Building a Vietnamese-Lao Machine Translation using Transformer architecture

[*]a project for UET Natural Language Processing class INT3406 1

Hoang Duy Anh
22028243 K67I-CS4
22028243@vnu.edu.vn

Nguyen Khoi Nguyen
22028032 K67I-CS2
22028032@vnu.edu.vn

Nguyen Tran Phuong Ha
22028325 K67I-CS4
22028325@vnu.edu.vn

***Abstract*—This paper documents the process on how we approach and create a Vietnamese-Lao machine translation using the transformer architecture and specifically without using any pre-trained models and modules.**

***Index Terms*—Vietnamese, Lao, machine translation, attention, transformer**

I. Task

The task for this final project is to create a machine translation model for two low-resource languages Vietnamese and Lao without using any pre-traned models. The goal of the machine translation model is to achieve high accuracy in translation for both Vietnamese to Lao and Lao to Vietnamese directions in BLEU and WER metrics.

II. Data

A. Maintaining the Integrity of the Specifications

Data for using in the model's training, validating and testing process are provided in VLSP2023.zip file, in which data is divided into 3 sets for training, validating and testing

- Training Data: stored in 'Train' folder, contains two text files that have 100000 sentences each for each language. In 'train2023.lo' file, Lao sentences are pre-processed and words are spaced.
- Validation Data: stored in 'Dev' folder, contains two text files that have 2000 sentences each for each language
- Testing Data: stored in 'Test' folder, and split into two folders named 'public test', contains 100000 Vietnamese-Lao sentence pairs, and 'private test', split into two folders for each translating direction Vietnamese to Lao and Lao to Vietnamese, contain a total number of 2000 sentence pairs

Our team tried looking for other training data sources for the two languages Vietnamese and Lao, the samples we got have high errors, blank sentences and sentence pairs that don't match their meaning due to being auto-generated from youtube subtitles.

III. Tokenization

We use the SentencePiece library to train separate tokenizers for each language, both using BPE method. For its fast running time and efficiency on the training datasets as Vietnamese sentences are already segmented by words due to the nature of the language's script, Lao sentences are also prepared and are segmented by words. Therefore we can apply BPE method for the tokenizer training process with expected high

After finishing training two tokenizers for two languages, apply tokenization on datasets and save as .csv files for next time uses.

IV. Model Creating Process

Overall, our team builds the models based on the standard Transformer architecture introduced in the 2017 paper *Attention Is All You Need* [1]. We utilize the built-in Transformer module provided by the PyTorch library, with modifications to certain classes and methods as needed to better suit our implementation preferences.

Specifically, we construct two separate Transformer-based models to support bidirectional translation: one for translating from Lao to Vietnamese, and another for translating from Vietnamese to Lao.

A. Embedding and Positional Encoding

For the embedding process, we apply the nn.Embedding() initialization function twice—once in the Encoder class and once in the Decoder class—directly within their respective initialization methods, without defining a separate Embedder class. By default, the nn.Embedding() layer creates an embedding matrix initialized with values drawn from a normal distribution.

The positional encoding is implemented as a separate class, first following the formulation introduced in the 2017 paper *"Attention Is All You Need"*. Later on we switched and made our choice to implement Rotary Positional Encoding Method after considering its higher efficiency compared to the previous method.

B. Attention and Multi-Head Attention

Our team implement the standard *Scaled Dot-Product Attention* formula for the Attention class, and used the same formula defined in Attention class to define Multi-Head Attention. The formulas are defined as follow:

Scaled Dot-Product Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$

Where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

C. Encoder

The Encoder is composed of 6 identical layers. Each Encoder layer consists of the following sub-layers:

- Multi-Head Self-Attention
- Position-wise Feed-Forward Network
- Residual Connections followed by Layer Normalization

The output of each sub-layer is computed as:

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

The feed-forward network (FFN) is applied to each position separately and identically:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

D. Decoder

The Decoder is also composed of 6 identical layers. However, each Decoder layer includes an additional sub-layer to incorporate the output from the Encoder. Specifically, each layer consists of:

- Masked Multi-Head Self-Attention
- Multi-Head Encoder-Decoder Attention
- Position-wise Feed-Forward Network
- Residual Connections followed by Layer Normalization (after each sub-layer)

Each sub-layer applies residual connection and layer normalization as follows:

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

The feed-forward network used in the Decoder is identical to the Encoder:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

In the first sub-layer, masking is applied to the attention weights to prevent attending to subsequent positions during training (autoregressive property).

E. Transformer

After having done difining and overwriting classes needed for the Transformer architecture, we now define a Transformer class that contains the previous defined Encoder and Decoder.

V. Training Process

A. DataLoader

The training dataset consists of 100,000 sentence pairs and a significant variation in the number of tokens per sentence. The longest sentence contains up to 624 tokens, while the shortest has only 3 tokens.

Upon analyzing the token distribution in both the Vietnamese and Lao subsets, we observed that the average number of tokens per sentence is 27 for Lao and 30 for Vietnamese. The 50th percentile (median) and 75th percentile values for the entire dataset are 20 and 33 tokens, respectively.

These statistics indicate that sentence lengths and token counts are unevenly distributed across the dataset, which could help us make good choices and strategy in batching and padding sentences.

Our team came to a conclusion that processing the batching and padding process will fit the most with batch size being 32, padding are automatically adjusted according the longest sentence of each batch with maximum padding length being 128.

B. Training Loop

For the training process, we implement a standard sequence-to-sequence training loop where in each training step, the model receives a batch of tokenized source sentences and corresponding target sentences. The source batch is passed through the encoder to obtain a sequence of encoded hidden representations. The decoder then takes the output of the encoder as input and calculate the attention matrices.

The loss function used is the cross-entropy loss applied to the training and validation dataset. Gradients are computed via backpropagation, and parameters are updated using the Adam optimizer with a learning rate scheduler.

We also apply gradient clipping to prevent exploding gradients, which is common in training deep neural networks on long sequences. The training process continues for a predefined number of epochs or until convergence, with validation loss monitored at the end of each epoch.

Applying the training loop for two models correspond with two translation direction from Vietnamese to Lao and Lao to Vietnamese then save the 'best' model with least loss values.

VI. Result

With the total runtime of our python notebook being 3866.1s, with accelerator GPU T4 x 2 on Kaggle. We achieve the following result (*see table 1*):

a) Vi to Lo: the model showed a decent performance on both public and private test dataset with the SacreBLEU score being 24.81 for both datasets.

b) Lo to Vi: the model performed poorly on public test dataset with the SacreBLEU score being 18.58. The SacreBLEU for private test is 0 for some reasons, we believe there could be some errors occuring during the evaluate process.

c) Conclusion: Without using pretrained models and modules and with using training and validation sets that have total number of 102000 sentence pairs, the model's performance overall isn't as great as we expected from the amount of training data it has and has lots of place for improvements.

Test set		VI-LO	LO-VI
Public test set	BLEU	24.81	18.58
	Inference time	236.69s	275.89s
	Tokens/s	500.02	498.79
Private test set	BLEU	24.81	0
	Inference time 2	237.28	276.66s
	Tokens/s	467.59	466.28

Table 1: Evaluation results on public and private test sets for both translation directions.

MEMBER CONTRIBUTION

Hoang Duy Anh: coding and doing report
Nguyen Khoi Nguyen: coding and training model
Nguyen Tran Phuong Ha: doing report and presentation

REFERENCES

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need" June 12, 2017.