Attention Is All You Need (Transformer)

(https://arxiv.org/pdf/1706.03762)

Abstract: The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

I. Introduction

A. Bối cảnh trước đó: Trước Transformer, các bài toán xử lý chuỗi như dịch máy hay mô hình ngôn ngữ chủ yếu dựa vào RNN, đặc biệt là LSTM và GRU. RNN tính toán theo từng vị trí của chuỗi. Ở mỗi bước t, trạng thái ẩn h_t phụ thuộc vào h_{t-1} và đầu vào tại vị trí t.

Do đó, việc tính toán bị ràng buộc theo thứ tự thời gian. Không thể song song hóa các bước trong cùng một chuỗi. Tác giả nhấn mạnh rằng dù đã có các cải tiến như factorization tricks hoặc conditional computation giúp tăng hiệu quả, nhưng ràng buộc tuần tự vẫn tồn tại về mặt bản chất.

→ Cách xử lý tuần tự này khiến việc tính toán khó song song hóa. Khi chuỗi dài, việc huấn luyện trở nên chậm và tốn tài nguyên, dù đã có nhiều cải tiến để tăng hiệu quả.

B. Vai trò của attention:

- **Attention** đã trở thành thành phần quan trọng trong các mô hình sequence modeling vì nó cho phép mô hình hóa phụ thuộc giữa các vị trí mà không quan tâm đến khoảng cách.
- Tuy nhiên, trong hầu hết các công trình, attention vẫn được sử dụng cùng với RNN. Nó **không thay thế phần tuần tự, mà chỉ hỗ trợ**.

C. Đề xuất Transformer

Từ các quan sát trên, tác giả đề xuất **Transformer**:

- Loại bỏ hoàn toàn recurrence
- Không sử dụng convolution
- Dựa hoàn toàn vào attention để học quan hệ toàn cục giữa input và output

Hệ quả là:

- Tăng mức độ song song hóa đáng kể
- Huấn luyện nhanh hơn
- Vẫn đạt hoặc vượt state-of-the-art trong dịch máy

D. Bằng chứng thực nghiệm và tổng hợp:

Những kết quả thu nghiệm được:

- **28.4 BLEU trên WMT 2014 English–German**, vượt hơn 2 BLEU so với kết quả tốt nhất trước đó
- **41.8 BLEU trên English–French**, đạt state-of-the-art cho single model
- Thời gian huấn luyện **chỉ 3.5 ngày trên 8 GPU**
- Mô hình còn **tổng quát tốt sang bài toán parsing**

→ Tóm lại, vấn đề cốt lõi của RNN là tính tuần tự, attention đã cho thấy tiềm năng, và Transformer là bước tiến khi xây dựng toàn bộ kiến trúc chỉ dựa trên attention để đạt hiệu quả tính toán và chất lượng vượt trội.

II. Background

A. Hướng tiếp cận giảm tính tuần tự trước Transformer

Một số mô hình như Extended Neural GPU, ByteNet và ConvS2S đã tìm cách giảm tính tuần tự bằng cách dùng **CNN** thay cho RNN. Nhờ convolution, các vị trí trong chuỗi có thể được tính toán song song.

Tuy nhiên, để liên hệ hai vị trí cách xa nhau:

- ConvS2S cần số bước tăng tuyến tính theo khoảng cách
- ByteNet cần số bước tăng theo log khoảng cách
  → **Điều này khiến việc học phụ thuộc xa trở nên khó hơn.**

⇒ Transformer giải quyết vấn đề này của CNN và cả RNN bằng self-attention, nơi mọi cặp vị trí có thể tương tác chỉ với số bước hằng số. Đổi lại, có hiện tượng giảm độ phân giải do cơ chế trung bình theo trọng số attention. Tác giả khắc phục bằng **Multi-Head Attention.**

B. Self-attention là gì và đã được dùng ở đâu

Self-attention, còn gọi là intra-attention, là cơ chế cho phép các vị trí trong cùng một chuỗi tương tác để tạo ra biểu diễn của chuỗi đó.

Cơ chế này đã được dùng thành công trong nhiều tác vụ như đọc hiểu, tóm tắt trừu tượng, textual entailment và học biểu diễn câu độc lập với tác vụ.

C. **Memory network và attention lặp:** End-to-end memory networks sử dụng cơ chế attention lặp thay vì recurrence theo vị trí chuỗi. Chúng đạt kết quả tốt trên các bài toán hỏi đáp đơn giản và language modeling.

D. **Điểm mới của Transformer:** Transformer là **mô hình transduction đầu tiên hoàn toàn dựa vào self-attention** để tính biểu diễn cho cả input và output, không dùng RNN theo chuỗi cũng không dùng convolution.

III. Model Architecture

Encoder–Decoder Structure

Các mô hình sequence transduction cạnh tranh nhất sử dụng cấu trúc encoder–decoder. Encoder ánh xạ chuỗi đầu vào dạng ký hiệu $(x_1,..., x_n)$ thành chuỗi biểu diễn liên tục $(z_1,..., z_n)$ Từ z, decoder sinh chuỗi đầu ra $(y_1,..., y_n)$ từng phần tử một. Tại mỗi bước, mô hình là auto-regressive, tức là sử dụng các ký hiệu đã sinh trước đó làm input bổ sung khi dự đoán ký hiệu tiếp theo.

Transformer giữ nguyên khung này nhưng thay thế toàn bộ thành phần bên trong bằng self-attention và các lớp fully connected theo vị trí.

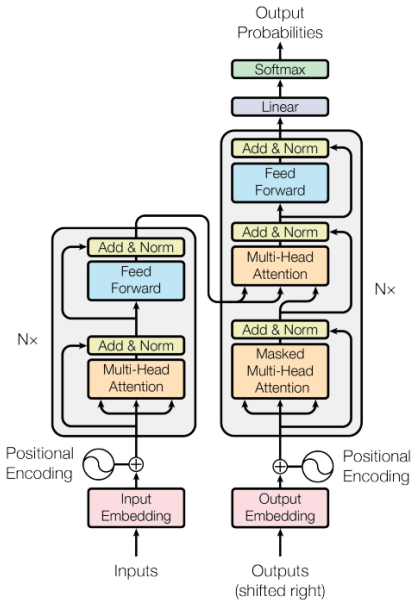

Figure 1: The Transformer - model architecture.

A. Encoder and Decoder Stacks

Encoder
Encoder gồm một stack gồm N=6 lớp giống hệt nhau.
Mỗi lớp gồm hai sub-layer:

1. Multi-head self-attention
2. Position-wise fully connected feed-forward network

Mỗi sub-layer được bao quanh bởi residual connection, sau đó là layer normalization. Công thức: $LayerNorm(x + Sublayer(x))$

Để residual connection hoạt động thuận lợi, tất cả sub-layer và embedding layer đều cho output cùng kích thước: $d_{model} = 512$

Decoder

Decoder cũng gồm một stack N=6 lớp giống hệt nhau.
Ngoài hai sub-layer giống encoder, decode**r thêm một sub-layer thứ ba**:

- Multi-head attention trên output của encoder stack

Tương tự encoder, mỗi sub-layer đều có residual connection và layer normalization.

Self-attention trong decoder được điều chỉnh để ngăn mỗi vị trí attend tới các vị trí phía sau. Việc masking này, kết hợp với việc dịch output embeddings lệch một vị trí, đảm bảo rằng dự đoán tại vị trí iii chỉ phụ thuộc vào các output đã biết tại các vị trí nhỏ hơn i.

B. Attention: Một attention function ánh xạ một query và một tập các cặp key–value sang một output. Query, keys, values và output đều là vector. Output được tính như weighted sum của values, trong đó trọng số được tính bởi compatibility function giữa query và key tương ứng.

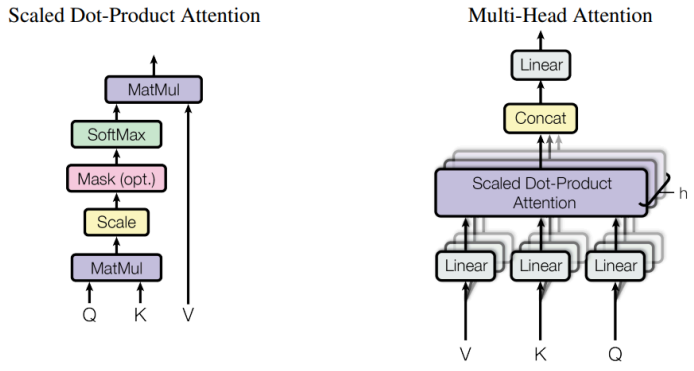

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Scaled Dot-Product Attention

Input gồm:

- Queries và Keys có kích thước d_k
- Values có kích thước d_v

Dot product giữa query và toàn bộ keys được tính, sau đó chia cho $\sqrt{d_k}$, rồi áp dụng softmax để lấy trọng số trên values.

Khi tính đồng thời nhiều query, gom thành ma trận Q, keys và values thành K và V. Khi đó:

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

So với additive attention và dot-product attention thông thường, phiên bản này thêm hệ số scale $\frac{1}{\sqrt{d_k}}$

Giả sử các thành phần của q và kđộc lập, có trung bình 0 và phương sai 1. Khi đó:

$$q \cdot k = \sum_{i=1}^{d_k} q_i k_i$$

có trung bình 0 và phương sai d_k. Khi d_k lớn, dot product có độ lớn lớn, đẩy softmax vào vùng gradient nhỏ. Do đó cần scale bởi $\frac{1}{\sqrt{d_k}}$

Multi-Head Attention

Thay vì thực hiện một attention duy nhất với keys, values và queries kích thước d_{model}, mô hình chiếu tuyến tính chúng h lần với các ma trận khác nhau, sang các không gian con kích thước d_k,d_k,d_v.

Trên mỗi phiên bản chiếu này, attention được thực hiện song song, cho output kích thước d_v. Các output này được nối lại và chiếu tuyến tính lần nữa.

Công thức:

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O$$

Với:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Trong đó:

$$W_i^Q \in R^{d_{model} \, x \, d_k}$$
$$W_i^K \in R^{d_{model} \, x \, d_k}$$
$$W_i^V \in R^{d_{model} \, x \, d_v}$$
$$W^O \in R^{hd_v \, x \, d_{model}}$$

Trong paper:

$$h = 8$$
$$d_k = d_v = \frac{d_{model}}{h} = 64$$

⇒ Multi-head attention cho phép mô hình attend đồng thời tới thông tin từ các representation subspaces khác nhau tại các vị trí khác nhau. Với một head duy nhất, việc averaging sẽ hạn chế điều này.

Applications of Attention in our Model

Transformer dùng multi-head attention theo ba cách:

1. **Encoder–Decoder Attention**
   Query từ decoder layer trước đó. Keys và values từ output của encoder. Mỗi vị trí decoder attend tới toàn bộ input sequence.

2. **Encoder Self-Attention**
   Keys, values và queries đều từ output của layer encoder trước đó. Mỗi vị trí encoder attend tới mọi vị trí khác trong layer trước.

3. **Decoder Self-Attention có Mask**
   Mỗi vị trí decoder attend tới các vị trí trong decoder tới và bao gồm chính nó.
   Để giữ tính auto-regressive, các kết nối không hợp lệ được mask bằng cách đặt giá trị tương ứng thành −∞ trước khi softmax.

C. Position-wise Feed-Forward Networks

Ngoài attention, mỗi layer có một fully connected feed-forward network áp dụng độc lập và giống nhau cho từng vị trí:

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Các linear transformation giống nhau giữa các vị trí trong cùng một layer, nhưng khác nhau giữa các layer.
Kích thước:

$$d_{model} = 512$$
$$d_{ff} = 2048$$

⇒ Có thể xem như hai convolution với kernel size 1.

D. Embeddings and Softmax

Input và output tokens được chuyển thành vector kích thước d_{model} bằng learned embeddings.

Decoder output được đưa qua learned linear transformation và softmax để dự đoán xác suất next token.

Transformer chia sẻ cùng một ma trận trọng số giữa:

- Hai embedding layers
- Lớp linear trước softmax

Trong embedding layers, các trọng số được nhân với $\sqrt{d_{model}}$

E. Positional Encoding

Do không có recurrence và convolution, mô hình cần thông tin thứ tự. Positional encodings được cộng vào input embeddings tại đáy encoder và decoder stacks. Positional encoding có cùng kích thước d_{model}.

Paper sử dụng hàm sin và cos:

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$
$$PE(pos, 2i + 1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}})$$

Trong đó pos là vị trí và i là chiều. Mỗi chiều của positional encoding là một sinusoid với bước sóng tạo thành cấp số nhân từ 2π đến 10000.2π

Lý do lựa chọn dạng này là vì với offset cố định k, PE_{pos+k} có thể được biểu diễn như một hàm tuyến tính của PE_{pos}. Điều này được giả thuyết giúp mô hình dễ học quan hệ vị trí tương đối.

Paper cũng thử learned positional embeddings và cho kết quả gần như tương đương, nhưng chọn sinusoidal encoding vì có thể giúp mô hình tổng quát tới độ dài chuỗi lớn hơn so với lúc huấn luyện.

IV. Why Self-Attention

Trong phần này, tác giả so sánh self-attention với recurrent layers và convolutional layers khi ánh xạ một chuỗi biến độ dài $(x_1,..., x_n)$ sang một chuỗi cùng độ dài $(z_1,..., z_n)$, với $x_i, z_i \in R^d$, như trong các hidden layer của encoder hoặc decoder điển hình.

Ba tiêu chí được xem xét:

1. Tổng độ phức tạp tính toán trên mỗi layer
2. Mức độ song song hóa, đo bằng số bước tuần tự tối thiểu
3. Độ dài đường truyền giữa các phụ thuộc xa trong mạng

Học phụ thuộc xa là thách thức trọng yếu trong sequence transduction. Một yếu tố quan trọng ảnh hưởng đến khả năng học các phụ thuộc này là độ dài đường mà tín hiệu forward và backward phải đi qua. Đường truyền càng ngắn giữa các vị trí trong chuỗi input và output, việc học phụ thuộc xa càng dễ.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

A. Độ phức tạp tính toán mỗi layer

Theo Table 1:

- **Self-Attention:** $O(n^2 \cdot d)$
- **Recurrent:** $O(n \cdot d^2)$

Self-attention nhanh hơn recurrent khi $n < d$. Tác giả nói đây thường là trường hợp trong MT hiện đại với word-piece và byte-pair.

Với chuỗi rất dài, có thể giới hạn self-attention trong vùng lân cận kích thước

r. Khi đó:

- **Complexity**: $O(r \cdot n \cdot d)$
- **Maximum path length tăng lên**: $O(n/r)$

B. Mức độ song song hóa

Self-attention kết nối mọi vị trí với số bước tuần tự: $O(1)$
Trong khi recurrent cần: $O(n)$
Nghĩa là self-attention song song hóa tốt hơn rõ rệt.

C. Đường truyền cho phụ thuộc xa

Học phụ thuộc xa khó khi tín hiệu phải đi qua đường truyền dài. Tác giả so sánh maximum path length:

- **Self-Attention**: $O(1)$
- **Recurrent**: $O(1)$
- **Convolutional**: $O(log_k(n))$

Với convolution $k < n$, một lớp không nối được mọi cặp vị trí. Muốn nối hết cần chồng nhiều lớp: $O(n/k)$ (contiguous) hoặc $O(log_k(n))$(dilated), làm đường truyền dài hơn.

D. So sánh với Convolution

Convolutional layers thường đắt hơn recurrent layers một hệ số k

Separable convolutions làm giảm độ phức tạp xuống: $O(k \cdot n \cdot d + n \cdot d^2)$
Ngay cả khi $k = n$, độ phức tạp của separable convolution bằng với tổng của:

- Một self-attention layer
- Một point-wise feed-forward layer

→ Đây chính là cấu trúc được dùng trong Transformer.

E. Khả năng diễn giải

Một lợi ích phụ của self-attention là tính diễn giải cao hơn. Quan sát phân bố attention cho thấy:

- Các attention head khác nhau học các chức năng khác nhau
- Nhiều head thể hiện hành vi liên quan đến cấu trúc cú pháp và ngữ nghĩa của câu

Điều này được trình bày thêm trong appendix của paper.

Kết luận: Self-attention được lựa chọn vì:

- Yêu cầu số bước tuần tự tối thiểu
- Có lợi thế tính toán khi n<dn < dn<d

- Có maximum path length nhỏ nhất
- Cho phép học phụ thuộc xa hiệu quả hơn

⇒ Đây là cơ sở lý thuyết chính để t**hay thế recurrent và convolutional layers bằng self-attention trong Transformer.**

V. Training

A. Training Data and Batching

English–German (WMT14)

- 4.5M câu
- Byte-Pair Encoding
- Vocabulary chung source–target: ~37K tokens

English–French (WMT14)

- 36M câu
- Word-piece vocabulary: 32K tokens

Batching:

- Gom câu theo độ dài xấp xỉ nhau
- Mỗi batch chứa khoảng:
    - 25K source tokens
    - 25K target tokens

→ Cách batch này giúp tận dụng GPU tốt hơn.

B. Hardware and Schedule

Huấn luyện trên:

- 1 máy
- 8 NVIDIA P100 GPUs

Base model

- 0.4 giây / step
- 100K steps
- 12 giờ

Big model

- 1.0 giây / step
- 300K steps
- 3.5 ngày

→ Transformer đạt kết quả mạnh với thời gian huấn luyện thấp hơn đáng kể so với các mô hình trước đó.

C. Optimizer

Dùng Adam với:

$$\beta_1 = 0.9$$

$$\beta_1 = 0.98$$

$$\beta_1 = 10^{-9}$$

Learning rate thay đổi theo công thức:

$$lrate = d_{model}^{-0.5} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

Với: $warmup\_steps = 4000$

Ý nghĩa:

- 4000 bước đầu: learning rate tăng tuyến tính
- Sau đó giảm theo $1/\sqrt{step}$

  Đây là learning rate schedule đặc trưng của Transformer.

D. Regularization

Trong quá trình huấn luyện, mô hình sử dụng ba kỹ thuật regularization sau:

- **Residual Dropout**

Dropout được áp dụng:

- Lên **output của mỗi sub-layer**, trước khi cộng với input qua residual connection và trước khi thực hiện layer normalization
- Lên **tổng của embedding và positional encoding** ở cả encoder và decoder

Với base model:

$$P_{drop} = 0.1$$

→ Mục đích là giảm overfitting và ổn định quá trình huấn luyện trong kiến trúc nhiều lớp.

- **Attention Dropout**

  Dropout được áp dụng lên attention weights trong cơ chế scaled dot-product attention, tức là sau khi tính softmax của ma trận attention.

  Điều này giúp tránh việc mô hình phụ thuộc quá mạnh vào một số kết nối attention cụ thể.

- **Label Smoothing**

  Trong huấn luyện, sử dụng label smoothing với hệ số:

$$\epsilon_{ls} = 0.1$$

  Thay vì dùng nhãn one-hot cứng, phân phối mục tiêu được làm “mềm” hơn.

  Kết quả là:

  - Perplexity có thể tăng nhẹ do mô hình ít chắc chắn hơn
  - Tuy nhiên accuracy và BLEU score được cải thiện

  → Label smoothing giúp mô hình tổng quát tốt hơn và giảm overconfidence.

VI. Results

A. Machine Translation

Trên bài toán **WMT 2014 English-to-German**, Transformer (big) đạt **BLEU = 28.4**, vượt hơn 2.0 BLEU so với tất cả các mô hình trước đó, kể cả ensemble, thiết lập state-of-the-art mới. Mô hình được huấn luyện trong 3.5 ngày trên 8 GPU P100. Ngay cả Transformer (base) cũng vượt toàn bộ các mô hình đã công bố, trong khi chi phí huấn luyện chỉ bằng một phần nhỏ.

- English–German (WMT14)
    + Transformer (big): BLEU = 28.4
    + Huấn luyện: 3.5 ngày, 8 P100
    + Base model cũng vượt mọi mô hình trước đó

Trên **WMT 2014 English-to-French**, Transformer (big) đạt **BLEU = 41.8**, vượt tất cả các single model trước đó với chi phí huấn luyện nhỏ hơn 1/4 so với state-of-the-art cũ. Với English–French, mô hình big dùng

$$P_{drop} = 0.1$$

thay vì 0.3.

- English–French (WMT14)
    + Transformer (big): BLEU = 41.8
    + Chi phí huấn luyện < 1/4 SOTA trước đó
    + Dropout: P_{drop} = 0.1

Đối với base model, kết quả được lấy bằng cách trung bình 5 checkpoint cuối cùng, ghi cách nhau 10 phút. Với big model, trung bình 20 checkpoint cuối. Khi suy luận, sử dụng beam search với beam size = 4 và length penalty

$$\alpha = 0.6$$

Độ dài output tối đa đặt bằng input length + 50 và dừng sớm khi có thể.

Chi phí huấn luyện được ước lượng bằng cách nhân thời gian huấn luyện, số GPU và năng lực tính toán dấu chấm động đơn chính xác duy trì của mỗi GPU. Bảng 2 cho thấy Transformer đạt BLEU cao hơn các kiến trúc trước đó trong khi FLOPs thấp hơn đáng kể.

B. Model Variations

Để đánh giá vai trò của từng thành phần, tác giả thay đổi base model và đo hiệu năng trên English-to-German newstest2013, không dùng checkpoint averaging.

- Ở nhóm (A), thay đổi số attention heads và kích thước d_k,d_v trong khi giữ nguyên tổng chi phí tính toán. Single-head attention kém hơn cấu hình tốt nhất 0.9 BLEU. Tuy nhiên, khi số head quá lớn, chất lượng cũng giảm.
- Ở nhóm (B), giảm kích thước key d_k làm giảm chất lượng mô hình. Điều này cho thấy việc xác định độ tương thích không đơn giản, và dot product có thể chưa phải là hàm tương thích tối ưu.
- Ở nhóm (C) và (D), khi tăng kích thước mô hình, chất lượng cải thiện như kỳ vọng. Đồng thời, dropout đóng vai trò quan trọng trong việc tránh overfitting.

- Ở nhóm (E), thay positional encoding dạng sinusoidal bằng learned positional embeddings cho kết quả gần như tương đương base model.

C. **English Constituency Parsing**

Để kiểm tra khả năng tổng quát hóa, Transformer được áp dụng cho bài toán English constituency parsing, nơi output dài hơn input và chịu ràng buộc cấu trúc mạnh. Trước đó, RNN sequence-to-sequence không đạt state-of-the-art trong chế độ ít dữ liệu.

Tác giả huấn luyện một Transformer 4 lớp với $d_{model} = 1024$ ở chế độ semi-supervised với thêm khoảng 17M câu. Vocabulary gồm 16K token cho WSJ-only và 32K cho semi-supervised.

Chỉ tinh chỉnh một số ít siêu tham số như dropout, learning rate và beam size trên tập phát triển; các tham số khác giữ nguyên từ base translation model. Khi suy luận, đặt độ dài output tối đa bằng input length + 300, beam size = 21 và α=0.3

Kết quả trên Section 23 của WSJ cho thấy Transformer đạt F1 = 92.7 trong thiết lập semi-supervised, vượt tất cả các mô hình đã công bố trước đó ngoại trừ Recurrent Neural Network Grammar. Ngay cả khi chỉ huấn luyện trên 40K câu WSJ, Transformer vẫn vượt BerkeleyParser, cho thấy khả năng tổng quát hóa mạnh mẽ sang tác vụ ngoài dịch máy.