Group 5

Machine Translation Viet - Lao

INT3406_1

Table of contents

01 Introduction

02 Data processing

03 Model structure

04 Training process

05 Evaluate

06 Conclusion

Introduction

Our team is introducing a machine translation model for two low-resource languages Vietnamese and Lao without using any pre-trained models.

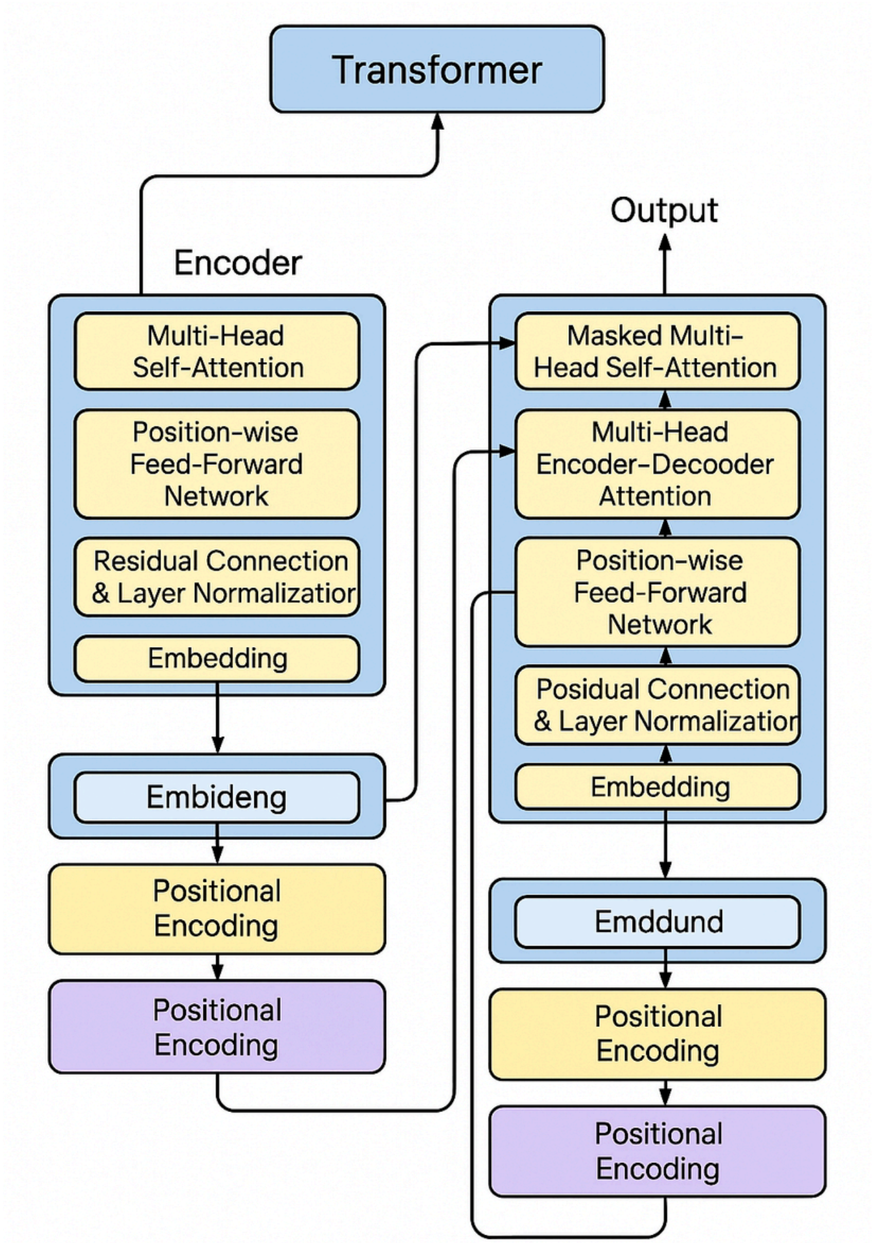
Our key goal is to achieve high accuracy in translation for both Vietnamese to Lao and Lao to Vietnamese directions in BLEU and WER metrics.

Data processing

Out of concern for maintaining the integrity of the specifications, we avoid training data sources having high errors. Data is collected in VLSP2023.zip file and divided into 3 sets for training, validating and testing

We applied diferent methods to train tokenizers for each language: BPE for Vietnamese tokenizer and Unigram for Laos tokenizer.

Model structure



We build the models based on the standard Transformer architecture

- Multi-head Attention is implemented for both Encoder and Decoder, dividing the embedding vector into many "heads" to learn different relationships.
- Embedding Layers: Convert tokens into representation vectors.
- Positional Encoding: Add positional information of tokens so that the model understands the order of sentences.
- Output Layer: Map the output of the Decoder to the probability on the target vocabulary.

Training process

Data

01

From the dataset prepared earlier, we sort the data by the maximum length of sentence pairs to optimize batch processing, before generate DataLoader. DataLoader returns pairs (source, target).

Model

02

The model is transferred to GPU, take parameters:

- Embedding dimension
- Number of heads in Multi-Head Attention.
- Vocabulary: 8000 tokens for both source and target.

Training config

03

We implemented Entropy-Cross Function to compute loss and Adam to optimize the model. We set 30 epoch for training. The best model is defined by the lowest validation loss it can gain.

Training process

```python
# Huấn luyện model
best_val_loss = float('inf')
for epoch in range(EPOCHS):
    print(f"Epoch {epoch+1}/{EPOCHS}")

    # Training phase
    train_loss, train_time = train_epoch(model, train_loader, optimizer, criterion, device)
    print(f"Train Loss: {train_loss:.4f} | Time: {train_time:.2f}s")

    # Validation phase
    val_loss = evaluate(model, val_loader, criterion, device)
    print(f"Validation Loss: {val_loss:.4f}")

    # Lưu model nếu validation loss giảm
    if val_loss < best_val_loss:
        best_val_loss = val_loss
        torch.save(model.state_dict(), 'best_model.pth')
        print("Model saved!")

    print('-' * 50)

print("Training completed!")
```

Training loop over 5 epoch:

- Training phase: Iterates through batches in *train_loader,* calculates loss using Cross-Entropy Loss, performs backpropagation. Finally, updates weights.

- Validation phase: Iterates through *dev_loader*, calculates validation loss and compares validation loss to the previous best.

Evaluate

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

To evaluate the model, we implemented metrics including BLEU score, inference time, and tokens per second (Tokens/s).

VI-LO: The BLEU score of 24.81 on both Public and Private Test sets indicates consistent and stable performance across the two datasets.

Meanwhile, for translation LO-VI, BLEU score is lower and the model failed to o produce accurate or relevant translations for the private test set.

Evaluate

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 is acceptable, but struggles with the Lo-Vi direction.

These results highlight the need for improvements in data quality, training processes, and evaluation methods to ensure robust performance in both directions.

This is the end of our presentation,

THANKS FOR WATCHING!