ỨNG DỤNG TRANSFORMER TRONG NHIỆM VỤ DỊCH MÁY NGÔN NGỮ ANH-VIỆT

Lý Gia Bảo[1], Trần Thanh Phương[2]

[1]*Trường ĐH Kinh tế-Tài chính TPHCM, baolg220@uef.edu.vn*

[2]*Trường ĐH Kinh tế-Tài chính TPHCM, phuongtt20@uef.edu.vn*

***Tóm tắt**: Bài báo này tập trung vào việc áp dụng mô hình Transformer, một trong những tiến bộ quan trọng trong lĩnh vực học sâu và dịch máy, vào nhiệm vụ dịch máy ngôn ngữ Anh-Việt. Mô hình Transformer đã chứng minh khả năng hiệu quả trong việc xử lý chuỗi dữ liệu dài và phức tạp thông qua cơ chế attention, giúp cải thiện chất lượng dịch với tốc độ xử lý nhanh hơn. Bằng cách tập trung vào việc cải thiện hiệu suất và chất lượng của mô hình dịch máy, nghiên cứu này đưa ra các đánh giá và so sánh việc áp dụng các kỹ thuật khác nhau vào mô hình Transformer cho nhiệm vụ phiên dịch. Kết quả cho thấy sự ảnh hưởng của từng kỹ thuật trên là khác nhau đối với của việc áp dụng Transformer trong dịch máy, đồng thời mở ra cơ hội nghiên cứu và phát triển tiếp theo trong lĩnh vực này.*

***Từ khóa**: Transformer, dịch máy, Cross Entropy Loss, tokenization, encoder-decoder, beamsearch.*

1. Giới thiệu chung

1.1. Mô hình Transformer:

Trong những năm gần đây, lĩnh vực dịch máy đã phát triển đáng kể nhờ sự ra đời của các mô hình học máy tiên tiến, đặc biệt là khi mô hình Transformer, một kiến trúc không sử dụng tính tuần hoàn mà dựa hoàn toàn vào cơ chế attention, đã xuất hiện và tạo ra sự chú ý lớn trong cộng đồng nghiên cứu và ứng dụng. Các mô hình truyền thống như mạng nơ-ron tuần hoàn (RNNs) với long short-term memory (LSTM) hay gated recurrent neural networks (GRUs) đã chiếm vị thế hàng đầu từ lâu trong việc mô hình hóa chuỗi dữ liệu và giải quyết các vấn đề dịch máy. Tuy nhiên, điểm hạn chế lớn của chúng là tính tuần tự trong việc tính toán, điều này đã giới hạn khả năng xử lý song song trong quá trình huấn luyện, đặc biệt khi đối mặt với việc xử lý các chuỗi dữ liệu dài hoặc phức tạp.

Transformer là một kiến trúc mạng nơ-ron được giới thiệu bởi Vaswani và các cộng sự vào năm 2017 nhằm tránh sự lặp lại (hồi quy). Thay vào đó, Transformer dựa hoàn toàn vào cơ chế tự chú ý để thu hút sự phụ thuộc tổng thể giữa input và output. Transformer đặc biệt phổ biến trong lĩnh vực xử lý ngôn ngữ tự nhiên và thị giác máy tính. Nó có những cải tiến đáng kể trong hiệu suất của các mô hình dựa trên mạng nơ-ron trong nhiều ứng dụng. Transformer ban đầu được tạo nên chủ yếu tập trung vào việc xử lý chuỗi dữ liệu như văn bản, âm thanh hoặc chuỗi thời gian. Cách tiếp cận này không sử dụng các kiến trúc mạng nơ-ron thông thường như LSTM hay GRU mà thay vào đó, sử dụng cơ chế chính là tự chú ý (self-attention).

Bài báo này đặt trọng điểm vào việc khám phá và áp dụng Transformer trong nhiệm vụ dịch máy từ tiếng Anh sang tiếng Việt, đồng thời đưa ra đánh giá, so sánh về các kỹ thuật liên quan hỗ trợ cho nhiệm vụ dịch máy. Các so sánh đó bao gồm: việc lựa chọn các hàm loss khác nhau cho

mô hình giữa Cross Entropy Loss và Label Smoothing Loss; kỹ thuật Beamsearch hay Greedy Decoder Search và cách lựa chọn quá trình token như thế nào, theo ký tự hay theo từ. Kết quả cho thấy sự ảnh hưởng đến hiệu suất và độ chính xác của từng kỹ thuật trên là khác nhau đối với mô hình Transformer trong dịch máy. Việc sử dụng mô hình này và các kỹ thuật phù hợp sẽ giúp cái thiện được hiệu suất cũng như độ chính xác trong nhiệm vụ phức tạp này.

1.2. Ứng dụng và thành công của Transformer trong nhiệm vụ dịch máy:

Transformer đã đem lại những cải tiến đáng kể trong nhiệm vụ dịch máy do khả năng xử lý các chuỗi dữ liệu phức tạp. Transformer ưu việt hơn là do những nguyên nhân sau:

- Transformer đã thay đổi cách tiếp cận trong dịch máy bằng cách áp dụng kiến trúc encoder-decoder dựa trên tự chú ý. Mô hình Transformer có thể học các mô hình ngôn ngữ phức tạp và tạo ra các dự đoán dịch chính xác hơn cho các cặp ngôn ngữ khác nhau.
- Tính linh hoạt trong đa ngôn ngữ: Transformer cho phép huấn luyện mô hình cho nhiều ngôn ngữ song song một cách hiệu quả. Điều này giúp tạo ra các mô hình dịch có thể xử lý nhiều ngôn ngữ, từ các cặp ngôn ngữ phổ biến đến các ngôn ngữ ít phổ biến hơn.
- Kiến trúc tự chú ý trong Transformer giúp mô hình tập trung vào các phần quan trọng của câu trong quá trình dịch, từ đó cải thiện hiệu suất và chất lượng của bản dịch.
- Transformer có khả năng học được cấu trúc ngữ pháp của các ngôn ngữ mục tiêu và nguồn, không chỉ dịch từng từ một mà còn hiểu được cấu trúc và ngữ cảnh của câu.

Trong nhiệm vụ dịch thuật từ Tiếng Anh sang Tiếng Pháp trên tập dữ liệu WMT 2014 English-to-French, mô hình Transformer (big) đã đạt được những kết quả vượt trội hơn hẳn những mô hình trước đó. Mô hình đạt điểm BLEU 41.0 vượt qua tất cả các mô hình đơn lẻ được công bố với chi phí huấn luyện chỉ bằng ¼ so với mô hình tốt nhất trước đó. Trong nhiệm vụ dịch thuật từ Tiếng Anh sang Tiếng Đức trên tập dữ liệu WMT 2014 English-to-German, mô hình Transformer (big) vượt qua tất cả các mô hình tốt nhất được báo cáo trước đó (bao gồm cả những mô hình kết hợp) với BLEU cải thiện hơn 2.0 điểm và đạt được điểm mới là 28.4 cho bài toán dịch máy. Dù cho mô hình được sử dụng là mô hình Transformer cơ sở cũng vượt qua hết tất cả các mô hình trước đó.

2. Cơ sở lý thuyết

2.1. Mô hình Transformer

2.1.1 Cấu trúc mô hình Transformer:

Transformer tuân theo kiến trúc tổng thể như hình bên bằng cách sử dụng các lớp self-attention (tự chú ý) được xếp chồng lên nhau và các lớp fully-connected, point-wise cho cả bộ mã hóa (encoder) và bộ giải mã (decoder). Sơ đồ tổng quan về kiến trúc tổng thể của Transformer:

- ***Embedding Layer:*** Đầu vào input được biểu diễn bằng các vector embedding. Trong mô hình này, các tokens được biểu diễn dưới dạng vector trong không gian nhiều chiều để có thể được xử lý bởi mạng nơ-ron.
- ***Encoder:*** Cấu trúc gồm 6 lớp (N=6) encoder tương tự được xếp chồng lên nhau. Trong đó, mỗi lớp gồm 2 cơ chế phụ (module) là Multi-head self-attention được sử dụng để tập trung vào các phần quan trọng của inputs và Position-wise fully-connectd feed-forward network, một mạng kết nối đầy đủ đơn giản được áp dụng ngay sau cơ chế self-attention. Lớp mạng nơ-ron truyền thẳng giúp cho mô hình học được các mối quan hệ phi tuyến tính trong dữ liệu.

Xung quanh mỗi cơ chế đều sử dụng kỹ thuật kết nối theo phần dư (residual connection) kết hợp với chuẩn hóa lớp (layer normalization).

$$LayerNorm(x + Sublayer(x))$$

Đầu ra của mỗi lớp phụ có kích thước là 512 chiều (dmodel = 512)

- ***Decoder:*** Tương tự như encoder. Cấu trúc gồm 6 lớp decoder tương tự được xếp chồng lên nhau. Ngoài 2 cơ chế phụ tương đồng như trong lớp encoder, decoder còn có thêm một lớp phụ thứ ba là multi-head cross self-attention để kết hợp đầu ra từ encoder và kết quả tính toán được từ decoder. Lớp này cho phép decoder tập trung vào các phần của chuỗi đầu vào mà nó cần để dự đoán các từ tiếp theo. Đồng thời, cơ chế multi-head self-attention cũng được điều chỉnh để đảm bảo rằng các vị trí trong tương lai không được phép biết trước bởi masked.
- ***Output Layer:*** Cuối cùng, sau các lớp decoder, một lớp Softmax được áp dụng để dự đoán xác suất xuất hiện của từ tiếp theo trong chuỗi đầu ra.

Kiến trúc tổng thể của Transformer là một mạng nơ-ron sâu (deep neural network) với nhiều lớp Encoder và Decoder được xếp chồng lên nhau, mỗi lớp chứa nhiều mô-đun như Self-Attention và mạng nơ-ron truyền thẳng.

2.1.2. Các cơ chế quan trọng trong Transformer:

2.1.2.1. Cơ chế Attention:

Cơ chế attention hoạt động bằng cách tính toán một trọng số cho mỗi phần tử của đầu vào (input). Trọng số này thể hiện mức độ quan trọng của phần tử đó đối với đầu ra của mô hình. Các phần tử có trọng số cao hơn sẽ được mô hình tập trung nhiều hơn. Cơ chế attention có 3 giá trị cần được quan tâm lần lượt là các vecto query (Q), key (K) và value (V). Thông qua quá trình này, cơ chế self-attention có thể quan trọng hóa các thông tin quan trọng trong inputs bằng cách ánh xạ Q vào các giá trị có liên quan tương ứng.

Một cách phổ biến là sử dụng phép tính ***scaled dot product attention***. Phép tính này hoạt động bằng cách tính tích vô hướng giữa hai vector, một vector đại diện cho các phần tử của đầu vào và một vector đại diện cho các trọng số attention.

Scaled Dot-Product Attention:

Đầu tiên, tính điểm chú ý (attention scores) bằng cách sử dụng 3 ma trận trọng số: Q, K và V, mỗi từ đều được biểu diễn bởi 3 ma trận vector trên. Input sẽ bao gồm Q và K có kích thước là d_k và V có kích thước là d_v. Tích ma trận của Q và K sẽ được tính toán và chia cho $\sqrt{d_k}$ nhằm mục đích ngăn chặn hiện tượng suy giảm gradient khi tích vô hướng trở nên quá lớn làm cho hàm softmax đưa các giá trị này vào vùng có độ dốc rất nhỏ, và áp dụng một lớp softmax để thu được các trọng số của từng giá trị V.

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

Multi-head Attention:

Thay vì thực hiện một hàm attention duy nhất đối với các K, V và Q thì việc thực hiện ánh xạ tuyến tính các K, V, Q nhiều lần với các phép chiếu tuyến tính khác nhau một cách song song sẽ mang lại hiệu quả cao hơn. Bởi ở mỗi lần ánh xạ tuyến tính sẽ cho phép mô hình cùng lúc chú ý tới thông tin từ các không gian biểu diễn khác nhau tại các vị trí khác nhau, giúp cho mô hình học được thêm nhiều khía cạnh khác của thông tin đầu vào. Thông qua việc sử dụng nhiều đầu attention, mô hình có khả năng học được các mối quan hệ phức tạp và tương tác linh hoạt hơn giữa các thành phần trong dữ liệu. Ở đây, 8 đầu attention được thực hiện cùng một lúc.

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

$$where\ head_1 = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Trong đó: $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_v}, W^O \in \mathbb{R}^{hd_v \times d_{model}}$

Cơ chế Multi-head Attention được áp dụng trong Transformer theo 3 cách khác nhau cho 3 thành phần khác nhau.

- ***Cross-attention giữa encoder và decoder***: Các Q được lấy từ lớp encoder trước đó, và K, V được lấy từ đầu ra của decoder. Điều này cho phép mỗi vị trí trong decoder chú ý đến tất cả các vị trí trong input. Đây là cách thức giúp mô hình xử lý chú ý giữa encoder và decoder.
- ***Self-attention trong encoder:*** Các lớp self-attention trong encoder cho phép mỗi vị trí trong encoder chú ý đến tất cả các vị trí trong lớp trước đó.
- ***Self-attention trong decoder:*** Tương tự, các lớp self-attention trong decoder cho phép mỗi vị trí trong decoder chú ý đến tất cả các vị trí trong decoder. Để duy trì tính chất tự hồi quy (auto-regressive), cần ngăn chặn thông tin tương lai bằng cách đặt lại các giá trị -∞ cho các giá trị tương lai trong đầu vào của hàm softmax. Điều này đảm bảo rằng mỗi vị trí chỉ có thể chú ý đến các vị trí trước đó trong chuỗi giải mã.

2.1.2.2. Position-wise Feed-forward Networks

Ở mỗi lớp trong encoder và decoder sẽ chứa một lớp Fully-connected feed-forward network được áp dụng vào mỗi vị trí riêng biệt và xác định. Lớp này bao gồm 2 lớp biến đổi tuyến tính và 1 hàm kích hoạt ReLU ở chính giữa.

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

Lớp mạng truyền thẳng theo vị trí sẽ giúp mô hình học các mối quan hệ phi tuyến tính giữa các từ trong chuỗi đầu vào. Nó giúp cho mô hình có thể thực hiện được các biến đổi đơn giản nhưng quan trọng, giúp tăng cường khả năng học và biểu diễn thông tin trong chuỗi. FFNs cho phép mô hình nắm bắt được các mối quan hệ phức tạp giữa các phần tử trong chuỗi.

2.1.2.3. Embeddings và Softmax

Các input tokens và output tokens cần được nhúng (embedded) thành các vectors có chiều tương ứng với chiều của mô hình để đảm bảo mô hình có thể giao tiếp với đầu vào và đầu ra. Hàm biến đổi tuyến tính và softmax cũng được sử dụng để chuyển đổi output của decoder thành xác suất của từ được dự đoán tiếp theo.

2.1.2.4. Positional Encoding

Trong mô hình Transformer, không có sự truyền tiến (recurrence) hoặc tích chập (convolution), do đó để mô hình có thể sử dụng thông tin về thứ tự của chuỗi, chúng ta cần thêm thông tin về vị trí tương đối hoặc tuyệt đối của các token trong chuỗi. Để làm điều này, chúng ta thêm "mã hóa vị trí" (positional encodings) vào các vector embedding ở đầu vào. Các mã hóa vị trí có cùng chiều với các nhúng. Nó cho phép xử lý thông tin theo thứ tự đúng của chuỗi. Positional encoding được tạo ra dựa trên công thức sin, cos như sau:

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

Trong đó:

- PE là vecto nhúng vị trí
- i là chiều
- pos là vị trí
- d_{model} là chiều của mô hình

2.2. Cross Entropy Loss và Label Smoothing Loss

2.2.1. Hàm Cross Entropy Loss

Cross entropy là một khái niệm trong lý thuyết thông tin và machine learning, thường được sử dụng trong việc đo lường sự khác biệt giữa hai phân phối xác suất. Trong bài toán phân loại, cross entropy thường được sử dụng như là hàm mất mát (loss function) để đo lường sự khác biệt giữa phân phối xác suất dự đoán bởi mô hình và phân phối xác suất thực tế của các nhãn. Mục tiêu của quá trình huấn luyện là cực tiểu hóa cross entropy, tức là làm cho dự đoán của mô hình gần với phân phối xác suất thực tế.

Cho một chuỗi đích (target sequence) được biểu diễn dưới dạng one-hot encoding với chiều dài tương ứng với số lượng từ vựng (vocabulary size), và xác suất dự đoán của mô hình, cross entropy được tính bằng:

$$Cross\ Entropy\ Loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N} y_{i,j}\log(p_{i,j})$$

Trong đó:

- N là số lượng mẫu
- V là số lượng từ trong từ vựng
- $y_{i,j}$ là giá trị one-hot encoding của từ (hoặc nhãn) thực tế
- $p_{i,j}$ là xác suất được dự đoán bởi mô hình cho từ (hoặc nhãn) tương ứng.

2.2.2. Hàm Label Smoothing Loss:

Label smoothing là một kỹ thuật được sử dụng trong học máy để giảm bớt sự chắc chắn của mô hình đối với các nhãn dự đoán. Thay vì mã hóa nhãn là một one-hot vector (0 hoặc 1), nhãn này sẽ được thay đổi bằng cách được phân bố lại một lượng nhỏ lỗi (loss) vào các nhãn dự đoán và thực

tế. Label smoothing có thể giúp giảm thiểu overfitting của mô hình bằng cách làm cho nó ít chắc chắn hơn về các nhãn dự đoán, giúp cho mô hình cải thiện hiệu suất trên tập dữ liệu kiểm tra.

$$p_s = (1 - \varepsilon) \times p + \frac{\varepsilon}{n}$$

Trong đó:

- p_s là phân phối thực tế đã được làm mờ
- p là phân phối thực tế ban đầu
- ε là lượng entropy (lỗi) được thêm vào
- n là số nhãn

2.3. Beam Search và Greedy Decoder Search

2.3.1. Greedy Decoder Search

Greedy Decoder Search (Tìm kiếm tham lam) là phương pháp giải mã đơn giản nhất. Greedy decoder search bắt đầu với một chuỗi rỗng. Sau đó, ở mỗi bước, nó chọn từ hoặc token có xác suất cao nhất và thêm nó vào chuỗi để đưa vào mô hình làm đầu vào cho bước dự đoán tiếp theo. Quá trình này tiếp diễn cho đến khi đáp ứng được điều kiện dừng, thường là khi gặp mã token kết thúc <eos> hoặc độ dài chuỗi tối đa.

2.3.2. Beam Search

Beam Search là một biến thể của Greedy Decoder Search và là phương pháp giải mã phức tạp hơn giúp theo dõi nhiều chuỗi tiềm năng hơn ở mỗi bước. Beam Search tìm kiếm qua không gian giả định các chuỗi có thể tiếp theo dựa trên mô hình dự đoán. Ở mỗi bước, beam search tạo ra nhiều chuỗi mới bằng cách thêm mọi từ tiếp theo có thể có vào mỗi chuỗi mà nó hiện đang xem xét. Sau đó, beam search chọn các chuỗi có khả năng xảy ra cao nhất ‘k’ từ các chuỗi mới này, trong đó ‘k’ là độ rộng tìm kiếm (beam width) (giữ lại các ứng viên tiềm năng nhất). Quá trình mở rộng và chọn lựa tiếp tục cho đến khi đạt được điều kiện dừng (ví dụ: khi đạt đến độ dài tối đa hoặc gặp ký tự kết thúc). Sau đó, Beam Search chọn ra chuỗi có xác suất cao nhất trong tập hợp các ứng viên.

3. Bộ dữ liệu và tiền xử lý dữ liệu

3.1. Bộ dữ liệu

Dữ liệu sử dụng trong nghiên cứu này được tạo thành từ 6 tệp văn bản, tổng cộng chứa gần 300.000 câu. Những dữ liệu này đã được thu thập và xử lý từ các đoạn hội thoại của TED. Bộ dữ liệu này được phân chia thành ba phần chính để phục vụ cho quá trình huấn luyện và đánh giá mô hình dịch máy sử dụng Transformer:

- Tập huấn luyện (train.vi và train.en): Bao gồm hơn 130 nghìn câu cho mỗi ngôn ngữ, tiếng Việt và tiếng Anh tương ứng.
- Tập xác thực (tst2013.en và tst2013.vi): Được dùng để đánh giá hiệu suất của mô hình trong quá trình huấn luyện.
- Tập kiểm tra (tst2012.en và tst2012.vi): Sử dụng để đánh giá kết quả cuối cùng của mô hình dịch.

Đối với token cấp bậc ký tự, 97% các câu trong tập dữ liệu train có số lượng ký tự là 284 trở xuống tương ứng với ngôn ngữ tiếng Việt và 280 cho ngôn ngữ tiếng Anh. Vì vậy số lượng token tối đa cho một câu được đặt là 300 để thuận tiện cho việc huấn luyện mô hình.

Còn đối với token cấp tự, 99% số câu trong tập này có số lượng từ là 80 từ trở xuống mỗi câu và độ dài tối đa của một câu là 628 từ.

3.2. Tiền xử lý dữ liệu

Quá trình tiền xử lý dữ liệu cho việc huấn luyện mô hình Transformer bao gồm việc đọc, xử lý và chuẩn bị dữ liệu dưới dạng dataset và iterator. Dữ liệu sau khi được đọc vào sẽ được cắt câu nếu câu có số lượng token vượt quá ngưỡng được đặt ra ban đầu. Đối với token dạng ký tự, ngưỡng giới hạn được đặt ra là 300 ký tự. Giới hạn này là 200 đối với token dạng từ.

Các ký tự dư thừa cũng được xử lý, ví dụ như các ký tự đặc biệt trong HTML và XML "'d", "'s",... Ngoài ra còn áp dụng các xử lý khác như chuyển đổi chữ hoa thành chữ thường, loại bỏ dấu câu,... Cuối cùng, quá trình chuẩn bị dữ liệu hoàn tất bằng việc tạo vòng lặp huấn luyện mô hình.

4. Thực nghiệm và kết quả

4.1. Thực nghiệm

Training Data và Batching: Chúng tôi huấn luyện mô hình trên tập dữ liệu 'train.en' với nhãn là tập 'train.vi' với gần 130.000 câu. Các câu có độ dài hay số lượng ký tự xấp xỉ sẽ được gom vào cùng một batch để tối ưu hóa hiệu suất huấn luyện của mô hình

Hệ số tối ưu (Optimizer): Chúng tôi sử dụng hệ số tối ưu Adam với $\beta_1 = 0.9$, $\beta_2 = 0.98$ và $\varepsilon = 10^{-9}$. Tỷ lệ học (learning rate) được điều chỉnh trong suốt quá trình training theo công thức sau:

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

Residual dropout: Chúng tôi áp dụng dropout vào đầu ra của mỗi lớp con (sub-layers). Trước khi kết quả đầu ra của mỗi lớp con được cộng vào đầu vào của tầng đó và được chuẩn hóa thì kỹ thuật dropout được áp dụng nhằm tránh overfitting và tăng tính tổng quát hóa. Ngoài ra, dropout cũng được áp dụng vào tổng của embeddings (biểu diễn vector cho từ) và positional encodings (biểu diễn vị trí của từ) trong cả tầng encoder và decoder của mô hình Transformer. Tỷ lệ dropout được thiết lập xuyên suốt quá trình huấn luyện là 0.1

Hàm loss: Hàm loss được lần lượt sử dụng là:

- Cross Entropy Loss, với tham số ignore_index là 'src_pad'
- Label Smoothing Loss với số lượng lớp (classes) cần phân loại là kích thước từ vựng của tập đích. Độ làm mịn là 0.1

Phương pháp sinh chuỗi: Có 2 phương pháp:

- Greedy Decoder Search
- Beam search với beam width = 5

4.1.1. Mô hình Transformer dành cho token là ký tự

Các câu được token theo cấp độ ký tự và được mã hóa bằng thư viện TorchText của PyTorch.

Batch_size tối đa được thiết lập là 7500 câu. Từ đó bộ huấn luyện gồm 2026 batch trong một vòng lặp (epoch). Mỗi batch huấn luyện bao gồm 86 tokens nguồn và 155 token đích.

Tập dữ liệu được huấn luyện qua 10 epochs và tính điểm BLEU sau mỗi epoch cho 100 câu đầu tiên của tập validate. Khi quá trình training được hoàn tất thì sẽ tiến hành tính điểm BLEU cho toàn bộ tập validate và tập test

4.1.2. Mô hình Transformer dành cho token là từ

Các câu được token theo cấp độ từ và được mã hóa bằng thư viện TorchText của PyTorch kết hợp với module vi_core_news_lg của Spacy dành cho tiếng Việt và module en_core_news_lg dành cho tiếng Anh:

Batch_size tối đa được thiết lập là 7500 câu. Từ đó bộ huấn luyện gồm 1994 batch trong một vòng lặp (epoch). Mỗi batch huấn luyện bao gồm 41352 tokens nguồn và 32979 tokens đích.

Tập dữ liệu được huấn luyện qua 20 epochs và tính điểm BLEU sau mỗi epoch cho 100 câu đầu tiên của tập validate. Khi quá trình training được hoàn tất thì sẽ tiến hành tính điểm BLEU cho toàn bộ tập validate và tập test

4.2. Kết quả

Sau khi cho mô hình Transformer huấn luyện xong với từng cấp độ token từ khác nhau ta có thể thấy, với token cấp độ ký tự mặc dù thời gian huấn luyện cho mỗi epoch gấp 7 đến 8 lần so với thời gian huấn luyện mô hình với token là từ. Tuy nhiên, kết quả đánh giá điểm BLEU của 100 câu đầu tiên thuộc tập validate lại cho thấy kết quả của ký tự vượt trội hơn hẳn so với token từ, mặc dù số lượng epoch duyệt qua của ký tự chỉ là 10 epochs, trong khi của từ là 20 epochs. Số điểm BLEU của ký tự luôn nhỉnh hơn xấp xỉ 0.3 điểm so với việc sử dụng phương thức chia văn bản thành từ.

Mặc dù điểm BLEU của phương thức token bằng ký tự cao hơn so với phương thức token bằng từ nhưng không có nghĩa là token bằng ký tự sẽ cho hiệu quả tốt hơn trong nhiệm vụ dịch máy văn bản. Bởi vì kích thước từ vựng của token bằng ký tự sẽ rất nhỏ trong khi của token bằng từ có thể lên đến hàng nghìn hay chục nghìn. Điều đó chứng tỏ, phân phối của token sẽ xuất hiện tiếp theo sẽ cao hơn nhiều so với từ. Đồng thời, nếu chia văn bản thành các từ thì các vecto đặc trưng có thể chứa và học được các mối quan hệ rõ ràng hơn trong câu. Trong khi đó, khi thử nghiệm mô hình

Token cấp độ ký tự					**Token cấp độ từ**				
Epoch	**Cross Entropy**		**Label Smoothing**		**Epoch**	**Cross Entropy**		**Label Smoothing**	
	BLEU score	***Time***	***BLEU score***	***Time***		***BLEU score***	***Time***	***BLEU score***	***Time***
1	0.1486	299.848	0.2310	200.227	2	0.0349	35.350	0.0332	30.546
2	0.2548	232.642	0.2419	221.559	4	0.0550	38.055	0.0679	33.042
3	0.2872	213.809	0.3024	203.964	6	0.0817	39.034	0.0756	35.734
4	0.3007	174.206	0.2812	172.018	8	0.0844	38.528	0.0877	33.952
5	0.3099	223.290	0.3295	205.713	10	0.0836	39.604	0.0941	34.274
6	0.3267	179.105	0.3332	191.217	12	0.0885	37.033	0.0892	34.436
7	0.3522	195.117	0.3412	195.056	14	0.0826	37.523	0.0927	34.508
8	0.3600	184.722	0.3473	206.805	16	0.0863	37.663	0.0878	34.062
9	0.3560	217.408	0.3398	218.961	18	0.079	37.306	0.0909	34.911
10	0.3805	187.848	0.3903	205.211	20	0.0913	38.941	0.0976	35.216
BLEU valid tổng	0.4022		0.3908		BLEU valid tổng	0.1044		0.1064	

Bảng 1: *Kết quả đánh giá mô hình trên 100 câu đầu của tập validate sau mỗi epoch*

Transformer cho tác vụ dịch máy với 2 hàm loss khác nhau là Cross Entropy và Label Smoothing cũng có sự chênh lệch nhỏ về điểm BLEU thu được. Càng huấn luyện mô hình với số lượng epoch càng cao thì độ chính xác của mô hình dùng Label Smoothing càng có sự cải thiện hơn so với hàm Cross Entropy Loss đối với tập dữ liệu nhỏ. Thời gian huấn luyện của Label Smoothing cũng ít hơn.

Sử dụng Beam search thay vì Greedy Decoder Search sẽ giúp cải thiện hơn điểm BLEU. Đối với phương thức token là ký tự, điểm đánh giá BLEU của toàn bộ tập validate sử dụng beam search là 0.4022 so với 0.3738 của greedy search. Đối với phương thức token là từ, beam search cũng cho điểm BLEU nhỉnh hơn so với tìm kiếm tham lam.

5. Kết luận

Transformer đã đóng một vai trò quan trọng và tạo ra sự đột phá trong lĩnh vực dịch máy, đặc biệt trong tác vụ phiên dịch từ tiếng Anh sang tiếng Việt. Mô hình này không chỉ vượt qua những hạn chế của các phương pháp truyền thống mà còn mang đến sự linh hoạt, hiệu suất cao và khả năng học tập sâu trong việc biểu diễn và xử lý ngôn ngữ tự nhiên.

Trong nghiên cứu này, chúng tôi đã tiếp cận Transformer thông qua 2 cách thức tokenization khác nhau: token cấp độ từ và token cấp độ ký tự. Kết quả cho thấy rằng thời gian huấn luyện trên token ký tự lâu hơn so với token từ. Ngoài ra, chúng tôi đã thực hiện so sánh giữa hai phương pháp tìm kiếm, Beam Search và Greedy Search sau mô hình Transformer, để tìm ra phương pháp nào tốt hơn trong quá trình dự đoán đầu ra. Kết quả cho thấy rằng Greedy Search thường xử lý nhanh hơn, nhưng Beam Search thường xuyên mang lại kết quả tốt hơn về mặt chất lượng dịch thuật với khả năng có thể xem xét nhiều ứng viên hơn cho dự đoán từng vị trí. Tuy nhiên, Beam Search thường tốn nhiều tài nguyên tính toán hơn. Label Smoothing Loss cũng giúp hạn chế overfitting cho mô hình Transformer so với Cross Entropy và mang đến hiệu suất tính toán tốt hơn trên tập kiểm nghiệm (test, validate). Nhưng Cross Entropy lại cho thấy ưu thế trong việc huấn luyện mô hình khi thời gian huấn luyện nhanh hơn và độ chính xác của tập train cao.

Trong tương lai, chúng tôi hướng đến việc tối ưu hóa hiệu suất của mô hình Transformer trên các bộ dữ liệu lớn. Cải thiện vấn đề về cấu hình máy tính và tối ưu hóa tài nguyên tính toán để có thể huấn luyện mô hình trên các dữ liệu đa dạng và lớn hơn. Ngoài ra chúng tôi hướng đến mở rộng nghiên cứu về việc sử dụng các loại token khác nhau trong mô hình, thay vì tập trung chỉ vào token từ và token ký tự, chúng tôi quan tâm đến việc thử nghiệm với các loại token khác.

Tài liệu tham khảo

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017), *"Attention is all you need"*, In Advances in neural information processing systems, Tr. 5998-6008.

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2014). *"Neural machine translation by jointly learning to align and translate."* arXiv preprint arXiv:1409.0473.

[3] Luong, M. T., Pham, H., & Manning, C. D. (2015). *"Effective approaches to attention-based neural machine translation."* arXiv preprint arXiv:1508.04025..

[4] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *"Sequence to sequence learning with neural networks."* In Advances in neural information processing systems (pp. 3104-3112).

[5] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *"Learning phrase representations using RNN encoder-decoder for statistical machine translation."* arXiv preprint arXiv:1406.1078.

[6] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). *"Google's neural machine translation system: Bridging the gap between human and machine translation."* arXiv preprint arXiv:1609.08144.

[7] Sennrich, R., Haddow, B., & Birch, A. (2016). *"Byte Pair Encoding (BPE).",* https://www.researchgate.net/publication/2310624_Byte_Pair_Encoding_A_Text_Compression_Scheme_That_Accelerates_Pattern_Matching

[8] Taku Kudo (2018), *"Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates"*, Annual Meeting of the Association for Computational Linguistics, DOI:10.18653/v1/P18-1007

[9] Kudo, T., & Richardson, J. (2018), *" SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing"*, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Conference, DOI:10.18653/v1/D18-2012

[10] Kudo, T. (2018). *"Unigram language model tokenization."*, arXiv:1804.10959v1

[11] Marta R. Costa-jussà and José A. R. Fonollosa (2016), *"Character-based Neural Machine Translation"*. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Association for Computational Linguistics, Tr. 357–361

[12] Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., ... & others. (2018). "*Character-level language modeling with deeper self-attention."*, Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, DOI:10.1609/aaai.v33i01.33013159