XÂY DỰNG MÔ HÌNH DỊCH MÁY ANH-VIỆT SỬ DỤNG KIẾN TRÚC TRANSFORMER

Vũ Việt Hùng - 22022585
Nguyễn Đức Anh - 22022661
Bùi Thế Huy - 22022667
Hà Kim Dương - 22022621
Nguyễn Trọng Huy - 22022545
Hồ Minh Hoàng - 22022567
Viện Trí tuệ nhân tạo
Trường Đại học Công Nghệ - ĐHQGHN
Hà Nội, Việt Nam

***Tóm tắt nội dung*—Bài toán dịch máy Anh-Việt và Việt-Anh là một trong những ứng dụng quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Trong báo cáo này, chúng em áp dụng kiến trúc Transformer, một phương pháp hiện đại và hiệu quả trong dịch máy, để huấn luyện mô hình trên tập dữ liệu do VinAI cung cấp. Mục tiêu của nghiên cứu là cải thiện chất lượng dịch giữa hai ngôn ngữ này, đồng thời phân tích các yếu tố ảnh hưởng đến độ chính xác của mô hình. Kết quả cho thấy kiến trúc Transformer có khả năng xử lý tốt các câu phức tạp và đạt được hiệu suất cao trong các bài kiểm tra dịch tự động.**

1. Bài toán dịch máy

1.1. Động lực

Dịch máy là một bài toán quan trọng và đã có lịch sử phát triển lâu dài trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Mục tiêu chính của dịch máy là tự động chuyển đổi các câu, đoạn văn từ một ngôn ngữ nguồn sang ngôn ngữ đích sao cho vẫn giữ được nghĩa và sắc thái của thông tin. Dịch máy không chỉ là một thách thức về ngữ nghĩa, mà còn liên quan đến các yếu tố như ngữ pháp, cú pháp và các vấn đề văn hóa, xã hội trong quá trình chuyển ngữ.

Trong bối cảnh hiện nay, dịch máy đang trở thành một công cụ hữu ích trong nhiều lĩnh vực, từ việc giúp người dùng giao tiếp giữa các ngôn ngữ khác nhau, đến việc hỗ trợ các dịch vụ khách hàng tự động và cải thiện các hệ thống tìm kiếm thông tin. Chính vì vậy, nghiên cứu và cải thiện các phương pháp dịch máy luôn là một chủ đề hấp dẫn, có ý nghĩa lớn đối với sự phát triển của công nghệ.

Chúng em quyết định lựa chọn chủ đề nghiên cứu về dịch máy, đặc biệt là dịch từ ngôn ngữ tiếng Anh sang tiếng Việt và ngược lại. Mục tiêu của chúng em là học hỏi và tìm hiểu sâu hơn về các phương pháp và kỹ thuật trong xử lý ngôn ngữ tự nhiên, đồng thời nghiên cứu các quy trình tiền xử lý dữ liệu, xây dựng các mô hình dịch máy và áp dụng các kiến trúc mạng nơ-ron, đặc biệt là mô hình deep learning, để giải quyết bài toán này. Qua đó, chúng em mong muốn nâng cao hiểu biết về các kiến thức cơ bản và nâng cao trong lĩnh vực học sâu (Deep Learning), từ đó mở rộng khả năng ứng dụng công nghệ trong các bài toán thực tế.

1.2. Mô tả bài toán

Bài toán dịch máy này có thể được mô tả đơn giản như sau:

- **Input:** Một câu đầu vào từ ngôn ngữ nguồn (có thể là tiếng Anh hoặc tiếng Việt).
- **Output:** Một câu đầu ra tương ứng trong ngôn ngữ đích (có thể là tiếng Anh hoặc tiếng Việt).

Mục tiêu của bài toán là xây dựng một hệ thống có thể tự động dịch chính xác giữa hai ngôn ngữ, đồng thời đảm bảo tính ngữ nghĩa và ngữ pháp của câu dịch sao cho phù hợp với văn cảnh và các quy tắc ngôn ngữ của ngôn ngữ đích.

1.3. Lịch sử các mô hình trước khi có Transformer (Mô hình - Nhược điểm)

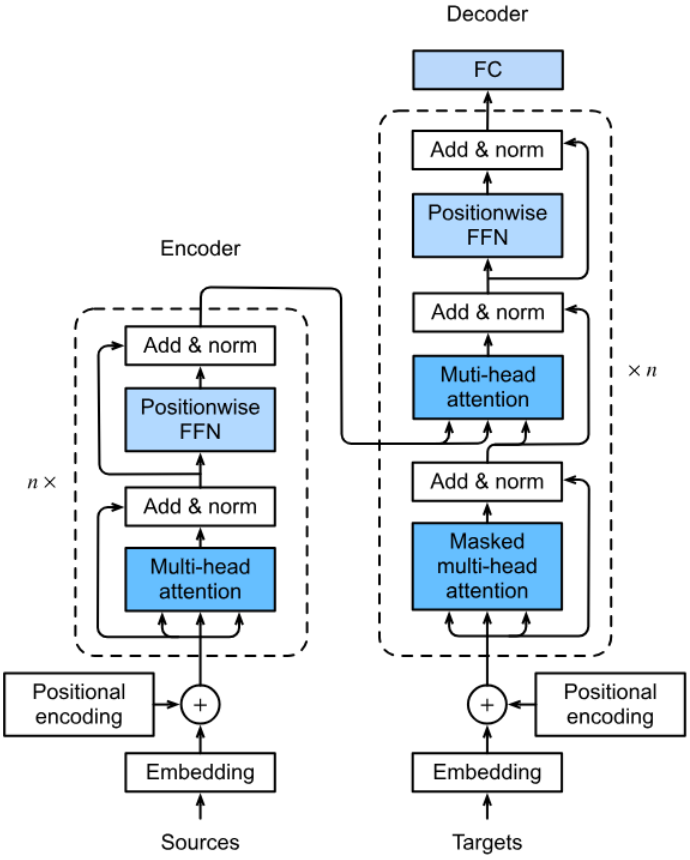
Trong bài tập này, chúng em sử dụng phương pháp dịch máy hiện đại, áp dụng mạng nơ-ron Deep Learning để học các vector ngữ cảnh, giúp hệ thống dịch không chỉ hiểu ngữ nghĩa mà còn nhận diện được ngữ cảnh của câu.

Trước khi Transformer ra đời, các hệ thống dịch máy chủ yếu sử dụng các mạng nơ-ron tuần tự như RNN. Tuy nhiên, phương pháp này gặp phải nhược điểm lớn, bao gồm việc làm chậm quá trình huấn luyện và hiện tượng vanishing gradient, khiến các biểu diễn ngữ nghĩa bị mất đi, đặc biệt là với các câu dài. Transformer đã giải quyết được những vấn đề này bằng cách cho phép tính toán song song và áp dụng cơ chế self-attention.

2. Transformer

Transformer là một kiến trúc mạng neural đột phá trong lĩnh vực xử lý ngôn ngữ tự nhiên, được giới thiệu trong bài báo nổi tiếng 'Attention Is All You Need'[2] vào năm 2017. Khác với các mô hình truyền thống như RNN hay LSTM, Transformer sử dụng cơ chế self-attention cho phép mô hình xử lý các từ trong một câu một cách song song và hiệu quả hơn. Kiến trúc này đã mở ra một bước ngoặt quan trọng trong các bài toán

dịch máy, sinh văn bản, và nhiều ứng dụng xử lý ngôn ngữ khác.



Hình 1. Kiến trúc Transformer

2.1. Embedding Layer với Position Encoding

Đây là layer không thể thiếu trong các bài toán xử lý ngôn ngữ tự nhiên đóng vài trò mã hóa các từ đầu vào thành các vector embadding trong không gian giúp mô hình có thể hiểu được mối quan hệ giữa các từ.

2.2. Position Encoding

Do transformer xử lý song song một câu vì vậy. Layer này đóng vai trò thêm vào các vector embadding các vector thứ tự để giúp mô hình phân biện được thứ tự các từ trong câu. Theo Vaswani et al. công thức mã hóa theo hàm sin và cos với các tần số khác nhau như sau:

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{\frac{2i}{d}}}) \quad (1)$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{\frac{2i}{d}}}) \quad (2)$$

Với :

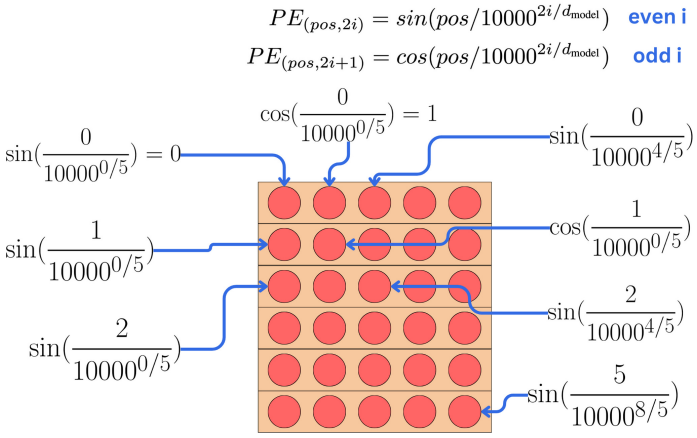
- pos: ví trí của từ trong câu
- i: Chỉ số chiều của vector mã hóa.
- d: Kích thước của vector mã hóa .

Vector đầu vào cuối cùng sẽ được tính theo công thức:

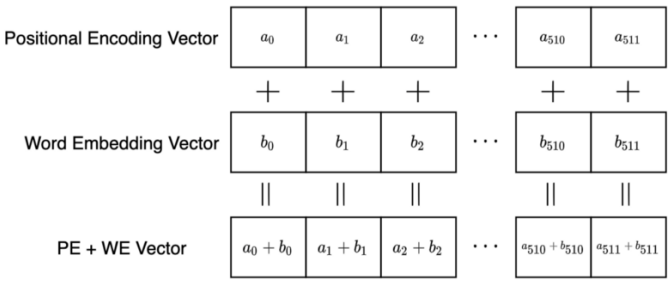
$$y = PE + WE \quad (3)$$

Với

- PE: là vector vị trí
- WE: là vector word embadding
- y: là vector mang biểu diễn cả ngữ nghĩa và ví trí để đưa vào mô hình



Hình 2. Hình ảnh minh họa cho Position Encoding.



Hình 3. Hình ảnh minh họa vector mang biểu diễn ngữ nghĩa và vị trí.

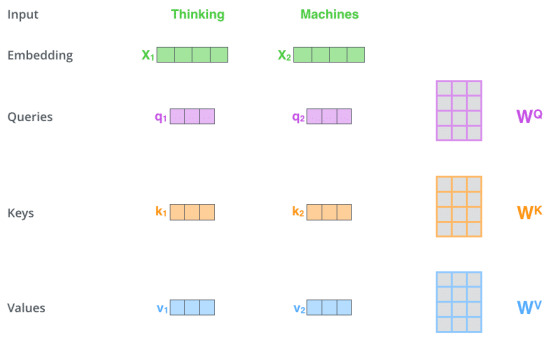
2.3. Encoder

Encoder gồm 2 thành phần chính là khối self-attention và feedforward đây có thể coi là phần quan trọng nhất trong transformer

1) Self Attention Layer

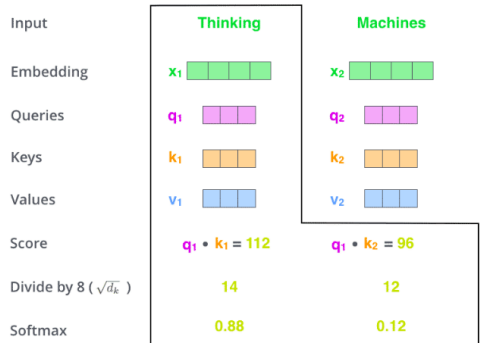
Vector y đầu vào sẽ được đi qua khối self-attention để học được ngữ cảnh của các từ trong câu mối liên hệ giữa các từ, nhấn mạnh các từ quan trọng cần chú ý đến, đầu ra sẽ là một vector z chứa thông tin giữa các từ trong ngữ cảnh các từ quan trọng sẽ được nhấn mạnh.

- *Cơ chế hoạt động của attention*: từ Vector đầu vào y mỗi từ i trong câu sẽ thực hiện phép nhân với 3 ma trận trọng số được khởi tạo ban đầu để tạo ra 3 vector tương ứng
  Query : Q_i ma trận trọng số tương ứng: W_Q
  Key : K_i ma trận trọng số tương ứng: W_k
  Value: V_i ma trận trọng số tương ứng: W_v



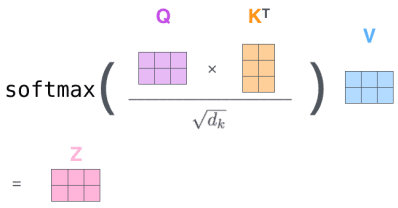
Hình 4. Hình ảnh minh họa cho cơ chế attention

– Sau đó với mỗi từ i vector Q_i sẽ được nhân với tất cả các vector K của các từ còn lại nhằm hiển thị được mối quan hệ giữa các từ



Hình 5. Hình ảnh minh họa thực hiện các phép nhân ma trận c
Q với K

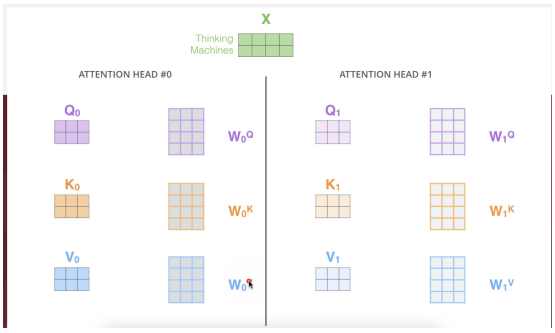
– Tiếp đến giá trị sẽ được chia cho căn kích thước vector embadding rồi sau đó softmax để biến đổi về phân khối xác xu
– Cuối cùng nhân phân khối xác xuất với V_i vector z_i tương ứng với mỗi từ



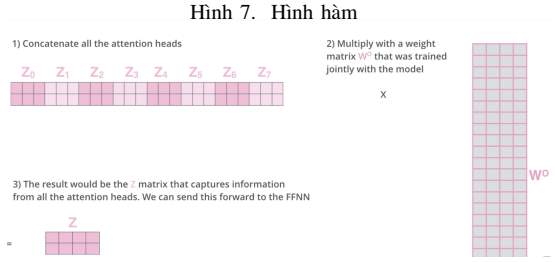
Hình 6. Hình ảnh minh họa vector attention được tính toán

Quá trình này được học để tìm ra các ma trận trọng số tối ưu nhất

- *Mutil-Head Attention*: Tương tự như Attention tuy nhiên mỗi từ sẽ có nhiều bộ ba ma trận trọng số tương ứng với mỗi head. Các phép tính toán bên trong được thực hiện tương tự sau đó đưa ra kết quả là một bộ vector z $(z_1, z_2, ...z_n)$. Bộ vector Z này được nhân với một ma trận Wo để đưa về kích thước ban đầu



Hình 7. Hình hàm

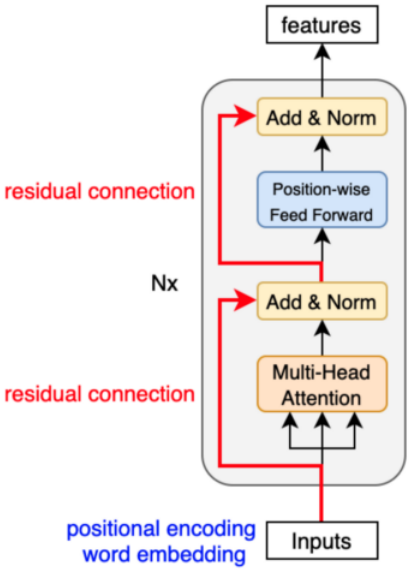


Hình 8. Hình ảnh minh họa

2) Residuals Connection và Normalization Layer

- Residuals Connection: Là cách kết nối dữ liệu đầu vào trực tiếp với dữ liệu đầu ra qua các layer:

$$AttentionOutput = MultiHeadAttention(x) + x \quad (4)$$



Hình 9. Hình ảnh minh họa

- Normalization Layer: giúp chuẩn hóa các đặc trưng của dữ liệu trong một layer.

$$LayerNorm(x) = \frac{x - \mu}{\sigma}.\gamma + \beta \quad (5)$$

– x : vector đầu vào tại mỗi từ
– μ : Trung bình các giá trị có trong x

- σ: Độ lệch chuẩn của x
- γ: Tham số học
- β: Độ lệch bias

3) Feed-forward

Cuối cùng các vector đầu ra sẽ được đi qua một mạng Feed-Forward đây là mạng no ron bình thường gồm các lớp tuyến tính và hàm kích hoạt. Giúp mô hình học các quan hệ phức tạp hơn giữa các từ trong ngữ cảnh.

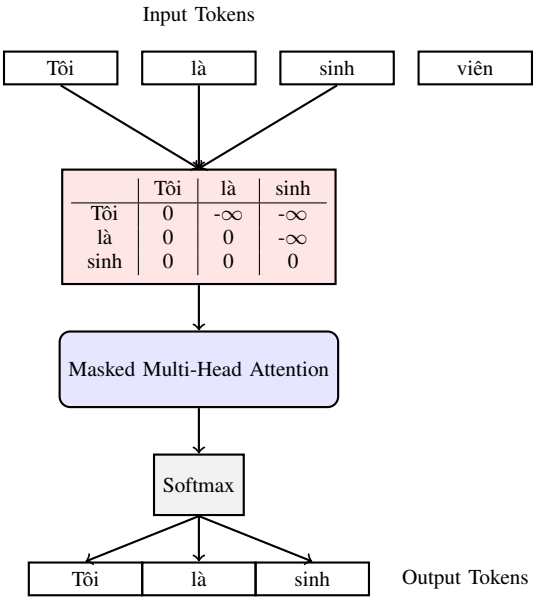
2.4. Decoder

Decoder có nhiệm vụ giải mã chuỗi đầu ra (ví dụ: tiếng Việt) dựa trên biểu diễn ngữ cảnh được cung cấp bởi Encoder và các từ đã được tạo ra trước đó. Decoder bao gồm ba thành phần chính: Masked Multi-Head Attention, Encoder-Decoder Attention và Feed-Forward.

1) Masked Multi-Head Attention Layer

Khác với Self-Attention trong Encoder, Masked Self-Attention trong Decoder áp dụng cơ chế masking để ngăn việc "nhìn vào tương lai". Điều này là cần thiết vì trong quá trình dịch, chúng ta chỉ có thể dựa vào các từ đã được dịch trước đó để dự đoán từ tiếp theo.

- *Cơ chế Masking*: Để ngăn việc "nhìn vào tương lai", một ma trận mask được tạo ra, trong đó các vị trí tương ứng với các từ ở tương lai được gán giá trị $-\infty$ (hoặc một giá trị rất lớn âm). Khi tính toán softmax, các giá trị này sẽ trở về 0, loại bỏ ảnh hưởng của các từ tương lai. Ví dụ: với câu đầu ra "Tôi là sinh viên", khi dịch đến từ "sinh", mask sẽ che các từ "viên"ở phía sau.



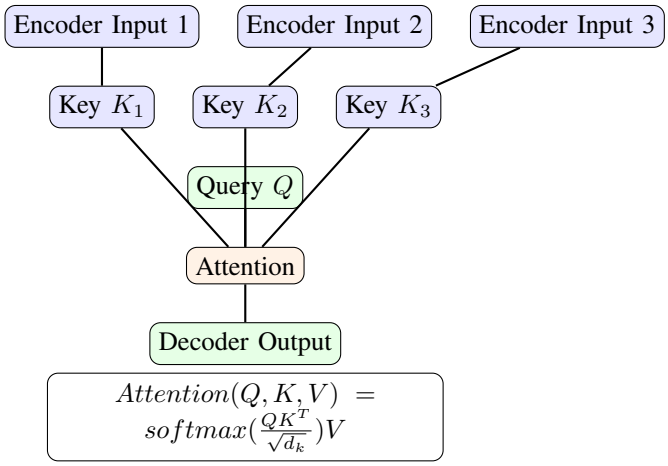
Hình 10. Minh họa quá trình Masked Multi-Head Attention trong mô hình Transformer. Các token đầu vào được xử lý qua các lớp Masking, Attention, và Softmax để tạo ra các token đầu ra.

- *Hoạt động của Masked Self-Attention*: Tương tự như Self-Attention trong Encoder, đầu vào là vector biểu diễn của các từ trong chuỗi đích. Mỗi từ i sẽ được nhân với ba ma trận trọng số W_Q, W_K, W_V để tạo ra Q_i, K_i, V_i. Tuy nhiên, trước khi tính softmax, ma trận attention sẽ được cộng với ma trận mask. Các bước tính toán còn lại tương tự như Self-Attention:
  - Tính attention scores: $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$
  - Áp dụng mask: $MaskedAttention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}} + Mask)V$
- *Multi-Head Attention*: Tương tự như Multi-Head Attention trong Encoder, nhưng sử dụng Masked Attention thay vì Self-Attention thông thường.

2) Encoder-Decoder Attention Layer

Lớp này giúp Decoder "chú ý"đến thông tin từ Encoder.

- *Cơ chế hoạt động*: Trong lớp này, Query Q được lấy từ output của lớp Masked Multi-Head Attention trong Decoder, còn Key K và Value V được lấy từ output của Encoder. Điều này cho phép Decoder tập trung vào các phần quan trọng của câu nguồn khi dịch. Công thức tính toán tương tự như Self-Attention: $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$



Hình 11. Hình ảnh minh họa Encoder-Decoder Attention

3) Residuals Connection và Normalization Layer

Tương tự như Encoder, Decoder cũng sử dụng Residuals Connection và Normalization Layer để ổn định quá trình huấn luyện và cải thiện hiệu suất.

- *Residuals Connection*:

$$DecoderOutput = AttentionOutput + Input \quad (6)$$

- *Normalization Layer*: Sử dụng công thức tương tự như Encoder:

$$LayerNorm(x) = \frac{x - \mu}{\sigma}.\gamma + \beta \quad (7)$$

4) Feed-Forward Network

Tương tự như Encoder, một mạng Feed-Forward được áp dụng cho mỗi vị trí.

- Mạng Feed-Forward bao gồm hai lớp fully connected với hàm kích hoạt ReLU ở giữa.

3. Thực nghiệm

3.1. Data

Bọn em sử dụng bộ dữ liệu được giới thiệu trong bài báo: PhoMT: A High-Quality and Large-Scale Benchmark Dataset for Vietnamese-English Machine Translation của Long Doan, Linh The Nguyen, Nguyen Luong Tran, Thai Hoang, Dat Quoc Nguyen [1]. Bộ dữ liệu chứa hơn 3000000 câu Anh - Việt tương ứng.

Trong bài tập lần này do giới hạn về tài nguyên nên chúng em chỉ sử dụng 100000 câu được trích xuất từ trong bộ dữ liệu.

1) Tiền xử lý dữ liệu

Dữ liệu được đọc từ file văn bản ban đầu và lưu trữ vào hai mảng tương ứng với tiếng Anh và Việt mỗi phần tử trong mảng tương ứng với một câu tiếng Anh hoặc tiếng Việt.

Tiếp đến dữ liệu được đi qua một hàm tokenizer để phân tách thành các thành các đơn vị từ nhỏ hơn như token trong bài tập bọn em sử dụng thư viện spacy để thực hiện phân tách câu thành các token.

Sau đó dữ liệu được tạo thành các data. Field trong torchtext để định nghĩa cách xử lý dữ liệu văn bản (text) trước khi đưa vào mô hình. Cung cấp các công cụ để chuẩn hóa, tokenize, xử lý các chuỗi văn bản, và tạo batch dữ liệu. Từ đó xây dựng ra bộ từ vựng của 2 ngôn ngữ

Sau cùng dữ liệu được sử dụng để tạo một dataset và train_iter chia dữ liệu thành các batch theo batch_size để train mô hình.

3.2. Implement

1) Tải dữ liệu lên và xử lý dữ liệu như mô tả bên trên.

2) Cài đặt kiến trúc các khối Encoder và Decoder của Transformer bằng thư viện pytorch.

3) Xử lý dữ liệu trước khi đưa vào mô hình để huấn luyện

- Dữ liệu được chia thành các batch
- Thêm padding vào các câu để đảm bảo các câu có độ dài bằng nhau
- Thêm lớp mask encoder để mô hình phân biệt được độ dài câu thực với padding và mask decoder để lúc dự đoán trong quá trình huấn luyện mô hình không nhìn thấy được các từ ở tương lai

4) Định nghĩa LabelSmoothingLoss

Hàm loss được sử dụng để tính toán độ sai lệch giữa nhãn và model dữ đoán dựa trên hàm loss cross-entropy nhưng thay vì gán xác suất 1.0 cho nhãn đúng, ta gán một giá trị xác suất nhỏ cho các nhãn khác, phân phối đều trên các lớp. Điều này giúp mô hình tránh bị quá khớp (overfitting) và xử lý tốt hơn trong các trường hợp không chắc chắn.

Sử dụng hàm optimizer Adam với learning rate được điều chỉnh trong quá trình học theo công thức:

$$new\_lr = init\_lr * (d_{model}^{-0.5} * min(n_{steps}^{-0.5}, n_{steps} * n_{\_warmup\_steps}^{-1.5})) \quad (8)$$

Với:

- new_lr: Learning rate mới được khởi tạo
- init_lr: Learning rate được khởi tạo ban đầu
- n_{steps}: Số bước mà model dùng để cập nhật learning rate
- n_{warmup_steps}: Số bước khởi đầu mà learning rate sẽ tăng dần trước khi bắt đầu giảm
- d_{model}: kích thước vector đầu vào.

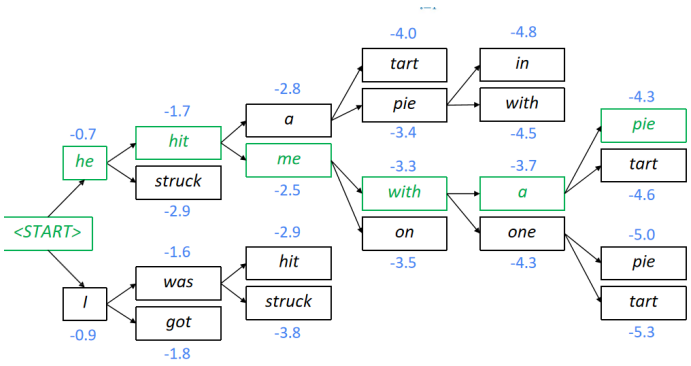
- Cập nhật cho mô hình nhiều câu hơn để có thể, tối ưu hiệu quả dịch ngôn ngữ của mô hình đối với các câu có độ phức tạp cao hơn.
- Thử nghiệm với nhiều loại ngôn ngữ khác.

5) Khởi tạo các siêu tham số

- Learning rate ban đầu : 1e-09
- n_{warmup_steps}: 4000
- d_{model}: 512
- max_strlen: 160
- $batchsize$: 1500
- $device$: cuda
- số lớp encoder và decoder: 6
- số heads attention: 8
- droput:0.1

6) Sử dụng beam search để decoder

Tại mỗi bước sẽ mở rộng k từ với xác xuất cao nhất và chọn ra 1 từ với xác xuất cao nhất trong k từ đó. Ví dụ:



Hình 12. Ví dụ về beam search với k = 2

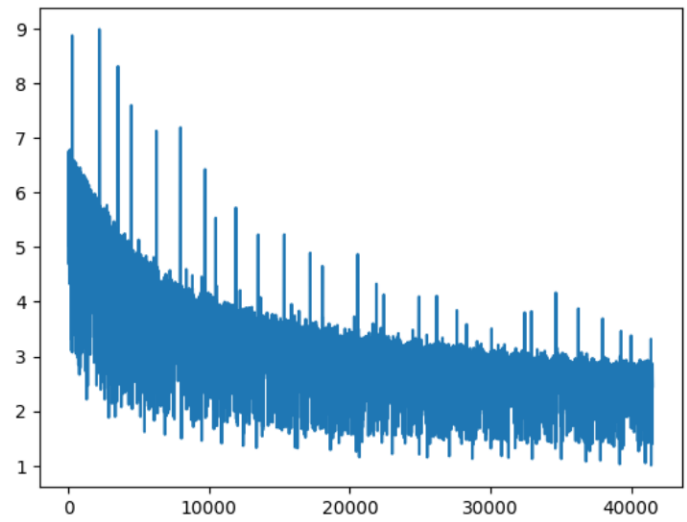
3.3. Kết quả

1) Sử dụng Transformer

a) Ở lần thử nghiệm đầu tiên

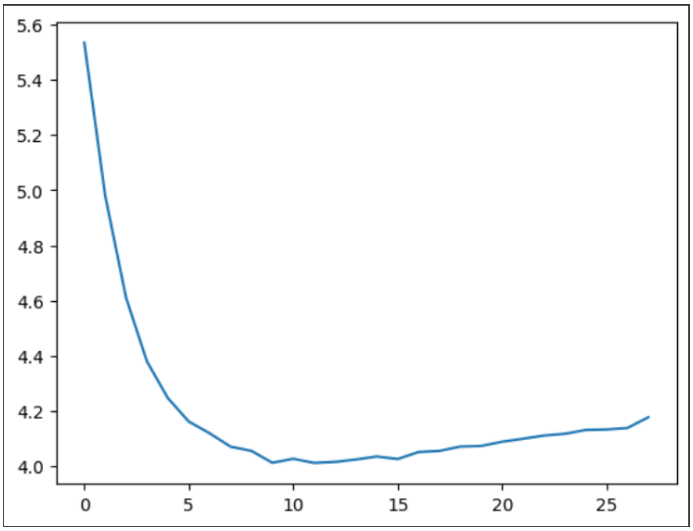
Bọn em thử nghiệm trên 25 epoch và train trong 3 tiếng dưới đây là đồ thị của 2 giá trị loss trên tập train và tập validate .

Nhìn chung loss đã có xu hướng giảm trên tập train tuy nhiên giảm không mượt có nhiều giao động, điều này có thể do kích thước batchsize, sự thay đổi của learning rate. Khoảng 10000 step ban đầu loss giảm khá nhanh và dần hội tụ ở khoảng 30000 step.



Hình 13. Biểu đồ Loss trên tập train

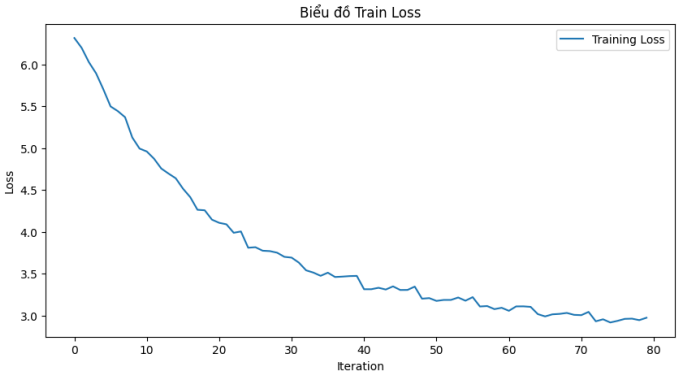
Với đồ thị loss trên tập validate ở những epoch đầu loss giảm đều và nhanh và hội tụ trong khoảng từ 10 - 15 epoch sau đó lại tăng nhẹ điều này cho thấy model đang bị overfit. Dẫn đến điểm BLEU thấp chỉ khoảng 0.002



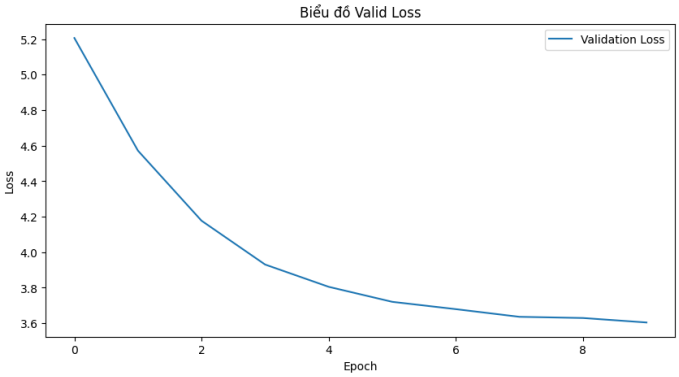
Hình 14. Biểu đồ loss trên tập validate

b) Train lại lần 2

Rút kinh nghiệm từ những lần trước lần này chúng em cho mô hình dừng lại ở epoch 10 và train trong khoảng hơn 1 tiếng.



Hình 15. Biểu đồ loss trong quá trình training



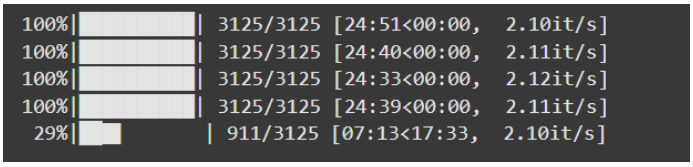
Hình 16. Biểu đồ loss trên tập validate

Đồ thị của loss trên tập validate đã ổn định hơn mô hình không bị tình trạng overfit.

Điểm BLEU cho mô hình dịch từ en - vi ở mức 0.3.

c) Thử nghiệm với RNN truyền thống

Bọn em cũng thử huấn luyện bộ dữ liệu nhưng sử dụng phương pháp RNN thì thấy tốc độ train rất chậm khoảng 25 phút cho 1 eps. Do hạn chế về tài nguyên tính toán nên bọn em mới thử train trên khoảng 5 eps. Do đó điểm BLEU chỉ ở mức 0.002299



Hình 17. Thời gian huấn luyện RNN

4. Hạn chế và hướng phát triển

4.1. Hạn chế

Trong quá trình thực hiện bài tập lớn, chúng em đã nhận thấy một số hạn chế sau đây, chủ yếu xuất phát từ hạn chế về tài nguyên tính toán và kinh nghiệm triển khai:

- Bộ dữ liệu huấn luyện được giới hạn ở 100.000 câu, chỉ là một phần nhỏ so với các bộ dữ liệu tiêu chuẩn trong

lĩnh vực dịch máy, chẳng hạn như WMT hoặc IWSLT, có kích thước hàng triệu câu. Điều này dẫn đến việc mô hình thiếu sự bao quát và khả năng học sâu đối với các câu có ngữ pháp phức tạp hoặc chứa các cụm từ ít xuất hiện.
- Do hạn chế tài nguyên phần cứng, chúng em chỉ triển khai mô hình với kiến trúc Transformer cơ bản, với số lượng lớp và kích thước embedding nhỏ hơn so với các mô hình tiên tiến. Điều này ảnh hưởng đến khả năng biểu diễn và hiệu quả học tập của mô hình.
- Các siêu tham số như learning rate, batch size, và số epoch chưa được tối ưu hóa triệt để do hạn chế về thời gian và không có sự hỗ trợ từ các công cụ tìm kiếm tự động như Optuna hoặc Ray Tune.
- Quá trình xử lý dữ liệu còn thủ công ở nhiều bước, đặc biệt là khi tách từ, mã hóa dữ liệu đầu vào, và xử lý các ký tự đặc biệt. Điều này làm gia tăng độ phức tạp trong triển khai, đồng thời có nguy cơ bỏ sót lỗi ở giai đoạn tiền xử lý.
- Việc đánh giá hiệu suất chỉ dựa trên BLEU score, một thước đo phổ biến nhưng không hoàn hảo, chưa phản ánh được các khía cạnh quan trọng khác như ngữ pháp, sự lưu loát, và ý nghĩa ngữ cảnh của bản dịch.
- Chưa có cơ chế điều chỉnh trọng số loss để xử lý các câu có độ dài khác nhau, dẫn đến kết quả dịch máy còn kém ổn định với các câu rất ngắn hoặc rất dài.
- Do thiếu kinh nghiệm, nhóm chưa thử nghiệm và so sánh nhiều biến thể của Transformer (chẳng hạn như Transformer-XL, BERT2BERT) để đánh giá mức độ cải thiện so với mô hình cơ bản.

4.2. Hướng phát triển

Nhằm khắc phục những hạn chế trên và tiến xa hơn trong nghiên cứu dịch máy, chúng em đề xuất các hướng phát triển cụ thể và có tính khả thi như sau:

- **Mở rộng quy mô dữ liệu huấn luyện**: Tăng kích thước bộ dữ liệu lên hàng triệu câu, đặc biệt bổ sung các câu có cấu trúc ngữ pháp phức tạp và các từ chuyên ngành, nhằm tăng khả năng tổng quát hóa của mô hình. Đồng thời, sử dụng các kỹ thuật làm sạch dữ liệu tự động để đảm bảo chất lượng dữ liệu đầu vào.
- **Áp dụng kỹ thuật tăng cường dữ liệu (Data Augmentation)**: Triển khai các phương pháp hoán đổi vị trí từ, sử dụng từ đồng nghĩa, hoặc thêm nhiễu có kiểm soát để tăng cường sự đa dạng của dữ liệu và nâng cao tính linh hoạt của mô hình.
- **Tối ưu hóa siêu tham số**: Sử dụng các thuật toán tối ưu hóa tự động như Grid Search, Random Search hoặc Bayesian Optimization để xác định giá trị tối ưu của các siêu tham số, qua đó cải thiện hiệu suất mô hình mà không làm tăng đáng kể chi phí tính toán.
- **Triển khai các biến thể của Transformer**: Thử nghiệm các mô hình tiên tiến hơn như Transformer-XL (xử lý các chuỗi dài tốt hơn), T5 (đào tạo đa nhiệm), hoặc mô hình dựa trên pre-trained như mBART để cải thiện khả năng dịch ngôn ngữ.
- **Kết hợp các thước đo đánh giá bổ sung**: Song song với BLEU score, áp dụng các thước đo khác như METEOR (tập trung vào độ chính xác ngữ nghĩa), TER (Tỷ lệ chỉnh sửa câu), hoặc Rouge để đánh giá toàn diện chất lượng bản dịch.
- **Tăng cường xử lý dữ liệu tự động**: Phát triển pipeline tự động hóa toàn bộ quy trình từ tải dữ liệu, tiền xử lý, huấn luyện, đến đánh giá. Việc này không chỉ tiết kiệm thời gian mà còn giảm thiểu lỗi do thao tác thủ công.
- **Khám phá đa ngôn ngữ**: Mở rộng bài toán dịch sang các cặp ngôn ngữ khác như Anh-Pháp, Anh-Nhật, hoặc giữa các ngôn ngữ có ít tài nguyên hơn như Việt-Myanmar. Đây là một hướng đi có tính ứng dụng cao, đặc biệt trong bối cảnh toàn cầu hóa.
- **Cải thiện việc xử lý câu dài và ngữ cảnh**: Điều chỉnh trọng số loss theo độ dài câu hoặc áp dụng các cơ chế bổ sung như Global Attention để giúp mô hình xử lý tốt hơn các câu có độ dài không đồng nhất hoặc chứa nhiều ý nghĩa ngữ cảnh.
- **Tăng hiệu quả sử dụng tài nguyên**: Tận dụng kỹ thuật nén mô hình như pruning (cắt tỉa trọng số không cần thiết) hoặc quantization (giảm độ chính xác số) để huấn luyện trên phần cứng hạn chế mà vẫn duy trì hiệu suất cao.

Những cải tiến trên không chỉ giúp mô hình giải quyết được các hạn chế hiện tại mà còn góp phần đặt nền tảng cho các nghiên cứu sâu hơn trong lĩnh vực xử lý ngôn ngữ tự nhiên. Với hướng đi rõ ràng và phương pháp tiếp cận khoa học, chúng em tin rằng chất lượng bài toán dịch máy sẽ ngày càng được nâng cao và mở rộng khả năng ứng dụng thực tiễn.

5. Kết luận

Qua quá trình thực hiện bài tập lớn, em đã học hỏi được nhiều kiến thức và kinh nghiệm quý báu trong lĩnh vực xử lý ngôn ngữ tự nhiên và huấn luyện mô hình deep learning. Cụ thể, em đã rút ra được những bài học sau:

- Biết cách chuẩn bị và điều chỉnh dữ liệu phù hợp với tài nguyên tính toán hiện có, từ việc thu thập, tiền xử lý đến tổ chức lại dữ liệu để tối ưu hóa hiệu suất mô hình.
- Hiểu rõ hơn về cơ chế hoạt động của mô hình Transformer, bao gồm cấu trúc encoder-decoder, cơ chế attention, và cách chúng giúp cải thiện hiệu quả trong bài toán dịch máy.
- Nắm bắt được quy trình xử lý dữ liệu cho bài toán dịch máy, từ token hóa, xây dựng từ điển, cho đến xử lý các trường hợp đặc biệt như ký tự ngoài từ vựng (OOV).
- Hiểu sâu hơn về các kiến trúc mạng nơ-ron nhân tạo, cách các thành phần như hàm mất mát (loss function) và bộ tối ưu hóa (optimizer) hoạt động để cải thiện quá trình huấn luyện mô hình.
- Biết cách điều chỉnh các tham số trong quá trình huấn luyện để giảm thiểu hiện tượng overfitting, đảm bảo mô hình đạt được sự cân bằng giữa hiệu quả trên tập huấn luyện và tập kiểm tra.

- Học được cách đánh giá chất lượng của mô hình dịch máy thông qua BLEU score, hiểu rõ ý nghĩa của thước đo này trong việc phản ánh mức độ tương đồng giữa đầu ra mô hình và bản dịch tham chiếu.
- Rèn luyện kỹ năng triển khai mô hình trên môi trường thực tế như Google Colab, tối ưu hóa việc sử dụng tài nguyên phần cứng, và quản lý quá trình huấn luyện dài hơi.

Bên cạnh đó, quá trình làm bài còn giúp em cải thiện kỹ năng lập kế hoạch, xử lý vấn đề và làm việc với các công cụ mã nguồn mở như PyTorch và Hugging Face.

Tuy nhiên, em cũng nhận ra rằng lĩnh vực deep learning còn nhiều thách thức, từ việc xử lý dữ liệu lớn đến việc tối ưu hóa mô hình trên các bài toán phức tạp. Những trải nghiệm này đã giúp em có cái nhìn tổng quan và sâu sắc hơn, tạo nền tảng để tiếp tục nghiên cứu và phát triển trong các dự án tương lai.

Tài liệu

[1] Long Doan **andothers**. "PhoMT: A High-Quality and Large-Scale Benchmark Dataset for Vietnamese-English Machine Translation". **in***Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*: 2021, **pages** 4495–4503.

[2] Ashish Vaswani **andothers**. "Attention Is All You Need". **in***Advances in Neural Information Processing Systems (NeurIPS)*: 2017, **pages** 5998–6008.