sum_{n=1}^{n-grams} w_n\,logp_n\right) \tag{2}$$

where P_n is the modified *n-grams* precision and w_n denotes the uniform weight. BP refers to the brevity, which can be calculated as follows:

$$BP = \begin{cases} 1, & \text{if c¡r.} \\ e^{1-\frac{r}{c}}, & \text{otherwise.} \end{cases} \tag{3}$$

where c and r refers the candidate and reference sentences, respectively.

Word Error Rate (WER) is a common metric of the performance of speech recognition or machine translation system, which is calculated as:

$$WER = \frac{S + D + I}{S + D + C} \tag{4}$$

where S, D, I, C are the number of substitutions, deletions, insertions, and correct words, respectively.

4.4 Result Analysis

Table 4 shows the comparison results of our experiments on the test set, which bold parts are the best results.

Table 4. Comparison of models on test set with BLEU and WER scores. Bold texts indicate the best results.

Lang	Training data	Metric	RNN	Trans	Our(RNN)	Our(Trans)
English	100k	BLEU	0.7405	0.7048	**0.8334**	0.741
		WER	0.1467	0.1561	**0.1017**	0.152
	200k	BLEU	0.7909	0.7919	**0.8353**	0.8188
		WER	0.1129	0.1033	0.112	**0.1003**
	500k	BLEU	0.7706	**0.8558**	0.8377	0.8394
		WER	0.128	**0.0708**	0.1009	0.0874
	1000k	BLEU	0.7959	**0.9138**	0.848	0.8933
		WER	0.1087	**0.0405**	0.0953	0.0517
Vietnamese	100 k	BLEU	0.6422	0.6755	**0.7019**	0.6774
		WER	0.2495	0.2447	**0.1897**	0.2029
	200 k	BLEU	0.6794	0.7201	0.7144	**0.7286**
		WER	0.2111	0.2101	0.184	**0.1774**
	500 k	BLEU	0.6921	0.7447	0.718	**0.7667**
		WER	0.2016	0.1767	0.1784	**0.1327**
	1000 k	BLEU	0.6777	0.7594	0.711	**0.7885**
		WER	0.2103	0.1821	0.1817	**0.1199**

Note that for the BLEU score, the higher is the better, contrary to the case of WER score. Accordingly, there are several issues we can summarize based on the results as follows:

– Our method outperforms baseline models in the case of low resource scenarios (i.e., 100 k and 200 k) and is able to achieve competitive results in the case of higher resources (i.e., 500 k and 1000 k) with the English language. The experiment results indicate that when the number of the training sample is low, using two stages is able to cover several prediction errors in end-to-end models.

- For the Vietnamese language, our method is able to achieve the best results in all cases. The reason is that different from English, the numerical classes in Vietnamese are more ambiguous in Spoken-form (as shown in Table 1), in which the end-to-end modes are not able to distinguish the ambiguous problem between numerical classes in Vietnamese.
- Recurrent-based seq2seq models with attention achieve better performance compared with Transformer in the case of low resource scenarios. Meanwhile, Transformer-based models are able to achieve the best results by increasing the number of training samples. Therefore, combining two methods (hybrid models) is able to improve the performance. We take this issue as our future work regarding this study.

5 Conclusion

In this study, we introduce a new method for the neural ITN approach. Specifically, the difference from previous works, we divide the neural ITN problem into two stages. Particularly, in the first stage, neural models are used to detect numerical segments. Sequentially, the written form is extracted based on a set of rules in the second stage. In this regard, our method is able to deal with the low resource scenarios, where there is not much available data for training. Furthermore, we showed that our method can be easily extended to other languages without linguistic knowledge requirements. The evaluation of two different language datasets (i.e., English and Vietnamese) with different sizes of training samples (i.e., 100 k, 200 k, 500 k, and 1000 k) indicates that our method is able to achieve comparable results in the English language, and the highest results in Vietnamese languages. Regarding the future work of this study, we focus on extending the model with deeper architectures, for instance, 12 layers of Transformer, and combine models, including pre-trained models for improving the performance of Neural ITN problem.

References

1. Chan, W., Jaitly, N., Le, Q.V., Vinyals, O.: Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In: Proceeding of the 41st International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4960–4964. IEEE (2016). https://doi.org/10.1109/ICASSP.2016.7472621
2. Ebden, P., Sproat, R.: The kestrel TTS text normalization system. Nat. Lang. Eng. **21**(3), 333–353 (2015). https://doi.org/10.1017/S1351324914000175
3. Ihori, M., Takashima, A., Masumura, R.: Large-context pointer-generator networks for spoken-to-written style conversion. In: Proceeding of the 45th International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8189–8193. IEEE (2020). https://doi.org/10.1109/ICASSP40776.2020.9053930
4. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: Proceeding of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1412–1421. The Association for Computational Linguistics (2015). https://doi.org/10.18653/v1/d15-1166

5. Mansfield, C., Sun, M., Liu, Y., Gandhe, A., Hoffmeister, B.: Neural text normalization with subword units. In: Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), pp. 190–196. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/N19-2024
6. Pramanik, S., Hussain, A.: Text normalization using memory augmented neural networks. Speech Commun. **109**, 15–23 (2019). https://doi.org/10.1016/j.specom.2019.02.003
7. Pusateri, E., Ambati, B.R., Brooks, E., Plátek, O., McAllaster, D., Nagesha, V.: A mostly data-driven approach to inverse text normalization. In: Proceeding of the 18th Annual Conference of the International Speech Communication Association (Interspeech), pp. 2784–2788. ISCA (2017)
8. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 1715–1725. Association for Computational Linguistics (2016). https://doi.org/10.18653/v1/P16-1162
9. Sproat, R., Jaitly, N.: An RNN model of text normalization. In: Proceeding of the 18th Annual Conference of the International Speech Communication Association (Interspeech), pp. 754–758. ISCA (2017)
10. Sunkara, M., Shivade, C., Bodapati, S., Kirchhoff, K.: Neural inverse text normalization. In: Proceeding of the 46th International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 7573–7577. IEEE (2021). https://doi.org/10.1109/ICASSP39728.2021.9414912
11. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Proceeding of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing System (NeurIPS), pp. 3104–3112 (2014)
12. Vaswani, A., et al.: Attention is all you need. In: Proceeding of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing System (NeurIPS), pp. 5998–6008 (2017)
13. Yolchuyeva, S., Németh, G., Gyires-Tóth, B.: Text normalization with convolutional neural networks. Int. J. Speech Technol. **21**(3), 589–600 (2018). https://doi.org/10.1007/s10772-018-9521-x
14. Zhang, H., et al.: Neural models of text normalization for speech applications. Comput. Linguist. **45**(2), 293–337 (2019). https://doi.org/10.1162/coli_a_00349
15. Zhang, Y., Bakhturina, E., Gorman, K., Ginsburg, B.: Nemo inverse text normalization: from development to production. CoRR abs/2104.05055 (2021)