TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Quốc Duy - 52200196
Huỳnh Tấn Nhã - 52200026
Đoàn Nguyễn Minh Khoa - 51900358

BÁO CÁO CUỐI KỲ
NHẬP MÔN
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN

Nguyễn Quốc Duy - 52200196
Huỳnh Tấn Nhã - 52200026
Đoàn Nguyễn Minh Khoa - 51900358

BÁO CÁO CUỐI KỲ
NHẬP MÔN
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Người hướng dẫn
PGS.TS. Lê Anh Cường

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Trước tiên, chúng em xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến **PSG.TS Lê Anh Cường** - Thầy là người đã luôn hỗ trợ và hướng dẫn tận tình cho chúng em trong suốt quá trình nghiên cứu và hoàn thành bài báo cáo của môn "Nhập môn Xử lý ngôn ngữ tự nhiên".

Tiếp theo, chúng em xin gửi lời cảm ơn đến **Khoa Công nghệ thông tin - Trường Đại học Tôn Đức Thắng** vì đã tạo điều kiện cho chúng em được học tập và nghiên cứu môn học này. Khoa đã luôn sẵn sàng chia sẻ các kiến thức bổ ích cũng như chia sẻ các kinh nghiệm tham khảo tài liệu, giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu mà còn giúp ích cho việc học tập và rèn luyện trong quá trình thực hành tại **trường Đại học Tôn Đức Thắng**.

Cuối cùng, sau khoảng thời gian học tập trên lớp chúng em đã hoàn tất bài báo cáo nhờ vào sự hướng dẫn, giúp đỡ và những kiến thức học hỏi được từ Quý thầy cô. Do giới hạn về mặt kiến thức và khả năng lý luận nên tôi vẫn còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của Quý thầy cô giáo để bài báo cáo của chúng em được hoàn thiện hơn. Hơn nữa, nhờ những góp ý từ thầy cô và các bạn hữu, chúng em sẽ hoàn thành tốt hơn ở những bài nghiên cứu, tiểu luận trong tương lai.

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 06 tháng 04 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Nguyễn Quốc Duy

Huỳnh Tấn Nhã

Đoàn Nguyễn Minh Khoa

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của PGS.TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 6 tháng 4 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Nguyễn Quốc Duy

Huỳnh Tấn Nhã

Đoàn Nguyễn Minh Khoa

TÓM TẮT

Báo cáo tập trung vào hai nhiệm vụ chính: (1) nghiên cứu các thuật toán Học tăng cường (Reinforcement Learning- RL) áp dụng trong mô hình ngôn ngữ lớn và (2) xây dựng mô hình dịch máy từ tiếng Anh sang tiếng Việt.

Ở nhiệm vụ đầu tiên, các thuật toán RL được phân tích chi tiết qua công thức, giải thuật, ý nghĩa và so sánh về ưu, nhược điểm. Một thuật toán RL được minh họa bằng mã nguồn chạy được, kèm dữliệu huấn luyện để tối ưu hóa hiệu suất mô hình ngôn ngữ.

Ở nhiệm vụ thứ hai, hai loại mô hình được phát triển: Encoder-Decoder Transformer và GPT (Generative Pretrained Transformer), với hai cách tiếp cận: sử dụng mô hình pretrained và huấn luyện từ đầu. Quá trình tiền xử lý dữ liệu được trình bày qua tokenizer và thuật toán Byte Pair Encoding (BPE). Hiệu suất các mô hình được đánh giá trên tập kiểm tra bằng các chỉ số ROUGE (ROUGE-1, ROUGE-2, ROUGE-L) và BLEU, từ đó so sánh chất lượng dịch của các mô hình.

Báo cáo này cung cấp cái nhìn toàn diện về ứng dụng RL trong mô hình ngôn ngữ và hiệu quả của các mô hình dịch máy, kết hợp lý thuyết, triển khai thực tiễn và đánh giá định lượng

MỤC LỤC

DANH MỤC HÌNH VẼ

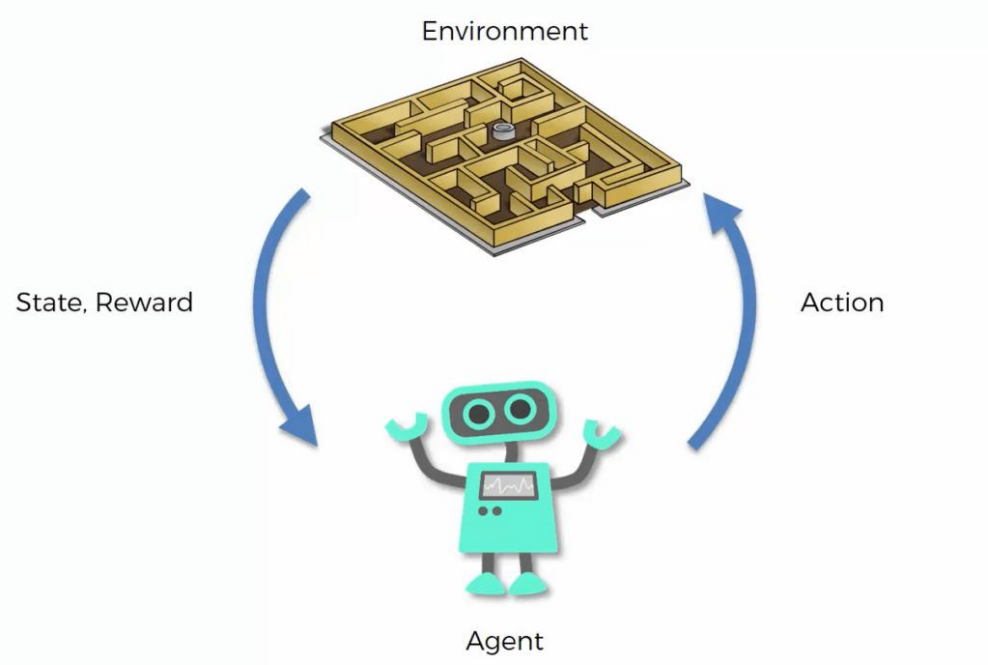
DANH MỤC CÁC CHỮ VIẾT TẮT

SMT	Statistical Machine Translation
RBMT	Rule-based Machine Translation
NMT	Neural Machine Translation
LSTM	Long Short-Term Memory
LLMs	Large Language Models

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1 Tìm hiểu Reinforcement Learning mô hình ngôn ngữ lớn

1.1.1 Khái quát về Reinforcement Learning (RL)



Hình 1.1: Mô tả về các hoạt động Reinforcement Learning (RL)

Reinforcement Learning (RL) là một nhánh của học máy, trong đó một tác nhân (agent) học cách thực hiện hành động thông qua tương tác với môi trường (environment). Mỗi lần tác nhân thực hiện hành động, môi trường phản hồi dưới dạng một phần thưởng (reward) và trạng thái mới (next state). Mục tiêu của tác nhân là tìm chính sách (policy) tối ưu để tối đa hóa tổng phần thưởng tích lũy.

Mô hình hóa MDP (Markov Decision Process) là một khung lý thuyết toán học để mô tả quá trình ra quyết định trong một môi trường mà kết quả tương lai phụ thuộc vào hành động hiện tại, nhưng không phụ thuộc vào quá khứ – tức là có tính Markov (không nhớ quá khứ).

Một MDP được định nghĩa bởi 5 thành phần:

1. Tập trạng thái ***S***: Mô tả tất cả các trạng thái có thể mà agent có thể rơi vào. Ví dụ: Trong ngôn ngữ, một trạng thái có thể là một chuỗi văn bản đang sinh ra.

2. Tập hành động $\boldsymbol{A}$: Tập hợp các hành động mà agent có thể thực hiện tại một trạng thái. Trong LLM, hành động có thể là chọn từ tiếp theo.
3. Hàm chuyển tiếp $\boldsymbol{P(s'|s,a)}$**:** Xác suất chuyển từ trạng thái $\boldsymbol{s}$ sang trạng thái $\boldsymbol{s'}$ khi thực hiện hành động $\boldsymbol{a}$. Mô hình hóa động lực môi trường.
4. Hàm phần thưởng $\boldsymbol{R(s,a)}$: Trả về phần thưởng nhận được khi hành động $\boldsymbol{a}$ được thực hiện tại trạng thái $\boldsymbol{s}$. Trong LLM, phần thưởng có thể đến từ đánh giá của người dùng hoặc mô hình đánh giá.
5. Hệ số chiết khấu $\boldsymbol{\gamma} \in [\boldsymbol{0},\boldsymbol{1}]$: Đo lường mức độ ưu tiên của phần thưởng hiện tại so với phần thưởng tương lai. $\boldsymbol{\gamma}$ càng gần 1 thì tác nhân càng quan tâm đến phần thưởng dài hạn.

Mô hình hóa MDP cho phép biến quá trình sinh văn bản thành một chuỗi các hành động có thể đánh giá và học hỏi.

Trong RLHF (Reinforcement Learning from Human Feedback), mỗi phản hồi từ người dùng về đầu ra của mô hình sẽ được quy đổi thành phần thưởng trong MDP.

Từ đó, thuật toán RL (như PPO) có thể tối ưu chính sách sinh văn bản sao cho đáp ứng tốt hơn kỳ vọng của người dùng.

1.1.2 Ứng dụng Reinforcement Learning (RL) trong Large Language Models (LLMs)

Trong quá trình huấn luyện các mô hình ngôn ngữ lớn (Large Language Models – LLMs), một bước quan trọng giúp mô hình phản hồi tốt hơn với con người là học tăng cường từ phản hồi của con người – gọi tắt là RLHF (Reinforcement Learning from Human Feedback).

Thông thường, LLMs được huấn luyện qua ba giai đoạn:

1. Tiền huấn luyện (Pre-training): Mô hình học cách dự đoán từ tiếp theo dựa trên hàng tỷ câu trong các văn bản tiếng người.

2. Huấn luyện có giám sát (Supervised Fine-tuning): Mô hình được chỉnh lại bằng các ví dụ đã được con người gán nhãn, để thực hiện các nhiệm vụ cụ thể hơn.
3. Học tăng cường từ phản hồi con người (RLHF): Đây là bước mô hình được tinh chỉnh bằng phản hồi đánh giá của con người, để tạo ra câu trả lời tự nhiên, hữu ích và đúng mực hơn.

Trong RLHF, khi mô hình tạo ra nhiều câu trả lời khác nhau cho một câu hỏi, con người sẽ đánh giá và xếp hạng những câu trả lời này. Từ đó, một mô hình phần thưởng (reward model) sẽ học cách "bắt chước" sự đánh giá của con người. Cuối cùng, thuật toán học tăng cường như PPO (Proximal Policy Optimization) sẽ được dùng để cập nhật mô hình sinh câu trả lời sao cho những câu có chất lượng cao được ưu tiên sinh ra nhiều hơn.

Nhờ cách tiếp cận này, mô hình không chỉ trả lời đúng ngữ pháp hay sát nghĩa, mà còn thể hiện sự lịch sự, hữu ích, và phù hợp với tình huống thực tế. RLHF cũng giúp giảm nguy cơ mô hình tạo ra nội dung sai lệch, thiên kiến hoặc không an toàn – điều rất quan trọng trong các ứng dụng như chatbot, trợ lý ảo, chăm sóc khách hàng,...

Tóm lại**,** Reinforcement Learning, đặc biệt thông qua RLHF, là bước "tinh chỉnh mang tính con người", giúp các mô hình ngôn ngữ không chỉ mạnh mẽ mà còn đáng tin cậy và phù hợp với xã hội hơn.

1.1.3 Các thuật toán Reinforcement Learning (RL) phổ biến trong Large Language Models (LLMs)

1.1.3.1 REINFORCE (Monte Carlo Policy Gradient)

Mục đích: Học chính sách trực tiếp thông qua gradient của tổng phần thưởng.

Cơ chế: Thu thập toàn bộ tập trải nghiệm (episode) rồi tính gradient:

$$\nabla_\theta J(\theta) = E[G_t \nabla_\theta \log \pi_\theta (a_t|s_t)]$$

Trong đó: $G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$

Ý nghĩa: Học chính sách bằng cách ước lượng trực tiếp gradient của kỳ vọng phần thưởng. Dựa vào tập dữ liệu thu thập đầy đủ (episodes) để điều chỉnh chính sách.

Giải thuật:

1. Chạy mô hình để thu thập toàn bộ episode: chuỗi (s_t , a_t , r_t)
2. Tính tổng phần thưởng sau thời điểm t: $\boldsymbol{G_t = \sum_{k=t}^{T} \gamma^{k-t} r_k}$
3. Cập nhật chính sách theo:

$$\boldsymbol{\nabla_\theta J(\theta) = E[G_t \nabla_\theta \log \pi_\theta (a_t|s_t)]}$$

Ưu điểm: Triển khai đơn giản.

Nhược điểm: High variance, cần nhiều episode để hội tụ.

1.1.3.2 Actor–Critic (A2C/A3C)

Mục đích: Kết hợp ưu điểm của Policy Gradient và Value-Based để giảm phương sai.

Cơ chế:

- Actor: Cập nhật chính sách theo hướng gradient của log xác suất.
- Critic: Ước lượng hàm giá trị $\boldsymbol{V(S)}$, tính Ưu thế (Advantage):
  - $\boldsymbol{\widehat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t)}$
- Cập nhật: Actor sử dụng $\boldsymbol{\nabla_\theta \log \pi_\theta (a_t|s_t) \widehat{A}_t}$

Ý nghĩa: Kết hợp chính sách (actor) và giá trị trạng thái (critic) để giảm phương sai, giúp học ổn định hơn.

Giải thuật:

1. **Critic** ước lượng hàm giá trị $V(s_t)$
2. Tính advantage (ưu thế): $\boldsymbol{\widehat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t)}$
3. Actor cập nhật chính sách theo: $\boldsymbol{\nabla_\theta \log \pi_\theta (a_t|s_t) \widehat{A}_t}$

A2C: Đồng bộ.

A3C: Phi đồng bộ (multi-thread, chạy song song).

Ưu điểm: Phương sai thấp hơn REINFORCE, hội tụ nhanh hơn.

Nhược điểm: Phải đồng bộ hai mạng actor và critic và có thể phức tạp khi triển khai.

1.1.3.3 Trust Region Policy Optimization (TRPO)

Mục đích: Bảo đảm mỗi bước cập nhật chính sách không đi quá xa so với chính sách trước, duy trì ổn định.

Cơ chế: Giới hạn khoảng cách KL-divergence giữa chính sách mới và cũ:

$\mathbf{maximize}_{\theta}\, \boldsymbol{E_t}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{\boldsymbol{A}}_t\right]$ điều kiện $\boldsymbol{E_t}\left[\boldsymbol{KL}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_{old}}, \boldsymbol{\pi}_{\boldsymbol{\theta}}\right)\right] \leq \boldsymbol{\delta}$

Ý nghĩa: Giới hạn cập nhật chính sách trong một "vùng tin cậy" bằng cách kiểm soát độ chênh lệch (KL divergence) giữa chính sách cũ và mới.

Giải thuật:

1. Tối đa hóa: $\boldsymbol{E_t}\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{\boldsymbol{A}}_t\right]$
2. Với điều kiện: $\boldsymbol{E_t}\left[\boldsymbol{KL}\left(\boldsymbol{\pi}_{\boldsymbol{\theta}_{old}}, \boldsymbol{\pi}_{\boldsymbol{\theta}}\right)\right] \leq \boldsymbol{\delta}$

Ưu điểm: Đảm bảo ổn định cao.

Nhược điểm: Tính toán phức tạp, kém linh hoạt

1.1.3.4 PPO (Proximal Policy Optimization)

Mục đích: Giản lược TRPO, giữ ổn định mà đơn giản hóa tính toán.

Cơ chế: Sử dụng hàm mất mát cắt (clipped loss):

$$\boldsymbol{L(\theta)} = \boldsymbol{E_t}\left[\mathbf{min}\left(\boldsymbol{r_t(\theta)}\widehat{\boldsymbol{A}}_t, \boldsymbol{clip}(\boldsymbol{r_t(\theta)}, \mathbf{1}-\boldsymbol{\epsilon}, \mathbf{1}+\boldsymbol{\epsilon})\widehat{\boldsymbol{A}}_t\right)\right]$$

Với $\boldsymbol{r_t(\theta)} = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

Ý nghĩa: Là phiên bản đơn giản hoá của TRPO nhưng vẫn giữ được độ ổn định cập nhật.

Giải thuật:

1. Tính tỷ lệ chính sách: $\boldsymbol{r_t(\theta)} = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$
2. Tối ưu hàm mất mát:

$$\boldsymbol{L(\theta)} = \boldsymbol{E_t}\left[\mathbf{min}\left(\boldsymbol{r_t(\theta)}\widehat{\boldsymbol{A}}_t, \boldsymbol{clip}(\boldsymbol{r_t(\theta)}, \mathbf{1}-\boldsymbol{\epsilon}, \mathbf{1}+\boldsymbol{\epsilon})\widehat{\boldsymbol{A}}_t\right)\right]$$

Ưu điểm: Dễ triển khai, hiệu quả, ổn định.

Nhược điểm: Cần điều chỉnh chính xác giá trị epsilon.

1.1.3.5 SAC (Soft Actor–Critic)

Mục đích: Kết hợp tối đa hoá phần thưởng và entropy để cân bằng khai thác và khám phá.

Cơ chế: Actor–Critic off-policy, thêm thành phần entropy vào hàm mất mát:

$$J(\pi) = \sum_t E_{(s_t,a_t)\sim p_\pi}[r(s_t,a_t)+\alpha H(\pi(\cdot|s_t))]$$

Ý nghĩa: Khuyến khích chính sách vừa nhận phần thưởng cao, vừa đa dạng hành động (entropy cao) để tăng khả năng khám phá.

Giải thuật:

1. Mục tiêu tối đa:

$$J(\pi) = \sum_t E_{(s_t,a_t)\sim p_\pi}[r(s_t,a_t)+\alpha H(\pi(\cdot|s_t))]$$

2. Cập nhật cả Actor và Critic thông qua gradient ngẫu nhiên (stochastic updates).

Ưu điểm: Khám phá hiệu quả, ổn định trong nhiều môi trường.

Nhược điểm: Phức tạp hơn, thường ít sử dụng trong RLHF.

1.1.4 So sánh tổng hợp các thuật toán

Thuật toán	**Ý nghĩa chính**	**Ưu điểm**	**Nhược điểm**	**Ứng dụng**
REINFORCE	Học chính sách qua gradient phần thưởng	Dễ cài đặt, đơn giản	Phương sai cao, cần nhiều episode	Cơ bản, lý thuyết nền
Actor–Critic (A2C/A3C)	Kết hợp actor và critic để giảm phương sai	Hội tụ nhanh hơn REINFORCE	Cần đồng bộ hóa 2 mạng	Tốt cho môi trường phức tạp
TRPO	Giữ chính sách mới	Ổn định, kiểm soát tốt	Tính toán KL phức tạp	Nghiên cứu cao cấp

	không lệch quá xa			
PPO	Đơn giản hóa TRPO, clip cập nhật	Ổn định, dễ triển khai	Cần chọn ε phù hợp	GPT, ChatGPT (OpenAI)
SAC	Kết hợp phần thưởng + entropy (khám phá)	Khám phá mạnh, off-policy	Phức tạp, ít dùng trong NLP	Nhiều môi trường robot, điều khiển

1.2 Mô hình Sequence-to-Sequence sử dụng RNNs

1.2.1 Khái niệm và kiến trúc cơ bản

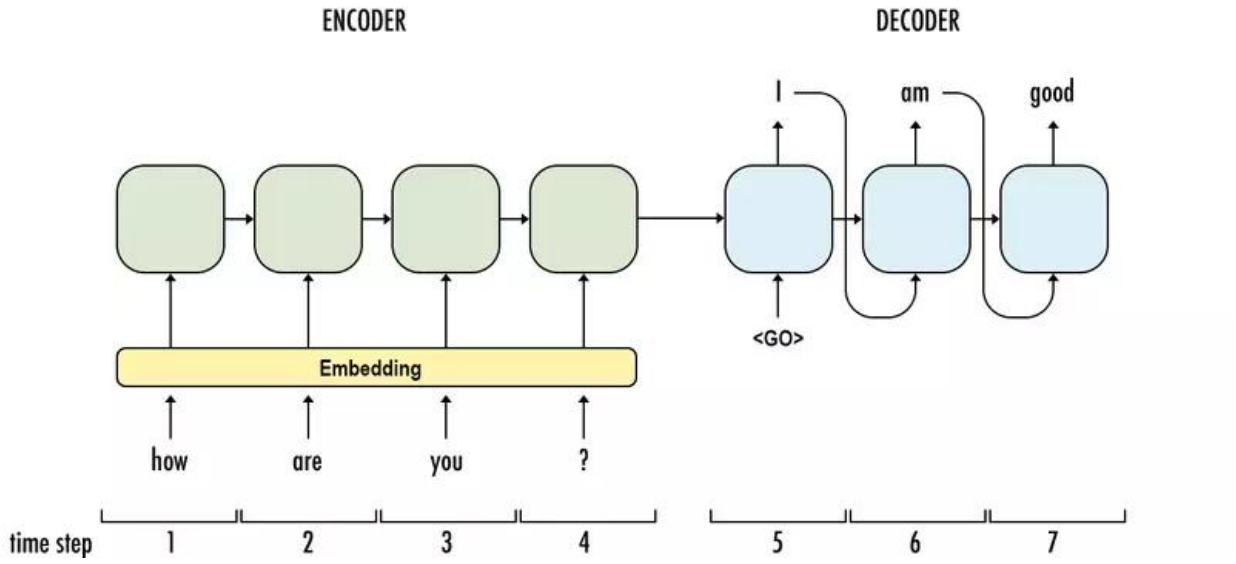
Mô hình Sequence-to-Sequence (Seq2Seq) là một kiến trúc mạng nơ-ron được thiết kế để xử lý các bài toán mà cả đầu vào (input) và đầu ra (output) đều là các chuỗi (sequences) có độ dài thay đổi. Điểm đặc trưng của mô hình này là khả năng ánh xạ một chuỗi đầu vào có độ dài T_x thành một chuỗi đầu ra có độ dài T_y, trong đó T_x và T_y có thể khác nhau. Nhiều bài toán quan trọng trong Xử lý ngôn ngữ tự nhiên (NLP) thuộc loại này, ví dụ như:

- Dịch máy (Machine Translation): Input là một câu trong ngôn ngữ nguồn (ví dụ: “how are you?”) và output là câu tương ứng trong ngôn ngữ đích (ví dụ: “bạn khoẻ không?”).
- Hỏi đáp (Question Answering): Input là một câu hỏi và output là câu trả lời.
- Tóm tắt văn bản (Text Summarization): Input là một đoạn văn bản dài và output là một bản tóm tắt ngắn gọn.
- Tạo chú thích ảnh (Image Captioning): Input là đặc trưng của ảnh (có thể xem như một chuỗi) và output là mô tả về ảnh đó.

Kiến trúc Seq2Seq truyền thống thường được xây dựng dựa trên Mạng Nơ-ron Hồi quy (Recurrent Neural Networks - RNNs) hoặc các biến thể của nó như

LSTM (Long Short-Term Memory) hay GRU (Gated Recurrent Unit). Mô hình bao gồm hai thành phần chính:

- Bộ mã hóa (Encoder): Có nhiệm vụ đọc tuần tự chuỗi đầu vào (x_1,x_2 ,...,x_{Tx}) và nén thông tin của nó thành một vector biểu diễn ngữ cảnh (context vector), thường ký hiệu là c. Vector này được kỳ vọng sẽ tóm lược được ý nghĩa của toàn bộ chuỗi đầu vào. Thông thường, Encoder là một mạng RNN, và vector ngữ cảnh c chính là trạng thái ẩn cuối cùng của RNN sau khi đọc hết chuỗi đầu vào.
- Bộ giải mã (Decoder): Cũng là một mạng RNN khác, nhận vector ngữ cảnh c từ Encoder và sinh ra tuần tự chuỗi đầu ra (y_1,y_2,...,y_{Ty}). Tại mỗi bước thời gian t, Decoder dự đoán phần tử tiếp theo y_t dựa trên vector ngữ cảnh c, trạng thái ẩn s_{t-1} của chính nó ở bước trước, và phần tử y_{t-1} đã được sinh ra trước đó. Quá trình này bắt đầu bằng một token đặc biệt (ví dụ: <SOS>) và kết thúc khi sinh ra token kết thúc (ví dụ: <EOS>) hoặc đạt độ dài tối đa cho phép.



Hình 1.2: Encoder và Decoder

1.2.2 Hạn chế của Mô hình Seq2Seq dựa trên RNNs

Mặc dù đạt được những thành công nhất định, kiến trúc Seq2Seq sử dụng RNNs bộc lộ một số hạn chế đáng kể:

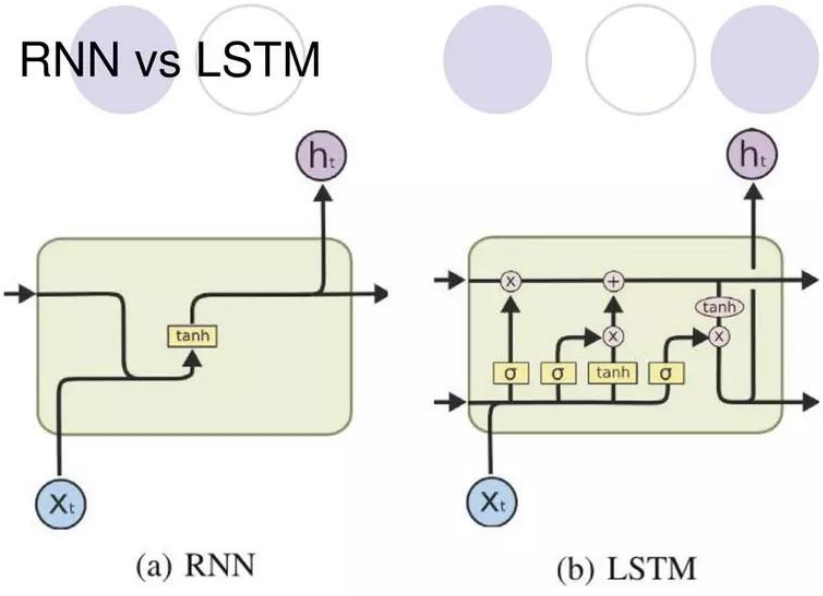
- Tốc độ huấn luyện chậm và khó song song hóa: Bản chất tính toán tuần tự của RNNs (trạng thái ẩn tại bước t phụ thuộc vào trạng thái ẩn tại bước t−1) gây khó khăn cho việc tận dụng khả năng tính toán song song mạnh mẽ của các bộ xử lý đồ họa (GPU). Việc huấn luyện mô hình RNN trên các chuỗi dài đòi hỏi phải thực hiện lan truyền ngược qua thời gian (Backpropagation Through Time - BPTT). Do hạn chế về bộ nhớ và tính ổn định, người ta thường phải sử dụng phiên bản cắt ngắn (Truncated BPTT), nhưng điều này cũng không giải quyết triệt để vấn đề tốc độ. Thời gian huấn luyện các mô hình này, đặc biệt với dữ liệu lớn, thường rất lâu.
- Khó khăn trong việc xử lý phụ thuộc xa (Gradient Vanishing/Exploding): Trong quá trình BPTT, gradient (đạo hàm của hàm mất mát theo các tham số) được lan truyền ngược từ cuối chuỗi về đầu chuỗi. Với các chuỗi dài, gradient có xu hướng nhỏ dần đến mức gần bằng 0 (Gradient Vanishing) hoặc lớn đến mức vô hạn (Gradient Exploding) do phép nhân lặp đi lặp lại trong công thức đạo hàm chuỗi. Hiện tượng Gradient Vanishing đặc biệt phổ biến và nghiêm trọng, nó làm cho các tham số ở những bước thời gian đầu được cập nhật rất ít hoặc không được cập nhật, khiến mô hình không học được các phụ thuộc ngữ nghĩa hoặc cú pháp giữa các từ cách xa nhau trong câu
- "Nút thắt cổ chai" thông tin (Information Bottleneck): Toàn bộ thông tin của chuỗi đầu vào, dù dài hay phức tạp đến đâu, đều bị buộc phải nén vào một vector ngữ cảnh c có kích thước cố định. Vector này trở thành "nút thắt cổ chai", hạn chế khả năng truyền tải đầy đủ thông tin cần thiết từ Encoder sang Decoder, đặc biệt là với các chuỗi đầu vào dài. Thông tin ở phần đầu của chuỗi có nguy cơ bị "lãng quên" khi Encoder xử lý đến cuối chuỗi.

1.2.3 Giải pháp tạm thời của LSTM và GRU

Để giải quyết phần nào vấn đề Gradient Vanishing, các biến thể phức tạp hơn của RNN như Long Short-Term Memory (LSTM) [Hochreiter & Schmidhuber, 1997] và Gated Recurrent Unit (GRU) [Cho et al., 2014] đã được đề xuất.

LSTM: Sử dụng một cấu trúc cell phức tạp với các "cổng" (gate) - input gate, forget gate, output gate - và một "trạng thái cell" (cell state, có thể hiểu là "nhánh C" như bạn đề cập) hoạt động như một bộ nhớ dài hạn. Các cổng này điều khiển luồng thông tin đi vào, đi ra và lưu trữ trong cell state, cho phép LSTM học và duy trì thông tin qua các khoảng thời gian dài hơn nhiều so với RNN cơ bản, giảm thiểu đáng kể ảnh hưởng của Gradient Vanishing.

GRU: Là một biến thể đơn giản hóa của LSTM với ít cổng hơn (update gate và reset gate) nhưng vẫn giữ được hiệu quả trong việc xử lý phụ thuộc xa.



Hình 1.3: RNN và LSTM

1.2.4 Đặt vấn đề cho kiến trúc mới

Tóm lại, các mô hình Seq2Seq dựa trên RNN/LSTM/GRU tuy là một bước tiến lớn nhưng vẫn đối mặt với những thách thức cố hữu: tốc độ huấn luyện bị giới hạn bởi tính tuần tự và khó khăn trong việc nắm bắt hoàn hảo các phụ thuộc rất xa trong dữ liệu. Nhu cầu về một kiến trúc mới có khả năng:

- Tận dụng tối đa khả năng tính toán song song của phần cứng hiện đại (GPU/TPU) để tăng tốc độ huấn luyện.
- Xử lý hiệu quả các phụ thuộc xa trong chuỗi mà không bị giới hạn bởi cơ chế truyền thông tin tuần tự hay "nút thắt cổ chai" của vector ngữ cảnh.

đã trở nên cấp thiết và chính trong bối cảnh này, kiến trúc Transformer đã ra đời và tạo nên một cuộc cách mạng trong lĩnh vực NLP.

1.3 Mô hình Transformer

1.3.1 Giới thiệu về vượt qua giới hạn của RNNs

Như đã trình bày ở mục 1.2, các mô hình Seq2Seq dựa trên RNNs, mặc dù hiệu quả, nhưng gặp phải những hạn chế về tốc độ huấn luyện do tính tuần tự và khó khăn trong việc nắm bắt các phụ thuộc xa. Để giải quyết những thách thức này, Vaswani và cộng sự đã giới thiệu kiến trúc Transformer trong bài báo nổi tiếng "Attention Is All You Need" vào năm 2017 [Vaswani et al., 2017].

Transformer cũng duy trì cấu trúc Encoder-Decoder tổng thể như các mô hình Seq2Seq trước đó, nhưng loại bỏ hoàn toàn các kết nối hồi quy (recurrent connections). Thay vào đó, nó dựa chủ yếu vào một cơ chế gọi là "Self-Attention" (hay tổng quát hơn là Attention) để mô hình hóa các phụ thuộc giữa các từ trong chuỗi đầu vào và đầu ra, cũng như giữa chuỗi đầu vào và đầu ra.

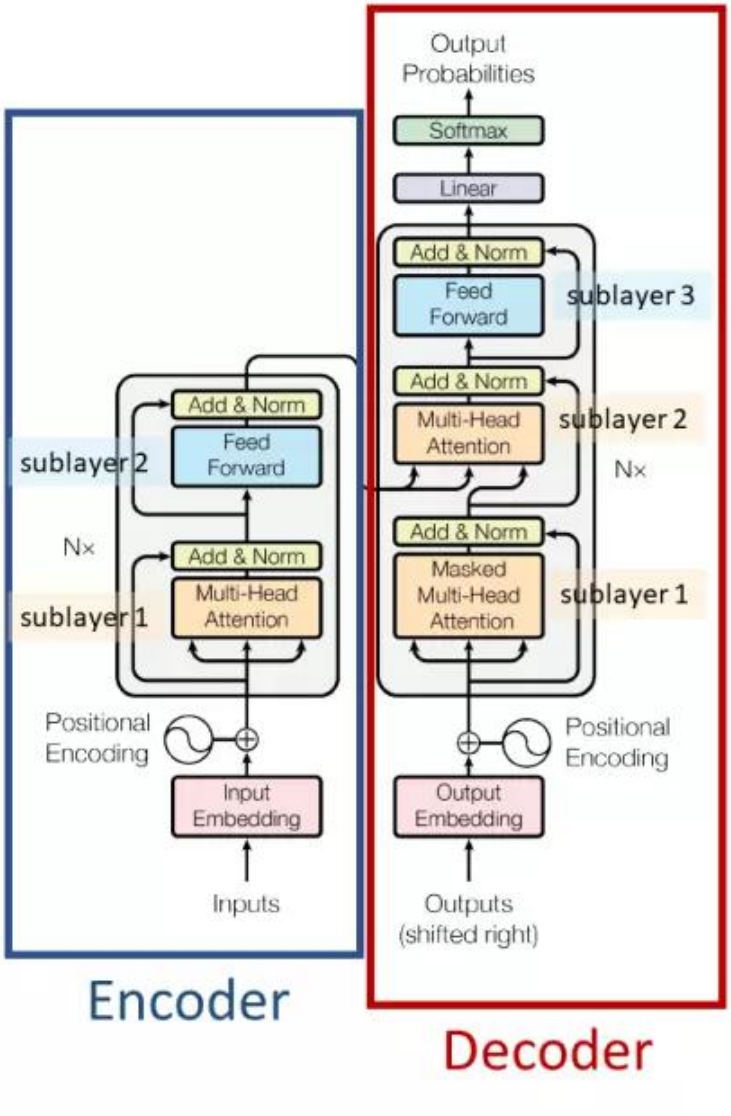
Điểm đột phá chính của Transformer là khả năng xử lý đồng thời tất cả các phần tử trong chuỗi đầu vào (thay vì tuần tự như RNNs). Điều này cho phép tận dụng tối đa khả năng tính toán song song của phần cứng hiện đại như GPU/TPU, giúp giảm đáng kể thời gian huấn luyện. Cơ chế Self-Attention chính là yếu tố thay thế cho sự "hồi quy" (recurrent) của RNNs trong việc nắm bắt ngữ cảnh và mối quan hệ giữa các từ, đúng như tinh thần của tên bài báo gốc.

1.3.2 Kiến trúc tổng quan

Transformer bao gồm hai thành phần chính: một chồng các Encoder giống hệt nhau và một chồng các Decoder giống hệt nhau.

- Encoder: Mỗi Encoder bao gồm hai lớp con chính: một lớp Multi-Head Self-Attention và một lớp mạng Feed-Forward kết nối đầy đủ (Fully Connected Feed-Forward Network).
- Decoder: Mỗi Decoder cũng bao gồm hai lớp con đó, nhưng có thêm một lớp Multi-Head Attention thứ ba thực hiện attention trên đầu ra của chồng Encoder.

Cả trong Encoder và Decoder, xung quanh mỗi lớp con đều có một kết nối tắt (residual connection) theo sau bởi một lớp chuẩn hóa lớp (Layer Normalization).



Hình 1.4: Kiến trúc của Transformer

1.3.3 Thành phần Encoder

Encoder có nhiệm vụ xử lý chuỗi đầu vào và tạo ra một chuỗi các biểu diễn ngữ cảnh (contextual representations) $z=(z_1,...,z_{Tx})$. Mỗi Encoder trong chồng bao gồm các lớp con sau:

- Input Embedding: Giống như các mô hình NLP khác, bước đầu tiên là chuyển đổi các từ (tokens) trong chuỗi đầu vào thành các vector có số chiều cố định d_{model}. Quá trình này gọi là embedding. Máy tính không hiểu ngôn ngữ tự nhiên, do đó ta cần biểu diễn mỗi từ dưới dạng một vector số thực sao cho các từ có ngữ nghĩa tương đồng sẽ có vector biểu diễn gần nhau trong không gian vector. Có thể sử dụng các kỹ thuật embedding được huấn luyện trước (pretrained word embeddings) như GloVe, FastText, hoặc Word2Vec, hoặc huấn luyện embedding từ đầu cùng với mô hình.
- Positional Encoding: Do Transformer xử lý các từ đồng thời và không có kết nối hồi quy, nó không có thông tin vốn có về thứ tự hoặc vị trí của các từ trong chuỗi. Để khắc phục điều này, thông tin về vị trí tương đối hoặc tuyệt đối của các token trong chuỗi được "tiêm" vào input embedding. Nhóm tác giả gốc đề xuất sử dụng các hàm sin và cos với tần số khác nhau:

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

  Trong đó:

  - ✓ pos là vị trí của token trong chuỗi (bắt đầu từ 0).
  - ✓ i là chỉ số của chiều trong vector embedding (từ 0 đến $d_{model}/2-1$).
  - ✓ PE(pos,k) là giá trị của thành phần thứ k trong vector Positional Encoding cho vị trí pos.
  - ✓ d_{model} là số chiều của embedding.

  Vector Positional Encoding này có cùng số chiều d_{model} với vector embedding và được cộng trực tiếp vào vector embedding của từ tương ứng. Việc sử dụng hàm sin/cos cho phép mô hình dễ dàng học cách

tham chiếu đến các vị trí tương đối, vì PE_{pos+k} có thể được biểu diễn như một hàm tuyến tính của PE_{pos}.

- Lớp Self-Attention: Đây là thành phần cốt lõi giúp Transformer hiểu được mối quan hệ giữa các từ khác nhau trong *cùng một chuỗi* (ví dụ: trong câu đầu vào cho Encoder, hoặc trong câu đầu ra đã được sinh ra cho Decoder). Ý tưởng là cho mỗi từ, mô hình sẽ tính toán một "điểm số chú ý" (attention score) đến tất cả các từ khác (bao gồm cả chính nó) trong chuỗi, sau đó tạo ra một biểu diễn mới cho từ đó dưới dạng trung bình có trọng số của các biểu diễn của tất cả các từ, với trọng số chính là điểm số chú ý đã được chuẩn hóa.

  Để thực hiện điều này, từ mỗi vector input x (là tổng của embedding và positional encoding, hoặc output của lớp trước), lớp Self-Attention tạo ra ba vector: Query (Q), Key (K), và Value (V) bằng cách nhân x với ba ma trận trọng số khác nhau WQ, WK, WV (các ma trận này được học trong quá trình huấn luyện)

  Cách tính Attention được sử dụng trong Transformer là Scaled Dot-Product Attention:

$$\boldsymbol{Z = softmax\left(\frac{Q \cdot K^T}{\sqrt{Dimension\ of\ vector Q, K\ or\ V}}\right) \cdot V}$$

  Quá trình tính toán diễn ra như sau:

  - ✓ Tính điểm tương đồng (Score): Tính tích vô hướng (dot product) giữa vector Query của một từ với tất cả các vector Key của các từ trong chuỗi (QK^T). Phép nhân này đo lường mức độ "liên quan" hoặc "tương hợp" giữa Query và từng Key. Một cách trực quan, Key giống như một "nhãn" đại diện cho thông tin của từ đó, còn Query là một "yêu cầu" tìm kiếm các thông tin liên quan.
  - ✓ Scale: Chia các điểm số vừa tính được cho $\sqrt{d_k}$, với d_k là số chiều của vector Key (và Query). Việc chia này giúp ổn định

gradient trong quá trình huấn luyện, tránh trường hợp tích vô hướng quá lớn khiến hàm softmax rơi vào vùng bão hòa (gradient gần bằng 0).

- ✓ Chuẩn hóa (Softmax): Áp dụng hàm softmax lên các điểm số đã scale để thu được các trọng số attention. Các trọng số này tạo thành một phân phối xác suất, tổng bằng 1, cho biết mức độ "chú ý" mà một từ (đại diện bởi Query) nên dành cho mỗi từ khác (đại diện bởi Key).
- ✓ Tính đầu ra (Weighted Sum): Nhân các trọng số attention vừa tính được với các vector Value tương ứng và cộng chúng lại. Kết quả là một vector biểu diễn mới (vector attention Z) cho từ đó, chứa thông tin tổng hợp từ các từ khác trong chuỗi dựa trên mức độ liên quan đã được tính toán. Những từ có trọng số attention cao (liên quan nhiều) sẽ đóng góp nhiều hơn vào vector Z, trong khi những từ có trọng số thấp (ít liên quan) sẽ bị "loại bỏ" hoặc giảm ảnh hưởng.

- Lớp Multi-Head Attention: Thay vì chỉ thực hiện một phép tính attention duy nhất với các vector Q, K, V có chiều dmodel, tác giả nhận thấy rằng việc chiếu (project) Q, K, V vào các không gian con khác nhau với số chiều nhỏ hơn và thực hiện attention song song trên các không gian con này ("multi-head") sẽ hiệu quả hơn.

  Ý tưởng là mỗi "head" attention có thể học cách tập trung vào các khía cạnh hoặc kiểu quan hệ khác nhau trong dữ liệu. Ví dụ, một head có thể tập trung vào quan hệ cú pháp, head khác tập trung vào quan hệ ngữ nghĩa gần,... Điều này cũng giúp giải quyết phần nào vấn đề self-attention có xu hướng chú ý quá mạnh vào chính từ đang xét.

  Cụ thể, với h head, Q, K, V đầu tiên được chiếu tuyến tính h lần bằng các ma trận trọng số khác nhau W_i^Q, W_i^K, W_i^V (cho head thứ i, i=1 ... h) vào các không gian có chiều $d_k=d_v=d_{model}/h$. Sau đó, phép tính Scaled

Dot-Product Attention được thực hiện song song cho từng head, tạo ra h vector output Z_i có chiều d_v.

$$\boldsymbol{head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)}$$

Các vector output $Z_1,...,Z_h$ này sau đó được ghép lại (concatenate) và chiếu một lần nữa qua một ma trận trọng số đầu ra W^O để thu được vector output cuối cùng của lớp Multi-Head Attention, có chiều d_{model}.

$$\boldsymbol{Multihead(Q,K,V) = Concat(head_1, \dots, head_h)W^O}$$

Tất cả các ma trận W_i^Q,W_i^K,W_i^V và W^O đều được học trong quá trình huấn luyện.

- Add & Norm (Residual Connections & Layer Normalization): Để giúp huấn luyện các mạng sâu hiệu quả, Transformer sử dụng kết nối tắt (residual/skip connection) xung quanh mỗi lớp con (Multi-Head Attention và Feed-Forward Network). Output của mỗi lớp con được cộng với input ban đầu của lớp con đó. Điều này cho phép gradient chảy trực tiếp qua mạng, giảm thiểu vấn đề vanishing gradient và cho phép xây dựng các mô hình sâu hơn.

  Sau kết nối tắt, một lớp Chuẩn hóa Lớp (Layer Normalization) được áp dụng. Khác với Batch Normalization, Layer Normalization chuẩn hóa các giá trị đầu vào trên các chiều của *từng mẫu* dữ liệu một cách độc lập, giúp ổn định quá trình huấn luyện.

  Công thức cho khối "Add & Norm":

  $$\boldsymbol{LayerNorm(x + Sublayer(x))}$$

  Trong đó: x là input của lớp con và Sublayer(x) là output của lớp con đó (ví dụ: Multi-Head Attention).

- Feed-Forward Network (FFN): Sau lớp Multi-Head Attention và Add & Norm, mỗi vị trí trong chuỗi được đưa qua một mạng nơ-ron truyền thẳng (Feed-Forward Network - FFN) kết nối đầy đủ một cách độc lập và giống hệt nhau. Mạng FFN này bao gồm hai lớp tuyến tính (linear transformations) với một hàm kích hoạt ReLU ở giữa**:**

$$\boldsymbol{FFN(Z) = Max(0, zW_1 + b_1)W_2 + b_2}$$

Trong đó z là output từ lớp Add & Norm trước đó, W_1,b_1,W_2,b_2 là các tham số được học. Số chiều của lớp ẩn bên trong thường lớn hơn d_{model} (ví dụ: $d_{ff}=4\times d_{model}$). Mạng FFN này cung cấp thêm khả năng biến đổi phi tuyến tính cho mô hình. Do FFN được áp dụng độc lập cho từng vị trí, nó có thể được tính toán song song trên toàn bộ chuỗi.

1.3.4 Thành phần Decoder

Decoder có nhiệm vụ sinh ra chuỗi đầu ra (y_1,...,y_{Ty}) dựa trên biểu diễn ngữ cảnh z từ Encoder. Kiến trúc của mỗi Decoder trong chồng khá giống với Encoder, nhưng có một số khác biệt quan trọng:

- Masked Multi-Head Attention (Lớp con đầu tiên): Decoder hoạt động theo cơ chế tự hồi quy (autoregressive), nghĩa là việc dự đoán token yt tại bước t chỉ được phép dựa vào các token đã được sinh ra trước đó (y_1,...,y_{t-1}) và output z của Encoder. Để đảm bảo điều này trong kiến trúc song song của Transformer, lớp Self-Attention đầu tiên trong Decoder được sửa đổi thành "Masked" Self-Attention. Trong quá trình tính điểm attention, một ma trận mặt nạ (mask) được áp dụng để gán giá trị âm vô cùng (-inf) cho tất cả các điểm số tương ứng với các vị trí sau vị trí hiện tại trong chuỗi đầu ra. Sau khi áp dụng softmax, các vị trí bị che này sẽ có trọng số attention bằng 0, đảm bảo rằng dự đoán tại vị trí i chỉ phụ thuộc vào các kết quả đã biết ở các vị trí $j \leq i$.
- Multi-Head Attention (Encoder-Decoder Attention - Lớp con thứ hai): Lớp này cho phép Decoder "nhìn" vào chuỗi đầu vào đã được mã hóa bởi Encoder. Nó hoạt động tương tự như Multi-Head Attention thông thường, nhưng Keys (K) và Values (V) được lấy từ output của lớp Encoder cuối cùng, trong khi Queries (Q) đến từ output của lớp Masked Multi-Head Attention (lớp con đầu tiên của Decoder). Điều này cho phép mỗi vị trí trong Decoder có thể tập trung chú ý vào các

vị trí phù hợp trong chuỗi đầu vào để đưa ra dự đoán chính xác cho token đầu ra tiếp theo.

- Feed-Forward Network (Lớp con thứ ba): Giống hệt như mạng FFN trong Encoder.

Các lớp Add & Norm cũng được sử dụng sau mỗi lớp con trong Decoder.

1.3.5 Lớp đầu ra (Linear & Softmax)

Sau khi chuỗi đầu ra đi qua chồng Decoder, vector output dec_out_t tại mỗi vị trí t được đưa qua một lớp tuyến tính (Linear layer) cuối cùng, theo sau bởi một hàm Softmax. Lớp tuyến tính này chiếu vector dec_out_t lên không gian có số chiều bằng kích thước của bộ từ vựng đích. Hàm Softmax sau đó chuyển đổi các điểm số này thành một phân phối xác suất trên toàn bộ từ vựng, cho biết xác suất của mỗi từ có thể là token tiếp theo y_{t+1}. Từ có xác suất cao nhất thường được chọn làm token dự đoán tiếp theo (trong quá trình greedy decoding).

1.4 Mô hình GPT (Generative Pretrained Transformer)

1.4.1 Giới thiệu về mô hình ngôn ngữ lớn dựa trên Transformer

Trong những năm gần đây, lĩnh vực Xử lý Ngôn ngữ Tự nhiên (NLP) đã chứng kiến sự trỗi dậy mạnh mẽ của các Mô hình Ngôn ngữ Lớn (Large Language Models - LLMs). Một trong những dòng mô hình tiên phong và có ảnh hưởng nhất chính là Generative Pre-trained Transformer, thường được biết đến với tên viết tắt GPT, do OpenAI phát triển.

Đúng như tên gọi, GPT là một họ các mô hình mạng nơ-ron tận dụng sức mạnh của kiến trúc Transformer (đã trình bày ở mục 1.3), nhưng được điều chỉnh và huấn luyện theo một phương pháp đặc biệt, bao gồm hai giai đoạn chính: Pre-training (Tiền huấn luyện) và Fine-tuning (Tinh chỉnh) (hoặc ứng dụng trực tiếp khả năng học trong ngữ cảnh - in-context learning). Khía cạnh "Generative" (Sinh) nhấn mạnh vào khả năng tạo ra dữ liệu mới (chủ yếu là văn bản) một cách mạch lạc và tự nhiên, trong khi "Pre-trained" (Tiền huấn luyện) chỉ ra rằng mô hình đã được huấn

luyện trên một lượng lớn dữ liệu văn bản không gán nhãn trước khi được áp dụng cho các tác vụ cụ thể.

GPT được coi là một tiến bộ quan trọng trong lĩnh vực trí tuệ nhân tạo (AI) nói chung và NLP nói riêng. Nó là nền tảng công nghệ đằng sau nhiều ứng dụng AI tạo sinh (Generative AI) đột phá, nổi bật nhất là ChatGPT. Các mô hình GPT cung cấp cho ứng dụng khả năng tạo ra văn bản giống như con người viết, soạn thảo nội dung sáng tạo (thơ, mã nguồn, kịch bản, email,...), tóm tắt thông tin, dịch thuật, và đặc biệt là khả năng tương tác, trả lời các câu hỏi của người dùng dưới dạng một cuộc trò chuyện tự nhiên. Ngoài văn bản, các nguyên lý của GPT cũng truyền cảm hứng cho việc tạo ra các nội dung khác như hình ảnh, âm nhạc.

1.4.2 Kiến trúc GPT - Decoder-only

Khác với kiến trúc Transformer gốc bao gồm cả Encoder và Decoder, các mô hình GPT chủ yếu dựa trên kiến trúc Decoder-only của Transformer. Lựa chọn này rất phù hợp với mục tiêu chính của GPT là một mô hình ngôn ngữ sinh: dự đoán từ tiếp theo trong một chuỗi dựa trên các từ đã có trước đó.

Kiến trúc GPT về cơ bản là một chồng (stack) gồm nhiều khối Decoder của Transformer giống hệt nhau. Mỗi khối bao gồm các thành phần chính sau:

- Masked Multi-Head Self-Attention: Đây là cơ chế attention cốt lõi, tương tự như lớp self-attention đầu tiên trong Decoder của Transformer gốc. Việc sử dụng "masking" đảm bảo rằng khi dự đoán tại một vị trí t, mô hình chỉ có thể "nhìn"" (attend to) các token ở vị trí $j \leq t$ trong chuỗi đầu vào. Điều này duy trì tính chất tự hồi quy (autoregressive) cần thiết cho việc sinh văn bản tuần tự.
- Add & Norm: Kết nối tắt (residual connection) và chuẩn hóa lớp (Layer Normalization) được áp dụng sau lớp Masked Multi-Head Self-Attention.
- Feed-Forward Network (FFN): Một mạng nơ-ron truyền thẳng giống hệt như trong kiến trúc Transformer gốc, được áp dụng độc lập cho từng vị trí.

- Add & Norm: Kết nối tắt và chuẩn hóa lớp một lần nữa được áp dụng sau lớp FFN.

Đầu vào của mô hình là chuỗi token (đã được embedding và cộng positional encoding). Đầu ra sau khi đi qua chồng các khối Decoder sẽ được đưa vào một lớp tuyến tính cuối cùng và hàm softmax để dự đoán phân phối xác suất trên từ vựng cho token tiếp theo.

Việc loại bỏ phần Encoder giúp đơn giản hóa kiến trúc và tập trung hoàn toàn vào khả năng mô hình hóa ngôn ngữ và sinh văn bản dựa trên ngữ cảnh trước đó.

1.4.3 Giai đoạn Pre-training học từ dữ liệu khổng lồ

Đây là giai đoạn quan trọng tạo nên sức mạnh nền tảng của GPT. Mô hình được huấn luyện trên một kho dữ liệu văn bản cực kỳ lớn và đa dạng, thu thập từ internet (ví dụ: Common Crawl), sách (ví dụ: BookCorpus), và các nguồn khác. Quá trình này là học không giám sát (unsupervised learning) hoặc tự giám sát (self-supervised learning), nghĩa là không yêu cầu dữ liệu phải được gán nhãn thủ công.

Nhiệm vụ huấn luyện cốt lõi trong giai đoạn pre-training của GPT là Mô hình hóa Ngôn ngữ Nhân quả (Causal Language Modeling - CLM), hay đơn giản là Dự đoán từ tiếp theo (Next Token Prediction). Cụ thể, với một chuỗi văn bản đầu vào (w_1,w_2,...,w_k), mô hình học cách dự đoán xác suất của từ tiếp theo w_{k+1} dựa trên các từ trước đó:

$$\boldsymbol{Tối\ đa\ hóa} \sum_{k} \mathbf{log}(\boldsymbol{w_k}|\boldsymbol{w_{k-n}}, \dots, \boldsymbol{w_{k-1}}; \boldsymbol{\Theta})$$

Trong đó Θ là tham số của mô hình và n là kích thước cửa sổ ngữ cảnh (context window size). Nhờ việc thực hiện nhiệm vụ này trên hàng tỷ, thậm chí hàng nghìn tỷ từ, mô hình GPT học được các mẫu phức tạp về ngữ pháp, ngữ nghĩa, các mối quan hệ logic, và thậm chí cả một lượng lớn kiến thức phổ quát về thế giới tiềm ẩn trong dữ liệu văn bản đó. Kết quả của giai đoạn này là một mô hình ngôn ngữ nền tảng (foundation model) có khả năng hiểu và tạo ra văn bản một cách đáng kinh ngạc.

1.4.4 Giai đoạn Fine-tuning và khả năng học ít mẫu (Few-shot Learning)

Sau khi có được mô hình nền tảng từ giai đoạn pre-training, có hai cách chính để khai thác sức mạnh của nó cho các tác vụ cụ thể:

- Fine-tuning (Tinh chỉnh): Đây là phương pháp truyền thống. Mô hình pre-trained được huấn luyện tiếp (thường là với tốc độ học nhỏ hơn) trên một tập dữ liệu nhỏ hơn và có gán nhãn, đặc thù cho tác vụ mong muốn (ví dụ: tập dữ liệu câu hỏi-trả lời cho tác vụ hỏi đáp, tập dữ liệu văn bản-nhãn tình cảm cho tác vụ phân tích tình cảm). Quá trình này giúp "chuyên môn hóa" mô hình cho tác vụ đó bằng cách điều chỉnh nhẹ các trọng số đã học mà không làm mất đi kiến thức nền tảng. GPT-1 và GPT-2 chủ yếu dựa vào phương pháp này.
- In-Context Learning (Zero-shot, One-shot, Few-shot Learning): Một khả năng đột phá được thể hiện rõ rệt từ GPT-3 trở đi là khả năng học trong ngữ cảnh. Với các mô hình đủ lớn, chúng có thể thực hiện nhiều tác vụ mới mà không cần cập nhật trọng số (tức là không cần fine-tuning). Thay vào đó, người dùng chỉ cần cung cấp mô tả về tác vụ và/hoặc một vài ví dụ minh họa (prompts) ngay trong đầu vào của mô hình.
    - ✓ Zero-shot: Chỉ cung cấp mô tả tác vụ.
    - ✓ One-shot: Cung cấp mô tả và 1 ví dụ.
    - ✓ Few-shot: Cung cấp mô tả và một vài ví dụ. Mô hình sẽ tự “hiểu” yêu cầu và thực hiện tác vụ dựa trên kiến thức đã học trong giai đoạn pre-training và thông tin từ prompt. Khả năng này giúp việc áp dụng GPT cho các tác vụ mới trở nên linh hoạt và nhanh chóng hơn rất nhiều.

1.4.5 Tokenizer và Thuật toán BPE

Trước khi đưa dữ liệu vào mô hình, văn bản cần được mã hóa thành chuỗi các token. Quá trình này được xử lý bởi tokenizer.

Trong dự án này, sử dụng thuật toán BPE (Byte Pair Encoding) – một kỹ thuật phân tách từ thành các đơn vị nhỏ (subword) giúp giảm kích thước từ vựng và xử lý tốt các từ chưa thấy (out-of-vocabulary).

Nguyên tắc chính của BPE là:

- Bắt đầu từ ký tự đơn lẻ, lặp lại việc ghép các cặp ký tự có tần suất cao nhất.
- Sau nhiều bước, thu được bảng mã hóa ổn định, vừa giữ được nghĩa ngữ pháp, vừa tránh phân mảnh từ không cần thiết.

Nhờ BPE, mô hình có thể học các mẫu ngôn ngữ linh hoạt hơn và dịch chính xác ngay cả với từ mới hoặc hiếm gặp.

1.4.6 Đánh giá BLEU và ROUGE

Để đánh giá chất lượng bản dịch, hai chỉ số phổ biến được sử dụng là BLEU và ROUGE:

- BLEU (Bilingual Evaluation Understudy): So sánh bản dịch máy với bản dịch tham chiếu bằng cách tính số lượng n-gram (chuỗi n từ liên tiếp) trùng khớp. Điểm BLEU càng cao thể hiện bản dịch càng giống bản gốc.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): Thường dùng trong đánh giá tóm tắt, nhưng cũng hữu ích trong dịch máy. ROUGE nhấn mạnh đến mức độ “gợi nhớ” (recall), tức là bản dịch máy đã bao phủ được bao nhiêu nội dung chính của bản dịch chuẩn.

Kết hợp BLEU và ROUGE giúp đánh giá cân bằng giữa độ chính xác và độ bao phủ trong bản dịch, từ đó phản ánh chân thực hơn chất lượng mô hình.

CHƯƠNG 2. BÀI TOÁN DỊCH MÁY (TRANSLATION MACHINE)

2.1 Giới thiệu Bài toán Dịch máy (Machine Translation)

2.1.1 Định nghĩa và mục tiêu

Dịch máy (Machine Translation) là một lĩnh vực con của Ngôn ngữ học Tính toán (Computational Linguistics) và Trí tuệ Nhân tạo (Artificial Intelligence), tập trung vào việc sử dụng các hệ thống máy tính để tự động chuyển đổi văn bản hoặc giọng nói từ một ngôn ngữ tự nhiên (được gọi là ngôn ngữ nguồn - source language) sang một ngôn ngữ tự nhiên khác (gọi là ngôn ngữ đích - target language) mà không cần sự can thiệp của con người trong quá trình dịch.

Mục tiêu cuối cùng của dịch máy không chỉ đơn thuần là chuyển đổi từng từ một cách riêng lẻ, mà là tạo ra một bản dịch chất lượng cao. Một bản dịch được coi là chất lượng khi nó đáp ứng được hai tiêu chí chính:

- Độ chính xác (Adequacy/Fidelity): Bản dịch phải truyền tải được đầy đủ và chính xác ý nghĩa, nội dung thông tin của văn bản gốc trong ngôn ngữ nguồn.
- Độ trôi chảy (Fluency): Bản dịch phải tự nhiên, mạch lạc, đúng ngữ pháp và dễ hiểu đối với người bản xứ của ngôn ngữ đích.

Việc cân bằng và đạt được cả hai tiêu chí này là một thách thức lớn, thúc đẩy sự phát triển không ngừng của các kỹ thuật và mô hình dịch máy.

2.1.2 Tầm quan trọng và sơ lược lịch sử

Trong một thế giới ngày càng kết nối, nhu cầu giao tiếp và trao đổi thông tin xuyên biên giới ngôn ngữ trở nên cấp thiết hơn bao giờ hết. Dịch máy đóng vai trò then chốt trong việc phá bỏ rào cản ngôn ngữ, thúc đẩy sự hiểu biết lẫn nhau giữa các nền văn hóa, hỗ trợ thương mại điện tử toàn cầu, tạo điều kiện tiếp cận thông tin khoa học kỹ thuật, và phục vụ nhiều mục đích quan trọng khác trong chính trị, giáo dục, du lịch, giải trí,...

Lịch sử của dịch máy là một hành trình dài với những bước tiến đáng kể, có thể tóm lược qua ba giai đoạn chính:

- Dịch máy dựa trên Luật (Rule-Based Machine Translation - RBMT): Xuất hiện từ những ngày đầu của kỷ nguyên máy tính (những năm 1950-1980s), RBMT dựa vào các bộ quy tắc ngôn ngữ học phức tạp (bao gồm từ điển song ngữ, quy tắc phân tích cú pháp, quy tắc chuyển đổi cấu trúc câu) do các chuyên gia ngôn ngữ xây dựng thủ công. Mặc dù có những thành công ban đầu, cách tiếp cận này tỏ ra cứng nhắc và khó mở rộng để bao phủ sự đa dạng và phức tạp của ngôn ngữ tự nhiên.
- Dịch máy Thống kê (Statistical Machine Translation - SMT): Bắt đầu phát triển mạnh mẽ từ cuối những năm 1980s và thống trị lĩnh vực trong khoảng hai thập kỷ sau đó (1990s-2010s), SMT thay đổi hoàn toàn cuộc chơi bằng cách học các mô hình xác suất trực tiếp từ dữ liệu song ngữ song song (parallel corpora) khổng lồ. Các hệ thống SMT, đặc biệt là Phrase-based SMT, đã mang lại sự cải thiện đáng kể về chất lượng dịch so với RBMT cho nhiều cặp ngôn ngữ phổ biến.
- Dịch máy Nơ-ron (Neural Machine Translation - NMT): Xuất hiện vào khoảng năm 2014 và nhanh chóng trở thành phương pháp chủ đạo từ khoảng năm 2016 trở đi, NMT sử dụng các kiến trúc mạng nơ-ron sâu (Deep Neural Networks), ban đầu là mạng nơ-ron hồi quy (RNN) với kiến trúc Sequence-to-Sequence, và sau đó là sự bùng nổ của kiến trúc Transformer. NMT có khả năng mô hình hóa ngôn ngữ một cách hiệu quả hơn, xử lý ngữ cảnh tốt hơn và tạo ra các bản dịch trôi chảy, tự nhiên hơn hẳn so với các phương pháp trước đó, đánh dấu một cuộc cách mạng thực sự trong lĩnh vực dịch máy.

2.1.3 Những thách thức chính trong Dịch máy

Mặc dù đã đạt được những tiến bộ vượt bậc, dịch máy vẫn là một bài toán vô cùng phức tạp và đối mặt với nhiều thách thức cố hữu của ngôn ngữ tự nhiên:

- Tính đa nghĩa của từ vựng (Lexical Ambiguity): Một từ trong ngôn ngữ nguồn có thể có nhiều nghĩa khác nhau, và việc chọn đúng nghĩa phù hợp trong ngữ cảnh để dịch sang ngôn ngữ đích là rất khó khăn (ví dụ: từ “bank” trong tiếng Anh có thể là “ngân hàng” hoặc “bờ sông”).
- Khác biệt về cấu trúc và ngữ pháp: Các ngôn ngữ khác nhau có thể có trật tự từ hoàn toàn khác nhau (ví dụ: SVO như tiếng Anh, SOV như tiếng Nhật, VSO như tiếng Ireland), các hệ thống thì, giống, số, cách,... phức tạp và không tương đồng. Việc tái cấu trúc câu sao cho đúng ngữ pháp và tự nhiên trong ngôn ngữ đích là một thách thức lớn.
- Thành ngữ, tiếng lóng và yếu tố văn hóa: Các cụm từ cố định, thành ngữ, tiếng lóng thường mang ý nghĩa khác xa nghĩa đen của từng từ cấu thành. Việc dịch đúng các yếu tố này đòi hỏi mô hình phải có khả năng "hiểu" được ngữ cảnh văn hóa sâu sắc.
- Duy trì ngữ cảnh và sự mạch lạc: Khi dịch các đoạn văn bản dài, việc duy trì sự nhất quán về thuật ngữ, đại từ tham chiếu (pronoun resolution), và luồng ý tưởng mạch lạc qua nhiều câu là rất khó khăn. Các mô hình dịch từng câu riêng lẻ thường thất bại trong việc này.
- Khan hiếm dữ liệu: Chất lượng của các mô hình SMT và đặc biệt là NMT phụ thuộc rất lớn vào số lượng và chất lượng của dữ liệu song ngữ song song dùng để huấn luyện. Tuy nhiên, việc xây dựng các bộ dữ liệu lớn, chất lượng cao là rất tốn kém và khó khăn, đặc biệt đối với các cặp ngôn ngữ ít phổ biến hoặc các ngôn ngữ ít tài nguyên (low-resource languages)

CHƯƠNG 3. TỔNG QUAN VỀ TẬP DỮ LIỆU

3.1 Giới thiệu về tập dữ liệu

Dataset “English-Vietnamese Translation”là một bộ dữ liệu song ngữ chất lượng cao, được thiết kế đặc biệt để hỗ trợ các nhiệm vụ liên quan đến dịch máy, xử lý ngôn ngữ tự nhiên (NLP), học sâu (Deep Learning) và trí tuệ nhân tạo (AI). Bộ dữ liệu này bao gồm hàng nghìn cặp câu song song giữa tiếng Anh và tiếng Việt, trong đó mỗi cặp bao gồm một câu bằng tiếng Anh và bản dịch tương ứng của nó sang tiếng Việt.

3.2 Nguồn gốc và mục đích

Dataset được thu thập và xây dựng nhằm đáp ứng nhu cầu ngày càng tăng về các công cụ dịch thuật tự động giữa tiếng Anh và tiếng Việt - hai ngôn ngữ có sự khác biệt lớn về cấu trúc ngữ pháp, từ vựng và văn hóa. Tiếng Việt, với hệ thống thanh điệu và cấu trúc ngữ pháp đặc thù, thường là một thách thức đối với các mô hình dịch máy. Do đó, bộ dữ liệu này được tạo ra để cung cấp một nguồn tài liệu phong phú, giúp các mô hình học máy có thể học hỏi và cải thiện khả năng dịch thuật một cách chính xác và tự nhiên hơn.

3.3 Bộ dữ liệu có sẵn trên Kaggle

Nguồn: Một bộ dữ liệu Anh-Việt được cung cấp sẵn trong môi trường Kaggle, bao gồm hai tệp văn bản riêng biệt: en_sents.txt (chứa các câu tiếng Anh) và vi_sents.txt (chứa các câu tiếng Việt tương ứng).

Phương pháp: Đọc trực tiếp nội dung từ hai tệp này bằng thư viện pandas trong Python, đảm bảo số lượng câu trong hai tệp khớp nhau và lưu vào một DataFrame ban đầu.

Kết quả: Thu được khoảng 254,090 cặp câu song ngữ từ nguồn này.

3.4 Tải và tích hợp dữ liệu từ kho OPUS

Nguồn: Kho dữ liệu mở OPUS (Open Parallel Corpus), một tài nguyên quý giá chứa rất nhiều bộ dữ liệu song ngữ cho các cặp ngôn ngữ khác nhau. Cụ thể, hai bộ dữ liệu con đã được chọn cho cặp Anh-Việt (en-vi):

- TED2020: Dữ liệu từ các bài nói chuyện TED Talks được dịch song ngữ.
- WikiMatrix: Dữ liệu được khai thác và căn chỉnh tự động từ Wikipedia đa ngôn ngữ.

Phương pháp:

- Sử dụng thư viện requests của Python để tự động tải về các tệp nén (.zip) chứa dữ liệu song ngữ từ các URL tương ứng trên máy chủ của OPUS (object.pouta.csc.fi).
- Sử dụng thư viện zipfile để giải nén các tệp đã tải về vào thư mục cục bộ.
- Đọc nội dung từ các tệp .en và .vi tương ứng trong mỗi bộ dữ liệu con đã giải nén.
- Kiểm tra số lượng dòng (câu) trong mỗi cặp tệp phải khớp nhau trước khi thêm vào danh sách dữ liệu tổng hợp.

Kết quả: Thu thập thêm khoảng 1,400,169 cặp câu song ngữ từ hai nguồn TED2020 và WikiMatrix trong OPUS.

3.5 Sinh Dữ liệu bổ sung bằng Mô hình Gemini

Nguồn: Sử dụng mô hình ngôn ngữ lớn gemini-2.0-flash của Google thông qua API google.generativeai.

Phương pháp:

- Thiết kế một "prompt" (câu lệnh yêu cầu) yêu cầu mô hình Gemini tạo ra các cặp câu song ngữ Anh-Việt ngắn (10-20 từ) về các chủ đề giao tiếp hàng ngày, theo một định dạng cụ thể (Anh: "..." - Việt: "...").
- Thực hiện gọi API lặp lại 400 lần, mỗi lần yêu cầu sinh 50 cặp câu.

- Xử lý (parse) văn bản trả về từ API để trích xuất chính xác các cặp câu tiếng Anh và tiếng Việt.
- Sử dụng time.sleep(1) giữa các lần gọi API để tránh gửi yêu cầu quá dồn dập.

Kết quả: Tạo thêm khoảng 20,192 cặp câu song ngữ mới bằng phương pháp này, giúp bổ sung dữ liệu về các chủ đề giao tiếp thông thường.

3.6 Tổng hợp Dữ liệu

Cuối cùng, tất cả dữ liệu từ ba nguồn trên (Kaggle, OPUS, Gemini) được tổng hợp lại bằng cách nối (concatenate) các DataFrame tương ứng.

Kết quả là một bộ dữ liệu song ngữ Anh-Việt lớn, bao gồm khoảng 1,674,451 cặp câu trước khi thực hiện các bước tiền xử lý tiếp theo.

Dữ liệu được lưu trữ dưới định dạng CSV và các file văn bản riêng biệt (en_sents_dataset.txt, vi_sents_dataset.txt), giúp dễ dàng tích hợp vào các dự án dịch máy. Để đảm bảo chất lượng, dataset đã trải qua các bước tiền xử lý như loại bỏ câu trống, trùng lặp, câu quá ngắn/dài (dưới 5 hoặc trên 100 từ), và các câu chứa ký tự nhiễu (ví dụ: URL, ký tự đặc biệt). Đường dẫn của datasets như sau: https://drive.google.com/drive/folders/1GgkYtHfUBx6Yy8e_8juCgCcuBA8mXc2N?usp=sharing.

1	Hello, how are you today?
2	I speak a little English.
3	How much does this cost?
4	I want to buy this.
5	I need to go now.
6	Goodbye, have a good day.
7	Of course, I can help.
8	How much does it cost?
9	The bathroom is over there.
10	I need to go home.
11	Can I help you find something?
12	I am looking for a shirt.
13	I only speak a little English.
14	How much does it cost?
15	Can I have the menu, please?
16	I would like some water.
17	I speak a little Vietnamese.
18	I need help with this.
19	Can you help me practice?
20	Goodbye, have a nice day.
21	I need to go shopping.
22	What is your phone number?
23	What is your favorite food?
24	Goodbye, have a nice day!
25	Excuse me, where is the restroom?
26	Can you speak slower, please?
27	Do you have any pets?
28	Thank you for your help.
29	I speak a little Vietnamese.
30	Can you help me with this?

1	Chào bạn, hôm nay bạn khỏe không?
2	Tôi nói được một chút tiếng Anh.
3	Cái này giá bao nhiêu?
4	Tôi muốn mua cái này.
5	Tôi cần phải đi bây giờ.
6	Tạm biệt, chúc một ngày tốt lành.
7	Tất nhiên, tôi có thể giúp.
8	Cái này bao nhiêu tiền?
9	Nhà vệ sinh ở đằng kia.
10	Tôi cần phải về nhà.
11	Tôi có thể giúp bạn tìm gì không?
12	Tôi đang tìm một cái áo sơ mi.
13	Tôi chỉ nói được một chút tiếng Anh.
14	Cái này giá bao nhiêu?
15	Cho tôi xin thực đơn được không?
16	Tôi muốn một chút nước.
17	Tôi nói một chút tiếng Việt.
18	Tôi cần giúp đỡ với việc này.
19	Bạn có thể giúp tôi luyện tập không?
20	Tạm biệt, chúc bạn một ngày tốt lành.
21	Tôi cần đi mua sắm.
22	Số điện thoại của bạn là gì?
23	Món ăn yêu thích của bạn là gì?
24	Tạm biệt, chúc một ngày tốt lành!
25	Xin lỗi, nhà vệ sinh ở đâu?
26	Bạn có thể nói chậm hơn không?
27	Bạn có nuôi thú cưng không?
28	Cảm ơn sự giúp đỡ của bạn.
29	Tôi nói được một chút tiếng Việt.
30	Bạn có thể giúp tôi việc này được không?

Hình 3.1: Hình ảnh về datasets “en_sents_dataset.txt” và “vi_sents_dataset.txt”

CHƯƠNG 4. CÁC HƯỚNG TIẾP CẬN BÀI TOÁN

Như đã đề cập, các mô hình dựa trên kiến trúc Transformer hiện đang là phương pháp hiệu quả nhất cho bài toán dịch máy. Trong phần này, chúng tôi sẽ trình bày các hướng tiếp cận cụ thể được áp dụng trong báo cáo này.

4.1 Sử dụng mô hình Transformer

Kiến trúc Transformer, với cơ chế Self-Attention và khả năng xử lý song song, đã tạo ra một bước đột phá trong NMT. Thay vì huấn luyện một mô hình Transformer từ đầu (training from scratch), một cách tiếp cận phổ biến và hiệu quả hơn là tận dụng các mô hình đã được huấn luyện trước (pre-trained) trên một lượng lớn dữ liệu.

4.1.1 Sử dụng Pre-trained Model và Fine-tuning

Hướng tiếp cận này dựa trên nguyên lý học chuyển giao (transfer learning). Chúng ta bắt đầu với một mô hình Transformer đã được huấn luyện trước cho nhiệm vụ dịch máy (thường là trên dữ liệu tổng quát hoặc đa ngôn ngữ), sau đó tinh chỉnh (fine-tuning) mô hình này trên tập dữ liệu song ngữ Anh-Việt cụ thể mà chúng ta đã thu thập. Quá trình này giúp mô hình "thích ứng" với đặc điểm của dữ liệu mới, kế thừa các "kiến thức" đã học từ giai đoạn pre-training và cải thiện hiệu suất trên tác vụ dịch Anh-Việt cụ thể.

Quy trình thực hiện trong notebook được tóm tắt như sau:

1. Lựa chọn Mô hình Pre-trained

- Một mô hình Transformer chuyên biệt cho dịch máy Anh-Việt từ cộng đồng Hugging Face đã được chọn làm điểm khởi đầu. Cụ thể là mô hình Helsinki-NLP/opus-mt-en-vi. Đây là một mô hình thuộc dòng MarianMT, được phát triển bởi nhóm Helsinki-NLP và đã được huấn luyện trên một phần lớn dữ liệu từ kho OPUS cho cặp ngôn ngữ Anh-Việt. Việc chọn mô hình pre-trained phù hợp giúp tiết kiệm đáng kể thời gian và tài nguyên huấn luyện so với việc đào tạo từ đầu.

2. Chuẩn bị và Tiền xử lý Dữ liệu

- Tập dữ liệu song ngữ Anh-Việt đã thu thập (từ Kaggle, OPUS, và Gemini) được tải và chuẩn bị.
- Một mẫu (sample) gồm 100,000 cặp câu được chọn ngẫu nhiên từ tập dữ liệu lớn để phục vụ cho quá trình fine-tuning và đánh giá trong giới hạn tài nguyên tính toán.
- Dữ liệu được chia thành ba tập: huấn luyện (train - 80%), kiểm định (validation - 10%), và kiểm tra (test - 10%) bằng cách sử dụng thư viện datasets. Việc chia tách này đảm bảo mô hình được huấn luyện trên một tập, tinh chỉnh siêu tham số trên tập thứ hai, và đánh giá hiệu năng cuối cùng trên một tập hoàn toàn độc lập.
- Tokenization: Đây là bước cực kỳ quan trọng. Cả câu nguồn (tiếng Anh) và câu đích (tiếng Việt) đều được mã hóa thành các chuỗi số (token IDs) bằng cách sử dụng Tokenizer tương ứng với mô hình pre-trained (AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-en-vi")).
    - ✓ Tokenizer này sử dụng kỹ thuật phân tách từ phụ (subword tokenization, cụ thể là SentencePiece) để xử lý từ vựng lớn và các từ hiếm gặp.
    - ✓ Một hàm tiền xử lý (preprocess_function) được định nghĩa để thực hiện việc tokenize cho cả câu nguồn (inputs) và câu đích (targets/labels), đồng thời cắt ngắn (truncation) các câu quá dài (độ dài tối đa được đặt là 128 tokens) và tạo ra các trường dữ liệu cần thiết cho mô hình (như input_ids, attention_mask, labels). Trường labels chính là input_ids của câu đích, được mô hình sử dụng để tính toán hàm mất mát trong quá trình huấn luyện.
- Toàn bộ quá trình tiền xử lý này được áp dụng cho cả ba tập train, validation, và test bằng phương thức .map() của thư viện datasets.

3. Tải Mô hình Pre-trained và Cấu hình Fine-tuning

- Mô hình Helsinki-NLP/opus-mt-en-vi được tải về bằng lớp AutoModelForSeq2SeqLM.from_pretrained(). Lớp này tự động nhận diện kiến trúc phù hợp (MarianMT) và tải các trọng số đã được huấn luyện trước.
- Các đối số huấn luyện (Training Arguments) được định nghĩa bằng Seq2SeqTrainingArguments. Các thiết lập quan trọng bao gồm:
  - ✓ Tên thư mục lưu trữ checkpoint.
  - ✓ Chiến lược đánh giá (eval_strategy="epoch"): Đánh giá mô hình trên tập validation sau mỗi epoch huấn luyện.
  - ✓ Tốc độ học (learning_rate=2e-5).
  - ✓ Kích thước lô (batch size) cho train và validation (per_device_train_batch_size=16).
  - ✓ Số epoch huấn luyện (num_train_epochs=5).
  - ✓ Sử dụng độ chính xác hỗn hợp (fp16=True) để tăng tốc độ huấn luyện trên GPU tương thích.
  - ✓ Bật chế độ predict_with_generate=True để cho phép tính toán các chỉ số như BLEU trong quá trình đánh giá.
- Một DataCollatorForSeq2Seq được sử dụng để tự động thực hiện việc đệm (padding) các câu trong mỗi lô về cùng một độ dài, giúp xử lý hiệu quả các câu có độ dài khác nhau.

4. Định nghĩa Hàm Tính toán Chỉ số Đánh giá

- Một hàm compute_metrics được viết để tính toán chỉ số BLEU trong quá trình huấn luyện và đánh giá. Hàm này nhận dự đoán và nhãn từ mô hình, giải mã (decode) chúng thành văn bản, thực hiện một số xử lý hậu kỳ đơn giản (như loại bỏ khoảng trắng thừa), và cuối cùng sử dụng thư viện evaluate (với metric sacrebleu) để tính điểm BLEU.

5. Huấn luyện (Fine-tuning)

- Một đối tượng Seq2SeqTrainer được khởi tạo, kết hợp mô hình, đối số huấn luyện, các tập dữ liệu đã token hóa, data collator, tokenizer và hàm compute_metrics.
- Lệnh trainer.train() (được comment lại trong notebook nhưng là bước logic tiếp theo) sẽ bắt đầu quá trình fine-tuning. Trong quá trình này, các trọng số của mô hình pre-trained được cập nhật dựa trên dữ liệu huấn luyện Anh-Việt, sử dụng hàm mất mát sequence-to-sequence và tối ưu hóa bằng các thuật toán như AdamW. Mô hình sẽ được lưu lại định kỳ và/hoặc khi đạt hiệu suất tốt nhất trên tập validation.

6. Đánh giá và Sử dụng Mô hình Fine-tuned

- Sau khi fine-tuning (hoặc tải một checkpoint đã được fine-tune từ trước, như trong phần cuối của notebook: /kaggle/input/weight-pretrained-transformer/opus-mt-en-vi-finetuned-en-to-vi/checkpoint-25000), mô hình có thể được sử dụng để dịch các câu mới.
- Notebook thực hiện việc tải mô hình và tokenizer từ checkpoint đã fine-tune.
- Thực hiện dịch thử một câu ví dụ (model.generate).
- Quan trọng hơn, notebook tiến hành đánh giá hiệu năng của mô hình đã fine-tune trên tập test (là tập dữ liệu chưa từng được mô hình nhìn thấy trong quá trình huấn luyện hay chọn checkpoint). Quá trình đánh giá bao gồm việc dịch toàn bộ câu nguồn trong tập test, sau đó so sánh kết quả dịch của mô hình với các bản dịch tham chiếu (câu đích) bằng các chỉ số chuẩn như BLEU và ROUGE (ROUGE-1, ROUGE-2, ROUGE-L). Các chỉ số này cung cấp một thước đo định lượng về chất lượng dịch của mô hình sau khi fine-tuning.

4.1.2 Huấn luyện từ đầu với mô hình Transformer (Training from Scratch)

Một hướng tiếp cận khác, thay vì sử dụng trọng số đã được huấn luyện trước, là xây dựng và huấn luyện mô hình Transformer hoàn toàn từ đầu trên tập dữ liệu song ngữ Anh-Việt đã thu thập. Hướng đi này cho phép kiểm soát hoàn toàn kiến

trúc và quá trình học của mô hình nhưng thường đòi hỏi nhiều dữ liệu và tài nguyên tính toán hơn đáng kể so với fine-tuning.

Quy trình thực hiện trong notebook final-nlp-scratch-transformer.ipynb bao gồm các bước chính sau:

1. Xây dựng Kiến trúc Transformer

- Mô hình Transformer theo kiến trúc Encoder-Decoder chuẩn (như trong bài báo "Attention Is All You Need"") được tự cài đặt (implement) bằng thư viện PyTorch (torch.nn).
- Các thành phần cốt lõi được định nghĩa thành các lớp (classes) riêng biệt:
  - ✓ Embedder: Lớp tạo vector embedding cho từ.
  - ✓ PositionalEncoder: Lớp tạo mã hóa vị trí sin/cos.
  - ✓ MultiHeadAttention: Cài đặt cơ chế Multi-Head Attention.
  - ✓ FeedForward: Cài đặt mạng nơ-ron truyền thẳng position-wise.
  - ✓ Norm: Cài đặt lớp Layer Normalization.
  - ✓ EncoderLayer và DecoderLayer: Kết hợp các thành phần trên để tạo thành một lớp Encoder và Decoder đơn lẻ (bao gồm cả kết nối tắt và chuẩn hóa lớp).
  - ✓ Encoder và Decoder: Chồng (stack) nhiều lớp EncoderLayer hoặc DecoderLayer (số lớp N được định nghĩa trong hyperparameters, ví dụ N=6).
  - ✓ Transformer: Lớp mô hình tổng thể, kết hợp Encoder, Decoder và một lớp Linear cuối cùng để đưa ra dự đoán trên từ vựng đích.

2. Chuẩn bị Dữ liệu (Sử dụng torchtext phiên bản cũ)

- Notebook này sử dụng thư viện torchtext (phiên bản 0.6.0) để xử lý dữ liệu, một cách tiếp cận phổ biến trước khi thư viện datasets và transformers của Hugging Face trở nên thịnh hành.

- Tokenizer tùy chỉnh: Một lớp tokenize được định nghĩa, sử dụng regex để làm sạch cơ bản và nltk.word_tokenize để tách từ cho cả tiếng Anh và tiếng Việt. Lưu ý: Mặc dù pyvi được cài đặt, tokenizer này không sử dụng nó.
- Định nghĩa Fields: Sử dụng torchtext.data.Field để định nghĩa quy trình xử lý cho ngôn ngữ nguồn (SRC) và ngôn ngữ đích (TRG). Các Field này đảm nhiệm việc:
  - ✓ Áp dụng tokenizer tùy chỉnh.
  - ✓ Chuyển thành chữ thường (lower=True).
  - ✓ Thêm các token đặc biệt (<sos>, <eos> cho ngôn ngữ đích).
  - ✓ Xây dựng từ vựng (Vocabulary): Tạo ra hai bộ từ vựng riêng biệt cho tiếng Anh và tiếng Việt trực tiếp từ tập dữ liệu huấn luyện (SRC.build_vocab(train), TRG.build_vocab(train)). Kích thước từ vựng được xác định tự động dựa trên dữ liệu.
- Tạo Dataset và Iterator:
  - ✓ Đọc dữ liệu từ các file .en và .vi.
  - ✓ Sử dụng torchtext.data.TabularDataset để tải dữ liệu.
  - ✓ Sử dụng một lớp MyIterator tùy chỉnh (kế thừa từ torchtext.data.Iterator) để tạo các lô (batches) dữ liệu cho quá trình huấn luyện và validation. Iterator này có thể bao gồm việc sắp xếp câu theo độ dài và sử dụng hàm batch_size_fn để tạo batch động, tối ưu hóa việc sử dụng bộ nhớ GPU.

3. Tạo Masks

- Các hàm nopeak_mask và create_masks được định nghĩa để tạo ra:
  - ✓ Mặt nạ đệm (padding mask) cho cả chuỗi nguồn và chuỗi đích: giúp mô hình bỏ qua các token <pad> được thêm vào để các câu trong batch có cùng độ dài.
  - ✓ Mặt nạ nhìn về phía trước (look-ahead mask hay subsequent mask) cho chuỗi đích: đảm bảo rằng tại mỗi bước dự đoán

trong Decoder, mô hình chỉ có thể “nhìn” thấy các token ở vị trí hiện tại và các vị trí trước đó, không được "nhìn" các token tương lai.

4. Thiết lập Huấn luyện

- Hyperparameters: Các siêu tham số quan trọng được định nghĩa trong dictionary opt, bao gồm: đường dẫn dữ liệu, kích thước embedding (d_model=512), số lớp Encoder/Decoder (n_layers=6), số lượng attention heads (heads=8), tỉ lệ dropout (0.1), tốc độ học (lr=0.0001), số epochs (5), kích thước batch (1500, có thể liên quan đến dynamic batching), thiết bị ('cuda').
- Khởi tạo Mô hình: Mô hình Transformer tự cài đặt được khởi tạo với kích thước từ vựng SRC và TRG, cùng các siêu tham số đã định nghĩa. Trọng số của mô hình được khởi tạo ngẫu nhiên.
- Optimizer: Sử dụng torch.optim.Adam với các tham số beta và epsilon cụ thể.
- Loss Function: Sử dụng nn.CrossEntropyLoss, có tùy chọn bỏ qua chỉ số của token padding (ignore_index=trg_pad).

5. Vòng lặp Huấn luyện

- Notebook triển khai một vòng lặp huấn luyện chuẩn:
  - ✓ Lặp qua số epochs đã định.
  - ✓ Trong mỗi epoch, lặp qua các batch dữ liệu từ train_iter.
  - ✓ Với mỗi batch, thực hiện một bước huấn luyện (step function): đưa dữ liệu lên GPU, tạo masks, thực hiện forward pass qua mô hình, tính loss, thực hiện backpropagation để tính gradients, và cập nhật trọng số bằng optimizer.
  - ✓ Sau mỗi epoch, tính toán và in ra loss trung bình trên tập huấn luyện.

- ✓ Thực hiện đánh giá trên tập validation (valid_iter) bằng hàm validate: tính loss trung bình trên tập validation mà không cập nhật trọng số.
- ✓ Lưu trữ Checkpoint: Lưu lại trạng thái của mô hình (model_state_dict) và optimizer (optimizer_state_dict) vào file (model_mt.pth) nếu validation loss của epoch hiện tại tốt hơn (thấp hơn) so với các epoch trước đó.

6. Inference (Dịch câu) và Đánh giá

- Beam Search: Cài đặt thuật toán Beam Search (k=5) để sinh câu dịch. Beam search tìm kiếm các chuỗi dịch có xác suất cao thay vì chỉ chọn từ có xác suất cao nhất tại mỗi bước (greedy decoding), thường cho kết quả tốt hơn.
- Hàm translate_sentence: Bao bọc quá trình dịch một câu đơn lẻ, bao gồm tokenization, numericalization (ánh xạ sang ID), chạy beam search, và chuyển đổi ID kết quả thành chuỗi văn bản, kèm theo hậu xử lý đơn giản về dấu câu.
- Tải Checkpoint: Tải lại trọng số từ file checkpoint (model_mt.pth) đã được lưu trong quá trình huấn luyện (checkpoint có validation loss tốt nhất).
- Đánh giá định lượng: Sử dụng hàm evaluate_model để:
  - ✓ Tải các metric BLEU và ROUGE từ thư viện evaluate.
  - ✓ Đọc lại tập dữ liệu validation (được dùng như tập test trong trường hợp này).
  - ✓ Dịch toàn bộ câu nguồn trong tập validation bằng hàm translate_sentence (sử dụng beam search).
  - ✓ So sánh các câu dịch được (candidates) với các câu tham chiếu (references) bằng metric BLEU và ROUGE.
  - ✓ In ra các điểm số cuối cùng.

4.2 Sử dụng mô hình GPT (Generative Pretrained Transformer)

Một hướng tiếp cận khác để giải quyết bài toán dịch máy là tận dụng các Mô hình Ngôn ngữ Lớn (LLMs) đã được tiền huấn luyện (pre-trained) như GPT. Thay vì sử dụng kiến trúc Encoder-Decoder chuyên biệt cho dịch máy, phương pháp này xem bài toán dịch như một bài toán sinh văn bản có điều kiện (conditional text generation).

4.2.1 Sử dụng Pre-trained Model và Fine-tuning (GPT-2)

Hướng tiếp cận này sử dụng một mô hình GPT đã được huấn luyện trước (trong trường hợp này là GPT-2, một mô hình generative mạnh mẽ chủ yếu được huấn luyện trên dữ liệu tiếng Anh) và tinh chỉnh (fine-tuning) nó để thực hiện việc dịch từ tiếng Anh sang tiếng Việt. Ý tưởng cốt lõi là "dạy" cho mô hình GPT-2 cách hoàn thành (generate) câu tiếng Việt khi được cung cấp (prompted) câu tiếng Anh tương ứng theo một định dạng cụ thể.

Quy trình thực hiện trong notebook final-nlp-pretrained-gpt.ipynb bao gồm các bước sau:

1. Chuẩn bị và Định dạng Dữ liệu

- Tập dữ liệu song ngữ Anh-Việt đã thu thập được tải và chuẩn bị. Một mẫu (sample) 100,000 cặp câu được sử dụng.
- Định dạng Prompt: Đây là bước then chốt. Mỗi cặp câu (Anh, Việt) trong dữ liệu huấn luyện được định dạng lại thành một chuỗi văn bản duy nhất theo cấu trúc: "English: [Câu tiếng Anh]\\nVietnamese: [Câu tiếng Việt]<|endoftext|>"
  - ✓ English: và Vietnamese: là các nhãn cố định giúp mô hình nhận biết ngôn ngữ.
  - ✓ \\n là ký tự xuống dòng, phân tách hai phần.
  - ✓ <|endoftext|> là token đặc biệt của GPT-2, báo hiệu kết thúc một mẫu dữ liệu huấn luyện. Định dạng này biến bài toán dịch thành việc học cách sinh ra phần "Vietnamese: ..." khi được cung cấp phần "English: ...".

- Dữ liệu đã định dạng được chia thành tập huấn luyện (train - 90%) và tập kiểm định (validation - 10%) bằng thư viện datasets.

2. Tải Mô hình và Tokenizer GPT-2

- Model: Mô hình gpt2 (phiên bản gốc của OpenAI) được tải từ Hugging Face Hub bằng lớp GPT2LMHeadModel. Đây là mô hình chỉ chứa các lớp Decoder của Transformer và có sẵn một lớp "đầu" (head) cho nhiệm vụ Language Modeling (LM), phù hợp cho việc sinh văn bản và fine-tuning.
- Tokenizer: Tokenizer tương ứng GPT2Tokenizer cũng được tải về.
- Thêm Padding Token: Do GPT-2 nguyên bản không có token đệm (padding token), một token đệm mới (<|pad|>) được thêm vào tokenizer (tokenizer.pad_token = '<|pad|>'). Điều này là cần thiết để xử lý các batch câu có độ dài khác nhau.
- Điều chỉnh Embedding: Sau khi thêm các token mới vào tokenizer, kích thước từ vựng của tokenizer đã thay đổi. Do đó, lớp embedding của mô hình cũng cần được điều chỉnh để phù hợp với kích thước từ vựng mới. Việc này được thực hiện bằng cách gọi phương thức: model.resize_token_embeddings(len(tokenizer))

3. Tokenization Dữ liệu

- Hàm tokenize_function được định nghĩa để áp dụng GPT2Tokenizer đã tải cho các chuỗi văn bản đã định dạng (ví dụ: "English: ...\\nVietnamese: ...<|endoftext|>").
- Quá trình này chuyển đổi toàn bộ chuỗi văn bản thành input_ids và attention_mask, có cắt ngắn (truncation) nếu độ dài vượt quá 512 tokens. Việc đệm (padding) sẽ được thực hiện sau bởi DataCollator.
- Hàm này được áp dụng cho cả tập train và validation.

4. Cấu hình Huấn luyện (Fine-tuning)

- Data Collator: Sử dụng DataCollatorForLanguageModeling với mlm=False (chỉ định đây là Causal Language Modeling, không phải Masked Language Modeling). Data Collator này sẽ tự động tạo ra nhãn (labels) bằng cách dịch chuyển input_ids và thực hiện padding động cho các batch.
- Training Arguments: Các tham số cho quá trình fine-tuning được thiết lập thông qua TrainingArguments, bao gồm:
  - ✓ Số epochs (5).
  - ✓ Kích thước batch (train: 4, eval: 8 - batch size nhỏ do GPT-2 tốn bộ nhớ).
  - ✓ Bước tích lũy gradient (gradient accumulation steps = 8): Giúp mô phỏng batch size lớn hơn (4 * 8 = 32) mà không cần nhiều bộ nhớ GPU.
  - ✓ Tốc độ học (5e-5), weight decay, warmup steps.
  - ✓ Tần suất lưu checkpoint (save_steps=500) và đánh giá (eval_steps=500).
  - ✓ Kích hoạt đánh giá (eval_strategy="steps") và lưu mô hình tốt nhất dựa trên validation loss (load_best_model_at_end=True, metric_for_best_model="loss").

5. Huấn luyện (Fine-tuning)

- Một đối tượng Trainer được khởi tạo với mô hình GPT-2, training arguments, các tập dữ liệu đã tokenize, data collator, và tokenizer.
- Lệnh trainer.train() được thực thi, bắt đầu quá trình fine-tuning. Mô hình GPT-2 học cách dự đoán token tiếp theo trong chuỗi văn bản đã định dạng, qua đó gián tiếp học cách sinh câu tiếng Việt theo sau câu tiếng Anh trong prompt. Quá trình huấn luyện theo dõi cả training loss và validation loss.

6. Lưu Mô hình

- Sau khi huấn luyện, mô hình có hiệu năng tốt nhất trên tập validation (dựa trên validation loss thấp nhất) được lưu lại cùng với tokenizer vào thư mục chỉ định.

7. Inference (Sinh bản dịch)

- Mô hình GPT-2 đã fine-tune được tải lại, thường thông qua transformers.pipeline('text-generation') để tiện lợi hơn.
- Để dịch một câu tiếng Anh mới, câu đó được đặt vào cấu trúc prompt: "English: [Câu tiếng Anh mới]\\nVietnamese:"
- Pipeline generator được gọi với prompt này để sinh phần tiếp theo (dự kiến là câu tiếng Việt). Các tham số như max_length, num_return_sequences, pad_token_id được cung cấp.
- Hậu xử lý: Kết quả trả về từ pipeline chứa cả prompt ban đầu và phần được sinh ra. Cần phải xử lý chuỗi này để trích xuất và làm sạch phần văn bản tiếng Việt mà mô hình đã sinh ra sau nhãn "Vietnamese:", loại bỏ các token đặc biệt như <|endoftext|> hoặc <|pad|>, và các phần không mong muốn khác nếu có. Hàm generate_vietnamese trong notebook thực hiện việc này.

8. Đánh giá Định lượng

- Tương tự như các phương pháp khác, hiệu năng dịch của mô hình GPT-2 fine-tuned được đánh giá trên tập validation (hoặc tập test riêng).
- Các bản dịch được sinh ra bởi mô hình (sau hậu xử lý) được so sánh với các bản dịch tham chiếu bằng các chỉ số BLEU (sử dụng sacrebleu) và ROUGE (ROUGE-1, ROUGE-2, ROUGE-L).

4.2.2 Huấn luyện từ đầu với mô hình GPT (Training from Scratch)

Bên cạnh hướng tiếp cận sử dụng các mô hình đã huấn luyện trước như BERT hay Transformer Encoder-Decoder, nhóm cũng tiến hành xây dựng một mô hình GPT (Generative Pre-trained Transformer) từ đầu để thực hiện bài toán dịch

máy Anh-Việt. Cách tiếp cận này giúp kiểm soát hoàn toàn cấu trúc mô hình, tokenizer và quá trình huấn luyện, từ đó hiểu sâu hơn về cơ chế hoạt động của GPT trong bài toán sinh chuỗi ngôn ngữ tự nhiên.

Quy trình được thực hiện trong notebook final-gpt-from-scratch.ipynb gồm các bước chính sau:

1. Xây dựng Tokenizer và Byte Pair Encoding (BPE)

- Trong bài toán dịch máy, việc xây dựng một tokenizer hiệu quả là cực kỳ quan trọng. Thay vì sử dụng các tokenizer có sẵn, mô hình của chúng ta sử dụng Byte Pair Encoding (BPE) để tạo ra một tokenizer tùy chỉnh. BPE giúp giảm thiểu số lượng từ vựng cần thiết, đồng thời tạo ra khả năng chia nhỏ từ ngữ chưa từng thấy thành các đơn vị nhỏ hơn (subword units).
- Để thực hiện, chúng tôi sử dụng SentencePiece, một thư viện mạnh mẽ giúp tạo ra từ vựng chung cho cả hai ngôn ngữ (Anh-Việt). SentencePiece tạo ra các đoạn mã hóa số cho mỗi từ hoặc từ ghép (subword token) trong dữ liệu, giúp mô hình dễ dàng học hỏi và suy luận các chuỗi ngữ nghĩa trong ngữ cảnh dịch máy.
- Dữ liệu song ngữ (gồm các cặp câu Anh-Việt) sau khi được xử lý bằng SentencePiece, sẽ được lưu trữ dưới dạng các tệp .npy — định dạng numpy — để có thể sử dụng trong các bước tiếp theo của mô hình.

2. Cài đặt Kiến trúc GPT bằng PyTorch

- Mô hình GPT được xây dựng dựa trên Transformer Decoder-only, lấy cảm hứng từ mô hình gốc trong bài báo “Attention Is All You Need”. Đặc trưng của GPT là chỉ sử dụng khối Decoder mà không có Encoder, nhằm tối ưu hóa khả năng sinh chuỗi (text generation) và dự đoán từ tiếp theo trong chuỗi.
    - ✓ GPTEmbedding: Lớp này chịu trách nhiệm tạo embedding cho từng token và các vị trí của nó trong chuỗi văn bản (positional

embedding). Điều này giúp mô hình nhận diện thứ tự các token trong chuỗi.

- ✓ MaskedMultiHeadAttention: Đây là lớp attention có mặt nạ (masking), ngăn không cho mô hình "nhìn" vào các token phía sau khi đang xử lý chuỗi hiện tại, giúp duy trì tính chất autoregressive của GPT.
- ✓ FeedForwardBlock: Là khối mạng nơ-ron truyền thẳng sử dụng hàm kích hoạt ReLU, giúp tăng khả năng phi tuyến của mô hình.
- ✓ DecoderBlock: Đây là khối cơ bản trong GPT, bao gồm các thành phần attention, chuẩn hóa lớp, và residual connection (kết nối tắt) để tối ưu hóa quá trình huấn luyện.
- ✓ GPTModel: Mô hình chính là sự kết hợp của nhiều DecoderBlock, với lớp Linear ở cuối cùng để ánh xạ các token dự đoán tới từ vựng đầu ra.

3. Chuẩn bị Dữ liệu Đầu vào

- Sau khi xử lý dữ liệu và mã hóa bằng tokenizer BPE, dữ liệu được chia thành các cặp (input, target). Ở đây, target sẽ là input dịch chuyển một bước, điều này giúp mô hình học dự đoán token tiếp theo trong chuỗi.
- Các cặp câu này sẽ được chia thành hai tập con:
  - ✓ Tập huấn luyện (training): Dùng để huấn luyện mô hình.
  - ✓ Tập validation: Dùng để đánh giá hiệu suất mô hình trong suốt quá trình huấn luyện.
  - ✓ Dữ liệu sẽ được nạp vào thông qua torch.utils.data.Dataset và DataLoader, cho phép việc batching và shuffle dữ liệu, giúp tối ưu hóa quá trình huấn luyện.

4. Tạo Masks và Padding

- Trong quá trình huấn luyện mô hình GPT, việc tạo và áp dụng masking là rất quan trọng. Mặt nạ tam giác (look-ahead mask) sẽ ngăn không cho decoder "nhìn" vào các token tương lai khi dự đoán token hiện tại. Điều này giúp mô hình học được tính chất autoregressive — mỗi token chỉ có thể dựa vào các token trước đó để dự đoán token tiếp theo.
- Ngoài ra, padding mask sẽ được áp dụng để tránh tính toán trên các token <pad>, điều này giúp mô hình tập trung vào các từ thực sự có ý nghĩa trong chuỗi đầu vào.

5. Thiết lập Huấn luyện

- Quá trình huấn luyện mô hình GPT đòi hỏi việc tinh chỉnh các siêu tham số (hyperparameters) quan trọng:
    - ✓ Kích thước embedding (d_model=512): Xác định số chiều của không gian embedding cho mỗi token.
    - ✓ Số lớp decoder (n_layers=6): Xác định số lượng các lớp DecoderBlock trong mô hình.
    - ✓ Số heads attention (n_heads=8): Xác định số lượng heads trong cơ chế attention.
    - ✓ Dropout (0.1): Dùng để giảm thiểu hiện tượng overfitting.
    - ✓ Batch size (64): Số lượng dữ liệu trong mỗi batch.
    - ✓ Learning rate (0.0001): Tốc độ học của optimizer.
    - ✓ Epochs (5): Số lần lặp lại qua toàn bộ dữ liệu huấn luyện
- Mô hình sẽ sử dụng hàm mất mát nn.CrossEntropyLoss với chỉ số ignore_index là token padding, giúp mô hình bỏ qua các vị trí không cần thiết. Adam Optimizer sẽ được sử dụng với các tham số beta và epsilon giống như trong mô hình GPT gốc

6. Vòng lặp Huấn luyện

- Mỗi epoch trong quá trình huấn luyện sẽ bao gồm các bước sau:

- ✓ Đưa batch dữ liệu lên GPU, tạo các mask tương ứng.
- ✓ Thực hiện một forward pass qua mô hình, tính toán loss.
- ✓ Tiến hành backpropagation và cập nhật trọng số của mô hình.
- ✓ Tính toán và in loss trung bình của mỗi epoch.

- Mô hình cũng sẽ được đánh giá trên tập validation sau mỗi epoch. Trong trường hợp validation loss của epoch hiện tại tốt hơn các epoch trước đó, mô hình sẽ lưu lại checkpoint (trạng thái mô hình).

7. Sinh câu và Đánh giá kết quả

- Sau khi huấn luyện hoàn tất, mô hình có thể được sử dụng để sinh câu dịch từ một chuỗi mã hóa. Greedy Decoding được áp dụng để sinh ra câu dịch, trong đó mô hình chọn từ có xác suất cao nhất tại mỗi bước.
- Mô hình cũng được đánh giá bằng các metric như BLEU và ROUGE để đo lường chất lượng của câu dịch. Quá trình đánh giá bao gồm:
  - ✓ So sánh câu dịch sinh ra với câu tham chiếu từ tập validation.
  - ✓ Tính toán điểm số BLEU và ROUGE.
  - ✓ In ra kết quả điểm số cuối cùng để so sánh với các mô hình khác.

CHƯƠNG 5. THỰC NGHIỆM

5.1 Kết quả của các mô hình dịch máy

5.1.1 Mô hình Transformer pre-trained

```
[19]
from transformers import MarianMTModel, MarianTokenizer
src_text = ['My name is Sarah and I live in London']

model_name = '/kaggle/input/weight-pretrained-transformer/opus-mt-en-vi-finetuned-en-to-vi/checkpoint-25000'
tokenizer = MarianTokenizer.from_pretrained(model_name)
print(tokenizer.supported_language_codes)

model = MarianMTModel.from_pretrained(model_name)
translated = model.generate(**tokenizer(src_text, return_tensors="pt", padding=True))
[tokenizer.decode(t, skip_special_tokens=True) for t in translated]

['>>vie<<']

Could not render content for 'application/vnd.jupyter.widget-view+json'
{"model_id":"8c2078fa1dda4b58bbf9c32b25ae815e","version_major":2,"version_minor":0}

['Tên tôi là Sarah và tôi sống ở London']
```

Hình 5.1: Kết quả dịch câu của mô hình Transformer pre-trained

Kết quả dịch: Mô hình dịch các câu với mức độ tự nhiên cao hơn, ví dụ: “My name is Sarah and I live in London.” được dịch thành “Tên tôi là Sarah và tôi sống ở London.” rất sát nghĩa và tự nhiên.

Nhận xét: Mô hình này tạo ra các bản dịch mượt mà, giữ được ý nghĩa và ngữ pháp, đặc biệt với các câu thông dụng.

5.1.2 Mô hình Transformer traning from Scratch

```
Original: Can I borrow your book?
Translated: tôi có thể mượn cuốn sách của bạn được không?
--------------------------------------------------
Original: I want to learn how solve this problem.
Translated: tôi muốn học giải quyết vấn đề này như thế nào.
--------------------------------------------------
Original: Could you help me find the way to the train station?
Translated: bạn có thể giúp tôi tìm đường đến ga xe lửa không?
--------------------------------------------------
Original: Thank you very much for your help.
Translated: cảm ơn các bạn rất nhiều vì sự giúp đỡ của bạn.
--------------------------------------------------
```

Hình 5.2: Kết quả dịch câu của mô hình Transformer traning from Scratch

Mô hình Transformer huấn luyện từ đầu dịch tốt các câu đơn giản, nhưng còn hạn chế với câu phức tạp hoặc sai cấu trúc. Một số câu dịch tự nhiên, nhưng vẫn xuất hiện lỗi ngữ pháp và thiếu mượt mà. Điều này cho thấy mô hình có khả

năng học từ vựng và cấu trúc cơ bản, nhưng chưa đủ để xử lý ngôn ngữ tự nhiên một cách hoàn chỉnh.

5.1.3 Mô hình GPT pre-trained

```
Hoàn tất sinh dự đoán.
Ví dụ một vài cặp dự đoán và tham chiếu:
  EN: It would come from the butcher's, it would come from presents.
  REF: từ những người bán thịt hoặc trong các hộp quà.
  PRED: Nó sẽ giữa t
--------------------
  EN: So the only things I could manage to obtain was a kind of a compromise.
  REF: Vậy nên điều duy nhất mà tôi đã xoay sở để có được là một sự thoả hiệp tế nhị.
  PRED: Một điều mà tô
--------------------
  EN: Rather than me putting a dish down, they were allowed to help themselves to as much or as little as they wanted.
  REF: chứ không phải là theo ý của tôi. mọi người được phép gọi cho mình nhiều hay ít như họ muốn.
  PRED: Thay vì tôi lại bỏ sánh
--------------------
  EN: And everyone agrees that trees are beautiful, and I've never met anyone who says, "I don't like trees."
  REF: Và mọi người đều thống nhất cây cối đều đẹp. Vì tôi chưa từng nghe ai nói "Tôi chẳng thích cây" bao giờ,
  PRED: trên mọi người rằng
--------------------
  EN: In our time right now, we shape the AI of tomorrow.
  REF: Những gì ta làm bây giờ sẽ quyết định AI tương lai.
  PRED: Trong lúc này, chúng ta gi
```

Hình 5.3: Kết quả dịch câu của mô hình GPT pre-trained

Mô hình GPT pre-trained dịch tốt các câu đơn giản nhưng thường sai ngữ nghĩa hoặc rời rạc ở câu phức tạp.

Dù có ưu điểm về tính tự nhiên trong câu dễ, mô hình vẫn gặp hạn chế về ngữ cảnh và xử lý cú pháp phức tạp, ảnh hưởng đến độ chính xác tổng thể.

5.1.4 Mô hình GPT training from scratch

```
Input : Some people turn into rockers like this.
Target: Thực kinh khủng. Một vài người trở thành những rocker như thế.
Pred  :  một người người người vào đá như như như này

Input : As had so often happened on the Eastern Front Hitler refused to allow a strategic withdrawal until it was too late.
Target: "Tương tự như mặt trận phía đông, Hitler lại không cho phép rút quân cho đến khi quá trễ."
Pred  :  " như như thường ra ra ra phía phía phía hit phía phía chối chối phép cho cho rút chiến rút rút cho cho đến đến đến.

Input : "At 290 days old males are about 14.1 kg, and females are about 12.6 kg."
Target: "Khi 290 ngày tuổi, con đực đạt chừng 14,1 kg, còn con cái đạt khoảng 12.6 kg."
Pred  :  "vào 290 290 tuổi tuổi đực tuổi tuổi 14... kg, con con 12.6.666."."."

Input : "As a result, costs in the amount of 14 million euros were assumed."
Target: "Tuy nhiên, ước tính mức phí là khoảng 14 triệu euro."
Pred  :  "lảết quả, phí phí trong lượng số triệu, 14 được được được được."."

Input : We should do this again sometime.
Target: đôi khi chúng ta nên làm điều này một lần nữa
Pred  :  chúng ta nên làm điều lần lần lần
```

Hình 5.4: Kết quả dịch câu của mô hình GPT training from scratch

Mô hình GPT được huấn luyện từ đầu vẫn còn nhiều hạn chế trong việc nắm bắt ngữ nghĩa và cấu trúc ngôn ngữ, dẫn đến các bản dịch thiếu chính xác và không mạch lạc. Dù đã học được một số quy tắc cơ bản về từ vựng, mô hình chưa thể hiện

khả năng xử lý tốt các mối quan hệ ngữ pháp phức tạp hoặc duy trì tính tự nhiên trong câu dịch, đặc biệt với các câu có cấu trúc dài hoặc nhiều lớp nghĩa.

5.2 Kết quả chỉ số đánh giá của Rouge và Bleu

5.2.1 Mô hình Transformer pre-trained

```
Kết quả đánh giá:
BLEU score: 0.3826
ROUGE-1: 0.7583
ROUGE-2: 0.5725
ROUGE-L: 0.6917
```

Hình 5.5: Kết quả đánh giá của mô hình Transformer pre-trained

BLEU score: 0.3826: Chỉ số BLEU đạt mức 0.3826, cho thấy mô hình có khả năng tạo ra các bản dịch có mức độ tương đồng ngữ pháp và từ vựng với bản dịch tham chiếu ở mức trung bình phản ánh rằng mô hình có thể dịch đúng một phần cấu trúc câu và từ vựng.

ROUGE-1: 0.7583: Chỉ số ROUGE-1 đạt mức cao (0.7583), cho thấy mô hình có khả năng tốt trong việc tái hiện các từ đơn (unigram) tương đồng với bản dịch tham chiếu. Điều này ám chỉ rằng mô hình giữ được phần lớn từ vựng chính xác và phù hợp với ngữ nghĩa của câu gốc.

ROUGE-2: 0.5725: ROUGE-2 thấp hơn ROUGE-1, đạt 0.5725, cho thấy khả năng tái hiện các cặp từ (bigram) của mô hình ở mức khá. Mặc dù không cao bằng ROUGE-1, giá trị này vẫn chấp nhận được, phản ánh rằng mô hình có thể duy trì một phần cấu trúc cụm từ và mối liên hệ ngữ pháp giữa các từ, nhưng chưa thực sự xuất sắc trong các câu phức tạp.

ROUGE-L: 0.6917: ROUGE-L đạt 0.6917, cho thấy mô hình có khả năng tốt trong việc duy trì chuỗi con chung dài nhất (longest common subsequence) với bản dịch tham chiếu. Giá trị này thể hiện rằng mô hình không chỉ giữ được từ vựng mà còn duy trì được cấu trúc tổng thể của câu ở mức đáng kể.

5.2.2 Mô hình Transformer traning from Scratch

```
--- Kết quả đánh giá ---
BLEU: 29.37
ROUGE-1: 0.7876
ROUGE-2: 0.6570
ROUGE-L: 0.7442
ROUGE-Lsum: 0.7442
------------------------
```

Hình 5.6: Kết quả đánh giá của mô hình Transformer traning from Scratch

BLEU: 29.37: Chỉ số BLEU đạt 29.37 (giả sử được biểu diễn dưới dạng phần trăm, tương đương với 0.2937 nếu chuẩn hóa về thang 0-1). Đây là một giá trị trung bình thấp, cho thấy mô hình có khả năng tạo ra các bản dịch với độ tương đồng ngữ pháp và từ vựng so với bản dịch tham chiếu ở mức hạn chế.

ROUGE-1: 0.7876: Chỉ số ROUGE-1 đạt mức cao (0.7876), thể hiện rằng mô hình có khả năng tái hiện tốt các từ đơn (unigram) so với bản dịch tham chiếu. Giá trị này cho thấy mô hình duy trì được phần lớn từ vựng chính xác và phù hợp với ngữ nghĩa của câu gốc, một điểm mạnh đáng kể.

ROUGE-2: 0.6570: ROUGE-2 đạt 0.6570, thấp hơn ROUGE-1 nhưng vẫn ở mức khá. Điều này cho thấy mô hình có khả năng tái hiện các cặp từ (bigram) ở mức tương đối, duy trì được một phần cấu trúc cụm từ và mối liên hệ ngữ pháp. Tuy nhiên, giá trị này cũng phản ánh hạn chế trong việc xử lý các cụm từ phức tạp hoặc cấu trúc câu dài.

ROUGE-L: 0.7442: ROUGE-L đạt 0.7442, cho thấy mô hình có khả năng tốt trong việc duy trì chuỗi con chung dài nhất (longest common subsequence) với bản dịch tham chiếu. Giá trị này thể hiện rằng mô hình không chỉ giữ được từ vựng mà còn duy trì được cấu trúc tổng thể của câu ở mức đáng kể.

ROUGE-Lsum: 0.7442: Giá trị ROUGE-Lsum bằng với ROUGE-L, cho thấy mô hình duy trì được tính liên kết trong toàn bộ câu hoặc đoạn văn, không có sự khác biệt đáng kể giữa các chuỗi con dài nhất và tổng hợp.

5.2.3 Mô hình GPT pre-trained

```
--- Kết quả BLEU ---
BLEU Score: 0.7050

--- Kết quả ROUGE ---
ROUGE-1 (F1): 0.3847
ROUGE-2 (F1): 0.2342
ROUGE-L (F1): 0.3517
```

Hình 5.7: Kết quả đánh giá của mô hình GPT pre-trained

BLEU Score: 0.7050: Chỉ số BLEU đạt 0.7050, một giá trị cao, cho thấy mô hình có khả năng tạo ra các bản dịch với độ tương đồng ngữ pháp và từ vựng rất tốt so với bản dịch tham chiếu. Giá trị này vượt trội so với các mô hình trước (Transformer pre-trained: 0.3826, Transformer training from scratch: 0.2937), thể hiện rằng mô hình GPT pre-trained có khả năng tái hiện cấu trúc câu và ngữ nghĩa một cách chính xác, phù hợp với các tác vụ dịch máy yêu cầu độ chính xác cao.

ROUGE-1 (F1): 0.3847: Chỉ số ROUGE-1 đạt 0.3847, mức khá thấp so với các mô hình trước (Transformer pre-trained: 0.7583, Transformer training from scratch: 0.7876). Điều này cho thấy mô hình có hạn chế trong việc tái hiện các từ đơn (unigram) tương đồng với bản dịch tham chiếu, có thể do mô hình tập trung vào ngữ nghĩa tổng thể hơn là sự khớp chính xác từng từ.

ROUGE-2 (F1): 0.2342: ROUGE-2 đạt 0.2342, cũng ở mức thấp (so với Transformer pre-trained: 0.5725, Transformer training from scratch: 0.6570). Giá trị này cho thấy mô hình gặp khó khăn trong việc tái hiện các cặp từ (bigram), tức là khả năng giữ cấu trúc cụm từ hoặc mối liên hệ ngữ pháp giữa các từ còn hạn chế.

ROUGE-L (F1): 0.3510: ROUGE-L đạt 0.3510, thấp hơn nhiều so với các mô hình trước (Transformer pre-trained: 0.6917, Transformer training from scratch: 0.7442). Điều này phản ánh rằng mô hình không duy trì tốt chuỗi con chung dài nhất (longest common subsequence) với bản dịch tham chiếu, cho thấy hạn chế trong việc giữ cấu trúc tổng thể của câu.

5.2.4 Mô hình GPT training from Scratch

```
--- Evaluation Results ---
BLEU: 6.64
ROUGE-1: 0.5054
ROUGE-2: 0.2493
ROUGE-L: 0.4181
ROUGE-Lsum: 0.4181
```

Hình 5.8: Kết quả đánh giá của mô hình GPT training from Scratch

BLEU: 6.64: Chỉ số BLEU đạt 6.64 (giả sử là phần trăm, tương đương với 0.0664 nếu chuẩn hóa về thang 0-1). Đây là một giá trị rất thấp, cho thấy GPT training from scratch gặp khó khăn trong việc tạo ra các bản dịch chính xác, có thể do thiếu dữ liệu huấn luyện hoặc thời gian huấn luyện chưa đủ để mô hình học được các đặc trưng phức tạp của tác vụ dịch máy.

ROUGE-1: 0.5054: Chỉ số ROUGE-1 đạt 0.5054, ở mức trung bình, cho thấy mô hình có khả năng tái hiện một phần các từ đơn (unigram) tương đồng với bản dịch tham chiếu. Tuy nhiên, giá trị này thấp hơn đáng kể so với các mô hình khác (Transformer pre-trained: 0.7583, Transformer training from scratch: 0.7876, GPT pre-trained: 0.3847), cho thấy mô hình chỉ giữ được một phần từ vựng chính xác.

ROUGE-2: 0.2493: ROUGE-2 đạt 0.2493, cũng ở mức thấp, tương tự hoặc hơi cao hơn so với GPT pre-trained (0.2342) nhưng thấp hơn nhiều so với Transformer pre-trained (0.5725) và Transformer training from scratch (0.6570). Giá trị này cho thấy mô hình gặp khó khăn trong việc tái hiện các cặp từ (bigram), tức là khả năng duy trì cấu trúc cụm từ và mối liên hệ ngữ pháp còn rất hạn chế.

ROUGE-L: 0.4181: ROUGE-L đạt 0.4181, thấp hơn so với các mô hình khác (Transformer pre-trained: 0.6917, Transformer training from scratch: 0.7442, GPT pre-trained: 0.3510). Điều này phản ánh rằng mô hình không duy trì tốt chuỗi con chung dài nhất (longest common subsequence) với bản dịch tham chiếu, cho thấy hạn chế trong việc giữ cấu trúc tổng thể của câu.

ROUGE-Lsum: 0.4181: Giá trị ROUGE-Lsum bằng với ROUGE-L, cho thấy mô hình duy trì tính liên kết trong câu hoặc đoạn văn ở mức tương đương với ROUGE-L, không có sự khác biệt đáng kể.

TÀI LIỆU THAM KHẢO

Tiếng Anh

[1] Opus, "TED2020 - A Multilingual Parallel Corpus," [Online]. Available: https://opus.nlpl.eu/TED2020/corpus/version/TED2020. [Accessed: Apr. 2025].

[2] GeeksforGeeks, "NLP | BLEU score for evaluating Neural Machine Translation in Python," [Online]. Available: https://www.geeksforgeeks.org/nlp-bleu-score-for-evaluating-neural-machine-translation-python/. [Accessed: Apr. 2025].

[3] A. Vaswani et al., “Attention Is All You Need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[4] D. Bahdanau, K. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in International Conference on Learning Representations (ICLR), 2015. [Online]. Available: https://arxiv.org/abs/1409.0473

[5] P. Koehn, “Neural Machine Translation,” arXiv preprint, 2020. [Online]. Available: https://arxiv.org/abs/1709.07809

[6] Y. Zhang et al., "DialoGPT: Large-scale Generative Pretraining for Conversational Response Generation," in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 270–278, 2020. [Online]. Available: https://aclanthology.org/2020.acl-demos.30/

[7] P. Koehn, Statistical Machine Translation. Cambridge, U.K.: Cambridge University Press, 2009.

[8] Y. Goldberg, Neural Network Methods in Natural Language Processing. San Rafael, CA: Morgan & Claypool Publishers, 2017. [Online]. Available: https://doi.org/10.2200/S00762ED1V01Y201703HLT037.