JAIST Repository
https://dspace.jaist.ac.jp/

Title	End-to-end Neural Inverse Text Normalization for Vietnamese Automatic Speak Recognition System
Author(s)	NGUYEN, Hieu Van
Citation	
Issue Date	2022-12
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/18163
Rights	
Description	Supervisor:NGUYEN, Le Minh, 先端科学技術研究科, 修士(情報科学)

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Master's Thesis

End-to-end Neural Inverse Text Normalization for Vietnamese Speech Recognition System

NGUYEN, Hieu Van

Supervisor NGUYEN, Minh Le

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

Nov, 2022

Abstract

Recently, the Inverse Text Normalization (ITN) problem has received significant attention when the Automatic Speech Recognition (ASR) system has applicability in the production environment. The task aims to convert spoken form text into the corresponding written form format. ITN plays an important role when it helps to improve the readability of the ASR's output and improve the efficiency of natural language processing tasks behind the ASR system.

This problem is challenging when it requires understanding the context to normalize the text correctly. Traditional methods based on grammar rules do not require training data but are less effective for cases with complex contexts. Developing a system based on grammar rules is often complicated due to language dependence and the need for language experts. This method is also difficult to apply to the two subproblems of ITN: Restoring punctuation and Capitalizing proper nouns, because they require deep context understanding. Recent studies using deep learning proved to be quite effective, and system development also becomes more accessible when it does not depend on language or grammar rules. This method also overcomes the limitation of the traditional approach when the input has a complex context. But deep learning requires many data to label the training process, which is expensive. In addition, studies often focus on the problem of converting text from spoken form to written form, ignoring two sub-problems: restoring punctuation marks and capitalizing proper nouns. The deep learning method can generate unrecoverable errors if the input sentence contains rare words. However, research based on deep learning gives results superior to traditional methods. However, it still has limitations: (1) Using a lot of labeled data, ignoring punctuation and capitalization restoration tasks, and (3) the model often generates unrecoverable errors.

To overcome the above limitations of the deep learning method, we propose to use a deep language model that is trained first and then refined for the ITN problem. Using a pre-trained language model makes it possible for the model to use available contextual information instead of learning from scratch. Pre-trained helps the ITN need very little labeled data compared to training from scratch. However, using less training data means the unrecoverable errors appear more. For this problem, we propose to apply the subword technique to separate rate words into smaller units before training the model. The subword method can handle the rare word problem in ITN thoroughly. However, using fewer data makes more unrecoverable errors. For

this problem, we propose to apply the subword technique to separate rate words into smaller units before training the model. Besides, we also train our model simultaneously to process two additional problems, including restoring punctuation marks and capitalizing proper nouns.

The experimental results of the proposed method are much higher than the baseline method. Above all, the experimental results show that the model achieves good results for two subproblems: Restoring punctuation marks and Capitalizing proper nouns. For English data, our proposed method gives competitive results with the state-of-the-art. Additionally, our solution using subwords shows significantly improved results than word units on both English and Vietnamese datasets.

Acknowledgement

First and foremost, I would like to express my sincerest thanks to my supervisor, Professor Nguyen Le Minh, who has always supported and guided me during my master's program. He gave me a lot of advice and feedback that motivated my research journey. Without his support, I would not have been able to complete this thesis.

I want to thank the committee members, including Professor Satoshi Tojo, Professor Kaneko Mineo, and Associate Professor Teeradaj Racharak, for their helpful discussions and comments on this thesis.

I would also like to thank the members of the NGUYEN laboratory. You guys have provided insightful reviews and comments that help me improve my thesis.

Last but not least, I want to express my gratitude to my family for their unwavering love and support. Without their assistance, I could not have finished this project.

Contents

1 Introduction **1**
- 1.1 Research background . . . 1
- 1.2 Problem statement . . . 3
- 1.3 Research motivation . . . 5
- 1.4 Research objective . . . 5
- 1.5 Thesis organization . . . 6

2 Literature Review **7**
- 2.1 Related works . . . 7
  - 2.1.1 Rule-based methods . . . 7
  - 2.1.2 Neural network methods . . . 9
  - 2.1.3 Hybrid methods . . . 11
- 2.2 Background knowledge . . . 12
  - 2.2.1 Encoder-Decoder model . . . 12
  - 2.2.2 Transformer . . . 15
- 2.3 Dataset . . . 17

3 Methodology **18**
- 3.1 Preliminaries . . . 18
  - 3.1.1 T5 . . . 18
  - 3.1.2 BART . . . 20
  - 3.1.3 BARTpho . . . 21
- 3.2 Baseline model . . . 21
- 3.3 Proposed model . . . 23
- 3.4 Subword processing . . . 24

4 Experiment and Result **26**
- 4.1 Data . . . 26
- 4.2 Metrics . . . 28
- 4.3 Result . . . 29
  - 4.3.1 Subword performance . . . 29

4.3.2 Performance on Vietnamese data 30
4.3.3 Performance on English data 32

5 Conclusion **34**
5.1 Summary 34
5.2 Future works 35

Publications **39**

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Vietnam National University, University of Engineering and Technology.

List of Figures

1.1 The ITN task in the speech processing system 3
1.2 The overview of ITN task 4

2.1 The NeMo ITN's pipeline [26] 9
2.2 High-level overview of the Encoder-Decoder model 13
2.3 Illustration of the encoder block 14
2.4 Illustration of the decoder block 15
2.5 The Encoder-Decoder of the Transformer architecture 16

3.1 A diagram of T5 framework [14] 19
3.2 BART is a combination of a bidirectional encoder and an autoregressive decoder. [6] 20
3.3 Overview of the baseline ITN model for Vietnamese 22
3.4 Illustrate the pipeline of the proposed method 23

4.1 Spoken-form's length distribution of vi_social (left), vi_tedtalks (center) and en_wiki (right). 27
4.2 Example of word tagging and WER calculation 29
4.3 Number of unique words in written form before and after apply subword 30

List of Tables

1.1 Example of Vietnamese spoken form and corresponding written form 2

3.1 Information about the pre-trained models used for the experiment 24
3.2 Example of encoding data with subwords on several categories 25

4.1 Information on how to use the data for the experiment 27
4.2 ITN performance on Vietnamese test sets ↓ 31
4.3 Result on capitalization and punctuation tasks ↑ 32
4.4 Result on Engish dataset ↓ 33

Chapter 1

Introduction

1.1 Research background

The most commonly used text format is written form (i.e., including numbers, punctuations, and syllables). This text format is also commonly used in most natural language processing (NLP) tasks. However, the written form of text is not used in speech-processing tasks (e.g., text-to-speech and automatic speech recognition). Because the written form has unlimited words, it also takes several syllables to read a number. These problems make it difficult for the speech-processing model to map and learn the relationship between input and output. Fortunately, we will no longer have to worry about these problems when using the spoken form.

Inverse text normalization is the task that normalizes the spoken form text to its written form. ITN is commonly used in the ASR system to improve user readability and performance on downstream NLP tasks, such as named entity recognition. A spoken form sentence can be divided into non-normalized words (plain tokens) and normalized words [18]. The plain tokens are words in written form preserved from the spoken form, while normalized words are the words/phrases that need to be converted from the spoken form to its corresponding written form (e.g., number, date, time).

ITN is closely related to the text normalization (TN) problem because its goal is to standardize the text from written to spoken form for the text-to-speech (TTS) system. Figure 1.1 shows the position of the TN problem in the TTS system and the ITN in the ASR system. It also shows that TN is TTS's preprocessing module while ITN is ASR's postprocessing module. Because of this relationship, some studies also show the use of existing TN systems to generate training data for the ITN task [13, 18, 26]. Besides, there are also differences between ITN and TN problems. Punctuation marks and

Spoken form	**Written form**
bốn phần trăm của năm ngàn là hai trăm bốn phần trăm của năm nghìn là hai trăm (four percent of five thousand is two hundred)	4% của 5000 là 200 (4% of 5000 is 200)
hôm nay là thứ ba ngày hai mươi tư hôm nay là thứ ba ngày hai mươi bốn hôm nay là thứ ba ngày hai bốn hôm nay là thứ ba ngày hai tư (today is tuesday the twenty-fourth)	Hôm nay là thứ 3 ngày 24 (Today is Tuesday the 24th)
a lô anh hai hai giờ chiều qua đón em nhé (hello hai please pick me up at two o'clock in the afternoon)	Alo, anh Hai 2 giờ chiều qua đón em nhé (Hello Hai, please pick me up at 2 pm in the afternoon)

Table 1.1: Example of Vietnamese spoken form and corresponding written form

capital words can be recovered from the written form at the input for the TN problem, while these two problems need to be solved in the ITN problem. Table 1.1 shows examples of spoken form sentences and their corresponding written form. In the first two examples, a sentence in the written form can have multiple variations in the spoken form. ITN needs to normalize all of these variants to their correct written form. In the third example, the written form also contains a proper noun (person's name) and a punctuation mark (comma). It is also a challenge with the ITN problem when compared with TN.

There are three main approaches to these two TN and ITN problems [26]. The first and most popular approach is based on weighted finite-state transducer (WFST) grammars. This approach proved to be effective but language dependent. Recently, the deep learning approach based on sequence-to-sequence model using Bi-LSTM [17] or Transformer [18] has shown better performance. Not only that, this method can be easily implemented for different languages. However, its biggest drawback is that it can cause unrecoverable errors. The third approach is to combine the above two methods to minimize the dependence on grammar rules as well as limit the errors generated by the deep learning model.

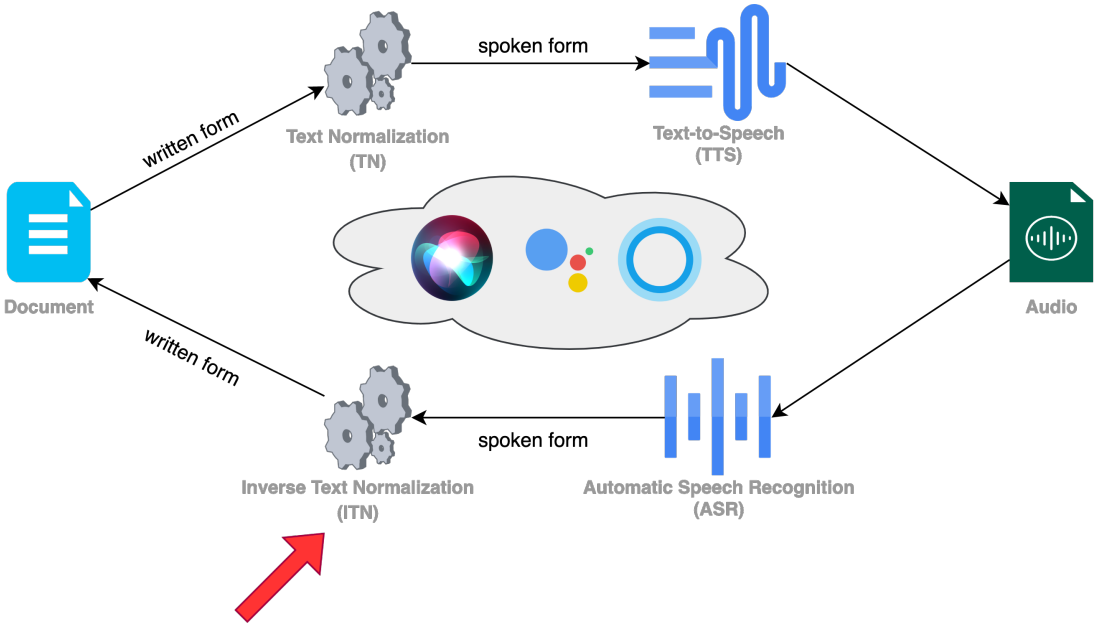

Figure 1.1: The ITN task in the speech processing system

1.2 Problem statement

A recent study shows that the deep learning method using the Transformer model gives more impressive results than the method based on WFST. It also shows superiority in building and developing the system because the deep learning method can be easily applied to different languages. Whereas WFST is language bound when it depends on grammar rules. The most significant difficulty when applying this deep learning method is that it requires a lot of labeled data. Therefore, it would be great to solve the data labeling problem partially. Several recent studies use a text normalization system based on WFST to generate labeled data. However, this method has certain limitations when using written data (e.g., Wikipedia, machine translation data), while ITN is a problem applied to spoken form data. In addition, the data generated from the WFST method, which is based on grammar rules, can cause the data to lose its diversity and naturalness.

Besides advantages, the use of transformer-based sequence-to-sequence (seq2seq) models also has the disadvantage of generating unrecoverable errors. This problem usually occurs when the input contains words or phrases that the model rarely encounters or has not encountered during training. This issue explains why we need a lot of labeled data for model training. Therefore, we need a solution to deal with those rare words to solve the

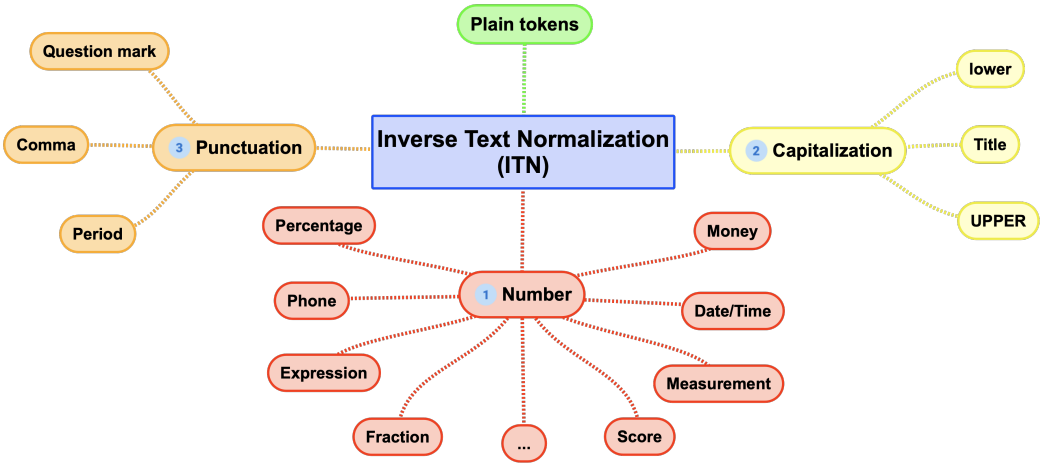

Figure 1.2: The overview of ITN task

problem of generating unrecoverable error sentences. The approach of recent studies is to apply the subword method (i.e., byte pair encoding) or combine it with WFST to reduce the error, showing remarkable effectiveness. Nevertheless, can we use a more straightforward method to deal with the rare word problem for ITN when the rare words consist only of numbers or words that contain digits?

ITN is a problem normalizing spoken text (ASR output) to the corresponding written form. Therefore, the input sentence will contain only words in lowercase and no punctuation. This missing is a challenge compared to the TN task, which still has punctuation information and proper nouns. The absence of these two pieces of information makes understanding the context more difficult. Restoring the punctuation and capitalization of proper nouns is also challenging, especially for proper noun information. Figure 1.2 shows the general picture of ITN, which can be divided into three subtasks: (1) normalization of entities, (2) punctuation restoration, and (3) capitalization restoration. Recent studies have used written data to train the model to overcome the problem of missing labeled data. Therefore, they do not have experiments with these two subproblems of ITN. It would be great if we could use a single model for all three of these subtasks.

1.3 Research motivation

ITN is essential in normalizing ASR output into written forms people use daily. The deep learning-based problem approach using the seq2seq model has proven more efficient than the traditional approach. Using the Bidirectional LSTM model (Bi-LSTM) to recent Transformer architecture shows its performance in the ITN task, including accuracy and scalability compared with WFST. The Transformer's ability to capture contextual information and non-sequential processing combined with positional encoding can explain why the method achieved a good performance. Additionally, the introduction of pre-trained language models based on Transformer architecture makes it much more feasible to train natural language processing problems. Because these pre-trained models significantly reduce the amount of labeled data to finetune the model compared to having to train from scratch.

Studies show that using the seq2seq model based on Transformer architecture on ITN task has achieved good results. In addition, studies using pre-trained language models such as BERT to finetune the target task give good results on many NLP problems. Finetune from pre-trained is significant for NLP problems in low-resource languages like Vietnamese. For the problem from the seq2seq model generating unrecoverable errors when encountering rare words, using a subword method like byte-pair-encoding (BPE) or sentencepiece will improve significantly. For this ITN task, we believe that the rare word problem can be solved more simply when the ASR output is definitely the words from its vocabulary list. Inspired by the above successes, we propose an end-to-end model based on a pre-trained language model combined with subword processing to solve the ITN task. This proposal helps to train the model with very little labeled data and still achieve good performance with a low error rate.

1.4 Research objective

We proposes a seq2seq model based on the transformer architecture that uses the pre-trained models to finetune ITN task using less labeled data for all three ITN subtasks: (1) normalization of entities, (2) punctuation restoration, and (3) capitalization restoration.

Since ITN is a specific task for ASR systems, it has only been interested in the last few years, so there are no public data sources. Some studies use WFST for automatic data generation from machine translation datasets, Wikipedia. However, this has specific differences from the actual problem and is affected by the performance of WFST. Our goal is to test the proposed

model on speech data. Although labeled speech data is limited, a solution using a pre-trained language model will assist in making this goal possible.

Besides, reducing the unrecoverable error rate on the seq2seq model is also an objective of this work. unrecoverable errors will be encountered more when the data size used for training is smaller. Nevertheless, the characteristic of ITN is the limited vocabulary. Our proposed method using subwords together with the use of a pre-trained language model set the goal to reduce unrecoverable errors.

1.5 Thesis organization

We organize the structure of this report into six chapters. The first chapter introduces the problem of this work, while the last is a conclusion about this work. The remaining chapters of this thesis, in turn, present related works, proposed methods, experimentation, and evaluation. Specifically, the remaining chapters of this thesis are summarized as follows:

Chapter 2 presents recent studies related to TN and ITN tasks. We divide recent research into three main directions. In each research direction, we summarize the solutions of related works and discuss each method's advantages and limitations. After that, we present the background knowledge related to this work, and the final is the dataset information for the ITN task.

Chapter 3 describes our method in detail in this thesis. First, we introduce the baseline model architecture using FST-based for Vietnamese. Next, we present the proposed method using the seq2seq architecture based on the pre-trained Transformer architecture. Along with that is the subword processing method to reduce the unrecoverable error rate of the neural network on the ITN.

Chapter 4 illustrates our experimental process on the ITN task. The content of this chapter includes detailed information about the data sets used, the metrics used to evaluate the proposed model, and the configuration details used in our model. Finaly, we shows the results of our experiments on the proposed method for Vietnamese and English data sets, respectively. For English, we also compared our results with recent studies. Next, we discuss and analyze the experimental results.

Chapter 2

Literature Review

2.1 Related works

While several studies have contributed to the TN task, the ITN task has only received attention in the last few years when the ASR problem has the potential for use in production. In addition, ITN is closely related to the TN task, so we will explore studies related to both tasks in this literature review.

We can divide the literature on the ITN and TN into three main directions:

- The rule-based methods, e.g., based on the language's grammar rules.
- Neural network models, typical using sequence-to-sequence models.
- Hybrid methods that aims to overcome the weaknesses of the two above methods.

The following subsections discuss each approach to the problem in detail.

2.1.1 Rule-based methods

Using WFST grammars is a typical method for TN and ITN [3, 26]. This method works based on a set of grammar rules of a particular language that help find and classify words or phrases in an input sentence. Based on the classification results, entities that are not plain tokens will be processed to their correct written form. Studies using this method achieve high accuracy.

Ebden et al. [3] proposed a TN system called Kestrel that achieved overall accuracy on the Google Text Normalization dataset [17] of 91.3% and 93.1% on the English and Russian subsets, respectively. The author evaluates the

system's accuracy by category (e.g., date, decimal, money, telephone, address), showing that many categories have absolute accuracy. However, high accuracy in a category on the English set does not mean high accuracy in Russian and vice versa. Besides evaluating the model's accuracy, the author also shows that more than 90% of the system errors reported by users have been fixed. This number shows that errors caused by WFST can easily find the cause of the error and fix it.

Recently, Yang Zhang et al. [26] announced an open-source Python WFST library for ITN called NeMo ITN. This library makes building and deploying ITNs more convenient and efficient. Figure 2.1 shows that the NeMo ITN system pipeline consists of two main stages: (1) The classify step uses WFST to detect and classify the tokens in the input sentence; and (2) the verbalize step is responsible for converting the tokens into the corresponding written form. Meanwhile, the two intermediate steps, parse and generate permutations, act as support for the verbalize step. The Google Text Normalization dataset was used to evaluate the performance of the proposed method. The author has identified some problems when using TN data to evaluate ITN. Specifically, several samples in the dataset are similar in spoken form but different in written form (e.g., the spoken form of "2000" and "2,000" are "two thousand" or "5:00 pm" and "5 pm" has the same spoken form as "five p m"). Because ITN always only generates one single written form for each spoken form input, this will result in lower accuracy (e.g., "two thousand" $\rightarrow$ "2000" or "2,000" are both correct). To overcome this problem, the author uses regex rules to handle the above cases, if possible, and obtain a "cleaned" version of the original dataset. The author uses the Word Error Rate (WER) metric to evaluate the method's performance. The experiment shows results of 12.7% and 10.14% on the original dataset and the "cleaned" version, respectively. The WER significantly improved on the "cleaned" dataset compared with the original across many categories, especially measure, money, and number.

Using WFST does not require labeled data or model training. This is an advantage of the WFST approach compared to the neural network approach. This solution also shows good accuracy in recent studies. However, this method requires expert linguists to build the system because it depends on linguistic rules. Besides, it is not easy to reuse an existing system for another language because the languages have different grammar. The author of Kestrel [3] has noted that system development time depends on language complexity. Russian or Slavic languages will take longer than English due to morphological complexity. The scalability limitation is confirmed when Kestrel's research results have no similarity in English and Russian.

In conclusion, WFST is suitable for a single-language system. Deployment and debugging are possible with language experts without the cost of making

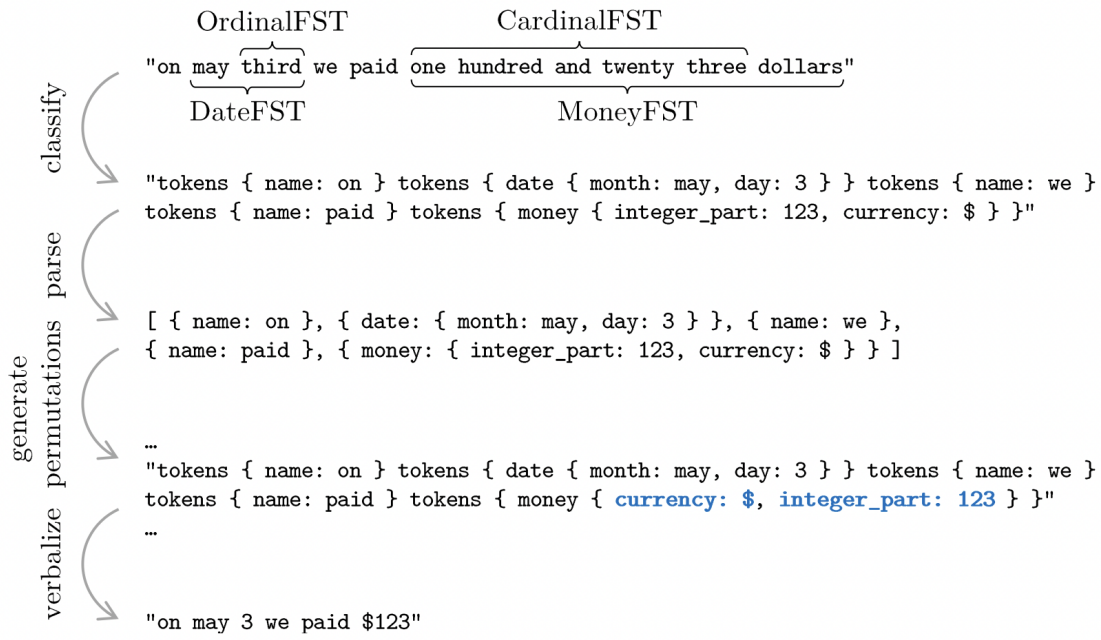

Figure 2.1: The NeMo ITN's pipeline [26]

labeled data. However, this solution will be complex with a system that supports multiple languages, and the development cost will increase with the number of languages that need to be supported.

2.1.2 Neural network methods

Unlike the WFST grammars-based approach, training ITN or TN on a neural network model requires a large amount of labeled data to train the model. Studies in this direction often use the seq2seq model [9, 12, 4, 7].

Mansfield et al. [9] consider TN as a machine translation task and solve it using the seq2seq model with Bi-LSTM architecture. This proposed method helps the model to have more context information when the whole input sentence is used instead of just words or phrases like the FST/WFST method. Besides, the author proposes using subwords instead of word level to improve the out-of-vocabulary (OOV) problem. Two different baseline models were tested, including the window-based model (i.e., wrapping the tokens in blocks and using the keyword "self" to represent non-normalize words) and the sentence-based (i.e., using full-sentence for training). The proposed model uses the sentencepiece[1] toolkit to apply subword processing. Besides, the au-

[1]https://github.com/google/sentencepiece

thor also added linguistic features for the model training process. The author divided the English subset in the Google Text Normalization dataset [13] to train and evaluate the model. When subwords were not applied, the window-based model outperformed the sentence-based training. However, the author also implies that using the window-based has limitations and is only used as a baseline. When applying subwords, the model's results reached 0.91% and outperformed both baseline models, and when adding four linguistic features, the WER was only 0.17%.

Pramanika et al. [12] use two models to solve the TN task. The first model uses XGBoost to classify whether each word in a sentence needs to be normalized or not. The characters of the current word and its neighboring will be represented as vectors of Unicode values for training the classifier. If a word is determined to be normalized, it is passed through the second model. The second is a seq2seq model that uses the Differentiable Neural Computer (DNC) architecture to replace Bi-LSTM. This model normalizes words from written to spoken form. Experimental results show that changing from Bi-LSTM to DNC with much smaller data used for training, the model still gives competitive results with the previous study of Sproat et al. [17].

Besides the direct study of ITN, it is also considered in a larger problem of ASR post-processing for readability (APR) proposed by Liao et al. [7]. Their goal is to convert noisy ASR output into readable text for humans and downstream tasks while preserving the original meaning of the speaker. The author has trained the model on different Transformer architectures and open-source pre-trained models. The study also built a pipeline to build data for APR from public data sources to evaluate the effectiveness of models. The experimental results show that RoBERTa [8] is the more suitable model for most evaluations.

In conclusion, studies using neural networks for ITN give better results than the traditional WFST method. The model learned from a large corpus can understand the context better than WFST is why we get a better result. Some studies incorporate additional entity boundaries (e.g., <s> entity here </s>) in the data to help the model learn better [9, 17]. Besides the advantage of efficiency, the method also shows that it can solve the scaling problem of WFST. Neural network models do not depend on a particular language. We can implement the same successful network architecture from one language to another if we have labeled data with very little modification compared to WFST. However, the neural network approach also has its limitations. First, the lack of labeled data impedes model training, especially deep learning. This problem is proven when many studies have built an automatic pipeline to generate ITN data using text normalization system [13, 18, 26, 7]. Another limitation of neural networks is that the model can generate

unrecoverable errors. While WFST errors are usually easy to find and fix, errors in deep learning models are often empirical. The easiest solution for the error of the neural model is to use more data to train the model.

2.1.3 Hybrid methods

For the ITN task, using WFST grammars requires a linguistic expert, and the ITN performance depends heavily on handwritten grammar rules. An error occurs if the grammar rules are missing or do not cover all cases. Meanwhile, the most significant limitation when using neural networks is that the model can generate unrecoverable errors. Therefore, some studies propose hybrid methods to limit the weaknesses of WFST and neural networks [16, 17, 1, 25, 18].

Shugrina [16] proposed a method of combining hand-crafted grammar with the language model to restore punctuation, capitalization, and normalization of numeric entities. Specifically, hand-crafted grammars find all potential variations of the written form of tokens that need to be normalized. Then, finding the most suitable variant was evaluated by a language model trained from written text. Although using a language model trained on written text is appropriate for evaluating written form variations, this approach encounters the obstacle of sparse numerical data. To overcome this obstacle, the author classifies numeric values into different classes based on the range of values before training the language model. The Numeric Entity Error Rate (NEER) metric evaluates the system's performance in their work. The results on their own dataset show that the system achieves NEER of 16.1% on exact match and 11.2% when ignoring spaces. For punctuation and capitalization, the F-score of their system achieved 0.64, 0.40, and 0.67 on the capitals, commas, and periods, respectively.

To overcome the unrecoverable error limitation of the neural network for TN, Sproat et al. [17] use a hybrid of an RNN model with an FST-based filter. The author implies that although RNN gives better accuracy than FST, it still produces unrecoverable errors. FST was used to detect and correct those error cases on entities related to money and measures categories. Experimental results show that this combination helps to increase the accuracy of money and measures categories by about 2% when compared to using only RNN.

To minimize dependence on grammar rules, Alphonso et al. [1] proposed a ranking-based approach of written texts to the ITN. First, Each output hypothesis of ASR's output will be put in the candidate generation stage. Second, the generated candidates' written form and features are put into the candidate selection stage. In the first stage, written form candidates are

generated by a WFST-based model along with the entity category information. In the ranking stage, n-gram and LSTM language models generate numeric features for each candidate. Experimental results show that the proposed method achieves an 18.48% relative reduction in WER compared to the baseline model (using FST-based). This result indicates that using additional features gives better results than just using the language model for ranking.

Sunkara et al. [18] have proposed using a combination of transformer-based seq2seq model and FST-based to solve the ITN task. The limitation of the traditional method is reusability and extensibility. The author has tested the proposed method in English, German, Spanish and Italian. Experiments show that the success of neural networks in ITN tasks in different domains and languages has solved the limitation of WFST. Besides, the author proposes using FST as a fallback result in case the transformer model returns a low probability (unrecoverable errors seem to occur). This hybrid has helped the ITN system in production when the FST can partially assist in reducing the errors produced by the neural network model.

2.2 Background knowledge

To explain why we chose to use the pre-trained seq2seq model based on transformer architecture to finetune the ITN task, we would like to present an overview of the knowledge related to our work. The content in this section includes the transformer architecture, pre-trained language models based on the transformer, and the subword methods. They form the basis for the proposed method that we will present in the next chapter.

2.2.1 Encoder-Decoder model

The encoder-decoder model was first introduced by Sutskever et al. [19] to solve the seq2seq problems. This model aims to map inputs with outputs of different lengths. These seq2seq problems can be machine translation, text generation, and question answering. TN and ITN are also two tasks typically considered seq2seq tasks. Figure 2.2 illustrates the high-level overview of the encoder-decoder model. At a high level, an encoder-decoder model consists of two blocks, the "encoder block" and the "decoder block", which are connected through a vector called the "context vector".

- **Encoder**

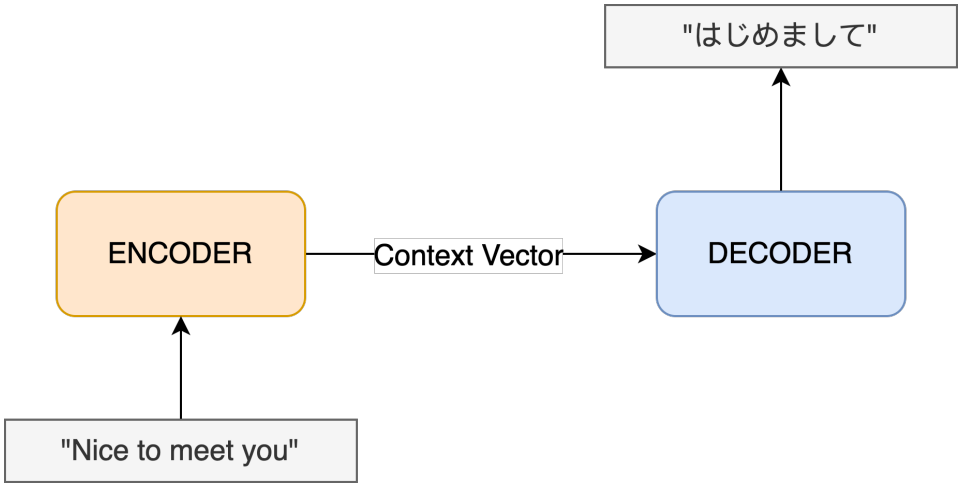

Figure 2.2: High-level overview of the Encoder-Decoder model

The encoder block is a list of recurrent units, such as Long short-term memory (LSTM). The input is passed into the encoder block as a sequence, and each cell takes part of the input (i.e., a token of the input sentence), captures its information, and forwards it to the next cell. Each cell attempts to learn all its information and hold its value in its final internal state. The final internal state of the last cell denoted the entire encoder block and passed to the first cell of the decoder block, and this is the context vector we mentioned above. In the encoder block, the outputs at each step are all discarded. Figure 2.3 shows the illustration of the encoder block.

The following formula 2.1 tells us how the cell in the encoder block calculates its final internal state.

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t) \tag{2.1}$$

Where:

- h_t is the current cell,
- W^{hh} is the weight of the previous cell,
- h_{t-1} is the previous cell,
- W^{hx} is the weight of the current cell,
- x_t is the current input cell.

- **Decoder**

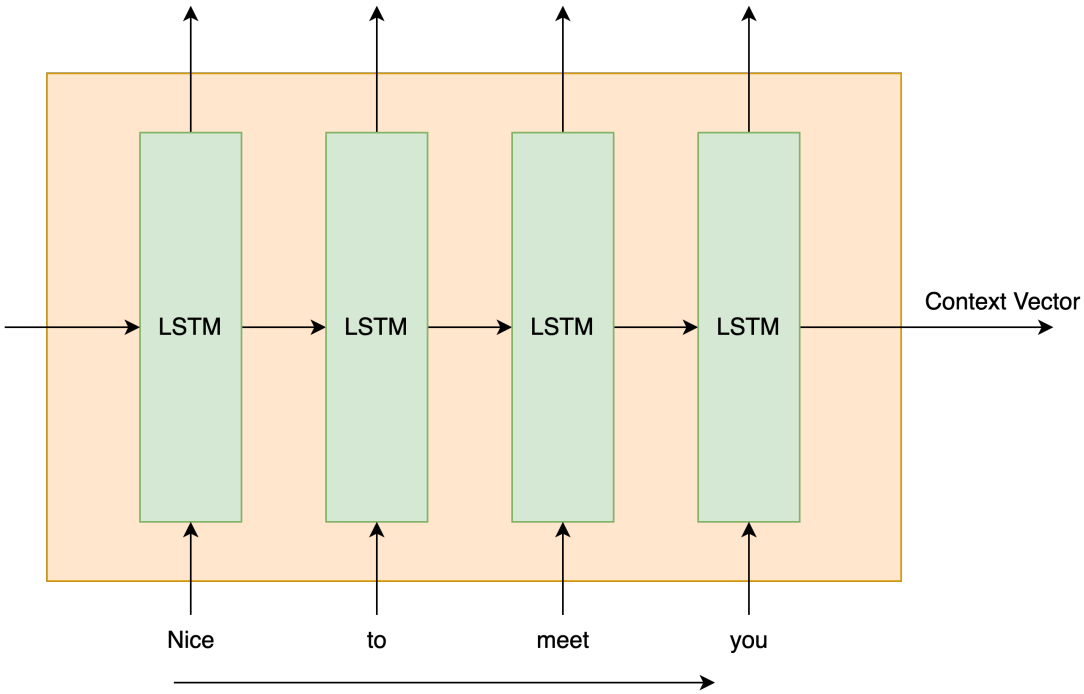

Figure 2.3: Illustration of the encoder block

After reading the whole input sequence, the first cell in the decoder acquires the context vector from the encoder block, where the output-sequence prediction initiates. The decoder block is also a list of recurrent cells, for example, LSTM cells. The initial state of the decoder is the context vector taken from the last cell of the encoder. In the decoder block, we tended to the output and discarded the final internal state of the last cell. Figure 2.4 shows the illustration of the decoder block.

The following formula 2.2 calculates the final internal state of a cell in the decoder block.

$$h_t = f(W^{hh} h_{t-1}) \tag{2.2}$$

Furthermore, the output of the current cell is calculated by the formula 2.3:

$$y_t = \text{softmax}(W^s h_t) \tag{2.3}$$

Where:

- W^s is the weight of the current cell,
- The softmax function here generates a probability vector that helps

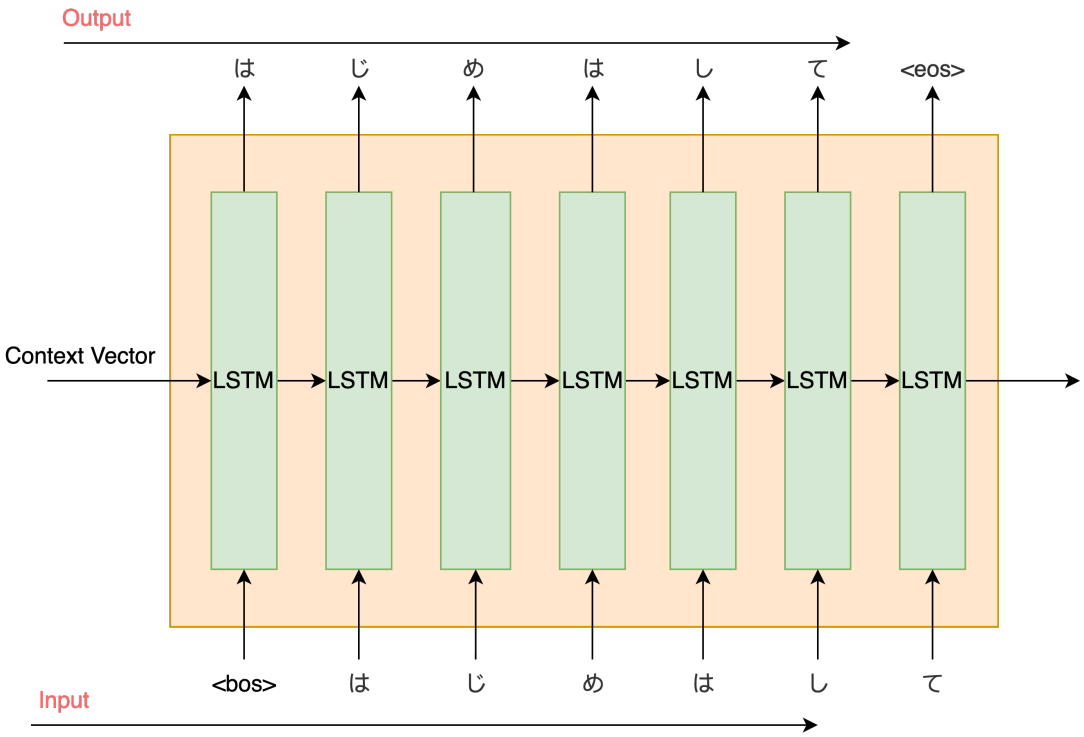

Figure 2.4: Illustration of the decoder block

evaluate the reliability of the final result.

2.2.2 Transformer

The Transformer architecture was proposed by Vaswani et al. [23]. At a high level, the transformer follows an encoder-decoder architecture without relying on the recurrent. Transformers are better than the encoder-decoder models above because this architecture can skip the recursion. Transformers process the whole input sentence at once and learn relationships among words by the multi-head attention mechanisms and positional embeddings.

In the paper, the encoder is a stack of six identical layers. In which each layer includes two sub-layers. The first sub-layer is a multi-head self-attention mechanism, while the remaining sub-layer is a fully connected feed-forward network. There is a residual connection in each of the two sub-layers, followed by a layer normalization. So, the output of each sub-layer is $\text{LayerNorm}(x + \text{Sublayer}(x))$, where $\text{Sublayer}(x)$ is the sub-layer itself function. The transformer encoder block does not capture any information about the relative positions of the token in the input sequence because it does not

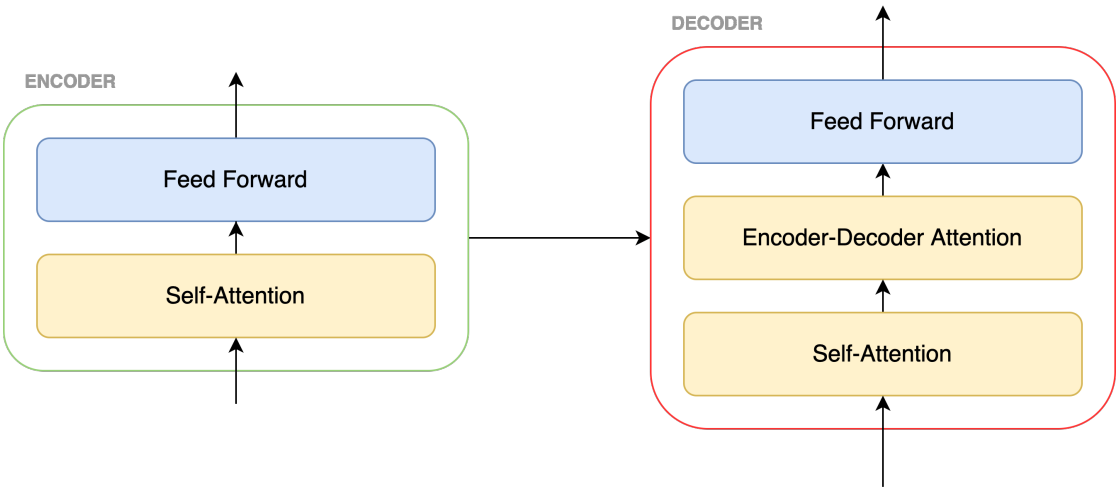

Figure 2.5: The Encoder-Decoder of the Transformer architecture

operate recurrence. Instead, the position information is replaced by new positional encodings vectors. The positional encodings are calculated by sine and cosine functions in different frequencies and return the vectors by the same length as the input embeddings. Then, positional encodings vectors are added to the input embeddings to supplement the positional information of the tokens in the sentence.

The decoder block also is a stack consists six identical layers. As shown in figure 2.5, each layer in the decoder also has two sub-layers like in the encoder. However, an additional attention layer is added between them to help the decoder attend to the output of the encoder stack. Similarly, the three sub-layers also have residual connections around each sub-layers followed by a normalization layer. Input embeddings in the decoder also added the positional embeddings in the same way as the encoder.

The self-attention mechanism in the Transformer helps the encoder look at other tokens in the input to find the relevant to the current token. This mechanism helps improve the encoding of the current token. Consider the following sentence:

"John took early retirement last year because of his bad health."

What is the word "his" mean in the above sentence? It is very easy for a human when we look at other words in the sentence, but not simple for a computer. That is why we need self-attention to embedding words in a sentence, which helps improve the current token's encoding.

The following formula 2.4 helps to calculate the self-attention of each

token in a given sentence.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.4}$$

Where:

- Q: The query matrix,
- K: The key matrix,
- V: The value matrix,
- d_k: The dimension of Q, K, V.

The query (Q), key (K), and value (V) vectors are calculated by multiplying the X matrix by the weight matrices we trained (W^Q, W^K, W^V), where X is a matrix whose row corresponds to a token in the input sentence.

The encoder-decoder attention layer performs similarly to multiheaded self-attention. However, this attention layer creates the queries matrix from the layer below, while the keys and values matrix are taken from the output of the encoder stack.

2.3 Dataset

There are not many public datasets for the ITN and TN tasks. Studies often use text normalization developed based on WFST to generate data for ITN automatically. Currently, the only publicly available large dataset of the TN problem is the Google Text Normalization dataset [17]. This dataset is collected from Wikipedia in written form, and its corresponding spoken form is semi-automatically generated by the Kestrel system [3] - a TN system developed based on WFST. The dataset consists of two sets, English with 1.1 billion words and Russian with 290 million words.

Because the Google Text Normalization dataset is a pair of spoken and written form sentences, we can use it to evaluate the ITN task. Recent studies have used this dataset's English set to evaluate the model's effectiveness. This work will also use this data set to evaluate and compare the results with previous studies.

Chapter 3

Methodology

In this chapter, we will present the proposed method that we will use for the ITN task focusing on the Vietnamese language. However, there is currently no research on ITN for Vietnamese similar to our method or experiments on public data. Specifically, there is only a study on ITN for Vietnamese proposed by Tran et al. [22] proposes a method using handcraft grammar rules combined with Convolutional Neural Networks (CNNs) to normalize ASR output. The author has experimented with restoring capitalization for proper nouns and normalizing numeric entities on the data they built. Therefore, we will use an FST-based model we developed as the baseline model for this work.

The content presented in this chapter will be in the following order:

1. First, we summarize the pre-trained language models used in this work.
2. We introduce the baseline model architecture that we built for Vietnamese.
3. We presented details of the proposed method for ITN.
4. The subword processing method combined with sentencepiece will be applied to improve the unrecoverable error rate of the seq2seq model.

3.1 Preliminaries

3.1.1 T5

Raffel et al. [14] have published the transformer-based encoder-decoder system called Text-to-Text Transfer Transformer (T5). T5 treats all NLP tasks as a unified text-to-text format where all input and output are strings. This

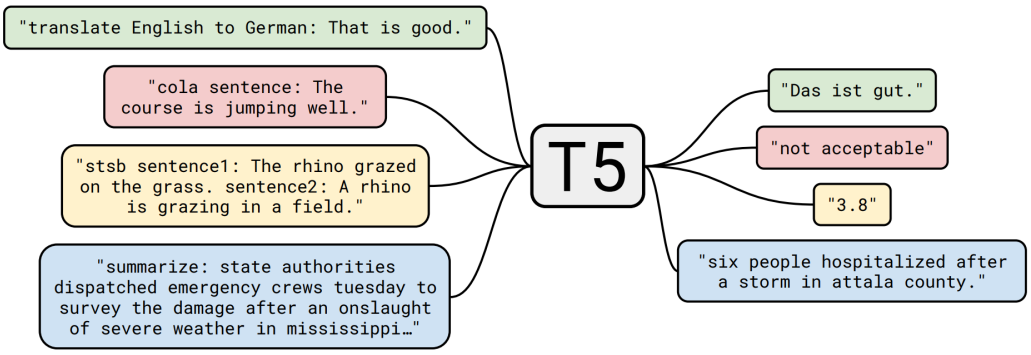

Figure 3.1: A diagram of T5 framework [14]

format makes T5 suitable for solving many NLP problems, such as text classification, machine translation, and question answering. The author also implies that T5 can be used to train regression tasks by training it to predict string representations of numeric values. Figure 3.1 is a diagram extracted from the article showing the ability of T5 when it comes to solving many NLP tasks.

Unlike BERT or GPT, which only uses the encoder or decoder part of the Transformer, T5 proves that using both encoder-decoders is more efficient than using only the decoder. Another factor of interest is the data used to pre-train the T5 model. The author published and used the C4[1] dataset, a cleaned version of Common Crawl[2] dataset with a size of about 800GB. A high-quality and diverse dataset makes the model's learning efficient.

Based on the original T5, it is currently available in other variations, including:

- T5v1.1: An upgraded version of T5 with some architectural modifications and is pre-trained on C4 without including the supervised tasks.
- mT5: The multilingual T5 version is pre-trained on the mC4 corpus and includes 101 languages.
- byT5: A T5 model pre-trained using byte sequences instead of SentencePiece subword processing.

We use the mT5 model to experiment with English and Vietnamese languages in this work.

[1]https://www.tensorflow.org/datasets/catalog/c4
[2]https://commoncrawl.org/

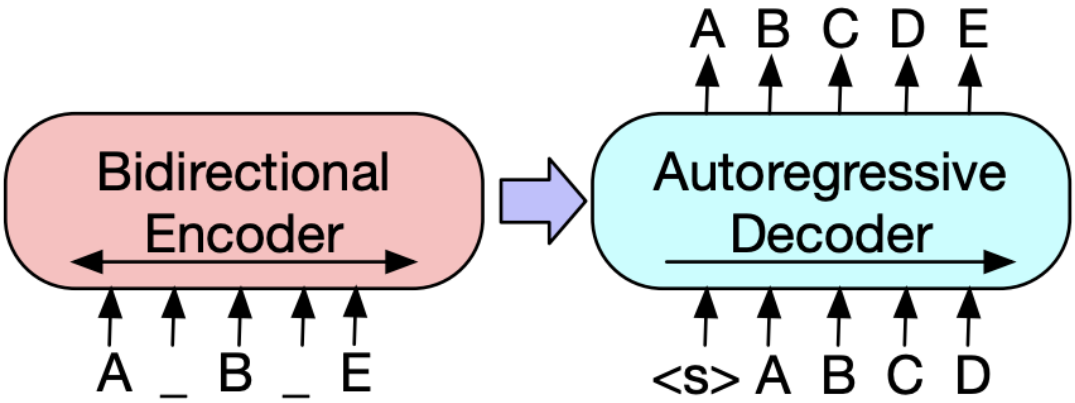

Figure 3.2: BART is a combination of a bidirectional encoder and an autoregressive decoder. [6]

3.1.2 BART

BART (Bidirectional Auto-Regressive Transformers) is an encoder-decoder model nearly identical to the transformer architecture proposed by Lewis et al. [6]. The difference between BART and the original transformer is:

- Replace RELU with GeLU
- Difference layer size, base version (6 encoder layers, 6 decoder layers), and large version (12 encoder layers, 12 decoder layers)
- The decoder layer has an additional cross-attention layer
- Without the feed-forward at the very end

BART uses both encoder and decoder, while BERT uses only a bidirectional encoder, and GPT uses only an autoregressive decoder. In the pre-training tasks, BART upgrades the complexity of the pre-training task to include five different noising approaches:

- Token masking like BERT
- Text infilling is like token masking, but one mask can hide a span of text (more than one token)
- Sentence permutation splits the document into sentences and shuffles the sentence order

- Document rotation predicts the starting position of a document after the start position and order of sentences are changed

BART is helpful when finetuning text generation tasks and works well for comprehension tasks. BART's performance matched RoBERTa's performance with comparable training resources on GLUE and SQuAD. In the experiments, BART performed new state-of-the-art results on a wide range of abstract dialogue, text summarization, and question-answering tasks. That is also the reason that we choose BART for our experiment. In this study, we used the BART multilingual version named mBART.

3.1.3 BARTpho

In this work, we use BARTpho, a pre-trained seq2seq model for Vietnamese published by Tran et al. [21]. BARTpho utilizes the pre-training strategy and "large" architecture of the seq2seq denoising autoencoder. To train the model, the author used PhoBERT data [10] that contains 20GB of uncompressed text data. The pre-training task is like BART and includes sentence permutation and text infilling. BARTpho comes in 2 different variants, word and syllable. BARTpho-word uses a Vietnamese word tokenizer at the word level. While BARTpho-syllable uses the sentencepiece toolkit to tokenize input.

To demonstrate the effectiveness of BARTpho in Vietnamese, the author compared BARTpho with multilingual BART (mBART) of the same size model on text summarization and punctuation and uppercase recovery tasks. Experimental results show that BARTpho outperforms mBART on all experiment tasks. Besides, the BARTpho-word variant gave slightly better results than the syllable variant.

3.2 Baseline model

The purpose of this model is to normalize entities related to numeric types (e.g., date, time, number, address, ...) from spoken to written form. The model is built based on handwritten grammar rules combined with language models. Figure 3.3 illustrates the architecture of this model, which consists of four main components: entity boundary detector, classifier, parser, and language model.

Entity Boundary Detector is a module responsible for finding all possible words or phrases that need to be normalized in the input sentence. This search relies on sentence matching against a set of pre-defined keywords. The results found by the entity boundary detector are usually not very accurate,

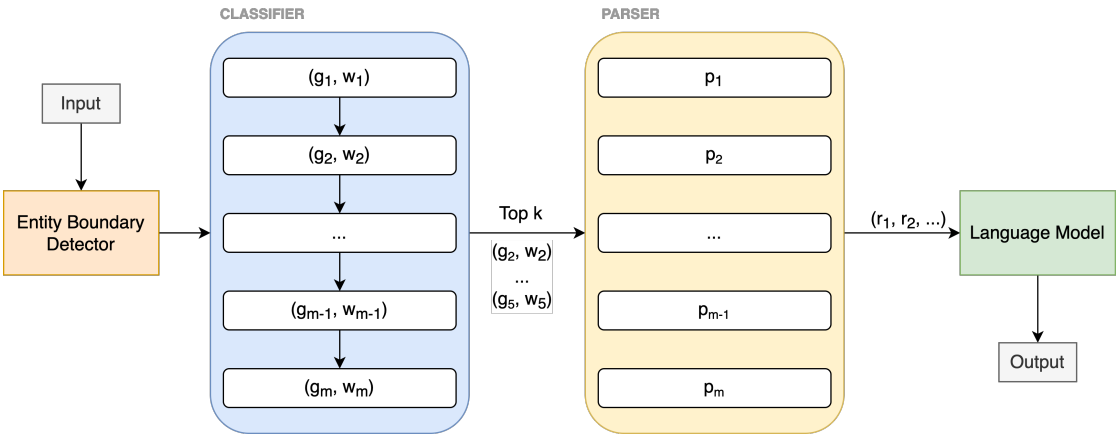

Figure 3.3: Overview of the baseline ITN model for Vietnamese

but the module has good coverage. Entities detected in this step will be passed to the classifier module for classification and processing.

The **classifier** consists of a set of m grammar rule groups $(g_1, g_2, ..., g_m)$ that classify entities detected in the previous step. Each group of grammar rules comes with a weight value $(w_1, w_2, ..., w_m)$. The weight is used to evaluate the priority of grammar rules. For example, each entity detected by the entity boundary detector will be checked against each rule according to weight priority. When various rules match an entity, weights are also used to rank and derive the top k results with the highest weight before moving on to the next step.

The **parser** is a module that converts spoken form entities provided by the classifier into their corresponding written forms. Each group of grammar rules g_i has a corresponding parser p_i. Therefore, each result obtained from the previous step will be passed directly to the corresponding parser for processing. These parsers perform normalization based on Vietnamese grammar rules. In some unsure cases, the parser may return more than one written form per entity, especially for number entities. Then, the language model in the next step will assist in finding the most suitable written form.

The working principle of **language model** in this system is inspired by Shugrina [16]. This language model is trained from written form sentences data. Suppose a spoken-form entity has more than one variant written form. In that case, the language model evaluates the variations in the context with the words surrounding it to find the most suitable variant. We use the language model's perplexity score to rank variants. Finally, the spoken form entities in the input sentence will be replaced by the most suitable written form to complete the normalization process.

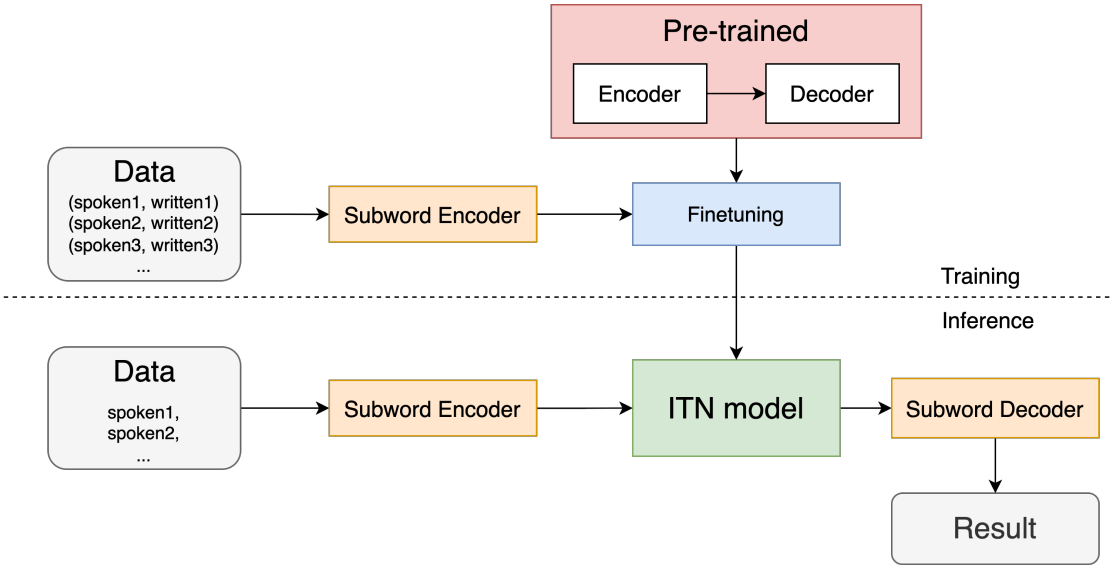

Figure 3.4: Illustrate the pipeline of the proposed method

3.3 Proposed model

We consider ITN as a machine translation task in this work and solve it using the seq2seq model. Formally, given an input as a sequence of m tokens in spoken form $x = (x_1, x_2, ..., x_m)$, and the output sequence consist n tokens in written form $y = (y_1, y_2, ..., y_n)$.

Inspired by the success of research using neural networks, especially transformers on TN and ITN tasks, we proposed using a pre-trained language model based on the transformer architecture combined with a data-driven approach to solving the ITN task. The goal of this approach is to use very little labeled data while ensuring competitive performance compared to previous work. Figure 3.4 presents an overview of our proposed method for the ITN task.

In the training phase, labeled data consisting of many sentence pairs $(x, y) \in (X, Y)$ where X, Y are the sources and targets data sets, is sent to the data preprocessing step. The data encoder and decoder are where we apply the subword technique that we present detailed in the next section. We then finetune the language models based on the transformer architecture for the ITN task. Similar to the inference phase, the input data has also applied a pipeline like the training process. However, at this stage, we only have the spoken form and will get the written form when moving through the trained ITN model.

We experiment with several pre-trained language models for the proposed

Name	Language	Size	Max length
mT5-base [24]	multilingual	12 encoders, 12 decoders	512
mT5-large [24]	multilingual	24 encoders, 24 decoders	512
BARTpho [21]	Vietnamese	12 encoders, 12 decoders	512
mBART-large [20]	multilingual	12 encoders, 12 decoders	512

Table 3.1: Information about the pre-trained models used for the experiment

method, including BERT [2], T5 [14], and BART [6]. Because we use the method for both English and Vietnamese languages, the multilingual versions of the above language models are used. However, we don't get good results on BERT. In addition, we experimented with pre-trained language models for existing monolingual Vietnamese, including viT5 [11], and BARTpho [21] but did not achieve good results compared to the multilingual version. Finally, we summarize information on pre-trained language models we used in Table 3.1. The max_length column specifies the max_length parameter that we use for the experiments in this work.

3.4 Subword processing

The seq2seq model performs poorly when encountering OOV or rare words. OOV and rare words also cause neural network models to generate unrecoverable errors. Previous studies also show that the most significant limitation of the neural network method on ITN is the unrecoverable errors [18]. Although the output of ASR has a fixed lexical size, its written form has no lexical limit, and this is because numeric data will appear when converted to a written form. The goal of ITN is to normalize spoken form to written form, which is mainly related to numeric types (e.g., "nineteen ninety-six" → "1996". That leads to OOV and rare words problem on ITN happening more frequently than machine translation. Specific words like "ten, one hundred, one thousand" will appear many times, but most spoken form numbers will be scarce in the training data (e.g., "one thousand two hundred thirty-four").

To solve the problem of OOV and rare words, studies on neural machine translation represent these words as subwords [15, 5]. Studies on the ITN task also show the effectiveness of applying the same technique to solve the rare word problem (e.g., sentencepiece toolkit[3]) [18, 9].

Inspired by the research of Mansfield et al. [9] in applying the technique of inserting underscores between digits (e.g., "1324" → "_1 _3 _2 _4"). In this work, we extend this technique to most tokens containing digits (e.g.,

[3]https://github.com/google/sentencepiece

Category	Original (written form)	Encode (written form)
Number	12345 We need 123 more products The result is 10.78	1 _2 _3 _4 _5 We need 1 _2 _3 more products The result is 1 _0 _. _7 _8
Date/Time	19:56:03 The current time is 19:59 on October 16	1 _9 _: _5 _6 _: _0 _3 The current time is 1 _9 _: _5 _9 on October 1 _6
Measurement	The minimum running speed is 7.3km/h Single room area is 15m2	The minimum running speed is 7 _. _3 _ km/h Single room area is 1 _5 _m2
Percentage	I have reached 50% of the target	I have reached 5 _0 _% of the target
Money	It costs ¥234	It costs ¥ _2 _3 _4
Expression	1+1=2	1 _+ _1 _= _2
Punctuation	Saturday, February 11, 2023	Saturday _, February 1 _1 _, 2 _0 _2 _3

Table 3.2: Example of encoding data with subwords on several categories

date, time, fraction, measurement, ...). However, we changed the rules for inserting underscores to make encryption and decryption easier. This technique saves numeric entities from the rare word problem where every numeric value will be represented by the characters '_0' to '_9'. Besides, we also apply it to punctuation marks, this does not increase the model's ability to learn punctuation recovery, but it helps the preprocess easily. Table 3.2 provides examples of how we encode tokens involving numbers and punctuation.

Chapter 4

Experiment and Result

4.1 Data

In this work, we train and evaluate our method on Vietnamese and English data sets. We use part of the Google Text Normalization dataset for English to train and compare results with other studies. For Vietnamese, we use two transcript datasets in two fields: social networks and academics. Below are details about the data sets and how we split the data for model training and evaluation.

The original Google Text Normalization dataset [17] for the English language contains 1.1 billion words extracted from Wikipedia. Its spoken form version was obtained using the Kestrel TN system [3]. In this work, we use part of this dataset (about 31 million words), a version of the Text Normalization Challenge - English Language on Kaggle[1]. In the competition, only the training dataset has labels, so we only use this set for model training and evaluation. From now on, we will call this piece of Google Text Normalization data that we used in this work "en_wiki".

The first Vietnamese dataset we used to train and evaluate the model is speech data on the social domain. The written form of this dataset is labeled by humans from ASR output in spoken form. The dataset includes 47230 pairs of spoken form and written forms. The data is picked with 90% of this dataset containing tokens that need to be normalized (i.e., containing numbers or proper nouns). This dataset is challenging when the data is free-style speech over an open domain. From now on, we will call this Vietnamese social dataset "vi_social".

The second Vietnamese dataset we used to train and evaluate the model is

[1]https://www.kaggle.com/competitions/text-normalization-challenge-english-language

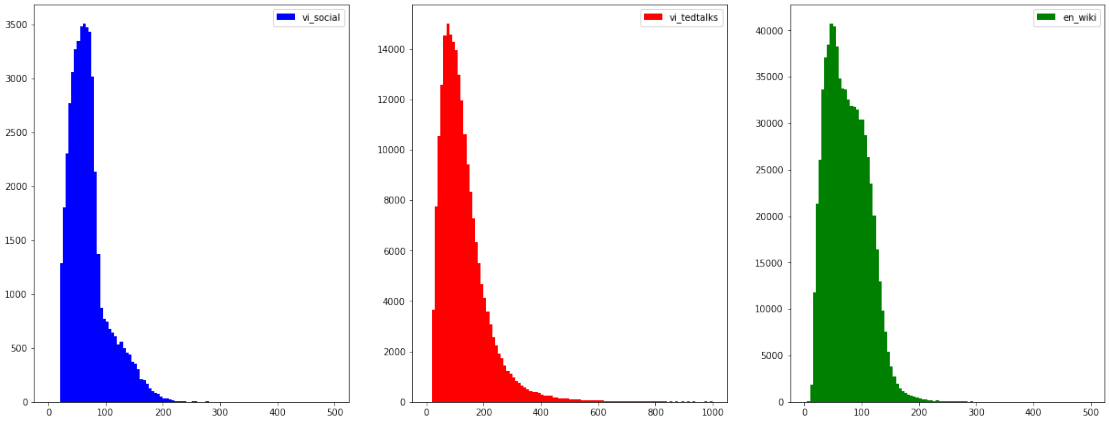

Figure 4.1: Spoken-form's length distribution of vi_social (left), vi_tedtalks (center) and en_wiki (right).

Dataset	**Train**	**Test (%)**	**Hard-test**
en_wiki	600000	148066 (20)	40069
vi_social	37784	9446 (20)	1020
vi_tedtalk	170000	46845 (21)	1185

Table 4.1: Information on how to use the data for the experiment

the Vietnamese transcript collected from TEDTalks[2]. It is a free educational video-sharing platform with subtitles. We have collected Vietnamese subtitles contributed by users on this platform. After collecting and processing based on audio chunks, we obtained 216845 subtitle segments. We then use an FST-based text normalization system to create spoken forms for the subtitles. We will name this dataset "vi_tedtalk".

To better understand the datasets, we observe the length of the data to ensure whether the proposed method is suitable or not. Figure 4.1 shows the length distribution (number of characters) across the data sets. Looking at the length distribution, we find that the lengths of the vi_tedtalks set have a wider distribution than the other two data sets. Generally, the length of these data sets does not exceed 1000 characters, which is why we chose max_length = 512 on pre-trained models.

We divide each dataset into approximately 80% for training and the remaining 20% for model evaluation purposes. Besides, to evaluate the effectiveness of the proposed subword method, we create additional "hard-test" sets, which are the complex items taken from the test set. A complex item

[2]https://www.ted.com/talks?language=vi

must contain at least a word with at least three digits and four different characters (e.g., It has increased by 123% from yesterday). Detailed information on how the data is divided in each data set (number of items) is presented in Table 4.1.

4.2 Metrics

We will first present the model evaluation method on the normalization of entities subtask. Following Sunkara et al. [18], we use Word Error Rate (WER), I-WER, and NI-WER to measure model performance. WER is used to measure the word error rate of the model on the entire target sentences, like previous works on ITN and TN. Meanwhile, I-WER only measures the word error rate on a group of words that need to be normalized, and NI-WER is the WER of words that need to copy from the source spoken form (non-normalized words). Figure 4.2 shows normalized and non-normalized words in a sentence. We use the Levenshtein distance to align the words in the source (spoken form) and target (written form). If a pair of words is the same, it will be labeled NI-words, and the rest will be I-words. Using I-WER and NI-WER helps us evaluate the model's performance on the group of words that need to be normalized while still observing the error rate on the group of words that need to be copied from the source.

Because our system not only normalizes entities but also predicts capitalization and punctuation. Therefore, before the WER evaluation, the punctuation will be removed, and the text will convert to lowercase. The WER on a pair of written form sentences, references and hypotheses is calculated using the equation 4.1 below.

$$\text{WER} = \frac{S + D + I}{N} = \frac{S + D + I}{C + S + D} \tag{4.1}$$

Where:

- C is the number of corrections,
- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- N is the number of words in the reference ($N = C + S + D$).

For the remaining subtasks, punctuation and capitalization restoration, we use the accuracy (%) metric to measure the model's performance on two tasks punctuation and capitalization restoration.

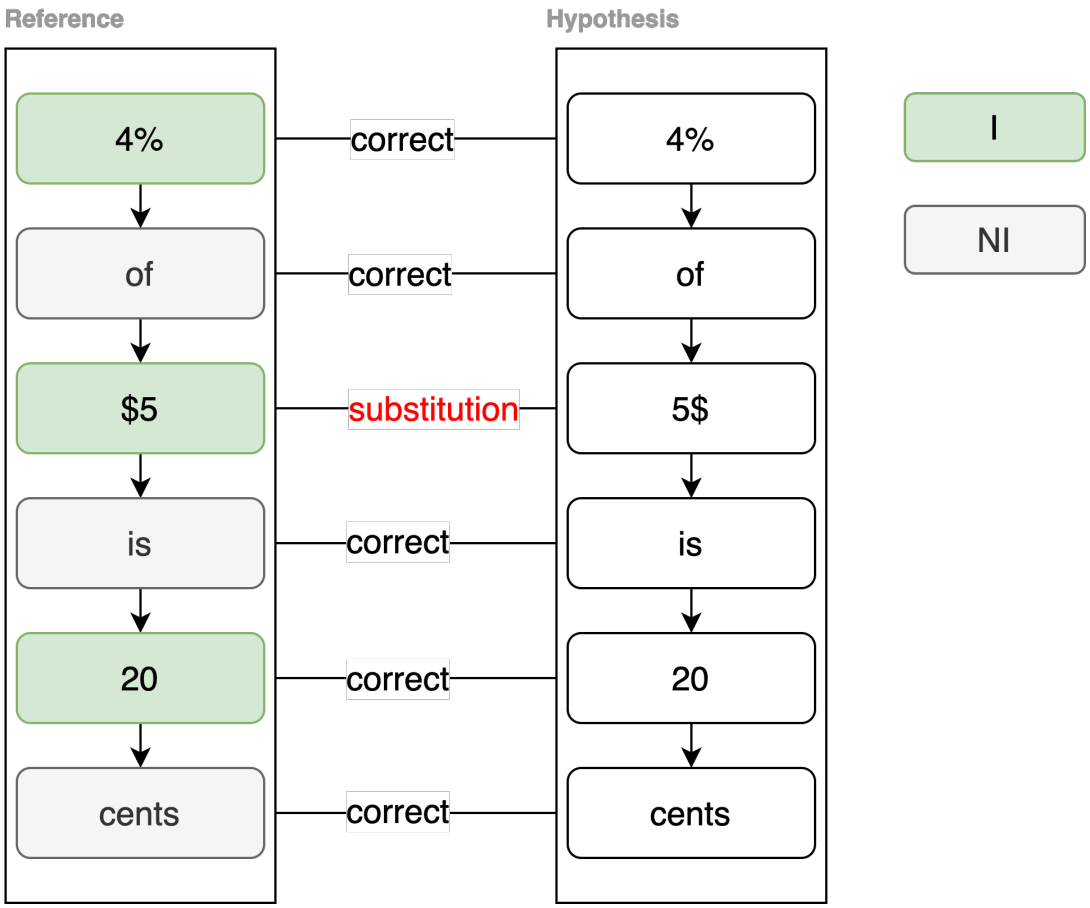

Figure 4.2: Example of word tagging and WER calculation

4.3 Result

This section will present the experimental results of the proposed method on the presented data sets. First, we want to show how the effect of the proposed subword method affects the vocabulary size. Then measure the results of the model on English and Vietnamese data sets.

4.3.1 Subword performance

The data of TN and ITN are pairs of spoken form and written form. As mentioned earlier, the advantage of using the spoken form is that it can limit the vocabulary size. Therefore, we only need to care about the vocabulary size of written form sentences. ITN and TN are tasks that involve a lot of numeric values, resulting in numeric values spanning a wide range. That is

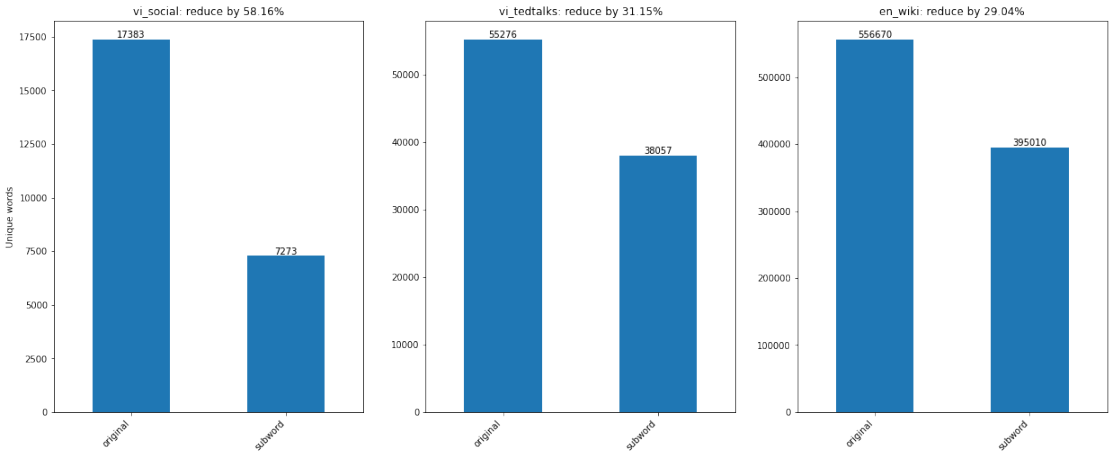

Figure 4.3: Number of unique words in written form before and after apply subword

why numeric vocabularies often appear less or do not appear in the training data, leading to an inefficient model. We propose applying subword encoding and decoding techniques to handle this problem.

To evaluate the subword method's impact on the vocabulary size, we counted the number of unique words in the written form of the data sets. Then we proceed to apply subword encoding and count again. Figure 4.3 displays the vocabulary size before and after applying the subword method. According to chart analysis, the vocabulary size is significantly reduced across all data sets. Specifically, the vocabulary size was reduced by 58% and 31% on the vi_social and vi_tedtalks datasets, and a 29% decrease on the en_wiki dataset.

4.3.2 Performance on Vietnamese data

This section will publish experimental results on Vietnamese data sets. Because the proposed method uses a model to train three different subtasks, normalize entities from spoken form to written form, restore punctuation, and restore capitalization. For the most detailed evaluation, we will first evaluate WER on the task of normalizing entities. And then measures the performance of the remaining two subtasks using the accuracy metric.

Table 4.2 summarizes the experimental results of the proposed method on two Vietnamese data sets. On each dataset, we first report the overall word error rate (WER) on the sentence level, then the word error rate (I-WER) on the words to be normalized, and finally, the word error rate (NI-WER) of words should be copied from the input sentence. The pre-trained encoder-

	vi_social			vi_social-hard			vi_tedtalks			vi_tedtalks-hard		
Method	**WER**	**I-WER**	**NI-WER**	**WER**	**I-WER**	**NI-WER**	**WER**	**I-WER**	**NI-WER**	**WER**	**I-WER**	**NI-WER**
Baseline (FST)	10.1	64.6	5.2	24.8	103.9	12.5	3.3	37.5	3.0	6.8	35.7	5.4
BARTpho-word	5.1	25.6	3.3	9.7	44.0	4.4	3.9	29.0	3.6	5.4	28.1	4.3
mT5-base	4.1	30.9	1.7	10.5	60.0	2.8	2.0	32.4	1.7	5.3	32.0	4.0
+ Subword	3.2	26.2	1.2	9.1	52.0	2.5	1.9	26.8	1.7	4.7	15.8	4.1
mBART-large	3.3	25.0	1.3	7.8	39.6	2.8	1.20	30.4	**0.9**	3.0	36.2	**1.4**
+ Subword	3.0	23.1	1.2	7.1	38.9	2.2	**1.15**	24.1	**0.9**	**2.0**	14.8	**1.4**
mT5-large	2.6	19.4	1.1	6.3	33.9	**2.0**	1.7	26.6	1.4	4.4	22.9	3.5
+ Subword	**2.5**	**18.5**	**1.0**	**5.6**	**28.0**	2.1	1.6	**23.5**	1.4	4.0	**13.2**	3.5

Table 4.2: ITN performance on Vietnamese test sets ↓

decoders we used to compare against the baseline model included BARTpho, mT5, and mBART. The model size of BARTpho, mT5-base, and mBART-large are the same (12 encoders, 12 decoders, 1024 hidden), while mT5-large uses 24 encoder-decoder layers.

Comparing the methods on the two data sets, we can see that WER on vi_tedtalks is always better than on vi_social. That's because the social domain is more complex than the scientific domain. Methods using pre-trained encoder-decoder models have outperformed the baseline. In particular, mT5-large has better results than other methods on most indicators. It is because the size of the mT5-large model is twice the size of the other three neural network models. When comparing three models of BARTpho, mT5-base, and mBART-large of the same size, mBART-large outperformed mT5 and BARTpho on all indicators. Although BARTpho is a pre-trained monolingual model for Vietnamese, the results are not as good compared to the multilingual version. Because ITN focuses on normalizing numeric entities, while BARTpho uses a word-level tokenizer. For the baseline model, we admit that the model only normalizes an entity if it has high confidence. Therefore, the model will skip normalizing the entities in the case of ambiguity, leading to a relatively high I-WER compared to methods using neural networks.

During the experiment, we recognized that the proposed subword method is only effective when pre-trained models use subword units. More specifically, mT5 and mBART use the sentencepiece toolkit (i.e., byte-pair-encoding), while BARTpho uses the word-level tokenizer. Therefore, we only experiment and publish the results using the proposed subword method on mT5 and mBART. The results show that WER is significantly improved on I-WER when using the proposed subword method, leading to improved WER. The effectiveness of the subword is more evident on the hard-test sets, where the data contains more tokens than is broken down by the subword method. For example, the subword method improves efficiency by more than 50% on the vi_tedtalks test in terms of I-WER.

For the punctuation and capitalization recovery tasks, we use the accuracy metric to evaluate the quality of the model. In this study, the task of recover-

	vi_social		vi_tedtalks				
Method	**lower**	**Title**	**Period**	**Comma**	**Question**	**lower**	**Title**
mBART-large	**99.03**	93.58	95.89	98.85	91.06	**99.32**	87.71
mT5-base	99.00	92.57	94.81	**98.95**	89.55	98.75	87.39
mT5-large	98.99	**94.00**	**96.22**	98.81	**93.65**	98.64	**90.20**

Table 4.3: Result on capitalization and punctuation tasks ↑

ing uppercase words only focuses on restoring proper nouns with capital letters. With the punctuation restoration task, we focus our experiments on the three most common punctuation marks, including punctuation marks, commas, and question marks. Table 4.3 summarizes our experimental results on these two tasks on Vietnamese test sets. The capitalization restoration task achieved relatively good results, with 94% and 90% accuracy on vi_social and vi_tedtalks, respectively. Again, the larger model mT5-large gives better results than the other two methods. The punctuation restoration task also achieved good results on the vi_tedtalks test set. The model's accuracy is 96%, 98%, and 93% on the period, comma, and question mark, respectively. This positive result is the premise for optimizing the proposed method to help the model gain better results on punctuation and capitalization restoration tasks along with ITN.

4.3.3 Performance on English data

We use the proposed approach to experiment with the English language to prove that the method is language-independent. In this experiment, we use part of the Google Text Normalization dataset used for a competition on Kaggle to train and evaluate the model. We also use the results of the two latest works using two different methods for ITN to compare with the proposed method. The first is a study using WFST-based proposed by Zhang et al. [26]. The second method proposed by Sunkara et al. [18] uses transformer architecture combined with FST. Both of these studies used the Google Text Normalization dataset for the experimental process and used the word error rate (WER) metric to measure the model's effectiveness.

Table 4.4 summarizes our experimental results on English data compared with related studies. More specifically, the work of Sunkara et al. [18] shows state-of-the-art results with a WER of 0.9%. Meanwhile, our proposed method combined with subword gives competitive results with state-of-the-art. Again, applying subwords combine with byte-pair-encoding contributes to a significant improvement in WER. The I-WER metric is greatly improved when applying the proposed subword method, decreasing from 10.9% to 4%

	en_wiki			en_wiki-hard		
Method	WER	I-WER	NI-WER	WER	I-WER	NI-WER
Zhang et al. [26]	12.70	N/A	N/A	N/A	N/A	N/A
Sunkara et al. [18]	**0.90**	**4.80**	**0.30**	N/A	N/A	N/A
mBART	2.60	14.10	1.80	4.30	10.90	2.90
mBART+Subword	1.70	7.80	1.30	**2.40**	**4.00**	**2.00**

Table 4.4: Result on Engish dataset ↓

on the hard-test set. In this experiment, we only used a part (31 million words) of the Google Text Normalization dataset [17] for the training process, while the full dataset consisted of 1.1 billion words. Experimental results show that our proposed method can be easily applied to different languages and does not require too much data for the training process.

Chapter 5

Conclusion

5.1 Summary

In this work, we propose to use a pre-trained language model based on transformer architecture combined with a data-driven approach to solving the ITN task. We have experimented with several pre-trained language models on English and Vietnamese data sets. The experiment results show that the proposed method has competitive performance with related works on English, although using much less training data. For Vietnamese, our method is highly effective on all three subtasks of ITN. Besides, the proposal to use subwords significantly improves the model's efficiency, especially for entities with digits.

Our contributions to this work are summarized as follows:

- Propose a method of using a pre-trained model based on transformer architecture for ITN using very little labeled data but still achieving high efficiency.
- Experimenting with the proposed method on all three subtasks of ITN gives good results.
- Proposal to use subwords together with sentencepiece shows a better result on entities containing digits.

We hope that this work will promote research in improving the performance of ITN in other low-resource languages like Vietnamese. The result shows that the proposed method is effective on ITN tasks in both English and Vietnamese, which mean that the method has potential in other natural language processing tasks such as text normalization, spelling correction, machine translation, or question-answering.

5.2 Future works

This work performs inverse text normalization for ASR output to improve readability and efficiency on downstream tasks. Besides the results already achieved, this task still has room for improvement. Therefore, we have a plan to continue to improve the proposed method:

- Using pre-trained models is an advantage over training the model from scratch. However, this method has limitations as the data used for pre-training BART, or T5 is written data while ITN serves spoken text data. Data domain differences are a limitation in improving model efficiency. Therefore, we intend to finetune the pre-trained mask language models with spoken form data before finetuning them for the ITN task. We can collect spoken form data from movie subtitles and video subtitles from online video-sharing platforms. However, this job also requires much time to perform.

- Data is always concerned with supervised learning tasks, including TN and ITN. The lack of labeled data is a significant barrier to model training. The ITN task primarily focuses on numeric and number-related entities (e.g., date, time). Smaller data means that numeric and number-related entities appear less frequently. Therefore, we plan to use data augmentation techniques for ITN, primarily numeric and number-related entities. The work on data augmentation can significantly improve ITN performance.

Bibliography

[1] Issac Alphonso, Nick Kibre, and Tasos Anastasakos. "Ranking Approach to Compact Text Representation for Personal Digital Assistants". In: *2018 IEEE Spoken Language Technology Workshop (SLT)* (2018), pp. 664–669.

[2] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *NAACL*. 2019.

[3] Peter Ebden and Richard Sproat. "The Kestrel TTS text normalization system". In: *Natural Language Engineering* 21 (2014), pp. 333–353.

[4] Mana Ihori, Akihiko Takashima, and Ryo Masumura. "Large-Context Pointer-Generator Networks for Spoken-to-Written Style Conversion". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), pp. 8189–8193.

[5] Taku Kudo. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates". In: *ACL*. 2018.

[6] Mike Lewis et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *ACL*. 2020.

[7] Junwei Liao et al. "Improving Readability for Automatic Speech Recognition Transcription". In: *Transactions on Asian and Low-Resource Language Information Processing* (2022).

[8] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *ArXiv* abs/1907.11692 (2019).

[9] Courtney Mansfield et al. "Neural Text Normalization with Subword Units". In: *NAACL*. 2019.

[10] Dat Quoc Nguyen and Anh Gia-Tuan Nguyen. "PhoBERT: Pre-trained language models for Vietnamese". In: *ArXiv* abs/2003.00744 (2020).

[11] Long Phan et al. “ViT5: Pretrained Text-to-Text Transformer for Vietnamese Language Generation”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*. Association for Computational Linguistics, 2022, pp. 136–142.

[12] Subhojeet Pramanik and Aman Hussain. “Text Normalization using Memory Augmented Neural Networks”. In: *Speech Commun.* 109 (2019), pp. 15–23.

[13] Ernest Pusateri et al. “A Mostly Data-Driven Approach to Inverse Text Normalization”. In: *INTERSPEECH*. 2017.

[14] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *ArXiv* abs/1910.10683 (2020).

[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. “Neural Machine Translation of Rare Words with Subword Units”. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162.

[16] Maria Shugrina. “Formatting Time-Aligned ASR Transcripts for Readability”. In: *NAACL*. 2010.

[17] Richard Sproat and Navdeep Jaitly. “An RNN Model of Text Normalization”. In: *INTERSPEECH*. 2017.

[18] Monica Sunkara et al. “Neural Inverse Text Normalization”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2021), pp. 7573–7577.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *NIPS*. 2014.

[20] Y. Tang et al. “Multilingual Translation with Extensible Multilingual Pretraining and Finetuning”. In: *ArXiv* abs/2008.00401 (2020).

[21] Nguyen Luong Tran, Duong Minh Le, and Dat Quoc Nguyen. “BARTpho: Pre-trained Sequence-to-Sequence Models for Vietnamese”. In: *INTERSPEECH*. 2022.

[22] Oanh T. K. Tran and Viet The Bui. “Neural Text Normalization in Speech-to-Text Systems with Rich Features”. In: *Applied Artificial Intelligence* 35 (2021), pp. 193–205.

[23] Ashish Vaswani et al. “Attention is All you Need”. In: *NIPS*. 2017.

[24] Linting Xue et al. “mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer”. In: *NAACL*. 2021.

[25] Hao Zhang et al. “Neural Models of Text Normalization for Speech Applications”. In: *Computational Linguistics* (2019), pp. 293–338.

[26] Yang Zhang et al. “NeMo Inverse Text Normalization: From Development To Production”. In: *Interspeech*. 2021.

Publications

- Hieu Nguyen Van, Dat Nguyen, Phuong Minh Nguyen and Minh Le Nguyen, "Miko Team: Deep Learning Approach for Legal Question Answering in ALQAC 2022", International Conference on Knowledge and Systems Engineering, 2022 (accepted)
- Dat Nguyen Tien, Hieu Nguyen Van, Tung Le Thanh and Minh Nguyen Le, "An Unsupervised Learning Method to improve Legal Document Retrieval task at ALQAC 2022", International Conference on Knowledge and Systems Engineering, 2022 (accepted)