HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: LẬP TRÌNH VỚI PYTHON
MÃ HỌC PHẦN: INT13162

ĐỀ TÀI: XÂY DỰNG MÔ HÌNH DỊCH ANH-VIỆT

Các sinh viên thực hiện:

B22DCAT130	Trương Huy Hoàng
B22DCAT158	Ngô Mạnh Kiên
B22DCAT098	Lê Tiến Trường Giang

Tên nhóm: 20

Tên lớp: D22-035

Giảng viên hướng dẫn: Ths Ninh Thị Thu Trang

HÀ NỘI 3-2025

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

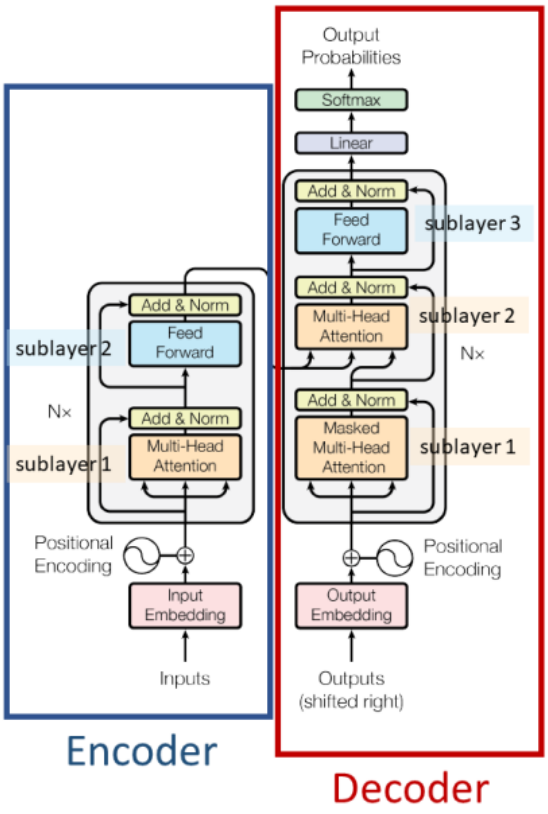
Trong bối cảnh toàn cầu hóa và sự phát triển mạnh mẽ của công nghệ, nhu cầu dịch thuật giữa các ngôn ngữ ngày càng trở nên quan trọng, đặc biệt là giữa tiếng Anh – ngôn ngữ quốc tế phổ biến – và tiếng Việt. Việc xây dựng một hệ thống dịch tự động có khả năng chuyển đổi chính xác và tự nhiên từ tiếng Anh sang tiếng Việt không chỉ giúp nâng cao khả năng tiếp cận tri thức cho người dùng Việt Nam mà còn mở rộng ứng dụng trong nhiều lĩnh vực như giáo dục, thương mại, và truyền thông.

Đề tài "Mô hình dịch tiếng Anh-Việt" nhằm mục tiêu nghiên cứu và phát triển một mô hình dịch máy sử dụng các kỹ thuật hiện đại trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), đặc biệt là các kiến trúc mạng nơ-ron sâu như Transformer. Mô hình được huấn luyện trên tập dữ liệu song ngữ lớn, có khả năng học ngữ cảnh và cú pháp giữa hai ngôn ngữ, từ đó tạo ra các bản dịch có chất lượng cao. Bên cạnh đó, đề tài cũng tập trung vào việc đánh giá hiệu quả của mô hình thông qua các chỉ số như BLEU score và thử nghiệm thực tế.

CHƯƠNG 2. CẤU TRÚC MÔ HÌNH

Mô hình dịch sử dụng kiến trúc Transformer- một kiến trúc mạng nơ-ron được thiết kế để xử lý chuỗi dữ liệu như văn bản, đặc biệt hiệu quả trong dịch máy, tóm tắt văn bản, sinh ngôn ngữ tự nhiên**,…**

Cấu trúc của mô hình gồm 2 phần chính: Encoder và Decoder. Cả Encoder và Decoder đều được xây dựng từ nhiều lớp, mỗi lớp có các thành phần khác nhau.



2.1 Embedding layer với positional encoding

2.1.1 Lớp Embedder

Lớp Embedder trong mô hình dịch có chức năng chính là ánh xạ các từ (tokens) trong câu đầu vào thành các vector thực (real-valued vectors) có kích thước cố định.

Cơ chế hoạt động:

- Khởi tạo (Constructor):

- vocab_size: Đây là kích thước của từ vựng, tức là số lượng token khác nhau có thể có trong bộ dữ liệu.
- d_model: Đây là số chiều của các vector nhúng (embedding vectors), tức là số lượng thông tin mô tả mỗi token.
- self.embed = nn.Embedding(vocab_size, d_model): Đoạn mã này tạo ra một lớp nhúng (embedding layer) có kích thước [vocab_size, d_model]. Mỗi token trong từ vựng sẽ được ánh xạ đến một vector nhúng có độ dài d_model.

- Phương thức forward:

- Nhận vào một tensor x (dữ liệu đầu vào, thường là các chỉ số đại diện cho các từ trong câu).
- Trả về các vector nhúng tương ứng với mỗi token trong đầu vào x bằng cách sử dụng lớp nhúng self.embed(x).

2.1.2 Lớp PositionalEncoder

Lớp PositionalEncoder (mã hóa vị trí) có vai trò bổ sung thông tin vị trí của các token trong chuỗi đầu vào để mô hình Transformer có thể hiểu được thứ tự của từ trong câu. Do Transformer không có cấu trúc tuần tự như RNN hay LSTM, nên nó không biết từ nào đến trước, từ nào đến sau nếu không có thêm thông tin về vị trí. Lớp này thực hiện việc mã hóa vị trí bằng cách tạo ra một ma trận vị trí (positional encoding matrix) dựa trên các hàm sin và cos theo công thức:

PE(pos, 2i) = sin(pos / 10000^(2i/d_model))

PE(pos, 2i+1) = cos(pos / 10000^(2i/d_model))

Trong đó, pos là vị trí token và i là chỉ số chiều trong vector nhúng. Ma trận này được tính sẵn và không cập nhật trong quá trình huấn luyện.

Trong phương thức forward, vector nhúng từ lớp Embedder sẽ được nhân với căn bậc hai của d_model nhằm duy trì độ lớn của giá trị nhúng, sau đó cộng thêm với positional encoding tương ứng. Kết quả là mô hình vừa có thông tin ngữ nghĩa từ vector nhúng, vừa có thông tin vị trí, giúp tăng khả năng học ngữ cảnh và cấu trúc câu của mô hình. Cuối cùng, một lớp dropout được áp dụng để tránh overfitting.

2.2 Self-Attention

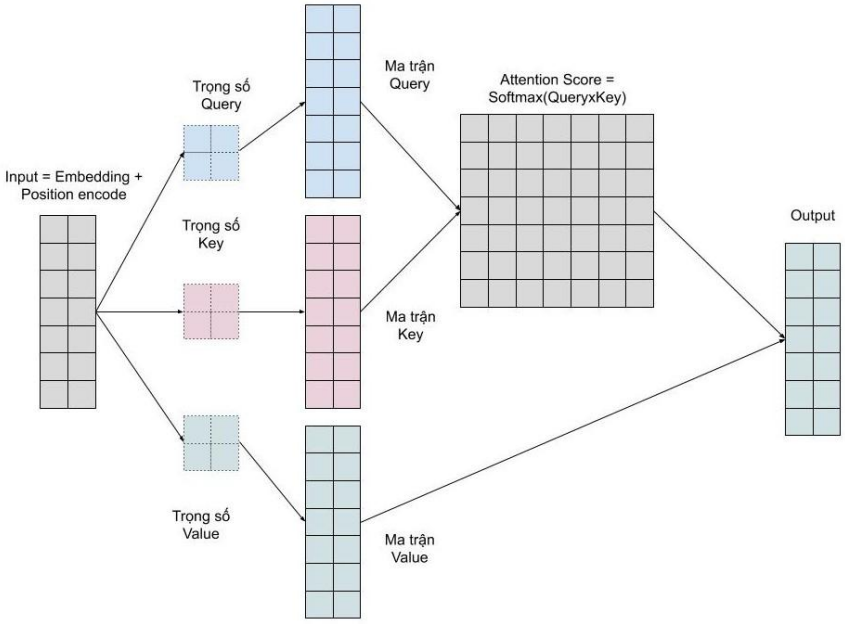
Cơ chế Self Attention cho phép mô hình khi mã hóa một từ có thể sử dụng thông tin của những từ liên quan tới nó.

Hàm attention tính toán điểm **scaled dot-product attention** giữa ba đầu vào chính là q (Query), k (Key) và v (Value). Sau đó, nó sử dụng softmax để chuẩn hóa các điểm chú ý và tính toán đầu ra của attention. Hàm còn hỗ trợ mask (lọc các vị trí không quan trọng) và dropout để giảm overfitting.

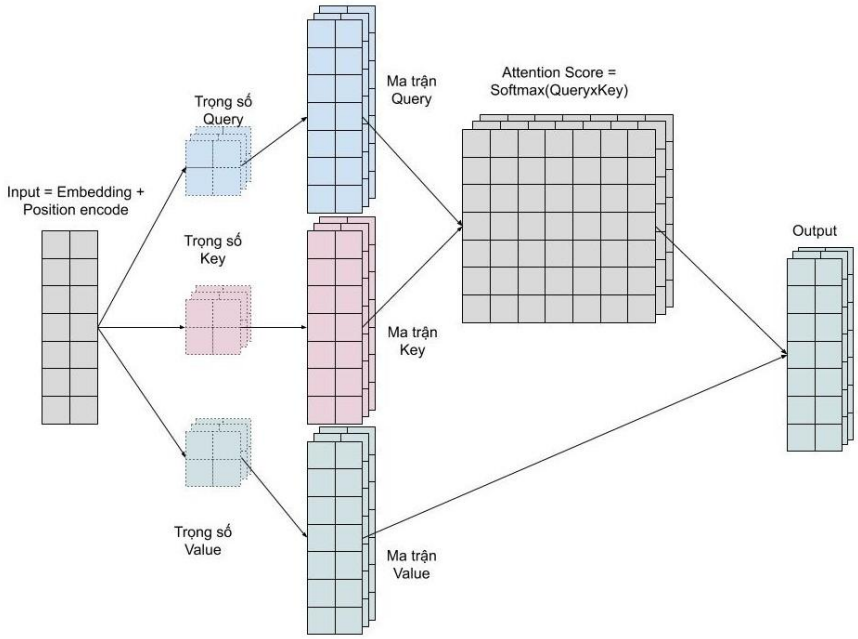
Các bước thực hiện:

- Tính toán điểm attention (scores): Tính toán điểm attention bằng cách nhân ma trận `q` (Query) với ma trận chuyển vị của `k` (Key) để tạo ra một ma trận điểm, rồi chia cho căn bậc hai của kích thước vector. Điều này giúp chuẩn hóa các giá trị và ngăn ngừa số quá lớn khi tính toán softmax.
- Áp dụng mask (nếu có): Nếu có mask (ví dụ như các vị trí padding hoặc các token không cần chú ý), các điểm attention tương ứng sẽ bị thay thế bằng một giá trị rất nhỏ (ví dụ -1e9), để chúng không ảnh hưởng đến quá trình tính toán softmax.

- Áp dụng hàm softmax lên các điểm attention để chuẩn hóa chúng thành các xác suất, sao cho tổng các giá trị trong mỗi hàng của ma trận scores bằng 1.
- Nếu dropout được kích hoạt, dropout sẽ được áp dụng lên các điểm attention để giảm overfitting.
- Cuối cùng, nhân điểm attention đã chuẩn hóa với ma trận v (Value) để tính toán đầu ra của cơ chế attention, phản ánh sự chú ý của từng token đối với các token khác trong câu đầu vào.



2.3 MultiHead Attention



Lớp MultiHeadAttention thực hiện cơ chế multi-head attention trong mô hình Transformer, cho phép mô hình tính toán sự chú ý giữa các token trong câu đầu vào bằng cách chia các vector attention thành nhiều phần nhỏ (đầu attention), mỗi đầu chú ý đến các mối quan hệ khác nhau trong dữ liệu.

Các bước thực hiện:

- Chuyển đổi đầu vào: Đầu vào q, k, và v được chuyển qua các lớp tuyến tính w_q, w_k, và w_v để thay đổi kích thước và chia nhỏ thành các phần theo số đầu attention.
- Tính toán attention cho từng đầu: Các vector q, k, và v được tách ra thành các phần (heads), sau đó tính toán điểm attention cho mỗi phần bằng cách sử dụng hàm attention.
- Kết hợp kết quả của các đầu attention: Kết quả từ các đầu attention được ghép lại và qua một lớp tuyến tính để tạo ra đầu ra cuối cùng.

2.4 Residuals Connection và Normalization Layer

Hai kỹ thuật giúp cho mô hình huấn luyện nhanh hội tụ hơn và trách mất mát thông tin trong quá trình huấn luyện mô hình, ví dụ như là thông tin của vị trí các từ được mã hóa.

2.4.1 Lớp LayerNorm

Lớp LayerNorm chuẩn hóa đầu vào theo từng chiều của tensor đầu vào (x) bằng cách sử dụng trung bình và phương sai của các giá trị trong mỗi chiều. Sau khi chuẩn hóa, lớp này áp dụng hai tham số học được là gamma (hệ số khuếch đại) và bias (độ dịch chuyển) để điều chỉnh kết quả chuẩn hóa. Điều này cho phép mô hình học được các thông số phù hợp thay vì chỉ dựa vào giá trị chuẩn hóa đơn giản.

Cơ chế hoạt động:

- Tính toán trung bình và phương sai:

- Tính toán trung bình (mean) và phương sai (var) của dữ liệu đầu vào x dọc theo chiều cuối cùng (theo chiều của các features).
- Sau đó tính toán độ lệch chuẩn (std), sử dụng eps để tránh chia cho 0 trong trường hợp phương sai bằng 0.

- Chuẩn hóa dữ liệu: Dữ liệu đầu vào được chuẩn hóa bằng cách trừ đi giá trị trung bình và chia cho độ lệch chuẩn. Điều này đảm bảo rằng dữ liệu đầu vào có trung bình bằng 0 và phương sai bằng 1.
- Áp dụng scaling và shifting: Sau khi chuẩn hóa, kết quả sẽ được nhân với hệ số khuếch đại gamma và cộng với độ dịch chuyển bias. Các tham số này được học trong quá trình huấn luyện, giúp mô hình có thể điều chỉnh kết quả chuẩn hóa để phù hợp với các đặc điểm của dữ liệu.
- Trả về kết quả chuẩn hóa: Kết quả cuối cùng là giá trị đã được chuẩn hóa và điều chỉnh bằng cách sử dụng các tham số gamma và bias. Đây là đầu ra của lớp LayerNorm và sẽ được truyền vào các lớp tiếp theo trong mô hình.

2.4.2 Lớp SublayerConnection

Lớp SublayerConnection trong mô hình đảm nhiệm vai trò kết nối giữa các tầng con (sublayers).

Cơ chế hoạt động:

- Chuẩn hóa đầu vào: Đầu vào x được chuẩn hóa bằng LayerNorm để đảm bảo ổn định giá trị khi đưa vào sublayer.
- Truyền qua tầng con (sublayer): Sau khi chuẩn hóa, đầu vào được truyền qua hàm sublayer, thường là một hàm attention hoặc feed-forward.
- Áp dụng Dropout: Dropout được sử dụng để giảm overfitting bằng cách ngẫu nhiên vô hiệu hóa một số phần tử trong đầu ra của sublayer trong quá trình huấn luyện.
- Thêm kết nối dư (residual): Đầu vào ban đầu x được cộng với đầu ra đã qua Dropout để tạo thành một kết nối dư — điều này giúp bảo tồn thông tin ban đầu và cải thiện việc lan truyền gradient.

2.4.3 Lớp FeedForward

Lớp FeedForward thực hiện một mạng nơ-ron truyền thẳng hai tầng áp dụng độc lập tại từng vị trí trong chuỗi đầu vào. Cấu trúc bao gồm hai lớp tuyến tính với một lớp ReLU ở giữa và dropout nhằm giảm overfitting. Lớp đầu tiên biến đổi đầu vào từ kích thước d_model sang d_ff, sau đó chuyển ngược về d_model, giúp mô hình học được các biểu diễn phức tạp hơn sau bước attention.

Dữ liệu đầu vào x được:

- Đưa qua lớp tuyến tính w_1 để tăng chiều không gian biểu diễn (từ d_model lên d_ff).
- Áp dụng hàm kích hoạt ReLU để tăng tính phi tuyến.
- Thực hiện dropout nhằm giảm quá khớp.
- Sau đó, đưa qua lớp w_2 để đưa kích thước trở lại d_model.

2.5 Encoder Layer

Lớp EncoderLayer là một thành phần cơ bản của khối mã hóa (Encoder) trong mô hình Transformer. Lớp này kết hợp hai thành phần chính: Multi-Head Attention và Feed Forward Network, được bao bọc bởi các Residual Connections và Layer Normalization thông qua lớp SublayerConnection.

Cơ chế hoạt động:

- Multi-Head Attention + Residual & LayerNorm:

- Đầu tiên, đầu vào x được đưa qua cơ chế self-attention bằng cách gọi self.attn(x, x, x, mask).

- Kết quả được bao bởi SublayerConnection, nghĩa là thực hiện chuẩn hóa đầu vào, truyền qua attention, rồi cộng residual (đầu vào ban đầu) vào kết quả đó.

- Feed Forward Network + Residual & LayerNorm:

- Kết quả từ attention tiếp tục được truyền qua mạng FeedForward với một lần chuẩn hóa khác.

- Cũng được bao bởi SublayerConnection, nên thực hiện residual + chuẩn hóa tương tự bước trên.

- Trả về đầu ra: là biểu diễn đã được học với ngữ cảnh và phi tuyến tính, có cùng kích thước với đầu vào.

2.6 Decoder Layer

Lớp DecoderLayer là một thành phần cơ bản trong bộ giải mã (Decoder) của mô hình Transformer, thực hiện ba bước xử lý chính gồm hai lớp attention và một lớp feed-forward, tất cả đều được bọc bởi Residual Connection và Layer Normalization.

Cơ chế hoạt động của lớp:

- Masked Multi-Head Self-Attention (attn1):

- self.attn1(x, x, x, trg_mask) áp dụng self-attention lên chính đầu vào đích (x).

- Sử dụng trg_mask để ngăn mô hình nhìn trước các token tương lai, đảm bảo tính tự hồi quy trong huấn luyện.

- Kết quả được chuẩn hóa và kết nối dư qua self.sublayer[0].

- Multi-Head Attention với đầu ra từ Encoder (attn2):

- self.attn2(x, mem, mem, src_mask) cho phép mô hình chú ý đến các thông tin từ đầu vào (encoder).

- Đây là nơi mô hình học cách liên kết giữa câu đầu vào và đầu ra.

- Cũng được chuẩn hóa và kết nối dư qua self.sublayer[1].

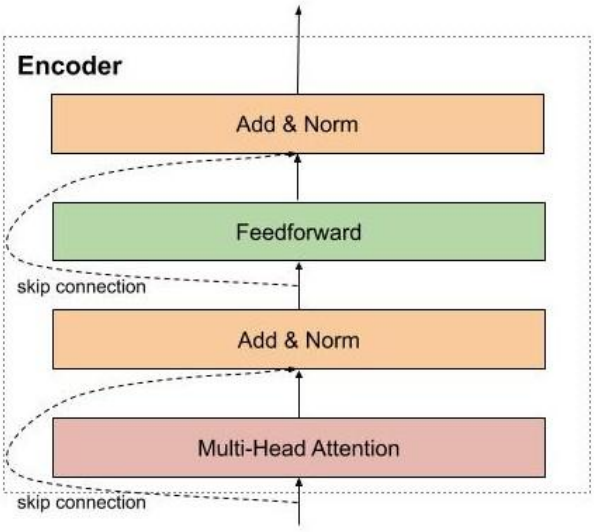
- Feed Forward Network (ffn):

- Áp dụng mạng nơ-ron hai tầng để đưa tính phi tuyến vào biểu diễn.

- Bọc trong self.sublayer[2] để thực hiện chuẩn hóa và residual connection.

2.7 Transformer hoàn chỉnh

2.7.1 Lớp Encoder



Lớp Encoder đóng vai trò xử lý và mã hóa chuỗi đầu vào thành các biểu diễn ngữ nghĩa giàu thông tin thông qua các bước: embedding, cộng positional encoding, rồi lần lượt qua nhiều lớp mã hóa (EncoderLayer). Kết quả được chuẩn hóa lần cuối để truyền sang bộ giải mã (Decoder). Đây là một phần cốt lõi trong mô hình dịch máy Transformer.

Cách thức hoạt động:

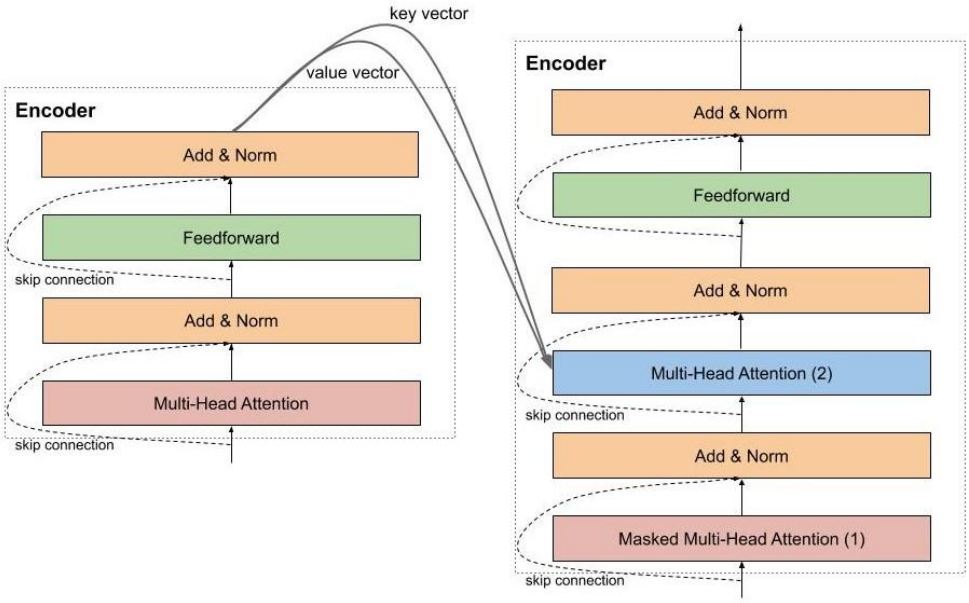
- Embedding Layer (self.embed): Chuyển đổi các token (chỉ số từ vựng) thành vector nhúng có chiều d_model.
- Positional Encoding (self.pe): Cộng thêm thông tin về vị trí của các token trong câu, giúp mô hình phân biệt vị trí từ trong chuỗi đầu vào.
- Encoder Layers (self.layers): Bao gồm N lớp EncoderLayer, mỗi lớp có:

- Multi-head self-attention: học mối quan hệ giữa các từ trong câu.
- Feed Forward Network: tăng độ phi tuyến và khả năng biểu diễn.
- Residual Connection + LayerNorm: giúp huấn luyện ổn định và hiệu quả.

- Layer Normalization cuối cùng (self.norm): Chuẩn hóa đầu ra cuối cùng của toàn bộ khối Encoder để ổn định gradient và tăng khả năng học.

2.7.2 Lớp Decoder

Kiến trúc của decoder rất giống với encoder, ngoại trừ có thêm một multi head attention nằm ở giữa dùng để học mối liên quan giữ từ đang được dịch với các từ được ở câu nguồn.



Lớp Decoder chịu trách nhiệm biến đổi đầu ra từng bước (câu đích) dựa trên thông tin ngữ cảnh từ Encoder và những từ đã sinh trước đó. Các lớp giải mã được tổ chức tuần tự, mỗi lớp gồm ba cơ chế chính (attention, cross-attention, FFN) cùng với chuẩn hóa.

Cơ chế hoạt động:

- Embedding Layer (self.embed): Chuyển các token đích thành vector nhúng kích thước d_model.
- Positional Encoding (self.pe): Bổ sung thông tin vị trí của token trong chuỗi đầu ra để mô hình biết vị trí tương đối của các từ.
- Decoder Layers (self.layers): Bao gồm N lớp DecoderLayer, mỗi lớp có 3 sub-layer:

- Masked Multi-head Self-Attention: chỉ cho phép mô hình nhìn thấy các token đã được sinh ra, ngăn việc "nhìn trước tương lai".
- Multi-head Attention với output từ Encoder (mem): giúp mô hình "chú ý" đến toàn bộ câu đầu vào.
- Feed Forward Network: tăng cường khả năng biểu diễn phi tuyến.
- Mỗi sub-layer đều sử dụng Residual Connection + Layer Normalization.

- Layer Normalization cuối cùng (self.norm): Chuẩn hóa đầu ra cuối cùng của khối Decoder để ổn định quá trình huấn luyện.

2.7.3 Transformer

Các thành phần chính:

- self.encoder: Khởi tạo lớp Encoder, nhận đầu vào là từ vựng của ngôn ngữ nguồn (src_vocab), số chiều của vector nhúng (d_model), số lớp Encoder (N), số đầu vào cho Multi-head Attention (heads), và tỉ lệ dropout (dropout).

- self.decoder: Khởi tạo lớp Decoder, nhận đầu vào là từ vựng của ngôn ngữ đích (trg_vocab), tương tự như Encoder.
- self.out: Một lớp Linear layer để chuyển đầu ra từ Decoder (một tensor có kích thước d_model) thành kích thước của từ vựng đích (trg_vocab), qua đó tạo ra dự đoán xác suất cho mỗi từ trong từ vựng đích.

Phương thức forward:

- Encoder: Đầu vào src (câu tiếng Anh) được đưa qua Encoder với src_mask để mã hóa thông tin.
- Decoder: Đầu vào trg được đưa qua Decoder, kết hợp với đầu ra từ Encoder (e_mem), và src_mask, trg_mask để tránh việc mô hình nhìn vào các token tương lai trong câu đích (tạo sự chặt chẽ trong quá trình học).
- Output Layer: Kết quả từ Decoder sau đó được đưa qua lớp out, giúp chuyển đổi vector thành xác suất phân phối cho từng từ trong từ vựng đích, làm cơ sở cho quá trình sinh từ (decoding).

CHƯƠNG 3. HUẤN LUYỆN MÔ HÌNH

3.1 Dữ liệu

Tải bộ dữ liệu song ngữ được thu thập trên TED bao gồm hơn 600k câu song ngữ anh-việt:
https://drive.google.com/file/d/1Fuo_ALIFKlUvOPbK5rUA5OfAS2wKn_95/view?usp=sharing

3.2 Tiền xử lý dữ liệu

Lớp Tokenize thực hiện:

- Tiền xử lý câu:
- Loại bỏ các ký tự không cần thiết: "*, ”, …, +, -, /, =, (,), :, [,], !, ;, etc.".
- Rút gọn các dấu câu lặp lại (!!! → !, ,,, → ,).
- Chuyển toàn bộ câu thành chữ thường (lowercase).
- Tách câu thành tokens (danh sách từ) bằng spaCy, loại bỏ các khoảng trắng dư thừa.

3.3 Thuật toán Beam Search

Sử dụng thuật toán Beam Search để sinh câu dịch đầu ra (chuỗi token ngôn ngữ đích) một cách hiệu quả bằng cách giữ lại k chuỗi có xác suất cao nhất tại mỗi bước, thay vì chỉ chọn từ xác suất cao nhất như trong Greedy Search.

Hàm beam_search: Sinh câu đầu ra bằng thuật toán Beam Search – mở rộng dần từng bước và chọn ra câu tốt nhất từ k ứng viên.

- Gọi init_vars() để khởi tạo.
- Ở mỗi bước thời gian i:
- Tạo trg_mask.
- Dự đoán token tiếp theo.
- Gọi k_best_outputs() để giữ lại k câu tốt nhất.
- Kiểm tra nếu cả k chuỗi đều đã sinh ra <eos>, dừng sớm và chọn câu tốt nhất theo công thức length penalty.
- Trả về câu sinh được dưới dạng chuỗi từ (itos).

3.4 Kỹ thuật huấn luyện

3.4.1 LabelSmoothing

Lớp LabelSmoothingLoss làm giảm hiện tượng overfitting và tăng khả năng tổng quát hóa của mô hình bằng kỹ thuật Label Smoothing.

Thay vì gán xác suất 1 cho nhãn đúng và 0 cho các nhãn còn lại, Label Smoothing sẽ:

- Gán nhãn đúng là confidence = 1 - smoothing
- Gán các nhãn sai một xác suất nhỏ đều nhau smoothing / (classes - 2)

Giúp mô hình không quá tự tin vào nhãn đúng → tổng quát hóa tốt hơn.

Chi tiết:

- self.confidence = 1.0 - smoothing: Tỷ lệ xác suất cho nhãn đúng.
- true_dist.fill_(self.smoothing / (self.cls - 2)): Gán xác suất nhỏ đều cho tất cả nhãn sai.
- true_dist.scatter_(1, target.data.unsqueeze(1), self.confidence): Gán xác suất cao (confidence) cho nhãn đúng.
- true_dist[:, self.padding_idx] = 0: Không tính loss với token padding.
- true_dist.index_fill_(0, mask.squeeze(), 0.0): Gán 0 hoàn toàn cho dòng có toàn padding.

Hàm forward:

- pred.log_softmax(...): Tính log-probability đầu ra.
- Tạo phân phối nhãn mịn true_dist.
- Trả về trung bình loss theo công thức cross-entropy mềm:

$$\text{loss} = -\sum(\text{true\_dist} \times \log(\text{pred}))$$

3.4.2 Optimizer

Learning rate được lớp ScheduledOptim điều chỉnh trong suốt quá trình học theo công thức sau:

$$lr\_rate = d_{d\_model}^{-0.5} * min(step\_num^{-0.5}, step\_num * warmup\_steps^{-1.5})$$

Cơ chế hoạt động:

- Khởi tạo optimizer có lịch điều chỉnh learning rate.
- step_and_update_lr(): Cập nhật learning rate và thực hiện bước huấn luyện.
- zero_grad(): Xóa gradient trước mỗi bước huấn luyện.
- _get_lr_scale(): Tính hệ số điều chỉnh learning rate theo công thức Transformer.
- _update_learning_rate(): Cập nhật learning rate mới sau mỗi bước.
- state_dict(): Lưu trạng thái optimizer để tiếp tục huấn luyện sau này.
- load_state_dict(): Tải lại trạng thái optimizer từ lần huấn luyện trước.

3.5 Kết quả

```
[ ] sentence='My family was not poor , and myself , I had never experienced hunger .'
    trans_sent = translate_sentence(sentence, model, SRC, TRG, opt['device'], opt['k'], opt['max_strlen'])
    trans_sent
```

```
'gia đình tôi không nghèo, và bản thân tôi, tôi chưa bao giờ trải qua nạn đói.'
```

KẾT LUẬN

Đề tài đã xây dựng thành công một mô hình dịch Anh–Việt dựa trên kiến trúc Transformer – một trong những kiến trúc mạnh mẽ và hiệu quả nhất hiện nay trong lĩnh vực xử lý ngôn ngữ tự nhiên. Mô hình có khả năng tiếp nhận câu đầu vào tiếng Anh và sinh ra câu dịch tiếng Việt một cách hợp lý về ngữ nghĩa và cú pháp trong nhiều trường hợp.

Trong quá trình xây dựng và huấn luyện, nhóm đã áp dụng một số kỹ thuật giúp cải thiện hiệu quả mô hình như: Label Smoothing để làm giảm hiện tượng quá khớp, Beam Search để tăng chất lượng câu dịch, và lịch trình điều chỉnh tốc độ học (learning rate scheduling) giúp tối ưu hóa quá trình huấn luyện. Các thành phần này đóng vai trò quan trọng trong việc nâng cao độ ổn định và khả năng sinh văn bản của mô hình.

TÀI LIỆU THAM KHẢO

[1] Transformer: https://arxiv.org/pdf/1706.03762

[2] Dữ liệu để huấn luyện: https://drive.google.com/file/d/1Fuo_ALIFKlUvOPbK5rUA5OfAS2wKn_95/view

ĐÓNG GÓP CÔNG VIỆC

Họ tên	MSV	Mức độ đóng góp
Trương Huy Hoàng	B22DCAT130	20%
Ngô Mạnh Kiên	B22DCAT158	70%
Lê Tiến Trương Giang	B22DCAT098	10%