Translate Han-Nom to Vietnamese using Neural Machine Translation Methods

Tan Chau[1,2*], Vu Ngo-Duc[1,2†], Tri Nguyen-Minh[1,2†] and Duc Nguyen-Tran-Anh[1,2†]

[1]Faculty of Information Science and Engineering, University of Information Technology.
[2]Vietnam National University Ho Chi Minh City, Linh Trung, Thu Duc, Ho Chi Minh city Vietnam.

*Corresponding author(s). E-mail(s): 20520926@gm.uit.edu.vn;
Contributing authors: 20520950@gm.uit.edu.vn; 20522052@gm.uit.edu.vn; 20521198@gm.uit.edu.vn;
†These authors contributed equally to this work.

Abstract

Han-Nom is the old language of Vietnam. Nom script began to establish and develop from the 10th to the 20th century in Vietnam. Initially, the Nom script was often used to record names of people and places, then gradually popularized and entered the nation's cultural activities. Nowadays, the Han-Nom language has gradually been forgotten. In this article, we introduce a translation tool that translates Han-Nom to Vietnamese using Deep Learning methods.

Keywords: Han-Nom, Vietnamese, Machine Translation, Deep Learning, Seq2Seq, Transformer

1 Introduction

The writing system is a sign of a country's civilization and an invention of its country. Vietnamese has a wonderful language with rich phonetics and the most affluent writing system in East Asia. In the course of history, Vietnamese writing has gone through a journey from *Han* letters to *Nom* letters.

Finally, the *Quoc Ngu* alphabet is based on the Latin writing system, and accompanying each type of script is glorious history worth reviewing of the nation.

After ending the thousand years of Northern domination in 939 CE, Vietnamese people created the *Nom* script based on the *Han*. Nom script used Chinese characters to represent both borrowed Sino-Vietnamese vocabulary and native words with similar pronunciation or meaning. And over the next 1000 years, from the 10th to the 20th century in Vietnam, *Nom* script was used to write most of the religion, cultural history, literature documents, philosophy, medicine, and many aspects of Vietnamese culture. Nevertheless, this language is now in risk of being forgotten by the shift to the *Quoc Ngu* modern script. According to the Vietnamese Nom Preservation Foundation - VNPF [**?**]: "Today, less than 100 scholars worldwide can read *Nom*. Nowadays, Nom script is being forgotten by the popularity of modern Vietnamese (Quoc-Ngu script). This project is the next project of NomNaOCR which we build a dataset and end-to-end text detection and recognition system for Nom-script. In this project, we build a Han-Nom translation app to Vietnamese using the data of NomNaNMT. We do this series of projects to try to reduce the forgetting of the Nom script and also help the researchers who are studying about Han-Nom script to make their research become easier.

2 Related Work

2.1 Research on translation approaches

Machine translation is the translation the text from one source language to another target language by using computer or machine. There are commonly 4 types of machine translation: Statistical machine translation, Rule-based machine translation, Hybrid machine translation and Neural machine translation.

Statistical machine translation works by alluding to statistical models. It expects to decide the correspondence between a word from the source language and a word from the objective language. Rule-based machine translation works by using the basics of grammatical rules. Hybrid machine translation is the combination of two approaches mentioned above.It uses a translation memory, so it makes unquestionably more successful regarding quality but it has a lot of downsides. The biggest downside is the requirement for enormous editing and human translators. Neural machine translation is a type of machine translation that uses neural network models or deep learning approaches to build statistical models with the end goal of translation.

We choose the Neural Machine Translation approach because of its neural network architecture, which allows it to learn from vast amounts of data and adapt to new contexts. Moreover, Neural machine translation also has higher accuracy, faster learning, simple integration and flexibility, more cost efficiency, and more scalable.

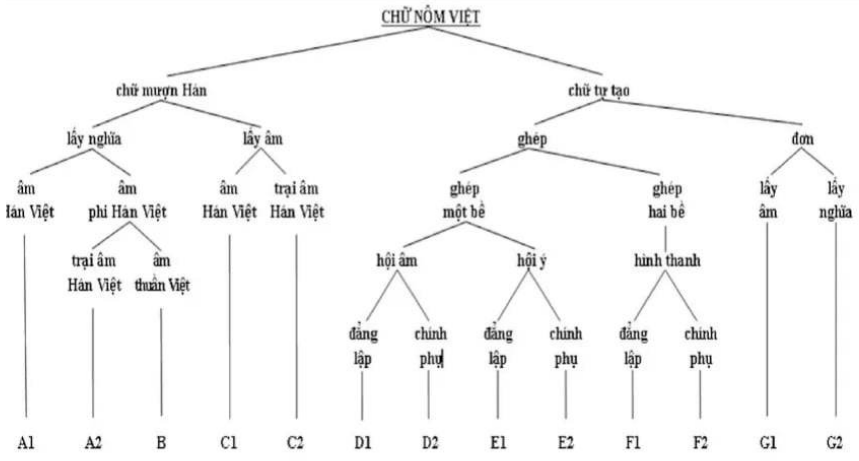


Fig. 1 Nom-script structure. Source: http://www.nomfoundation.org

2.2 Research on the foundation and the structure of Nom-script

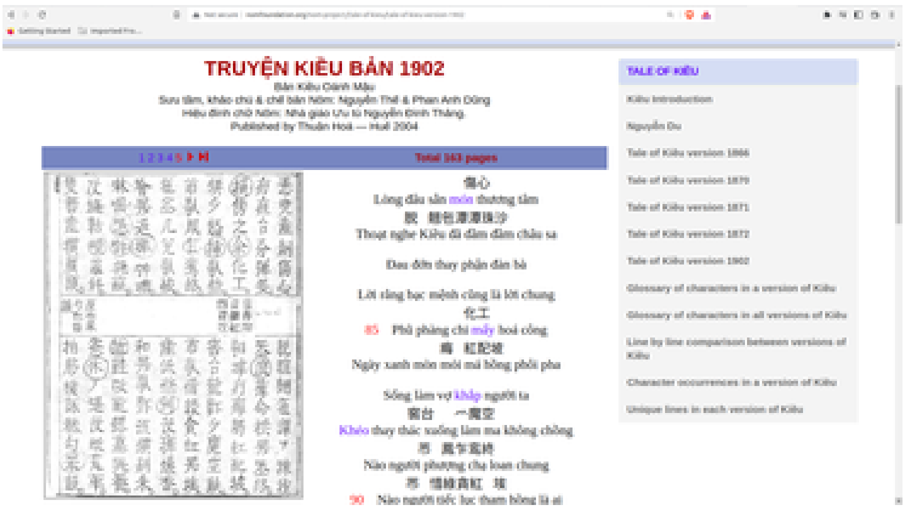
Chinese characters were introduced to Vietnam after the Han dynasty conquered the country in 111 BC. Vietnam gained independence in 938 AD, but Chinese literature was adopted for official purposes in 1010. Vietnamese scholars were thus intimately familiar with Chinese writing. Vietnamese applied the structural principles of Chinese characters to develop the Nom script in other to record a native language. The new script mainly was used to record folk songs and other popular literature. Nom script briefly replaced Chinese for official purposes under the Ho dynasty (1400–1407) and under the Tay-Son (1778–1802).

Nom script structure contains borrowed characters and locally invented characters.

3 Building NomNaNMT Dataset Progress

3.1 VNPF - The Vietnamese Nom Preservation Foundation

The Vietnamese Nom Preservation Foundation [**?**] arose from the postwar efforts of Vietnamese and Americans who feared that an entire literary culture–the 1000 years of writing in Nom script used to record the Vietnamese language and its vast heritage of poetry, history, medicine, royal edict, and religion–was about to go extinct. As a result of these efforts, a core set of Nom script characters was successfully encoded in Unicode / ISO 10646.



TRUYỆN KIỀU BẢN 1902
Bản Kiều Oánh Mậu
Sưu tầm, khảo chú & chế bản Nôm: Nguyễn Thế & Phan Anh Dũng
Hiệu đính chữ Nôm: Nhà giáo Ưu tú Nguyễn Đình Thắng
Published by Thuận Hoá — Huế 2004

1 2 3 4 5 Total 163 pages

[illegible]
Lòng đâu sẵn mối thương tâm
[illegible]
Thoạt nghe Kiều đã đầm đầm châu sa

Đau đớn thay phận đàn bà

Lời rằng bạc mệnh cũng là lời chung
化工
85 Phũ phàng chi mấy hoá công
[illegible]
Ngày xanh mòn mỏi má hồng phôi pha

Sống làm vợ khắp người ta
[illegible]
Khéo thay thác xuống làm ma không chồng
[illegible]
Nào người phượng chạ loan chung
[illegible]
90 Nào người tiếc lục tham hồng là ai

TALE OF KIỀU
Kiều Introduction
Nguyễn Du
Tale of Kiều version 1866
Tale of Kiều version 1870
Tale of Kiều version 1871
Tale of Kiều version 1872
Tale of Kiều version 1902
Glossary of characters in a version of Kiều
Glossary of characters in all versions of Kiều
Line by line comparison between versions of Kiều
Character occurrences in a version of Kiều
Unique lines in each version of Kiều

Fig. 2 VNPF texts documents site. Source: http://www.nomfoundation.org/nom-project/tale-of-kieu/tale-of-kieu-version-1902

Realizing that standardization alone was not sufficient, in 1999 the founders (James Phuoc Do-Ba, Nhan-Thanh Ngo, John Balaban, and later, Viet Ngo-Trung) incorporated the VNPF as a 501c3 public charity, raising funds to preserve and popularize this rich cultural heritage by developing software tools for its digitization, printing, study, physical conservation, and internet sharing of the many works in national libraries and Buddhist temples. At the same time, the VNPF developed programs to support students and senior scholars, while encouraging public awareness. A particular focus was to awaken the younger generation to the potential for employing the tools of high technology in the preservation and study of this great cultural heritage.

We use the VNPF site which includes some Han-Nom literature, historical documents, and poems like: The Tale of Kieu, the Tale of Luc Van Tien, and the History of Greater Vietnam. We use these materials to build the NomNaNMT dataset. The dataset is now published at the site: https://www.kaggle.com/datasets/quandang/nomnanmt

3.2 Data Collection and Preprocessing

The VNPF [?] site has a number of Han-Nom documents which have both in Han-Nom script and in Vietnamese script. We use BeautifulSoup to collect the data. We crawl the data from each episode of documents to each CSV file. Then we merge those files into the NomNaNMT dataset.

The NomNaNMT dataset after being merged still have some special characters. We use a regular expression to remove these characters and punctuation marks in both the Vietnamese script and Han-Nom script. Then, we split the Nom script and Vietnamese script into each column.

4 Methods

4.1 Sequence-to-Sequence method

The Seq2Seq model is designed based on Encoder-Decoder Architecture consisting of two components: an encoder that takes a variable-length sequence as input, and a decoder that acts as a conditional language model, taking encoded input and the leftwards context of the target sequence and predicts the subsequent token in the target sequence.

4.1.1 Encoder

The Encoder transforms an input sequence of variable length into a fixed-shape context variable c. Suppose that the input sequence is $x_1, x_2, ..., x_T$ such as x_t is the t token. At time step t, the RNN will have two input feature vectors X_t for x_t and the hidden state h_{t-1} from the previous time step into the current hidden state h_t. We use f function to express the transformation of the RNN's recurrent layer:

$$h_t = f(x_t, h_{t-1}) \tag{1}$$

In general, the encoder captures the information of all hidden states and encodes them into context variable c through a customized function q:

$$c = q(h_1, \ldots, h_T) \tag{2}$$

4.1.2 Decoder

As mentioned above, the context variable encoded input sequence is $x_1, x_2, \ldots, x_T$. Suppose the output of the training set is $y_1, y_2, \ldots, y_T$, for each time step t. At the time t, the conditional probability of the output y_t will depend on the previous output $y_1, y_2, \ldots, y_(t-1)$ and the context variable c, i.e.,

$$P(y_t \mid y_1, y_2, \ldots, y_{t-1}, c) \tag{3}$$

To predict the subsequent token $t+1$ in the target sequence, the RNN decoder takes the previous step's target token y_t, the context variable c as its input, the hidden RNN state from the previous time step s_{t-1}, and transforms them into the hidden state s_t, at the current time step. We can use a function g to indicate the transformation of the decoder's hidden layer:

$$s_t = g(y_{t-1}, c, s_{t-1}) \tag{4}$$

4.1.3 Attention mechanism

With classic seq2seq, if the input were a sentence consisting of a considerable number of words, the model would be in trouble.

The attention mechanism was proposed by Bahdanua et al., and Loung et al. This technique allowed for a considerable improvement in machine translation systems by focusing on the relevant parts of the input sequence. Thus, the context variable c is calculated by α weighted sum of the final hidden states:

$$c_t = \sum_{i=1}^{T} (\alpha_{ti}, h_i) \tag{5}$$

where the weight $\alpha_{t,j}$ is computed by an alignment model.

$$\alpha_{ti} = \frac{exp(score(s_{t-1}, h_i))}{\sum_{i=1}^{T} exp(score(s_{t-1}, h_i))} \tag{6}$$

The *score* function can be computed using a neural network. Given the predicted preceding words $y_1, y_2, \ldots, y_{t-1}$, context vector c_t, and the RNN hidden state s_t, the decoder calculates a probability of the next word y_t:

$$P(y_t \mid y_1, y_2, \ldots, y_{t-1}) = a(y_{t-1}, c, s_{t-1}) \tag{7}$$

where $a()$ is a softmax activation function and

$$s_t = g(y_{t-1}, c_t, s_{t-1}) \tag{8}$$

here, g is a non-linear function. It can be a logistic function, an LSTM unit, or a Gated Recurrent Unit (GRU).

4.2 Transformer method

4.2.1 Transformer Attention

The first is about the single-head transformer attention mechanism[1]. The structure of single-head engagement includes 3 input matrices as Query (Q), Key (K), and Value (V). The Query and Key matrix calculate the score distribution for word pairs (w_i, w_j). The Value matrix will be based on the score distribution combined with the softmax function to calculate the output probability distribution vector.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{9}$$

d is the scaling factor, depending on the size of the input of the attention layer. Dividing by d is the dimension of the key vector to avoid overflow if the exponent is too large.

After obtaining 3 attention matrices in the output, we will concatenate these matrices by columns to obtain a composite multi-head matrix with the same height as the input matrix. Each query, key, and value undergoes a separate linear transformation to the lower dimension to make the multi-head attention dimension smaller.

$$head_i = Attention(Q_i, K_i, V_i) \tag{10}$$

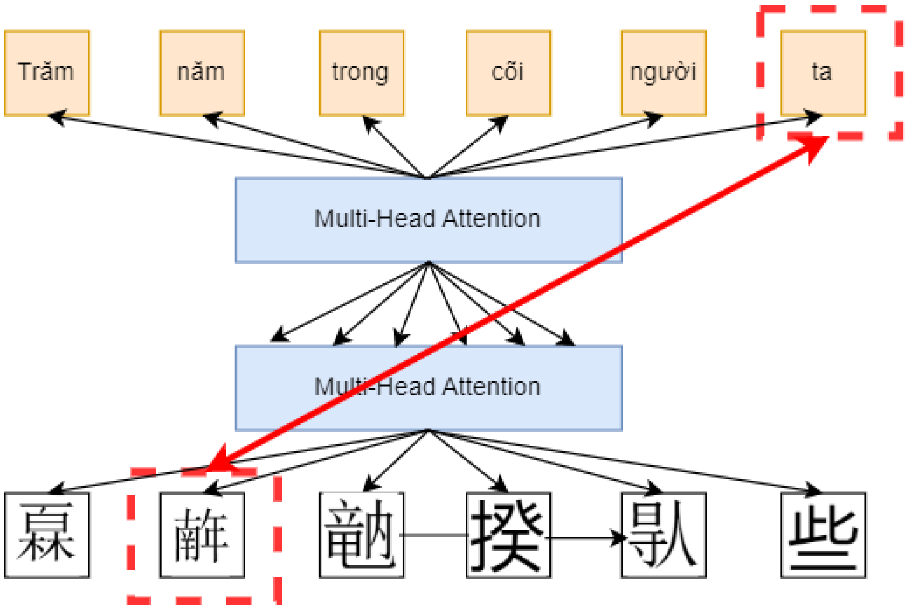


Fig. 3 Transformer attention handling dependencies between input or output tokens

$$Multihead(Q, K, V) = concatenate(head_1, head_2, ..., head_n)W_0 \tag{11}$$

Each linear layer in the Transformer model uses a Glorot uniform initializer, which draws samples from a uniform distribution range in [-l, l], limit is calculated from the formula:

$$l = \sqrt{\frac{6}{f_{in} + f_{out}}} \tag{12}$$

where f_{in}, f_{out} is the number of input and output units. The use of transformer attention can solve the **long-range dependencies** problem that traditional RNNs are facing. The dependence on input and output codes tends to decrease. The Transformer handles this flaw by allowing the encoder and decoder to see the entire input sequence all at once. In addition, this solves the problem of **sequential nature**. This slow down the whole process for GPU

4.2.2 Transformer Encoder

We initialize the encoder with a stacked composite of 2 defined layers. Each layer consists of 2 sub-layers and ReLU activation in it. The first sublayer is the multi-head self-attention. The second layer is merely fully-connected feed-forward layers. We will use a residual connection in each sublayer right after the normalization layer. Each sublayer is followed by a LayerNorm taking the sublayer residually as follows:

$$LayerNorm(x + sublayer(x)) \tag{13}$$

4.2.3 Transformer Decoder

The decoder is built according to the same architecture of the Encoder with a stacked composite of 2 defined layers, except that there are three sublayers instead of two: the multi-head self-attention layer (decoder-decoder), the multi-head encoder's attention layer (decoder-encoder) and a feedforward layer like the one in an encoder unit. Residual mechanism is also applied in the same way as in Encoder.

4.2.4 Positional Encoding

Unlike RNN, Multihead Attention Model cannot automatically use word position. To get information about the position of each token, we use sinusoidal to get positional encoding [2]. Sinusoidal function will get positional encoding that will be appended to encoder and decoder word embeddings. The function looks as follows:

$$PE_{(pos,2i)} = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{14}$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{15}$$

Where pos: Position of an object in the input sequence 0¡=pos¡=L/2, which L is the length of the sequence.
d_{model}: Dimension of the output embedding space.
i: Value for mapping to column indices (0 ¡ i ¡
The model will learn to take relative positions, each dimension of the position encoding is a wave of different frequency.

5 Metrics

5.1 Bilingual Evaluation Understudy - BLEU

The BLEU [3] score according to the introduction will have the effect of evaluating the higher score if the Machine Translation results are close to the translator's results. BLEU score will be calculated based on n-gram score P_1, P_2, P_3, P_4:

$$BLEU = \exp(\frac{P_1 + P_2 + P_3 + P_4}{4}) \tag{16}$$

The shorter the sentence, the higher the BLEU score value field tends to be. So you need to multiply by a Brevity Penalty(BP) value. Then the formula of BLEU:

$$BLEU = BP \times \exp(\frac{P_1 + P_2 + P_3 + P_4}{4}) \tag{17}$$

With a corpus consisting of many sentences, the calculation of the BLEU score will be different. Instead of using the average BLEU value of the sentences,

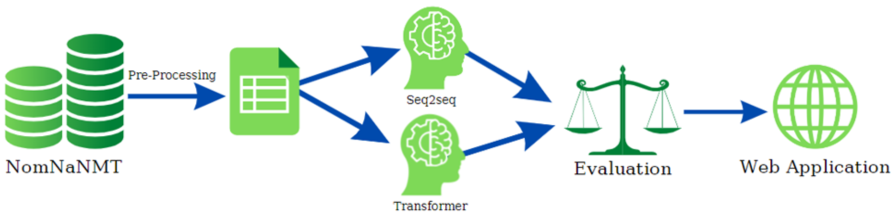


Fig. 4 Implementation progress of NomNaNMT progress

according to the original BLEU metric, they used micro-average precision. We use the *nltk* package to calculate the BLEU score for the corpus

6 Experiments and results

6.1 Data

The NomNaNMT corpus after being merged still has some problems such as special characters, punctuation marks, and missing whitespace, ... We use other methods for processing data for other attributes. With Vietnamese, we removed special characters and punctuation marks. Besides, we converted uppercase characters to lowercase characters, add whitespace, used the *wor_tokenize*() function of *underthesea* library to tokenize Vietnamese, and add tags $< sos >< eos >$. With Nom, We tokenize sentences by whitespace and add tags $< sos >< eos >$. Tags $< sos >< eos >$ for the model to recognize the start and the end of the sentence.

To prepare for the predictive model building step, we divide the data set of 32000 lines into 90% for training corpus, and 10% for result evaluation.

6.2 Models

6.2.1 Seq2Seq

In our model, the encoder has an Embedding layer that converts tokens IDs to vectors, and bi-directional LSTM layers to process those vectors sequentially. The decoder has an Embedding layer to convert token IDs to vectors, The LSTM layer keeps track of what's been generated so far, The LSTM output will be the query for the attention layer, and the Dense layer is a fully connected layer that produces the logits for each output token. For this model, we initialize $embedding\_size = 64, hidden\_units = 64, learning\_rate = 0.002, test\_split\_size = 0.1, epochs = 100, batch\_size = 128$. We also use *Adam* optimizer to learn weights and compute loss based on sparse softmax cross entropy between logits and labels with $sparse\_softmax\_cross\_entropy\_with\_logits()$ function of $TensorFlow$

6.2.2 Transformer

Each Transformer encoder includes a multi-head self-attention sublayer and a feedforward sublayer (two dense layers using ReLU activation). The decoder has a similar structure to an encoder, with an extra layer a multi-head encoder attention layer. The dropout layer is applied after each sub-layers before normalization.

We used $Glorotuniform$ initializer to $kernel\_initializer$ and $bias\_initializer$ for encoding and decoding layers. For the positional encoding task, we used sinusoidal encoding, The value of d_{model} we built is 64. Besides, we also use $embedding\_size = 64, hidden\_units = 64, learning\_rate = 0.002, test\_split\_size = 0.1, epochs = 200, batch\_size = 128$. We apply $Adam$ optimizer to learn weights and compute loss based on sparse categorical cross-entropy between logits and labels

6.3 Experimental Results

After the prediction process from the Han-Nom test corpus, we have 3000 predicted Vietnamese sentences. We use the BLEU scale to evaluate predicted Vietnamese sentences with labels. With the Seq2seq model, the BLEU score is 32%. The transformer model gives better results than Seq2seq with a BLEU score is 35%.

7 Conclusion

In this article, we use neural machine translation methods to translate the Han-Nom script into Vietnamese script. We build the NomNaNMT dataset from the VNPF site by crawling poems, historical documents, and Tales. Two methods we use are Seq2Seq and Transformer. The transformer model, which solves the **long-range dependencies** problem of RNNs, gives better results when having 35% BLEU score, while 32% is the BLEU score of Seq2Seq. We also build a translation tool by using Flask with a better model.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., ??? (2017). https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[2] Takase, S., Okazaki, N.: Positional encoding to control output sequence length. arXiv preprint arXiv:1904.07418 (2019)

[3] Papineni, K., Roukos, S., Ward, T., Zhu, W.-J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA (2002). https://doi.org/10.3115/1073083.1073135. https://aclanthology.org/P02-1040