A Comparison of several Deep Learning based Models for Diacritic Restoration Problem in Vietnamese Text

Quang-Linh Tran[1,2,*], Van-Binh Duong[1,2,*], Gia-Huy Lam[1,2,*], Trong-Hop Do[1,2,†]

[1] University of Information Technology
[2] Vietnam National University Ho Chi Minh City
Email:[*] {18520997, 18520505, 18520832}@gm.uit.edu.vn,
[†] *hopdt@uit.edu.vn*

Abstract. Diacritic restoration is a challenging problem in natural language processing (NLP). With diacritic restoration, one can text faster and easier. Diacritic restoration is also helpful in making use of diacritic-missing texts, which are normally discarded in many NLP applications. This paper deals with the diacritic restoration problem for Vietnamese text. Three state-of-the-art deep learning models including Gated Recurrent Unit, Bidirectional Long-short Term Memory and Bidirectional Gated Recurrent Unit have been examined for the problem and the last one turned out to be the best among them. Besides deep learning models, it was found in this paper that word tokenization, which is the final pre-processing step applied on the data before feeding it to deep learning models also have influences on the final accuracy. Between two examined word tokenization methods: morpheme-based tokenization and phrase-based tokenization, the former yield better results regardless of the applied deep learning models. The experimental results show that the combination of morpheme-based tokenization and Bidirectional-GRU achieve the best performance of diacritic restoration with the Bleu-score of 88.06%.

Keywords: Diacritic Restoration · Neuron Network · Machine Translation · Natural Language Processing · Word Tokenization.

1 Introduction

Several languages have diacritical marks such as Romanian, Czech, and Vietnamese. Among them, Vietnamese has arguably the most complicated diacritic system [1]. This is due two reasons. First, Vietnamese is a tonal language with six distinctive tones including “level” (ngang), “acute-angry” (sắc), “grave-lowering” (huyền), “smooth-rising” (hỏi), “chesty-raised” (ngã), and “chesty-heavy” (nặng). Besides "level", which has no accent, other five tones have their own diacritics. Second, Vietnamese is written using Latin based script with 26 standard Latin letters and seven modified letters including đ, ă, â, ê, ô, ơ, and ư. Consequently, a total of 11 diacritics are used in Vietnamese for marking tones and creating new letters. The combinations of these 11 tone marking and letter modifying diacritics make the diacritic system in Vietnamese even more complicated. For example, many diacritic combinations can be used in the word 'to-large' to create other words with different meaning like 'tô-bowl','tố- accuse','tộ-claypot','tơ-yarn','tớ-I', 'tỏ-clear', and so on. Because of this, learning Vietnamese is a real challenge for any foreigner. Even for some Vietnamese, typically children and elderly ones, typing these diacritics is tiresome or even difficult, especially when smartphones are used to prepare the text. To solve this problem, this paper introduces new models for diacritic restoration in Vietnamese to make typing Vietnamese less a headache problem for both foreigner and Vietnamese.

Besides helping people typing Vietnamese faster and easier, diacritic restoration can also be helpful for other natural language processing (NLP) problems such as sentiment analysis or machine translation on Vietnamese data. To be more specific, the data for these NLP problems might be collected from multiple sources including websites, online newspaper, forums, e-commerce sites, and especially social network. The collected data might contain lots of diacritic-missing texts, which are normally discarded in pre-processing step in many NLP problems. In other words, a large portion of the collected data is useless because of the missing of diacritics. With diacritic restoration, these data can be preprocessed to restore the missing diacritics in the text before applying further processing.

This paper focuses on building the best diacritic restoration model for Vietnamese text. To this end, three state-of-the-art deep learning models including Gated Recurrent Unit, Bidirectional Long-short Term Memory and Bidirectional Gated Recurrent Unit, which are normally applied for machine translation problem are examined. The experiment results show that Bidirectional Gated Recurrent Unit (GRU) achieved the best performance on the tested data set. Since Vietnamese is a monosyllabic language where compound words are constructed from two or more morphemes, there are different word tokenization methods that can be applied to separate words in the sentences before feeding them to deep learning models. This paper examined two common word tokenization methods: morpheme-based tokenization and phrase-based tokenization. The results show that the former achieve better results with all applied deep learning models. To sum up, morpheme-based tokenization and Bidirectional GRU create the best combination with the highest diacritic restoration accuracy of 88.06% in Bleu-score.

2 Related Works

There are several related works on this topic and they used different approaches and received some achievement.

In 2012, Luan Nghia Pham et al. introduced a paper Vietnamese Text Accent Restoration With Statistical Machine Translation[2]. This model used the MOSES framework to translate from sentences missing diacritics to completed sentences. The dataset which they used had 206000 Vietnamese sentences and their model got 92% accuracy. However, the major drawback of the MOSES toolkit is it needs computational resources and a long training time. This is the aspect that this paper will improve in our model.

Thai Hoang Pham et al. proposed a new approach for Vietnamese diacritic restoration[3]. This paper used the OpenNMT toolkit which is built from neuron network algorithms. This toolkit used the Encoder-Decoder network to restore diacritics for Vietnamese sentences. The toolkit got an accuracy of 96.15%. Training progress spent 8 hours with 140474 Vietnamese sentences.

In general, although these papers performed in diacritic restoration, these approaches are not modern, resulting in needing high computational resources and a long training and testing time. In addition, the datasets are not large enough, so they may not cover all cases of Vietnamese. There are reasons why we want to build a better translation model based on a big dataset and state-of-the-art algorithms to solve this problem.

3 Dataset

3.1 General Information

A Large-scale Vietnamese News Text Classification Corpus[3] is used for this paper to train and evaluate models. This dataset was published in the paper A Comparative Study on Vietnamese Text Classification Methods[4]. The dataset has over 26452 Vietnamese articles with 27 topics ranging from sports, music to the world. The initial purpose of this dataset is to classify Vietnamese news text but this dataset is large, covering all of the aspects of life so choosing this dataset for automatic diacritic restoration problem is suitable. We conduct some preprocessing to make this dataset suitable for our problems.

3.2 Constructing dataset

Large-scale Vietnamese News Text Classification Corpus dataset is separated into sentences by the dot and comma symbols (",", ".") to obtain 542107 Vietnamese sentences. Some preprocessing steps are applied to this dataset. Firstly, eliminating some special symbols (?, !, ", {, }, [,], etc.) as well as digits in this dataset because these symbols do not affect the meaning of sentences. Secondly, English words in this dataset are removed from sentences. Finally, converting all words with capital letters to lowercase because these words have the same meaning but different representation. The unidecode[4] framework is used to remove diacritics automatically on 542107 sentences and after that, a new dataset is created whose data are non-diacritical sentences and labels are the corresponding accented sentences. The new dataset is divided into three subsets which are train set, validation set, and test, and the size of the three sets are 500000, 21107, and 21100 sentences respectively. Table 1 shows some examples of data points in the new dataset.

[3] https://github.com/duyvuleo/VNTC
[4] https://pypi.org/project/Unidecode/

Table 1: Some examples of data points in the new dataset

No	Label	Data
1	tại câu lạc bộ nguyễn du	tai cau lac bo nguyen du
2	tham gia chương trình lần này có ca sĩ hồ ngọc hà	tham gia chuong trinh lan nay co ca si ho ngoc ha
3	khéo léo kết hợp thành viên chủ chốt của nhóm tay cự phách	kheo leo ket hop thanh vien chu chot cua nhom tay cu phach
4	âm thanh hiện đại nhất trong các đĩa nhạc từng phát hành	am thanh hien dai nhat trong cac dia nhac tung phat hanh

4 Methodlogies

In this paper, the problem of diacritic restoration is treated using the same approach as the machine translation problem. There is one minor difference that in the diacritic adding problem, the input always has the same length as the output. Three state-of-the-art deep learning models including Long Short-Term Memory (LSTM), Gated Recurrent Units (GRU), and Birectional RNN, which are usually used for machine translation, are examined to find which is the best model for the problem of diacritics restoration in Vietnamese text.

Although deep learning models might be a deciding factor, word tokenization, which is the final data preprocessing step applied to the data before fed into the deep learning model can also have certain impacts on the accuracy of the diacritic restoration. In this paper, two different word tokenization methods: morpheme-based tokenization and phrase-based tokenization are also examined with the three mentioned deep learning models to find which combination yields the best performance of diacritic restoration.

4.1 Word tokenization methods

Word tokenization is the final data preprocessing step used for separating words in each sentence. Since Vietnamese contains compound words that are constructed from two or more monosyllabic morpheme. So, there are two methods for word tokenization: morpheme-based tokenization and phrase-based tokenization.

4.1.1 Morpheme-based tokenization or Word tokenization by white space: The advantage of this method is fast, simple, and popular when it comes to sequence inputs. For example, when tokenizing a sequence: 'trường đại học công nghệ thông tin'->['trường', 'đại', 'học', 'công', 'nghệ', 'thông', 'tin'] ('information technology university' -> ['information', 'technology', 'university']).
The disadvantage of this method is the loss of meaning of the compound words, for instance, 'đại học' or 'công nghệ thông tin' ('university' or 'information technology') because a single word standing alone has a different meaning from a compound word. With this method, the training set has a vocabulary set of 1625 words.

4.1.2 Phrase-based tokenization or word tokenization by counpoud words: The vncorenlp[5] library is used to tokenize a sequence into pairs of single and compound words as: ['trường', 'đại_học', 'công_nghệ_thông_tin'] to handle the limitations of the first word tokenization method. And the preservation of the meaning of compound words in a sequence also helps the model understand the compound words in Vietnamese. With this method, the number of words for model training also increases significantly (the size of the vocabulary set on the training set is 27692 words).

4.2 Machine translation models

4.2.1 Long Short-Term Memory Units (LSTM): First introduced by Hochreiter and Schmidhuber[5]. This has made many improvements to traditional RNN models. When using LSTM, the gate architecture can be adjusted differently depending on the purpose to get different test results with the goal of building the best model.

The application of LSTM to machine translation problems has achieved positive results through evidence that the authors from the article LSTM Neural Networks For Language Modeling[6] have run experiments and achieved 8% improved results compared to the usual machine translation models.

The architecture of the basic LSTM[6] consists of the *update gate, forget gate* and *output gate* with the respective functions: adding information to memory selectively, removing unnecessary information, and determining what information is selected as output. The fine-tuning in LSTM is the tuning of these ports (adding ports or adjusting computational functions in gates). Information from each state is stored at c^t *memory cells*.

The candidate may be stored in *cell* with the given x^t.

$$\tilde{c}^t = tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c), \tag{1}$$

The formula of *update gate*:

$$\Gamma_u = \sigma(W_u[c^{t-1}, x^{<t>}] + b_u), \tag{2}$$

Calculation of *forget gate* is:

$$\Gamma_f = \sigma(W_f[a^{t-1}, x^t] + b_f), \tag{3}$$

Output gate is figured as:

$$\Gamma_o = \sigma(W_o[a^{t-1}, x^t] + b_o), \tag{4}$$

Memory cell is updated by the formula:

$$c^{<t>} = \Gamma_u * \tilde{c}^t + \Gamma_f * c^{t-1}, \tag{5}$$

Activation function:

$$a^{<t>} = \Gamma_o * tanh(c^t). \tag{6}$$

[5] https://github.com/vncorenlp/VnCoreNLP

[6] https://www.coursera.org/lecture/nlp-sequence-models/long-short-term-memory-lstm-KXoay

4.2.2 Gated Recurrent Units (GRU): Architecture of GRU is similar to LSTM. It was introduced to overcome the drawbacks of the traditional RNN. GRU was also tested and evaluated in the article Empirical Evaluation Of Gated Recurrent Neural Networks On Sequence Modeling[7]. The architecture of GRU is also constructed based on gates like LSTM but there are only two gates: *update gate* and *relevance gate* so that the calculation in GRU is also simpler. The function of the *relevance gate* is to find out whether the information from the beforehand state is important or not. *Memory cell* in GRU and the activation function are similar to each other:

$$c^{<t>} = a^{<t>}, \tag{7}$$

The formula of *relevance gate*:

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_c), \tag{8}$$

The $\tilde{c}$ and $c^{<t>}$ are also changed:

$$\tilde{c}^t = tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c), \tag{9}$$

$$c^{<t>} = \Gamma_u * \tilde{c}^t + (1 - \Gamma_u) * c^{t-1}. \tag{10}$$

4.2.3 Bidirectional RNN[8]: The idea of Bidirectional RNN (BRNN) is similar to the exercise on filling the blank with a suitable word, people have to consider both words from the front of the blank and the behind part in order to decide the right word to fill in and to be able to correctly choose the type of word (noun, adjective, verb or adverb). The basic BRNN network consists of 2 independent RNNs and the results of these two RNNs are aggregated into a single output for each state of the network.

In BRNN, the output of the current state depends on the information from the state before and the forward state. Therefore, the problem of adding diacritics to any word in a specific sequence requires information of all the surrounding words to pick the appropriate marks to create a meaningful sequence. So, if using only the RNN model to find out suitable signs for the word, the model predicts the sign without considering whether the following word matches the sign-added word or not (in the worst situation, the word is incorrectly added sign compared with the expected sign, as the result, the following word may also be accented to have a wrong meaning that matches the accented word before), this causes a serious mistake leading to the meaning of the sequence can be far away from the expectation. In conclusion, the information from the backward and forward of the current text is very valuable. Therefore, the BRNN model is expected to be a potential model for the problem.

5 Model preparation:

The models used in the paper are installed with the support of the Keras Framework[7].

In order to find the most suitable model based on running experiments on the dataset, there are four different models for the training phase (with embedding_layer = 256):

- **GRU-RNN:** The GRU-RNN layer has 256 units, using the Softmax activation function for the output layer.
- **Bidirectional GRU-RNN and Bidirectional LSTM-RNN:** These two models include the Bidirectional GRU and Bidirectional LSTM layers with the same as 128 units. And the output layer also uses Softmax as an activation function.
- The **TimeDistributed** layer (functioning as a fully-connected layer) is also added so as to aggregate the results of each LSTM unit for each time-step. There are two TimeDistributed layers that are sequentially implemented to optimize the predicted results of the model.
- In order to avoid the overfitting, a dropout = 0.5 layer is also used.

With the different architectures of the RNN network, this paper aims to explore the best model for the dataset which takes advantage of poor hardware resources and has the shortest training time.

6 Results analysis and discussion

6.1 Results

After conducting experiments on all the models, the results of two word tokenization methods: morpheme-based tokenization (Section 4.1.1) and phrase-based tokenization (Section 4.1.2) are given in Table 2 and 3.

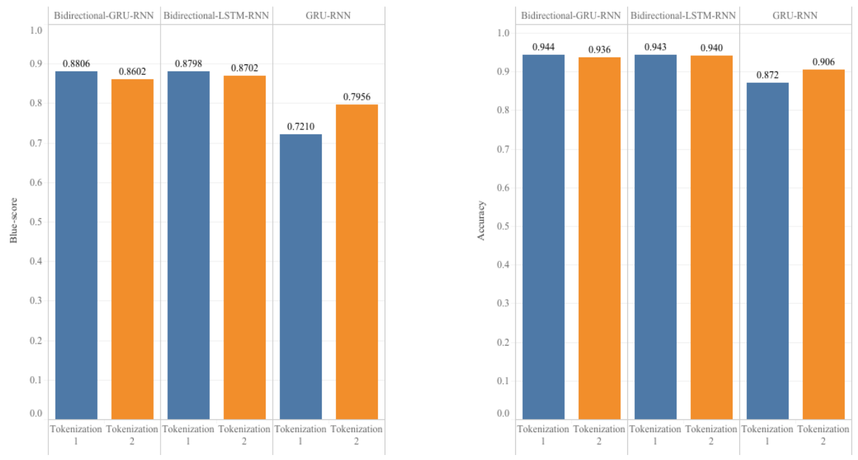
Table 2: The results of three models in the morpheme-based tokenization.

Model	Train accuracy	Test Accuracy	Bleu-score
GRU-RNN	0.9892	0.8790	0.7347
Bidirectional-LSTM-RNN	0.9956	0.9432	0.8798
Bidirectional-GRU-RNN	0.9954	0.9436	0.8806

[7] https://www.tensorflow.org/guide/keras/rnn

Table 3: The results of three models in the phrase-based tokenization.

Model	Train accuracy	Test Accuracy	Bleu-score
GRU-RNN	0.9925	0.9032	0.7909
Bidirectional-LSTM-RNN	0.9952	0.9399	0.8702
Bidirectional-GRU-RNN	0.9930	0.9357	0.8602



(a) Blue_score. (b) Test Accuracy.

Fig. 1: Measurement scores on models with two tokenization methods.

6.2 Discussion

It is clear that word tokenization methods have a large influence on the models' accuracy and the Bleu-score.

On model aspect: In both word tokenization methods, the performance of the Bidirectional-LSTM model is better than the other models' performance. The accuracy on the test set is 94%, this is a wonderful result. GRU-RNN model is the modest one that gives the lowest result among the three models with 87% and 90% on morpheme-based tokenization and phrase-based tokenization methods, respectively.

On Bleu-score aspect: The experiment results are not high but can be acceptable in a range from 72% to 88%. The Bidirectional-GRU model gives the highest result (88%) on the morpheme-based tokenization. In addition, the result of the Bidirectional-LSTM model is very close to that figure (87%).

When processing to analyze on different sequences, there is an insight that the sign of a word at any position depends a lot on its backward and forward words. So, the contextual information in the sequence plays an important role when the model chooses signs for a word. Therefore, the Bidirectional RNN models perform better than the RNN models.

On word tokenization methods aspect: the phrase-based tokenization gives a noticeable improvement on GRU-RNN model. In specific, the test accuracy is improved from 87% to 90% and the Bleu-score is also enhanced from 73% to 79%. Vietnamese has so many compound words, so the morpheme-based tokenization may not be good, but through the results in Table 3, it can be seen that the dataset is large enough to partially handle the lack of compound words.

With the phrase-based tokenization: The best model is Bidirectional-LSTM with accuracy on the test set and Bleu-score are 93% and 87%, respectively. It clearly shows that the method gives an effective solution for the problem. The accuracy on the test set of the models also shows fewer discrepancies compared to the second method.

6.3 Errors Analysis

In evaluating models, analysis of some errors is carried out in the validation set and find out some common errors in our three models. There are two major groups of errors, which are adding wrong diacritics due to lack of information from former letters and due to bias:

- Adding wrong diacritics due to lack of information from the forwarding words. For example, if the word 'thay' ('thầy') stands at the beginning of sentences, it will be adding diacritics to ('thấy') because the frequency of the word 'thấy' is higher than that of the word 'thầy'. However, when 'nguoi' or 'giao' stands next to 'thay', the word 'thay' will be added the right diacritic because the word 'thay' depends on two other words. This shows that models learn from phrases and predict diacritics from other words in sentences.
- Adding wrong diacritics due to bias. Because of the high frequency of some words in the dataset, the models always predict some words with the fixed diacritics. For example, the word 'ban than' is always adding diacritics to 'bản thân' even though the right word is 'bạn thân'.

The bias error only appears in the prediction of the GRU-RNN model, because of the specific features of this model. It can be seen that the Bidirectional RNN model performs as well as expected.

7 Conclusion

This paper introduces and compares three machine translation models based on neural network algorithms to solve the problem of adding diacritics for Vietnamese sentences without diacritical marks. After experimenting with these approaches, the Bidirectional-GRU model gives the best result with 86.06% Bleu-score and the Gated Recurrent Units give the worst result with 73.47% Bleu-score. These results are good for this problem despite short time training models and limited computational resources. Besides, this paper also compares the word tokenization methods in Vietnamese and finds out that there is a small difference in model performance between the two methods of tokenization. In morpheme-based tokenization, the highest Bleu-score is 88.06% and phrase-based tokenization has the highest Bleu-score at 87.02%. The Bidirectional GRU model gives significantly higher results than the normal GRU model with Bleu-score 15% higher in the morpheme-based tokenization and 8% higher in the phrase-based tokenization. This means that a single Vietnamese word needs information coming from both before and after the word to accurately predict the diacritic of this word.

In the future, we will aim to develop models to achieve the highest accuracy with the lowest cost to apply them in real-world life. Adding more data with context diversification is one way to improve the performance of models. Besides, applying new deep learning algorithms like Transformer can help machine translation models predict more accurately.

References

1. Jakub Náplava, Milan Straka, Pavel Straňák, and Jan Hajič. Diacritics restoration using neural networks. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).
2. Luan-Nghia Pham, Viet Hong Tran, and Vinh Van Nguyen. Vietnamese text accent restoration with statistical machine translation. In *Proceedings of the 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC 27)*, pages 423–429, 2013.
3. T. Pham, X. Pham, and P. Le-Hong. On the use of machine translation-based approaches for vietnamese diacritic restoration. In *2017 International Conference on Asian Language Processing (IALP)*, pages 272–275, 2017.
4. Vu Cong Duy Hoang, Dien Dinh, Nguyen Le Nguyen, and Hung Quoc Ngo. A comparative study on vietnamese text classification methods. In *2007 IEEE International Conference on Research, Innovation and Vision for the Future*, pages 267–273. IEEE, 2007.
5. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
6. Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.
7. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
8. Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.