TDP – A Hybrid Diacritic Restoration with Transformer Decoder

DANG Trung Duc Anh

ducanhhbtt@gmail.com
Hanoi University of Science and Technology
Hanoi, Vietnam

NGUYEN Thi Thu Trang*

trangntt@soict.hust.edu.vn
Hanoi University of Science and Technology
Hanoi, Vietnam

Abstract

Diacritic restoration plays an important role in Natural Language Processing (NLP) for many diacritical languages such as Vietnamese, Czech, Hungarian, etc. With the development of deep neural network, this task could reach a good accuracy, i.e. F1-score is up to 97.7% for the state-of-the-art models. However, the output of these models can include meaningless syllables, the processing time is rather long especially with the sequence-to-sequence method and beam search. This task is a very first step in any text (pre-)processing, which can be a part of another application. Therefore, the processing time is extremely important. To balance both accuracy and time consuming, this paper proposes a novel hybrid method which includes a transformer decoder and a diacritic penalty layer. The transformer decoder is good enough for this problem since an input character only corresponds to exact one output character. The purpose of the penalty layer is to guide the model to produce only possible diacritic letters of the language. The experimental results on Vietnamese corpus show that the proposed model helps the predicting time reduce from about eight to ten times compared to the previous methods. Whereas, the accuracy of the proposed method is better than (i.e. 1%) or equal to the state-of-the-art sequence-to-sequence without or with beam search.

*Corresponding author

1 Introduction

Diacritics are a vitally important component in many diacritical languages such as Vietnamese, Czech, Hungarian, etc. However, a large number of texts without diacritics serve many purposes.

In diacritic languages, typing a diacritic word is far more troubled than typing a non-diacritic one. For instance, in Vietnamese, “đường” can be typed as "dduongwf" (Telex system). The middle or old-aged who do not know the rules for typing or those who want to save time typing diacritics would prefer to type sentences without diacritics, although they can be misleading and incomprehensible to other people. Moreover, many public-originated foreign systems only support non-diacritic characters, leading to a huge amount of this data type to be processed.

There are numerous ways for restoring a sentence to its former full diacritical marks with different meanings. For example, in Vietnamese, the most suitable restoration version for the non-diacritic one “Toi muon mo the tin dung” is "Tôi muốn mở thẻ tín dụng” (I want to open a credit card). However, each syllable in the sentence has multiple ways to become the form with full diacritic marks. "Muon" can be restored as "muốn" (want), "muộn" (late), "muôn" (many) while "the" can be restored as "thẻ" (card), "thè" (put out), "the" (a kind of traditional cloth), "thé" (high pitch), "thế" (and), "thề" (swear), "thê" (wife). This leads to a number of restoration combinations corresponding to the input sentence that we need to disambiguate. Therefore, recovering diacritics is among the most necessary but challeng-

ing problems in natural language processing. It is particularly hard for Vietnamese, whose ratio of diacritical words is highest, i.e. approximately 90%, 80% of which contain ambiguity (Do et al., 2013).

A number of researches on restoring diacritic marks used both machine learning and deep learning approaches. For Vietnamese, three main ones have been proposed, i.e. (i) rule and dictionary-based, (ii) machine learning-based (Nguyen and Ock, 2010) and (iii) deep learning-based approach (Hung, 2018; Náplava et al., 2018; Nga et al., 2019). A typical example of the rule and dictionary-based approach is VietPad[1]. In that work, a dictionary with all Vietnamese syllables was built to restore diacritic marks. However, this tool could not solve a number of ambiguous cases, leading a limited accuracy of about 60% to 85% depending on the domain. The machine learning-based approach (Nguyen and Ock, 2010) achieved an accuracy of 94.7% on their dataset, using a combination of AdaBoost and C4.5 algorithms. Recently, deep learning-based methods with machine translation models have emerged as the state-of-the-art solution to the problem of diacritic restoration. The idea of this method is to treat non-diacritic and diacritic texts as the source and target languages in the machine translation formulation. The best work in this approach used a novel combination of a character-level recurrent neural network-based model and a language model applied to diacritics restoration and reached the highest accuracy of 97.73% on Vietnamese (Náplava et al., 2018).

However, there are several shortcomings of the above state-of-the-art methods, i.e. producing nonexistent outputs and time-consuming for the task. Since the output is generated based on the possibility that the model predicts character by character, the sequence of output text may be nonexistent or meaningless in the language. Moreover, the diacritic restoration is a very-first step of text (pre-)processing for any NLP application. For instance, in question answering or chatbot systems, users sometimes input with non-diacritical marks which should be recovered before many next steps. Diacritic restoration is only a small step in any NLP application. Therefore, the restoration time of this task is hence extremely important in the industry.

In this paper, we propose TDP – a novel hybrid diacritic restoration model which retains the Transformer Decoder at the character-level with Penalty layer. The penalty layer is a restriction mechanism of possible diacritical letters for the output sequence. We have experimented the model for Vietnamese datasets with a promising performance in both accuracy and predicting time. The rest of the paper is organized as follows. Section 2 describes the language orthography and theory of the transformer model. Session 3 presents our proposed hybrid model for restoring diacritical marks. Section 4 discusses on the related works to the model and techniques in our model. In section 5, the experiments on Vietnamese data-sets are described and discussed. Finally, the paper draws some conclusions and perspectives of the work.

2 Background

Since we have experimented with Vietnamese, we provide in this section some backgrounds on Vietnamese orthography with diacritical features. We also present the full transformer model, parts of which are used to construct our model.

2.1 Orthography

In any diacritic language, a limited number of diacritic letters can be restored for a specific non-diacritic one.

Table 1: Possible Vietnamese diacritical letters

Non-diacritical Letter	Possible Diacritical Letter
a	a, á, à, ả, ã, ạ, ă, ắ, ằ, ẳ, ẵ, ặ, â, ấ, ầ, ẩ, ẫ, ậ
e	e, é, è, ẻ, ẽ, ẹ, ê, ế, ề, ể, ễ, ệ
i	i, í, ì, ỉ, ĩ, ị
y	y, ý, ỳ, ỷ, ỹ, ỵ
o	o, ó, ò, ỏ, õ, ọ, ô, ố, ồ, ổ, ỗ, ộ, ơ, ớ, ờ, ở, ỡ, ợ
u	u, ú, ù, ủ, ũ, ụ, ư, ứ, ừ, ử, ữ, ự
d	d, đ

In this paper, we describe the orthography of Vietnamese, which has the highest ratio of diacritical

[1]http://vietpad.sourceforge.net/

words among diacritical languages. Based on the Latin alphabet, there are 29 letters in Vietnamese alphabet including 11 vowels and 18 consonants. 22 letters of them are Latin letters (“f”,“j”, “w” and “z” are removed), and the rest are newly created ones (Đoàn, 2016).

Those new ones are the combination of four diacritics and the Roman alphabets (breve, inverted breve, horn, d with stroke) (Đoàn, 2016) . The 5 tone markings (acute, grave, hook, tilde and dot-below) are used to describe the tone of a syllable that can be marked on the vowel. In a word, diacritics in Vietnamese are put on all vowel letters and one consonant letter (d). Therefore, there are 22 input characters without diacritics, from which 89 characters with diacritics are inferred. The rules for converting from non-diacritic to diacritic letters are shown in Table 1. Letters not in the table should be ignored when restoring diacritics, i.e. ’b’, ’c’, ’g’, ’h’, ’k’, ’l’, ’m’, ’n’, ’p’, ’q’, ’r’, ’s’, ’t’, ’v’, ’x’.

2.2 Transformer model

Transformer model (Vaswani et al., 2017) is a type of neural network architecture developed to solve the problem of sequence transduction, or neural machine translation. It is built based on Seq2seq architecture, comprising an encoder and a decoder. The encoder takes the input sequence and maps it into a higher dimensional space using something like an abstract of the input. It is then fed into the decoder, where it is turned into an output sequence.

Before the appearance of transformers, the encoder and the decoder of the Seq2Seq model relied on gated recurrent neural networks (RNNs), such as LSTMs, with added attention mechanisms to handle the input and output sequences without fixed length and avoided gradient vanishing problem. However, the transformer model with only attention-mechanisms without any RNN facilitates more paralleling during training computations, which brings better results with less time for training. Transformers currently have become the state-of-the-art architectures in NLP.

Self-Attention and Multi-Head Attention (Vaswani et al., 2017). Self-attention can be described as mapping a query and a set of key-value pairs to an output. Query, key, and value vector are calculated by multiplying the input by query, key, value, trainable matrix respectively. Then query, key, value vector is fed into Scaled Dot-Product Attention. The detailed description is shown by the following equation:

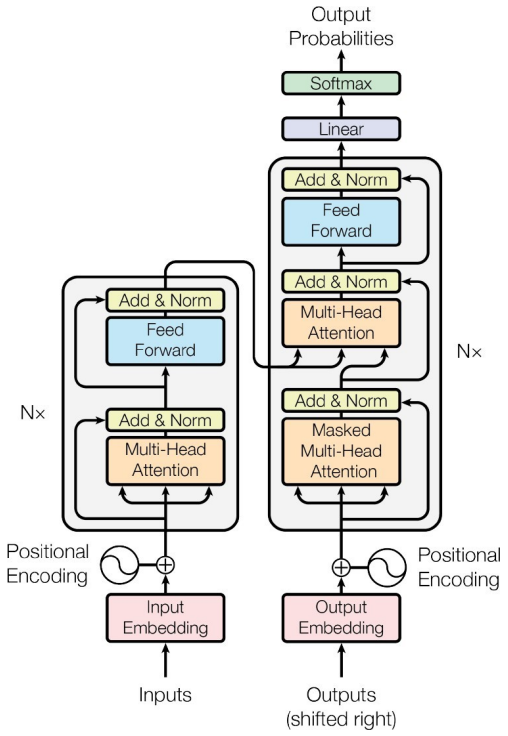

Figure 1: Architecture of transformer model (Vaswani et al., 2017).

Instead of calculating single attention at one time, we can calculate multiple attention in parallel, but each attention uses a different key, value, and query matrices. This technique is called multi-head attention. Each set of 3 key-value-query matrices is a head that pays “attention” to a certain piece of content of the input. The output of all heads will be concatenated together to form a complete vector output. Multi-head attention lets the model jointly attend to information from different representation sub-spaces at different positions.

Position embedding Because it does not use any CNN or RNN classes, the transformer needs to use a different way to handle the order of inputs, i.e. the effect of position encoding. Some information about the relative or absolute position of the tokens are injected into the input. Positional encoding can be learned or fixed. It has the same dimension as the embedding and the two are summed before being fed

into encoder or decoder block.

Model architecture The decoder and encoder are the main components of the transformer model, illustrated in Figure 1. Each part is a stack of the same blocks. Encoder block has two sub-layers: a multi-head self- attention mechanism, and a simple feed-forward network. A residual connection is deployed around each of the two sub-layers, followed by layer normalization. Similar encoder block, decoder block is built based on a multi-head attention and a feed-forward network. However, the decoder block inserts a third sub-layer to perform multi-head attention over the output of the encoder stack. Besides, the self-attention sub-layer is modified to make sure that the predictions for position i only depend on the known outputs at positions less than i.

3 Proposed model

We propose a novel model TDP (Transformer Decoder with Penalty layer) that only includes a transformer decoder at the character level with a penalty layer, whose architecture is illustrated in Figure 2. In this architecture, only the decoder blocks are kept instead of a full transformer model. As the origin full transformer, our model is a stack of 6 decoder blocks. With only the decoder, we can still solve the diacritic restoration problem since the length of the input is the same to that of the output, and an input character only corresponds to exact one output character. The encoder is redundant for this task. Moreover, the predicting time of the only decoder is expected to be much quicker than the full one.

When predicting, an output character corresponds to exact one input character in a position. Hence, we do not need to model the position in a separate layer. In self-attention, the memory keys and values come from the output of the previous decoder layer are used instead of that of the encoder. In the full architecture, the decoder has to be repeated every time step, corresponding to the number of input's characters. However, in the transformer decoder, we do not need to repeat the decode every time step. We can ignore the masking step of the decoder and only run it once. As the result, the predicting time of our model is expected to be reduced about x times compared to the full one, whereas x is the number of input's characters.

As mentioned in Table 1, each input character only has a specific number of output characters. For example, with the input 'i', the output can only be one of the six characters 'í', 'ì', 'ị', 'ĩ', 'ỉ', 'i'. If the input is a consonant like 'g' the output of the model must only be 'g'. Therefore, we propose a penalty layer which restricts the output with only some possible values of the input letter. This layer first looks up from a diacritic conversion dictionary and then calculates a penalty matrix for input characters. The penalty layer is executed in parallel to the decoder model, and then the penalty matrix will be added to the decoder output matrix to force the output character to be one of possible values, illustrated in Equation 1. This mechanism ensures that the output will not produce strange characters for the input. For example, when the user enters the word "co", the model sometimes predict to "ca". This issue can be addressed by the penalty layer.

$$Output = argmax(DecoderOutput + Input * PenaltyMatrix)$$

The penalty matrix works like an embedding matrix, each row of which is a penalty vector corresponding to a non-diacritic character. If the input character can be converted to the output character, the scalar at that corresponding position is 0; otherwise, it is an extremely negative number:

$$PenaltyMatrix_{i,j} = \begin{cases} 0 & \text{if input charater i can convert to j} \\ -\infty & \text{if input charater i can't convert to j} \end{cases}$$

We can feed the input as a sequence of syllables instead of characters with the expectation of reducing the processing time and making the input more meaningful. However, the amount of input and output vocabulary turned to be immense. That makes it complicated to guide the output of the model following the diacritical rules of the language. With the character-level approach, the vocabulary of diacritic conversion is small hence save much more time to construct the penalty matrix from the input. The penalty layer using vectorization makes the calculations much simpler and faster than using directly diacritic rules for the input and output sentences.

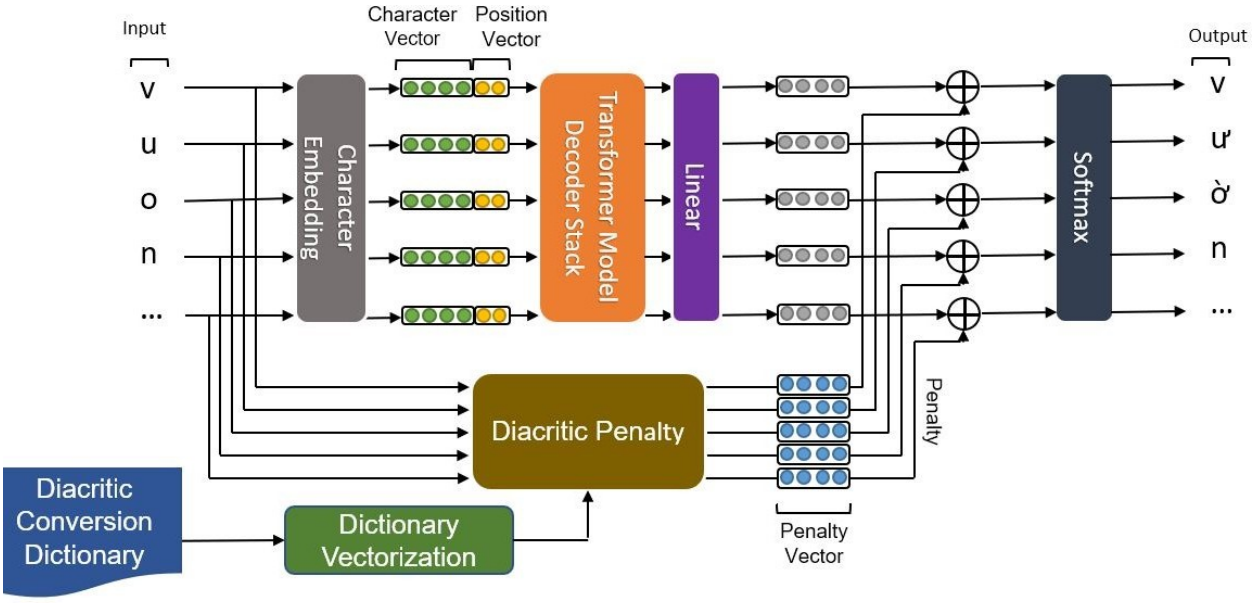

Figure 2: Architecture of the proposed hybrid model for diacritic restoration.

4 Related works

4.1 Transformer model

Since 2017, when the transformer was introduced in (Vaswani et al., 2017) , it has become the basic building block of most state-of-the-art architectures in NLP. In the original paper, transformer models achieve a new state of the art on both WMT 2014 English-to-German and WMT 2014 English-to- French translation tasks with a small fraction of the training costs of the best models from the literature. To solve diacritic restoration problem, the Transformer word-based model is only applied in Yorùbá Language (Orife, 2018) . When compared to the other methods mentioned in the paper (Orife, 2018) , the transformer model outperforms with word accuracy 95.4%, 5.3% higher than the second method on the test set.

4.2 Transformer decoder

In many NLP problems, instead of using the full transformer model, people only use the decoder part. This architecture was first used in generating English Wikipedia articles by summarizing long sequences (Liu et al., 2018). To handle this problem, they propose a two-state method. First, they use extractive summarization to coarsely identify salient information. Then, they use a neural abstractive model to generate the article. Authors affirmed that monolingual text-to-text tasks redundant information is re-learned about language in the encoder and decoder, so they only use the decoder. Their experiment showed that their model with decoder-only for the abstractive stage could handle very long input-output examples, better than using both traditional encoder-decoder architectures and recurrent neural network (RNN).

In addition, the transformer decoder is also used to create pre-trained language models such as GPT model (Radford, Narasimhan, et al., n.d.; Radford, Wu, et al., n.d.). There have been two versions of the GPT model released. GPT is generative pre-training of a language model on a diverse corpus of unlabeled text, followed by discriminative fine- tuning on each specific task. Both of two versions achieve great results in NLP tasks. GPT-1 improves the state of the art on 9 of the 12 datasets they study. GPT-2 is a direct scale-up of GPT-1, with more than 10X the parameters (1.5B) and trained on more than 10X the amount of data (40 GB text data). Due to author's concerns about malicious applications of the technology, they only release a much smaller model for researchers instead of the trained model.

To the best of our knowledge, this is the first time the transformer decoder is used for the diacritic

restoration task.

4.3 Hybrid method

When solving NLP problems, in order to improve the quality of neural networks, people often combine neural networks with different techniques, such as traditional machine learning models and rule-based models. A hybrid data-model parallel approach (Ono et al., n.d.) was used for reducing training time of sequence-to-sequence machine translation model. In abstractive summarization, to get better performance at content selection of neural network-based method, the work in (Gehrmann et al., 2018) combined a standard neural model with adata-efficient content selector to over-determine phrases in a source document that should be parts of the summary. Furthermore, the hybrid method requires much fewer data to train, which makes it more adaptable to new domains.

In our proposed model, we combine a transformer decoder with a diacritic penalty layer which restricts the output with all possible values corresponding to the input. This guides the model more accurate, reduces training time and gives reasonable outputs.

5 Experiment

In this section, we present some experiments for our proposed model with a Vietnamese corpus.

5.1 Dataset

The test set that we have to work in this paper is from banking domain. It includes 8,000 sentences.

To enhance the training set of this domain (i.e. 25,000 sentences), we retrieve more from Internet newspapers[2] . This corpus contains approximately 29 Gb of Vietnamese text files and approximately 160 millions of Vietnamese sentences. Nonetheless, due to a number of loan words and wrong spelling in that corpus, we only keep about 7% of sentences based on their types (i.e. interrogation, exclamation and affirmation) and constituent words. We use a Vietnamese dictionary VCL[3] as a filter. The final dataset contains about 11 millions of Vietnamese sentences. The valid set consists of 5,000 randomly selected sentences from the banking training data and 25,000 sentences from Internet newspapers. The rest is used for training.

All data in the corpus contain diacritics, which is the output that the model has to predict. The input is sentences after being stripped off all diacritics.

5.2 Evaluation method

Many input characters have only one candidate output, so the high character accuracy does not prove that the model works well. Therefore, although the model is at character level, we use evaluate the model at syllable level:

$$Accuracy = \frac{\#CorrectPredictedSyllable}{\#TotalPredictedSyllable}$$

5.3 Training

For the Transformer decoder, we reuse most of the hyperparameters proposed in (Vaswani et al., 2017). The decoder is composed of a stack of N = 6 identical blocks and each block contains 8 multi- head attention, but the dimension d model is 128 instead of 512 because of a small character vocabulary. The problem of recovering diacritics does not require the use of context too far, so we set the maximum sentence length to be 60 characters to avoid padding too long and save predicting time. Sentences longer than 60 characters will be broken down into sections of 60 characters, between which there will be an overlapping part with offset length = 10 characters.

Our model is trained on the hardware with the configuration as follows: 01 Tesla V100-PCIE-32GB GPU, Intel(R) Xeon(R) Silver 4210 CPU, 120GB RAM. We set the batch size=128 and use the default optimizer proposed in (Vaswani et al., 2017). The model converges after about 5 days. We evaluate the final model obtained by taking the average of the last 5 checkpoints.

5.4 Result

To compare with our model, we retrain the model architecture proposed in (Náplava et al., 2018) on our dataset. We compare this previous seq2seq model with or without beam search.

The result is shown in Table 2. Our TDP model with only transformer decoder and a penalty layer receives a better accuracy (i.e. 1.53%) and 16 times faster than the full transformer one. Compared to the previous seq2seq model, although the results were slightly lower than the model that used beam search

[2]https://github.com/binhvq/news-corpus#full-txt-v2

[3]https://vlsp.hpda.vn/demo/?page=vcl

(0.46%), the predicting time was reduced by approximately 10 times in both cases using CPU or GPU. Our model is 1% better and about 8 times faster than the one without beam search.

Table 2: Experimental results for hybrid diacritic restoration model. The prediction is executed on Tesla V100-PCIE-32GB GPU, Intel(R) Xeon(R) Silver 4210 CPU, 120GB RAM

Model	**Word accuracy (%)**	**Predicting time**	
		GPU(s)	CPU(s)
Seq2Seq re-run (Náplava et al., 2018)	97.52	0.372	0.423
Seq2Seq + Beam search re-run (Náplava et al., 2018)	98.83	0.433	0.533
Transformer model (full)	96.84	0.904	0.896
TDP model (our model)	98.37	0.043	0.055

To further evaluate how the model works in practice, we have performed an error analysis by statistically reporting the cases in which the model predicts incorrectly. The words which are wrongly predicted the most are listed in the table 3 below. The results show that most of mispredicted words are the ones that appear frequently in the banking domain but rarely appear in the others.For example, the word "thẻ" (card), "dùng"(use), "hủy" (cancel), "khóa" (key or stop), "vay" (loan), "lãi"(interest), ect, despite being small in number, are important words for the conversation. Inaccurate diacritic restoration of those words can lead to complete change of sentence meaning. For instance, the sentence "toi muon dung dich vu nay" can be restored to "tôi muốn dùng dịch vụ này" (I want to use this service) or "tôi muốn dừng dịch vụ này"(I want to stop this service), which are of diametrically opposite meanings. Therefore, it is essential that the domain adaptation technique be adopted in the future to bring enhanced efficiency to the industry.

Table 3: The most incorrectly predicted syllables

Expected output	**Number wrong predict**	**Confused with**
Thẻ	120	Thể, thế
dùng	87	Đúng, dụng, dừng, đứng
Hủy	59	huy
khóa	52	khoa
bạn	51	Bán, bàn, bẩn
thế	51	Thẻ, thể
vậy	44	vay
lãi	43	lại

6 Conclusion

In this work, we propose a hybrid diacritic restoration model TDP which includes a transformer decoder model and a diacritic penalty layer. The transformer decoder can solve this problem since an input character only corresponds to exact one output character. The only decoder also helps to decrease much predicting time since it does not need to repeat every time step. The purpose of the penalty layer is to guide the model to produce only possible diacritic letters of the language. The experimental results on a Vietnamese corpus show that our model TDP with only transformer decoders and a penalty layer helps the predicting time reduce from about eight to ten times compared to the state- of-the-art method. Whereas, the accuracy of the proposed method is better than (i.e. 1%) or equal to the sequence-to-sequence without or with beam search. Although the accuracy is quite high, the model wrongly predicts some important words in banking domain, e.g. “thẻ” (card) to “thế” (and), “mượn” (borrow) to “muốn” (want)... In the future, we will work on domain adaptation to solve this problem. In addition, we also consider using language model to improve the quality of the model.

References

Nguyen, K.-H., & Ock, C.-Y. (2010). Diacritics restoration in vietnamese: Letter based vs. syllable based model (B.-T. Zhang & M. A. Orgun, Eds.). In B.-T. Zhang & M. A. Orgun (Eds.), *PRICAI 2010: Trends in artificial*

intelligence, Berlin, Heidelberg, Springer. https://doi.org/10.1007/978-3-642-15246-7_61

Do, T. N. D., Nguyen, D. B., Mac, D. K., & Tran, D. D. (2013, August). Machine translation approach for vietnamese diacritic restoration, In *2013 international conference on asian language processing*. 2013 International Conference on Asian Language Processing. https://doi.org/10.1109/IALP.2013.30

Đoàn, T. T. (2016). *Ngữ âm tiếng việt* [Accepted: 2017-10-09T02:27:47Z]. H. : Đại học Quốc Gia Hà Nội. Retrieved September 12, 2020, from http://repository.vnu.edu.vn/handle/VNU_123/59688

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *arXiv:1706.03762 [cs]*, arxiv 1706.03762. Retrieved June 26, 2020, from http://arxiv.org/abs/1706.03762

Gehrmann, S., Deng, Y., & Rush, A. M. (2018). Bottom-up abstractive summarization. *arXiv:1808.10792 [cs]*, arxiv 1808.10792. Retrieved July 3, 2020, from http://arxiv.org/abs/1808.10792

Hung, B. T. (2018). Vietnamese diacritics restoration using deep learning approach. *2018 10th International Conference on Knowledge and Systems Engineering (KSE)*. https://doi.org/10.1109/KSE.2018.8573427

Liu, P. J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., & Shazeer, N. (2018). Generating wikipedia by summarizing long sequences. *arXiv:1801.10198 [cs]*, arxiv 1801.10198. Retrieved July 1, 2020, from http://arxiv.org/abs/1801.10198

Náplava, J., Straka, M., Straňák, P., & Hajič, J. (2018, May). Diacritics restoration using neural networks, In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*. LREC 2018, Miyazaki, Japan, European Language Resources Association (ELRA). Retrieved June 26, 2020, from https://www.aclweb.org/anthology/L18-1247

Orife, I. (2018). Attentive sequence-to-sequence learning for diacritic restoration of yor\‘ub\’a language text. *arXiv:1804.00832 [cs]*, arxiv 1804.00832. Retrieved June 27, 2020, from http://arxiv.org/abs/1804.00832

Nga, C. H., Thinh, N. K., Chang, P.-C., & Wang, J.-C. (2019, December). Deep learning based vietnamese diacritics restoration, In *2019 IEEE international symposium on multimedia (ISM)*. 2019 IEEE International Symposium on Multimedia (ISM). https://doi.org/10.1109/ISM46123.2019.00074

Ono, J., Utiyama, M., & Sumita, E. (n.d.). Hybrid data-model parallel training for sequence-to-sequence recurrent neural network machine translation, 9.

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (n.d.). Improving language understanding by generative pre-training, 12.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (n.d.). Language models are unsupervised multitask learners, 24.