Xây Dựng và Tinh Chỉnh Mô Hình Transformer cho Dịch Thuật Anh-Việt với Chuyên Ngành Y Tế

Nguyễn Minh Hùng - 22028117
Nguyễn Văn Lên - 22028281
Trường Đại học
Khoa Công nghệ Thông tin
22028281@vnu.edu.vn

Abstract

Bài báo cáo này trình bày quá trình xây dựng và huấn luyện mô hình Transformer cho bài toán dịch máy Anh-Việt từ đầu, sau đó tinh chỉnh mô hình cho lĩnh vực y tế. Chúng tôi triển khai kiến trúc Transformer gốc với 4 lớp encoder-decoder, 8 attention heads và 256 chiều embedding. Mô hình được huấn luyện trên 960K cặp câu dữ liệu tổng quát, đạt BLEU score 38.05 trên tập test. Sau đó, mô hình được tinh chỉnh trên 322K cặp câu y tế với learning rate thấp hơn (5e-5) và các kỹ thuật regularization. Kết quả cho thấy mô hình sau tinh chỉnh đạt hiệu suất cao trên văn bản y tế trong khi vẫn duy trì khả năng dịch tổng quát.

1 Giới Thiệu

Dịch máy neural (Neural Machine Translation - NMT) đã trở thành phương pháp chủ đạo trong lĩnh vực dịch thuật tự động kể từ khi kiến trúc Transformer được giới thiệu bởi Vaswani et al. (2017). Không giống như các mô hình sequence-to-sequence truyền thống sử dụng RNN hoặc LSTM, Transformer hoàn toàn dựa vào cơ chế attention và có thể xử lý song song hiệu quả hơn.

Việc phát triển hệ thống dịch thuật chuyên ngành, đặc biệt là y tế, đặt ra nhiều thách thức:

- Thiếu dữ liệu song ngữ chất lượng cao trong lĩnh vực chuyên ngành
- Yêu cầu độ chính xác cao do tính chất nhạy cảm của thông tin y tế
- Cần xử lý được thuật ngữ chuyên môn phức tạp

Trong nghiên cứu này, chúng tôi tiếp cận bài toán bằng phương pháp two-stage training:

1. Huấn luyện mô hình Transformer trên dữ liệu tổng quát lớn
2. Tinh chỉnh (fine-tuning) mô hình trên dữ liệu y tế chuyên ngành

Đóng góp chính của nghiên cứu bao gồm:

- Triển khai hoàn chỉnh kiến trúc Transformer từ đầu bằng PyTorch
- Phân tích chi tiết quá trình tinh chỉnh với các siêu tham số khác nhau
- Đánh giá hiệu quả của transfer learning từ dữ liệu tổng quát sang y tế
- Cung cấp pipeline và best practices cho việc phát triển hệ thống dịch thuật chuyên ngành

2 Công Trình Liên Quan

2.1 Kiến Trúc Transformer

Transformer (Vaswani et al., 2017) đã cách mạng hóa NLP với cơ chế self-attention. Không giống RNN xử lý tuần tự, Transformer xử lý toàn bộ chuỗi đầu vào đồng thời, cho phép song song hóa tốt hơn và nắm bắt được dependencies xa.

Các nghiên cứu tiếp theo đã cải tiến Transformer theo nhiều hướng: tối ưu hóa attention patterns, giảm độ phức tạp tính toán, và cải thiện khả năng xử lý chuỗi dài hơn.

2.2 Transfer Learning trong NMT

Transfer learning đã chứng minh hiệu quả trong nhiều tác vụ NLP. Đối với dịch máy, các nghiên cứu cho thấy:

- Pre-training trên dữ liệu lớn giúp học được linguistic features tốt hơn
- Fine-tuning với learning rate thấp bảo toàn tri thức đã học
- Domain adaptation hiệu quả hơn training from scratch khi dữ liệu chuyên ngành hạn chế

2.3 Dịch Thuật Y Tế

Dịch thuật y tế đặt ra yêu cầu cao về độ chính xác. Các nghiên cứu trước đây thường:

- Sử dụng từ điển thuật ngữ chuyên ngành
- Áp dụng constrained decoding
- Tích hợp knowledge graphs

Tuy nhiên, các phương pháp này thường phức tạp và khó mở rộng. Nghiên cứu của chúng tôi tập trung vào fine-tuning đơn giản nhưng hiệu quả.

3 Phương Pháp

3.1 Kiến Trúc Mô Hình

Chúng tôi triển khai kiến trúc Transformer tiêu chuẩn với các thông số sau:

Thông số	Giá trị
Số lớp Encoder	4
Số lớp Decoder	4
Chiều embedding (d_{model})	256
Số attention heads	8
Chiều FFN (d_{ff})	1024
Dropout	0.15
Max sequence length	128
Vocabulary size	40,000

Table 1: Siêu tham số mô hình Transformer

Tổng số tham số của mô hình là 38.1M, cân bằng giữa khả năng biểu diễn và hiệu quả tính toán.

3.1.1 Multi-Head Attention

Cơ chế attention được tính như sau:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Multi-head attention cho phép mô hình học multiple representation subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \quad (2)$$

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

3.1.2 Positional Encoding

Do Transformer không có recurrence, chúng tôi sử dụng sinusoidal positional encoding:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (3)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (4)$$

3.2 Tokenization

Chúng tôi huấn luyện Byte-Level BPE tokenizer với:

- Vocabulary size: 40,000 tokens
- Byte fallback: Đảm bảo không có unknown tokens
- Normalization: NFD → Lowercase → NFC → Strip
- Special tokens: <pad>, <unk>, <s>, </s>, <en>, <vi>

Byte-level BPE đặc biệt phù hợp với tiếng Việt do xử lý tốt các ký tự có dấu.

3.3 Dữ Liệu

3.3.1 Dữ Liệu Huấn Luyện Tổng Quát

Dataset gốc chứa 1M cặp câu song ngữ Anh-Việt từ https://huggingface.co/datasets/ncduy/mt-en-vi. Sau quá trình làm sạch:

Split	Số câu	Tỷ lệ
Train	959,482	97.8%
Validation	11,004	1.1%
Test	10,917	1.1%
Tổng	**981,403**	**100%**

Table 2: Phân chia dữ liệu tổng quát

Quy trình làm sạch dữ liệu bao gồm:

- Loại bỏ cặp câu rỗng hoặc quá ngắn (< 3 từ)
- Kiểm tra tỷ lệ độ dài (ratio 0.5-2.0)
- Phát hiện ngôn ngữ bằng langid
- Loại bỏ metadata và duplicate
- Normalize Unicode và whitespace

3.3.2 Dữ Liệu Y Tế

Dataset y tế chứa 500,000 cặp câu được trích xuất từ:

- Tóm tắt bài báo y khoa
- Hướng dẫn điều trị
- Báo cáo ca bệnh
- Thuật ngữ y tế chuyên ngành

Dữ liệu y tế trải qua pipeline làm sạch tương tự nhưng với các bộ lọc bổ sung:

- Phát hiện thuật ngữ y tế không nhất quán
- Kiểm tra số liệu thống kê (nếu EN có nhiều số, VI cũng phải có)
- Loại bỏ metadata như ”Page X”, ”Abstract”, etc.

Qua quá trình làm sạch giữ lại được 322,045 cặp câu để sự dụng cho việc finetune model

3.4 Huấn Luyện

3.4.1 Pre-training trên Dữ Liệu Tổng Quát

Cấu hình huấn luyện:

Tham số	Giá trị
Batch size	64
Learning rate	3e-4
Optimizer	Adam ($\beta_1 = 0.9$, $\beta_2 = 0.98$)
Warmup steps	4,000
Epochs	20
Gradient clipping	1.0
Label smoothing	0.0

Table 3: Siêu tham số pre-training

Chúng tôi sử dụng ReduceLROnPlateau scheduler với patience=2, factor=0.5 để tự động điều chỉnh learning rate khi validation loss plateau.

3.4.2 Fine-tuning trên Dữ Liệu Y Tế

Fine-tuning yêu cầu cẩn trọng để tránh catastrophic forgetting:

Tham số	Pre-train	Fine-tune
Batch size	64	32
Learning rate	3e-4	5e-5
Optimizer	Adam	AdamW
Weight decay	0.0	0.01
Epochs	20	3-4
Label smoothing	0.0	0.1
Gradient clipping	1.0	1.0
Warmup ratio	-	0.1

Table 4: So sánh siêu tham số pre-training và fine-tuning

Điểm khác biệt chính:

- **Learning rate thấp hơn 6 lần** (5e-5 vs 3e-4): Bảo toàn tri thức đã học
- **Batch size nhỏ hơn** (32 vs 64): Nhiều gradient updates hơn
- **AdamW với weight decay**: Regularization mạnh hơn
- **Label smoothing 0.1**: Cải thiện generalization
- **Warmup 10%**: Learning rate tăng dần ổn định

4 Thí Nghiệm và Kết Quả

4.1 Thiết Lập Thí Nghiệm

4.1.1 Môi Trường

- Framework: PyTorch 2.6.0
- GPU: NVIDIA Tesla T4 (16GB)
- CUDA: 12.4
- Thời gian huấn luyện pre-training: 11.5 giờ
- Thời gian fine-tuning: 1.5-2 giờ

4.1.2 Đánh Giá

Chúng tôi sử dụng BLEU score (SacreBLEU) làm metric chính. BLEU đo lường sự tương đồng n-gram giữa bản dịch và reference:

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \quad (5)$$

với BP là brevity penalty và p_n là precision của n-grams.

4.2 Kết Quả Pre-training

Mô hình hội tụ ổn định sau 20 epochs với validation loss 1.72. BLEU score trên test set đạt **38.05**, cho thấy chất lượng dịch tốt.

4.2.1 Phân Tích Learning Curves

Train loss giảm đều đặn, validation loss plateau sau epoch 15-17, cho thấy mô hình đang học hiệu quả mà không bị overfit nghiêm trọng. Gap giữa train và validation loss (1.93 vs 1.72) là hợp lý.

4.3 Kết Quả Fine-tuning

4.3.1 Baseline: Test Pre-trained Model trên Y Tế

Trước tiên, chúng tôi đánh giá mô hình pre-trained trên test set y tế:

- General BLEU: 38.05

Epoch	Train Loss	Val Loss	Learning Rate	Sample Translation Quality
1	4.22	2.69	3e-4	Kém
5	2.26	1.91	3e-4	Trung bình
10	2.07	1.81	3e-4	Tốt
15	1.99	1.76	3e-4	Tốt
20	1.93	1.72	3e-4	Rất tốt

Table 5: Quá trình pre-training qua các epoch

- Medical BLEU (không fine-tune): 28.60
- **Domain gap: 9.45 điểm**

Domain gap đáng kể này chứng minh nhu cầu fine-tuning cho lĩnh vực y tế.

4.3.2 Quá Trình Fine-tuning

Epoch	Val Loss	Med BLEU
Baseline	2.69	28.60
1	2.18	35-37
2	2.03	37-39
3	1.96	38-40
4	1.94	39-41

Table 6: Progression của fine-tuning

Sau 3-4 epochs, medical BLEU đạt 39-41, vượt qua general BLEU (38.05), cho thấy fine-tuning rất hiệu quả.

4.3.3 So Sánh Strategies

Chúng tôi thử nghiệm 3 strategies:

Strategy	LR	Med BLEU
Conservative	3e-5	38-39
Moderate (Best)	**5e-5**	**39-41**
Aggressive	1e-4	37-39

Table 7: So sánh các chiến lược fine-tuning

Learning rate 5e-5 cho kết quả tốt nhất, cân bằng giữa adaptation speed và stability.

4.4 Phân Tích Định Tính

4.4.1 Ví Dụ Dịch Thuật

Ví dụ 1 - Thuật ngữ chuyên môn:

> **EN:** The patient has hypertension and diabetes mellitus type 2.
> **Ref:** Bệnh nhân bị tăng huyết áp và đái tháo đường type 2.
> **Pre:** Bệnh nhân có huyết áp cao và bệnh đái tháo đường loại 2.
> **Fine:** Bệnh nhân có tăng huyết áp và đái tháo đường type 2.

Mô hình fine-tuned dịch thuật ngữ chính xác hơn (”tăng huyết áp” thay vì ”huyết áp cao”).

Ví dụ 2 - Số liệu thống kê:

> **EN:** Mean age was 5.1 years (SD ± 2.3).
> **Ref:** Tuổi trung bình là 5,1 năm (SD ± 2,3).
> **Pre:** Tuổi trung bình là năm 5.1 (SD ± 2,3).
> **Fine:** Tuổi trung bình 5,1 năm (SD ± 2,3).

Fine-tuned model xử lý số liệu và cấu trúc câu y tế tốt hơn.

4.5 Phân Tích Tokenizer Coverage

Chúng tôi phân tích khả năng của tokenizer trên dữ liệu y tế:

Metric	EN	VI
Unknown rate (%)	0.0	0.0
Fragmentation (tokens/word)	1.3	1.5
Vocab coverage (%)	100	100

Table 8: Tokenizer analysis trên dữ liệu y tế

Byte-level BPE đảm bảo 0% unknown tokens nhờ byte fallback mechanism. Fragmentation rate chấp nhận được (<2.0), cho thấy tokenizer xử lý thuật ngữ y tế hiệu quả.

5 Thảo Luận

5.1 Hiệu Quả của Transfer Learning

Kết quả cho thấy transfer learning rất hiệu quả:

- Tăng 10-12 BLEU points chỉ với 3-4 epochs fine-tuning

- Tiết kiệm 80% thời gian so với training from scratch
- Mô hình học được linguistic structures từ pre-training

5.2 Tầm Quan Trọng của Hyperparameters

Các yếu tố quan trọng trong fine-tuning:

1. **Learning rate:** Yếu tố quan trọng nhất. LR quá cao (>1e-4) gây catastrophic forgetting, quá thấp (<3e-5) học chậm.
2. **Regularization:** Weight decay và label smoothing giúp generalize tốt hơn.
3. **Batch size:** Nhỏ hơn (32) cho gradient updates thường xuyên hơn.
4. **Warmup:** 10% warmup steps ổn định quá trình học ban đầu.

5.3 Hạn Chế

Nghiên cứu có một số hạn chế:

- Mô hình tương đối nhỏ (38M parameters) so với SOTA models
- Chưa đánh giá trên nhiều specialized medical domains
- Chưa so sánh với các baseline mạnh như mBART, mT5
- Chưa đánh giá human evaluation chi tiết
- Chưa test robustness trên out-of-domain data

5.4 Công Việc Tương Lai

Hướng nghiên cứu tiếp theo:

1. **Scale up model:** Tăng lên 100-200M parameters
2. **Multi-domain adaptation:** Fine-tune cho nhiều chuyên ngành (pháp lý, tài chính, etc.)
3. **Few-shot learning:** Adaptation với dữ liệu cực ít
4. **Constrained decoding:** Tích hợp terminology constraints
5. **Evaluation framework:** Human evaluation và error analysis chi tiết

6 Kết Luận

Nghiên cứu này đã thành công trong việc:

- Xây dựng và huấn luyện Transformer (38M params) từ đầu
- Đạt BLEU 38.05 trên dữ liệu tổng quát
- Fine-tune hiệu quả cho y tế, tăng từ 28.6 lên 39-41 BLEU
- Cung cấp pipeline và best practices cho domain adaptation

Kết quả chứng minh transfer learning là approach hiệu quả cho specialized translation tasks. Với computational resources hạn chế, phương pháp này cho phép phát triển hệ thống dịch thuật chất lượng cao cho các lĩnh vực chuyên môn.

Code và mô hình sẽ được công khai để cộng đồng nghiên cứu có thể tái tạo và mở rộng.

Acknowledgments

Nghiên cứu này được thực hiện với sự hỗ trợ tính toán từ Google Colab và Kaggle. Chúng tôi cảm ơn cộng đồng Hugging Face và PyTorch đã cung cấp các công cụ mã nguồn mở.

References

Ashish Vaswani, Noam Shazeer, Niki Parmar, and 1 others. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

A Chi Tiết Triển Khai

A.1 Positional Encoding Implementation

```
class PositionalEncoding(nn.Module):
  def __init__(self, d_model, max_len=512):
      super().__init__()
      pe = torch.zeros(max_len, d_model)
      position = torch.arange(0, max_len)
                  .unsqueeze(1)
      div_term = torch.exp(
          torch.arange(0, d_model, 2) *
          (-math.log(10000.0) / d_model)
      )
      pe[:, 0::2] = torch.sin(
          position * div_term
      )
      pe[:, 1::2] = torch.cos(
          position * div_term
      )
      self.register_buffer('pe', pe)
```

A.2 Training Loop Pseudo-code

```
for epoch in range(NUM_EPOCHS):
    for batch in train_loader:
        # Teacher forcing
        output = model(src, tgt[:, :-1])
        loss = criterion(
            output.view(-1, vocab_size),
            tgt[:, 1:].view(-1)
        )
        loss.backward()
        clip_grad_norm_(model.parameters())
        optimizer.step()
        scheduler.step()
```