ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN

DỊCH MÁY ANH-PHÁP VỚI MÔ HÌNH ENCODER – DECODER LSTM

GVHD: PGS. Nguyễn Tuấn Đăng

Lớp: DCT122C5

Thành viên :

1. Võ Kiều Anh - 3122411009
2. Hồ Đăng Khoa - 3122411095

Thành Phố Hồ Chí Minh, Tháng 12 năm 2025

LỜI CAM ĐOAN

Chúng tôi xin cam đoan đây là kết quả nghiên cứu của riêng chúng tôi dựa trên sự hiểu biết và nghiên cứu tài liệu, các số liệu và kết quả nghiên cứu nêu trong báo cáo là trung thực và hoàn toàn khách quan. Chúng tôi xin chịu trách nhiệm hoàn toàn về tính trung thực trong bài làm của mình.

Sinh viên thực hiện

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường đại học Sài Gòn, Khoa Công nghệ thông tin đã tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành bài nghiên cứu này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên PGS.TS.Nguyễn Tuấn Đăng đã dày công truyền đạt kiến thức và hướng dẫn chúng em trong quá trình làm bài.

Trong thời gian tìm hiểu và nghiên cứu, chúng em đã tiếp thu được rất nhiều kiến thức, tiếp cận công nghệ mới. Nhưng do kiến thức hạn chế và không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý của bạn đọc và giảng viên PGS.TS.Nguyễn Tuấn Đăng để bài tiểu luận của chúng em được hoàn thiện hơn.

Một lần nữa, chúng em xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

LỜI MỞ ĐẦU

Trong kỷ nguyên toàn cầu hóa và bùng nổ thông tin, rào cản ngôn ngữ là một trong những thách thức lớn nhất đối với sự kết nối và phát triển tri thức. Sự chuyển dịch từ Dịch máy Thống kê (SMT) sang Dịch máy Nơ-ron (Neural Machine Translation - NMT) đã tạo ra một cuộc cách mạng thực sự, cho phép máy tính không chỉ dịch từng từ mà còn nắm bắt được ngữ nghĩa và văn phong của con người.

Tuy nhiên, các kiến trúc NMT nền tảng, điển hình là mô hình Encoder-Decoder sử dụng mạng nơ-ron hồi quy (RNN/LSTM), vẫn tồn tại một nhược điểm chí tử: "Nút thắt cổ chai thông tin" (Information Bottleneck). Trong kiến trúc này, toàn bộ nội dung của câu nguồn, bất kể độ dài hay độ phức tạp, đều bị buộc phải nén vào một vector ngữ cảnh (context vector) cố định duy nhất. Điều này dẫn đến hiện tượng mất mát thông tin cục bộ và suy giảm chất lượng dịch rõ rệt đối với các câu dài, đặc biệt là giữa các cặp ngôn ngữ có cấu trúc ngữ pháp khác biệt như tiếng Anh và tiếng Pháp.

Để giải quyết vấn đề trên, cơ chế Sự chú ý (Attention Mechanism) đã ra đời như một lời giải đột phá, cho phép mô hình "nhìn lại" và tập trung vào các từ khóa quan trọng trong câu nguồn tại mỗi bước giải mã, thay vì phụ thuộc vào một vector nén duy nhất.

Đồ án "Xây dựng hệ thống dịch máy Anh – Pháp sử dụng kiến trúc Seq2Seq và cơ chế Luong Attention" được thực hiện nhằm mục đích nghiên cứu sâu sắc hiệu quả của cải tiến này. Không chỉ dừng lại ở việc cài đặt mô hình, đề tài còn tập trung vào phân tích thực nghiệm đa chiều trên bộ dữ liệu chuẩn Multi30k. Cụ thể, nghiên cứu hướng tới ba mục tiêu chính:

1. Xây dựng lại mô hình Seq2Seq cơ bản làm baseline,
2. Triển khai phiên bản có Attention để cải thiện khả năng học ngữ cảnh,
3. Đánh giá và so sánh hiệu quả của hai hướng tiếp cận thông qua các thước đo thực nghiệm.

Kết quả của đồ án không chỉ là một hệ thống dịch thuật hoạt động tốt với chỉ số BLEU khả quan, mà quan trọng hơn, nó cung cấp một cái nhìn sâu sắc về bản chất của quá trình huấn luyện mô hình ngôn ngữ. Việc nắm vững nguyên lý hoạt động của Seq2Seq và Attention trong đồ án này chính là nền tảng lý thuyết vững chắc để tiếp cận các kiến trúc tối tân hiện nay như Transformer hay các Mô hình Ngôn ngữ Lớn (LLMs/GPT), đóng góp vào hành trang kiến thức cho các nghiên cứu chuyên sâu về Xử lý Ngôn ngữ Tự nhiên (NLP) trong tương lai.

Bái báo cáo được chia thành các chương:

Chương 1: Tổng quan về đề tài

Chương 2: Phân tích và tiền xử lý dữ liệu

Chương 3: Cơ sở lý thuyết và kiến trúc mô hình

Chương 4: Xây dựng thiết kế thực nghiệm

Chương 5: Đánh giá kết quả thực nghiệm

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Bối cảnh nghiên cứu

Sự phát triển nhanh chóng của công nghệ trí tuệ nhân tạo (AI) trong thập kỷ qua đã tạo ra bước tiến lớn trong lĩnh vực xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP). Một trong những ứng dụng tiêu biểu và mang tính thách thức nhất của NLP là dịch máy (Machine Translation – MT), với mục tiêu giúp máy tính hiểu và chuyển đổi chính xác ý nghĩa của ngôn ngữ này sang ngôn ngữ khác.

Các hệ thống dịch truyền thống, như Rule-Based hay Statistical Machine Translation, tuy đặt nền móng cho lĩnh vực này, nhưng còn hạn chế trong việc nắm bắt ngữ cảnh dài và cấu trúc phức tạp của câu. Sự ra đời của mạng nơ-ron hồi quy (Recurrent Neural Network – RNN), đặc biệt là mạng LSTM (Long Short-Term Memory), cùng với kiến trúc Encoder–Decoder, đã mở ra hướng tiếp cận mới cho dịch máy. Mô hình này cho phép hệ thống học được biểu diễn ngữ nghĩa và quan hệ giữa các chuỗi từ, thay vì dựa trên các quy tắc ngôn ngữ được xác định trước.

Trong bối cảnh đó, việc nghiên cứu và xây dựng mô hình Encoder-Decoder LSTM cho bài toán dịch từ tiếng Anh sang tiếng Pháp là một bước quan trọng để hiểu rõ cơ chế hoạt động của dịch máy nơ-ron (Neural Machine Translation – NMT). Tiếng Anh và tiếng Pháp đều là ngôn ngữ phổ biến, có cấu trúc ngữ pháp khác biệt, nên là cặp ngôn ngữ phù hợp để thử nghiệm khả năng học biểu diễn ngữ nghĩa của mô hình.

Đề tài này không chỉ giúp củng cố nền tảng về xử lý chuỗi, mô hình nơ-ron và kỹ thuật huấn luyện mô hình học sâu bằng PyTorch, mà còn hướng đến việc tìm hiểu cách AI có thể học để hiểu và sinh ngôn ngữ tự nhiên một cách hiệu quả hơn. Kết quả của nghiên cứu này góp phần vào mục tiêu dài hạn của lĩnh vực AI: phát triển các hệ thống có khả năng giao tiếp và hiểu ngôn ngữ con người một cách tự nhiên, chính xác và linh hoạt hơn.

1.2. Vấn đề đặt ra

Trong nhiều năm qua, dịch máy (Machine Translation) đã trở thành một trong những hướng nghiên cứu trọng tâm của lĩnh vực Xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP). Tuy nhiên, việc chuyển đổi chính xác ý nghĩa, cấu trúc

ngữ pháp và sắc thái biểu đạt giữa hai ngôn ngữ vẫn là một thách thức lớn đối với các mô hình dịch truyền thống.

Các phương pháp dựa trên quy tắc (Rule-Based Machine Translation – RBMT) và thống kê (Statistical Machine Translation – SMT) tuy đạt được những kết quả nhất định, nhưng phụ thuộc nhiều vào nguồn dữ liệu song ngữ và các quy tắc ngữ pháp được thiết kế thủ công. Chúng thường không thể nắm bắt ngữ cảnh toàn cục của câu, dẫn đến hiện tượng dịch rời rạc, thiếu tự nhiên, hoặc sai lệch ngữ nghĩa khi câu nguồn có độ dài lớn hay cấu trúc phức tạp.

Sự ra đời của dịch máy nơ-ron (Neural Machine Translation – NMT) đã mở ra bước ngoặt mới khi cho phép mô hình học trực tiếp mối quan hệ giữa chuỗi từ ngữ đầu vào và đầu ra thông qua kiến trúc Encoder–Decoder. Trong đó, mạng LSTM (Long Short-Term Memory) được sử dụng để khắc phục hiện tượng mất thông tin trong chuỗi dài, giúp mô hình ghi nhớ và biểu diễn ngữ nghĩa tốt hơn.

Tuy nhiên, mô hình Encoder-Decoder LSTM cơ bản với context vector cố định vẫn tồn tại những hạn chế đáng kể:

- Khi câu nguồn quá dài, toàn bộ thông tin bị nén vào một vector duy nhất, khiến decoder không thể tái tạo đầy đủ ngữ nghĩa của câu đích.
- Các hiện tượng ngữ pháp và từ vựng phức tạp, đặc biệt trong ngôn ngữ có sự khác biệt cú pháp lớn như tiếng Anh và tiếng Pháp, dễ gây ra lỗi dịch sai hoặc thiếu từ.
- Mô hình thiếu cơ chế tập trung (attention) nên không thể xác định chính xác phần nội dung nào cần được ưu tiên xử lý trong từng bước giải mã.
- Việc xử lý các từ hiếm (out-of-vocabulary) và độ phong phú biểu đạt ngôn ngữ tự nhiên vẫn là thách thức chưa được giải quyết triệt để.

Chính vì vậy, bài toán nghiên cứu và triển khai mô hình Encoder-Decoder LSTM cho dịch máy Anh–Pháp được đặt ra nhằm tìm hiểu:

- Mức độ hiệu quả của mô hình dịch nơ-ron tuần tự khi không sử dụng cơ chế attention.
- Những giới hạn của việc nén toàn bộ ngữ nghĩa vào một context vector cố định.

- Khả năng cải thiện chất lượng dịch thông qua việc phân tích lỗi, điều chỉnh tham số và mở rộng kiến trúc mô hình.

Thông qua đó, đề tài hướng đến việc hiểu sâu hơn cơ chế hoạt động của dịch máy nơ-ron, đánh giá năng lực biểu diễn của LSTM trong bài toán dịch song ngữ, và làm cơ sở cho các hướng nghiên cứu mở rộng như attention mechanism, beam search hay mô hình Transformer trong tương lai.

1.3. Mục tiêu và phạm vi nghiên cứu

1.3.1. Mục tiêu tổng quát

Đề tài hướng đến việc xây dựng và đánh giá một mô hình dịch máy Anh–Pháp sử dụng kiến trúc Encoder-Decoder dựa trên mạng LSTM, từ đó tìm hiểu khả năng của mô hình trong việc biểu diễn và chuyển đổi ngữ nghĩa giữa hai ngôn ngữ có cấu trúc cú pháp khác nhau.

Thông qua quá trình triển khai, mô hình được kỳ vọng sẽ giúp làm rõ hiệu quả của phương pháp học tuần tự (sequence-to-sequence learning) trong bài toán dịch ngôn ngữ tự nhiên, đồng thời đánh giá được những giới hạn khi không sử dụng cơ chế attention.

1.3.2. Mục tiêu cụ thể

a. Chuẩn hóa và tổ chức dữ liệu song ngữ (Multi30K, en–fr)

Thu thập và xử lý dữ liệu từ bộ Multi30K, gồm các cặp câu tương ứng giữa tiếng Anh và tiếng Pháp. Các bước xử lý như tách từ, chuẩn hóa và giới hạn từ vựng được thực hiện nhằm giúp mô hình dễ dàng học được mối quan hệ giữa hai ngôn ngữ.

b. Thiết kế và hiện thực mô hình Seq2Seq với LSTM

Phát triển mô hình dịch tự động dựa trên kiến trúc Encoder-Decoder sử dụng LSTM, trong đó phần mã hóa giúp mô hình hiểu ngữ nghĩa của câu tiếng Anh, còn phần giải mã tạo ra câu tiếng Pháp tương ứng. Ngoài mô hình cơ bản, đề tài cũng xem xét phiên bản có cơ chế tập trung (attention) để so sánh hiệu quả.

c. Huấn luyện mô hình và theo dõi quá trình học

Tiến hành huấn luyện mô hình trên tập dữ liệu đã chuẩn bị, điều chỉnh các thông số học phù hợp để đạt được kết quả ổn định. Quá trình này được theo dõi thông qua các chỉ số sai số và biểu đồ thể hiện mức độ cải thiện sau từng giai đoạn học.

d. Đánh giá hiệu quả của hệ thống dịch

Đánh giá chất lượng bản dịch bằng cách so sánh với bản dịch chuẩn, sử dụng thang điểm BLEU và quan sát sự khác biệt giữa mô hình có và không có attention. Kết quả được minh họa bằng các ví dụ cụ thể và biểu đồ trực quan hóa

e. Phân tích và rút ra nhận xét từ kết quả

Phân tích những điểm mạnh và hạn chế của mô hình, đặc biệt trong các trường hợp câu dài, câu phức tạp hoặc chứa từ hiếm. Từ đó, rút ra nguyên nhân dẫn đến lỗi dịch và xác định phạm vi mà mô hình hoạt động hiệu quả nhất.

f. Đề xuất hướng cải thiện cho nghiên cứu tiếp theo

Dựa trên kết quả và hạn chế quan sát được, đề xuất một số hướng phát triển như tăng kích thước mô hình, sử dụng kỹ thuật tách từ con (subword units), hoặc áp dụng cơ chế học tiên tiến hơn để nâng cao độ chính xác và tự nhiên của bản dịch.

1.3.3. Phạm vi nghiên cứu

Đề tài tập trung nghiên cứu bài toán dịch máy song ngữ từ tiếng Anh sang tiếng Pháp bằng cách sử dụng mô hình học sâu Encoder-Decoder dựa trên mạng LSTM.

Cụ thể, nghiên cứu được thực hiện trong các phạm vi sau:

- **Về dữ liệu**: sử dụng bộ Multi30K, gồm khoảng 29.000 cặp câu huấn luyện và 1.000 cặp cho mỗi tập kiểm thử và xác thực. Dữ liệu có độ dài câu trung bình từ 10–15 từ, phù hợp để huấn luyện trên máy tính cá nhân hoặc GPU phổ thông.
- **Về mô hình**: tập trung vào kiến trúc Encoder-Decoder cơ bản, trong đó thử nghiệm hai biến thể: có và không có cơ chế tập trung (attention).
- **Về công cụ**: sử dụng ngôn ngữ lập trình Python và thư viện PyTorch để xây dựng, huấn luyện và đánh giá mô hình.

- **Về đánh giá**: áp dụng thước đo BLEU score để định lượng mức độ tương đồng giữa bản dịch của mô hình và bản dịch chuẩn; đồng thời phân tích thêm một số ví dụ điển hình nhằm đánh giá chất lượng dịch ở mức định tính.

1.3.4. Giới hạn nghiên cứu

Mặc dù mô hình Encoder-Decoder LSTM là một bước tiến đáng kể trong dịch máy, đề tài vẫn tồn tại một số giới hạn nhất định:

Chưa áp dụng cơ chế học nâng cao như Transformer, beam search, hay subword tokenization (BPE), do phạm vi và thời gian nghiên cứu có hạn.

Không mở rộng sang các ngôn ngữ khác ngoài cặp Anh–Pháp.

Dữ liệu Multi30K tuy phù hợp cho nghiên cứu học thuật, nhưng quy mô nhỏ và chủ đề hẹp (mô tả hình ảnh), nên chưa phản ánh đầy đủ sự đa dạng ngôn ngữ trong thực tế.

Tài nguyên tính toán được giới hạn trong môi trường GPU phổ thông, vì vậy mô hình có kích thước nhỏ hơn so với các hệ thống dịch thương mại hiện nay.

Việc đánh giá chất lượng dịch chủ yếu dựa trên BLEU score và phân tích thủ công, chưa sử dụng các chỉ số ngữ nghĩa chuyên sâu hơn (như METEOR hoặc COMET).

Những giới hạn trên được xem là cơ sở để định hướng cho các nghiên cứu mở rộng trong tương lai, đặc biệt là trong việc ứng dụng attention toàn phần, mô hình Transformer, và dữ liệu quy mô lớn hơn để cải thiện độ chính xác và tính tự nhiên của bản dịch.

1.4. Giả thuyết nghiên cứu

Trong dịch máy nơ-ron, mô hình Encoder-Decoder sử dụng mạng LSTM được xem là một trong những nền tảng quan trọng giúp hệ thống học cách chuyển đổi giữa các ngôn ngữ có cấu trúc khác nhau. Tuy nhiên, việc biểu diễn toàn bộ thông tin của câu gốc bằng một vector ngữ cảnh duy nhất (context vector) có thể gây mất dữ liệu ngữ nghĩa, đặc biệt đối với các câu dài hoặc có cấu trúc ngữ pháp phức tạp.

Từ đặc điểm này, nghiên cứu này được hình thành dựa trên các giả thuyết như sau:

Giả thuyết 1:

Mô hình Encoder-Decoder LSTM không có cơ chế attention vẫn có thể học được mối quan hệ cơ bản giữa câu nguồn và câu đích trong cặp ngôn ngữ Anh–Pháp, nhờ khả

năng ghi nhớ ngữ cảnh của mạng LSTM. Tuy nhiên, hiệu quả dịch sẽ giảm dần khi độ dài câu tăng hoặc khi cấu trúc ngữ pháp trở nên phức tạp.

Giả thuyết 2:

Việc bổ sung cơ chế attention giúp mô hình tập trung vào các phần quan trọng của câu nguồn trong từng bước dịch, từ đó cải thiện độ chính xác và tự nhiên của bản dịch, đặc biệt với các câu dài hoặc chứa nhiều mệnh đề phụ.

Giả thuyết 3:

Chất lượng bản dịch phụ thuộc không chỉ vào kiến trúc mô hình mà còn vào chất lượng dữ liệu huấn luyện và kích thước từ vựng. Dữ liệu giới hạn và từ hiếm (OOV) sẽ ảnh hưởng tiêu cực đến khả năng diễn đạt của mô hình, dù cấu trúc mạng được tối ưu.

Giả thuyết 4:

Với tập dữ liệu Multi30K có quy mô vừa phải, mô hình Encoder-Decoder LSTM có thể đạt kết quả dịch chấp nhận được (theo BLEU score), nhưng chưa thể đạt đến mức tự nhiên của các hệ thống dịch thương mại sử dụng kiến trúc Transformer.

Từ các giả thuyết này, nghiên cứu hướng đến việc kiểm chứng mức độ chính xác và khả năng tổng quát hóa của mô hình Encoder-Decoder LSTM khi áp dụng cho bài toán dịch Anh–Pháp, đồng thời làm rõ tác động của cơ chế attention đối với hiệu suất dịch máy.

1.5. Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài là mô hình dịch máy nơ-ron (Neural Machine Translation – NMT) sử dụng kiến trúc Encoder-Decoder dựa trên mạng LSTM (Long Short-Term Memory).

Mô hình này được lựa chọn vì khả năng xử lý tốt chuỗi dữ liệu có độ dài thay đổi và khả năng ghi nhớ ngữ cảnh trong quá trình dịch.

Cụ thể, đề tài tập trung vào ba nhóm đối tượng chính sau:

1.5.1. Mô hình học sâu Encoder-Decoder LSTM:

Đây là thành phần cốt lõi của nghiên cứu, chịu trách nhiệm mã hóa câu tiếng Anh thành dạng biểu diễn vector (Encoder) và giải mã để sinh ra câu tiếng Pháp (Decoder). Hai biến thể được xem xét gồm mô hình không có attention và mô hình có attention (Luong), nhằm đánh giá sự khác biệt trong hiệu quả dịch.

1.5.2. Tập dữ liệu song ngữ Multi30K:

Bộ dữ liệu gồm các cặp câu mô tả hình ảnh bằng tiếng Anh và tiếng Pháp. Dữ liệu có độ dài câu trung bình ngắn, cấu trúc rõ ràng và chủ đề nhất quán, giúp thuận tiện cho việc huấn luyện và đánh giá mô hình dịch máy ở quy mô học thuật.

1.5.3. Quá trình học và đánh giá ngôn ngữ tự nhiên:

Bao gồm các hoạt động tiền xử lý dữ liệu, huấn luyện, kiểm thử, đánh giá kết quả bằng BLEU score, và phân tích lỗi dịch thuật để làm rõ mối quan hệ giữa cấu trúc mô hình, chất lượng dữ liệu và hiệu quả dịch.

Thông qua việc nghiên cứu ba nhóm đối tượng trên, đề tài hướng đến hiểu rõ cách mà mô hình học sâu tiếp nhận, lưu giữ và chuyển đổi thông tin ngữ nghĩa giữa hai ngôn ngữ, từ đó rút ra các yếu tố ảnh hưởng đến chất lượng bản dịch và định hướng cải tiến trong các mô hình dịch máy thế hệ tiếp theo.

1.6. Ý nghĩa khoa học và thực tiễn của đề tài

1.6.1. Ý nghĩa khoa học

Đề tài góp phần làm rõ cách mà mô hình học sâu Encoder-Decoder LSTM xử lý ngôn ngữ tự nhiên trong bài toán dịch máy. Thông qua việc xây dựng và đánh giá mô hình, nghiên cứu giúp:

Hiểu rõ hơn cơ chế học chuỗi – chuỗi (sequence-to-sequence learning) và khả năng biểu diễn ngữ cảnh của mạng LSTM.

So sánh được sự khác biệt về hiệu quả giữa mô hình có và không có cơ chế attention, từ đó nhận diện vai trò của attention trong việc cải thiện chất lượng bản dịch.

Cung cấp cơ sở tham khảo cho những nghiên cứu nâng cao sau này về các mô hình dịch hiện đại hơn như Transformer hoặc các hệ thống dựa trên mô hình ngôn ngữ lớn.

1.6.2. Ý nghĩa thực tiễn

Về mặt ứng dụng, kết quả của đề tài có thể:

Góp phần xây dựng các hệ thống dịch tự động quy mô nhỏ, phù hợp cho nghiên cứu, giảng dạy hoặc thử nghiệm trong môi trường học thuật.

Giúp người học, đặc biệt là sinh viên ngành AI hoặc ngôn ngữ học tính toán, nắm được quy trình tạo và huấn luyện một mô hình dịch cơ bản từ dữ liệu thật.

Tạo tiền đề cho việc phát triển các công cụ hỗ trợ giao tiếp đa ngôn ngữ, như trợ lý ảo, chatbot hoặc phần mềm học ngoại ngữ có khả năng dịch và phản hồi theo ngữ cảnh.

Nhìn chung, đề tài không chỉ có giá trị trong việc làm rõ cách AI hiểu và sinh ngôn ngữ, mà còn có tính thực hành cao, giúp người học ứng dụng được kiến thức vào các bài toán dịch và xử lý ngôn ngữ trong thực tế.

1.7. Cấu trúc báo cáo

Báo cáo được bố cục thành năm chương chính, kết luận và phụ lục, được trình bày như sau:

Lời cảm ơn và Lời mở đầu: Giới thiệu tổng quan về đề tài, lý do chọn hướng nghiên cứu và phương pháp tiếp cận tổng quát.

- Chương 1 – Tổng quan về đề tài: Trình bày bối cảnh nghiên cứu, vấn đề đặt ra, mục tiêu, phạm vi, giả thuyết và đối tượng nghiên cứu, cùng với ý nghĩa khoa học và thực tiễn của đề tài.
- Chương 2 – Phân tích và tiền xử lý dữ liệu: Mô tả bộ dữ liệu Multi30K, thực hiện phân tích thống kê, quyết định tham số, cùng các bước tiền xử lý như tokenization, padding và packing.
- Chương 3 – Cơ sở lý thuyết và kiến trúc mô hình: Trình bày kiến thức nền tảng về dịch máy, mạng LSTM và mô hình Encoder–Decoder; giới thiệu cơ chế attention và vai trò của nó trong cải thiện kết quả dịch.
- Chương 4 – Thiết kế thực nghiệm và cài đặt: Mô tả môi trường lập trình, cấu hình huấn luyện, các kịch bản thử nghiệm được triển khai và quy trình thực hiện mô hình.
- Chương 5 – Đánh giá kết quả thực nghiệm: Phân tích kết quả huấn luyện, so sánh mô hình cơ bản (No-Attention) với mô hình cải tiến (Attention), đánh giá bằng BLEU score và rút ra nhận xét.
- Kết luận và hướng phát triển: Tóm tắt những kết quả đạt được, chỉ ra hạn chế của mô hình và đề xuất hướng nghiên cứu mở rộng trong tương lai.
- Phụ lục và Tài liệu tham khảo: Cung cấp mã nguồn, biểu đồ, bảng kết quả chi tiết và danh mục các tài liệu được trích dẫn trong quá trình thực hiện đề tài.

CHƯƠNG 2: PHÂN TÍCH VÀ TIỀN XỬ LÝ DỮ LIỆU

2.1. Tổng quan bộ dữ liệu

Bộ dữ liệu được sử dụng trong nghiên cứu là Multi30K English–French, một tập dữ liệu song ngữ mở rộng từ Flickr30K, ban đầu được xây dựng cho bài toán image captioning. Multi30K được phát triển thêm các cặp câu mô tả hình ảnh bằng nhiều ngôn ngữ, trong đó nghiên cứu này lựa chọn cặp Anh – Pháp nhằm phục vụ bài toán machine translation.

Khác với các tập dữ liệu dịch quy mô lớn như WMT hoặc Europarl, Multi30K có ưu điểm là quy mô vừa phải, ngữ liệu đơn giản, và nội dung mang tính mô tả trực quan, rất phù hợp để huấn luyện và đánh giá các mô hình Encoder–Decoder sử dụng mạng Recurrent Neural Network (RNN) hoặc LSTM. Các câu trong tập dữ liệu thường ngắn (trung bình 10–12 từ), chủ yếu mô tả đối tượng và hành động trong ảnh, nên ít xuất hiện các cấu trúc cú pháp phức tạp.

2.1.1. Quy mô dữ liệu

Tập dữ liệu được chia theo ba phần:

- Tập huấn luyện (Train): 29,000 cặp câu
- Tập kiểm định (Validation): 1,014 cặp câu
- Tập kiểm tra (Test): 1,000 cặp câu

Việc phân chia này đảm bảo sự cân bằng giữa khối lượng dữ liệu phục vụ huấn luyện và khối lượng dùng để đánh giá khả năng tổng quát hóa của mô hình. Các tập con có phân bố tương đồng về độ dài và cấu trúc câu, giúp hạn chế sai lệch trong quá trình huấn luyện và kiểm thử.

Loại tập	Số lượng cặp câu	Độ dài TB (EN)	Độ dài TB (FR)
Train	29,000	11.2 từ	12.4 từ
Validation	1,014	11.0 từ	12.1 từ
Test	1,000	10.9 từ	12.3 từ

Bảng 2.1: Thống kê tổng quan về độ dài trung bình của câu trong từng tập

Các chỉ số cho thấy sự đồng đều giữa hai ngôn ngữ, đồng thời phản ánh đặc điểm hình thái của tiếng Pháp – thường dài hơn tiếng Anh do sự xuất hiện của các yếu tố ngữ pháp như giống, số, và chia động từ.

2.1.2. Đặc điểm ngôn ngữ

Thống kê	Tiếng Anh	Tiếng Pháp
Số lượng câu	29.000	29.000
Độ dài trung bình	13.11 từ	14.28 từ
Độ dài trung vị	12	13
Tối thiểu / Tối đa	4 / 41	4 / 54
Phân vị 5% / 95%	8 / 21	8 / 23

Bảng 2.2: Bảng thống kê đặc điểm ngôn ngữ của tập dữ liệu

Các câu trong tập dữ liệu Multi30K mang phong cách mô tả trực tiếp (descriptive style), thường xuất hiện trong các bài chú thích ảnh (image captions). Nội dung câu chủ yếu mô tả hành động, đối tượng và bối cảnh thị giác, ví dụ như:

A man is riding a bicycle.

Un homme fait du vélo.

Đặc điểm này khiến cho dữ liệu có độ phức tạp cú pháp thấp, vốn từ vựng cụ thể, và mức độ đa nghĩa hạn chế, rất phù hợp cho huấn luyện các mô hình dịch máy dựa trên học biểu diễn tuần tự.

Về mặt ngôn ngữ học, tiếng Anh và tiếng Pháp có sự tương đồng tương đối cao về nguồn gốc (cùng hệ Ấn–Âu) nhưng vẫn tồn tại nhiều khác biệt quan trọng ảnh hưởng đến quá trình học của mô hình:

1. **Khác biệt hình thái học (Morphology)**
   - Tiếng Pháp có biến đổi giống và số cho danh từ, tính từ, mạo từ (un homme / une femme / des enfants), trong khi tiếng Anh hầu như không.
   - Điều này làm tăng số lượng hình thức bề mặt của một từ gốc → mô hình phải học được ánh xạ giữa nhiều biến thể hình thái tiếng Pháp và cùng một từ tiếng Anh.
2. **Khác biệt cú pháp (Syntax)**

- Tiếng Anh tuân theo trật tự từ ổn định S–V–O (Subject–Verb–Object), trong khi tiếng Pháp có thể thay đổi vị trí tính từ và tân ngữ phụ thuộc vào ngữ cảnh.
- Các cụm danh từ tiếng Pháp thường dài hơn do bắt buộc có mạo từ, ví dụ: A girl wearing a red dress → Une fille portant une robe rouge.
- Do đó, mô hình phải học được hiện tượng dịch chuyển vị trí từ (word reordering), vốn là một trong những thách thức cơ bản trong dịch máy.

3. **Khác biệt về từ vựng (Lexicon)**
   - Các câu trong Multi30K có xu hướng lặp lại các động từ hành động phổ biến (run, sit, hold, wear, play), và các danh từ chỉ người hoặc đồ vật trong ảnh (man, woman, child, dog, ball, bicycle).
   - Tiếng Pháp biểu đạt tương ứng bằng các cụm động từ có cấu trúc chặt chẽ hơn, ví dụ: is sitting → est assis, is playing → joue.
   - Nhờ tính song song chặt chẽ này, bộ dữ liệu hỗ trợ tốt cho việc học liên kết ngữ nghĩa (semantic alignment) giữa hai ngôn ngữ.
4. **Phủ định, đại từ và hiện tượng đặc thù tiếng Pháp**
   - Tiếng Pháp có hệ thống đại từ clitic (se, me, le, la, y, en) và phủ định kép (ne … pas), khiến tokenizer cần xử lý đúng hiện tượng lược âm (elision): l'homme, qu'une, d'un.
   - Các hiện tượng này được SpaCy tokenizer nhận dạng chính xác, giúp giảm lỗi phân tách từ trong giai đoạn tiền xử lý.
5. **Đặc điểm phong cách và độ dài câu**
   - Câu mô tả trong Multi30K thường ngắn, trung bình khoảng 13 từ tiếng Anh và 14 từ tiếng Pháp, phần lớn dưới 25 từ.
   - Phân bố độ dài hẹp và cấu trúc đơn giản giúp mô hình học dễ hơn, giảm nguy cơ mất ngữ cảnh ở các chuỗi dài.
   - Đây là lý do tập dữ liệu này thường được chọn để đánh giá mô hình dịch cơ sở (baseline translation models).

Nhìn chung, đặc điểm ngôn ngữ của Multi30K thể hiện sự cân bằng giữa đơn giản (về độ dài và ngữ pháp) và đa dạng (về từ vựng và hình thái). Điều này giúp mô hình Encoder–Decoder LSTM có thể học hiệu quả mối quan hệ song ngữ mà không cần đến các kiến trúc phức tạp như Transformer hay attention mở rộng.

2.1.3. Phân tích từ vựng

Thông số	Tiếng Anh	Tiếng Pháp
Tổng số token	380.190	414.065
Số lượng từ loại (types)	10.833	11.505
TTR (Type-token Ratio)	0.0285	0.0278

Bảng 2.3: Thống kê token từ vựng

Giá trị TTR (Type–Token Ratio) được tính bằng tỷ lệ giữa số lượng từ loại (types) và tổng số token. TTR càng thấp thể hiện mức độ lặp lại cao của từ vựng, tức là ngôn ngữ có xu hướng sử dụng nhiều từ quen thuộc trong nhiều ngữ cảnh khác nhau.

Điều này hoàn toàn phù hợp với đặc trưng của ngữ liệu mô tả hình ảnh (caption data), nơi các câu thường xoay quanh một tập hạn chế các danh từ và động từ chỉ người, vật, hành động,

a. Đặc trưng từ vựng tiếng Anh

- Tập từ vựng tiếng Anh có 10.833 từ riêng biệt, bao gồm phần lớn là danh từ và động từ thường gặp trong miêu tả hình ảnh.
- Các động từ phổ biến nhất thường là “is”, “are”, “wearing”, “holding”, “sitting”, “standing”, phản ánh cấu trúc mô tả hành động.
- Các danh từ phổ biến là “man”, “woman”, “dog”, “child”, “people”, “shirt”, “street”, “table”, thể hiện bối cảnh vật thể quen thuộc trong ảnh.
- Đặc biệt, dữ liệu gần như không có câu phức hoặc ngôn ngữ trừu tượng, nên tần suất xuất hiện của liên từ, đại từ, hoặc mệnh đề phụ rất thấp.
- Điều này giúp mô hình dễ dàng học được cấu trúc ngữ pháp đơn giản và mối quan hệ vị trí giữa từ (subject – verb – object).

b. Đặc trưng từ vựng tiếng Pháp

- Bộ từ vựng tiếng Pháp gồm 11.505 từ loại riêng biệt, cao hơn nhẹ so với tiếng Anh.
- Sự gia tăng này bắt nguồn từ đặc điểm hình thái phong phú của tiếng Pháp: danh từ và tính từ biến đổi theo giống (masculin/féminin) và số (singulier/pluriel), ví dụ: grand/grande/grands/grandes.

- Ngoài ra, tiếng Pháp chứa nhiều hình thức elision (l'homme, d'un, qu'une, s'il) và từ ghép có dấu nháy hoặc gạch nối (aujourd'hui, t-shirt), làm tăng số lượng dạng bề mặt (surface forms).
- Các động từ như est, sont, porte, tient, joue và danh từ như homme, femme, enfant, chien, table, rue là những từ xuất hiện nhiều nhất — phản ánh tương đồng về chủ đề với tiếng Anh nhưng đa dạng hình thức hơn.

c. Mức độ lặp lại và tính bao phủ của từ vựng

- Tổng số token trên 380 nghìn (EN) và 414 nghìn (FR) trong khi chỉ có khoảng 10–11 nghìn loại từ cho thấy vốn từ lặp lại rất nhiều.
- Khi xây dựng từ điển huấn luyện, ngưỡng 10.000 từ cho mỗi ngôn ngữ được xem là đủ để bao phủ trên 98% dữ liệu thực tế, hạn chế đáng kể tình trạng OOV (Out-of-Vocabulary) trong tập kiểm tra.
- Việc giới hạn kích thước từ vựng ở mức này còn giúp giảm chi phí tính toán và dung lượng embedding, trong khi vẫn giữ được độ bao phủ ngữ nghĩa cần thiết cho mô hình.

d. Hiện tượng từ vựng đặc trưng

- Từ chức năng (function words) chiếm tỷ trọng cao, đặc biệt trong tiếng Pháp do sự hiện diện bắt buộc của mạo từ (un, une, le, la, des) và giới từ (dans, sur, à, de).
- Từ có dấu (diacritics) trong tiếng Pháp (é, è, à, ç, ô, ï) được giữ nguyên trong quá trình tiền xử lý để tránh gộp sai các từ không đồng nghĩa.

e. Tác động đến mô hình dịch

- Với vốn từ khoảng 10k–11k cho mỗi ngôn ngữ, mô hình Encoder–Decoder có thể học được các biểu diễn từ vựng (embeddings) ổn định mà không cần sử dụng kỹ thuật subword.
- Tuy nhiên, các biến thể hình thái tiếng Pháp có thể khiến mô hình gặp hiện tượng OOV nội bộ (từ hiếm nhưng có cùng gốc), gợi ý hướng mở rộng bằng BPE (Byte Pair Encoding) hoặc Unigram Tokenization ở các mô hình sau.
- Sự chênh lệch nhỏ về độ dài trung bình và số lượng token giữa hai ngôn ngữ cho thấy khả năng cân bằng dịch tốt, ít lệch về độ dài đầu ra (length bias).ví dụ: man, woman, child, dog, ball, run, sit, stand, wear.

2.1.4. Tỉ lệ từ ngoài tập (OOV)

Tập dữ liệu	EN	FR
Validation	1.47%	1.50%
Test	1.32%	1.40%

Bảng 2.4: Tỉ lệ OOV

Từ OOV (Out-of-Vocabulary) là những từ xuất hiện trong tập kiểm định hoặc kiểm tra nhưng không có trong tập huấn luyện, tức không nằm trong bộ từ vựng mà mô hình đã học được. Trong quá trình huấn luyện, các từ này được thay thế bằng token đặc biệt <unk> để mô hình vẫn có thể xử lý chúng trong chuỗi đầu vào hoặc đầu ra.

Kết quả thống kê cho thấy tỉ lệ OOV của tập Multi30K là rất thấp (dưới 2%) ở cả hai ngôn ngữ, cụ thể là 1.47%–1.50% trong validation và 1.32%–1.40% trong test. Điều này chứng minh rằng ngưỡng từ vựng 10.000 được lựa chọn trong quá trình xây dựng tokenizer là hợp lý và hiệu quả. Bộ từ vựng này có khả năng bao phủ gần như toàn bộ ngữ liệu thực tế, giúp mô hình không bị ảnh hưởng đáng kể bởi hiện tượng "từ lạ" khi suy luận.

a. Nguyên nhân dẫn đến OOV thấp

- Tỉ lệ OOV thấp đến từ nhiều yếu tố:
- Ngữ liệu có tính lặp cao: Các câu mô tả hình ảnh thường sử dụng lặp lại một tập nhỏ các danh từ và động từ hành động cơ bản.
- Tiền xử lý nhất quán: Quá trình chuẩn hóa (lowercase, loại ký tự đặc biệt, thống nhất dấu nháy và dấu câu) giúp giảm số dạng từ khác nhau cho cùng một từ gốc.
- Cấu trúc từ vựng gọn: Vocab 10k được xây dựng dựa trên tần suất xuất hiện trong tập train, đảm bảo cân bằng giữa kích thước từ điển và độ bao phủ.
- Ngữ cảnh đơn giản: Do các câu trong Multi30K ngắn và có cấu trúc lặp lại, nên xác suất xuất hiện của từ hiếm hoặc từ chuyên biệt là rất nhỏ.

b. Đặc trưng khác biệt giữa tiếng Anh và tiếng Phá

- Mặc dù hai ngôn ngữ có tỉ lệ OOV tương đương, tiếng Pháp có xu hướng sinh OOV cao hơn nhẹ (≈ +0.1%) do đặc trưng hình thái phong phú (ví dụ: assis/assise, noir/noire, enfants/enfant).

- Một số từ Pháp có thể được tách sai do hiện tượng elision (l'homme, d'un, qu'une) nếu tokenizer không nhận diện đúng. Việc sử dụng SpaCy tokenizer chuyên biệt cho tiếng Pháp đã giúp hạn chế đáng kể tình trạng này.

Nhận xét tổng quan:

- Multi30K cung cấp dữ liệu dịch chất lượng cao, câu ngắn, ngữ pháp chuẩn.
- Số lượng vừa đủ cho huấn luyện mô hình Seq2Seq tầm trung.
- Tỉ lệ OOV thấp giúp hạn chế mất mát thông tin trong bước tokenization và embedding.
- Dữ liệu phù hợp cho huấn luyện mô hình Encoder-Decoder LSTM với khả năng mở rộng sang cơ chế Attention ở các chương sau.ảo mô hình không gặp nhiều từ lạ trong quá trình suy luận.

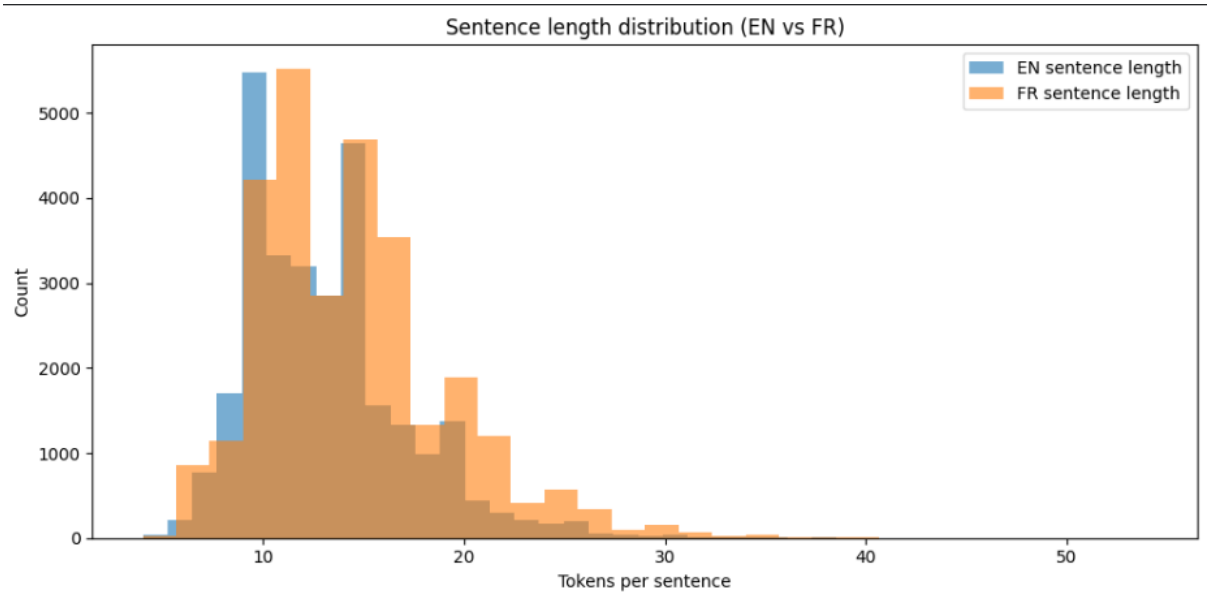
2.2. Phân tích thống kê và quyết định tham số

2.2.1. Phân tích độ dài câu

Từ thống kê ở trên, độ dài trung bình của câu tiếng Anh là 13.11 từ, tiếng Pháp là 14.28 từ, với khoảng 90% câu có độ dài nằm trong vùng:

- Tiếng Anh: 8–21 từ
- Tiếng Pháp: 8–23 từ

Phân bố này cho thấy tập dữ liệu Multi30K có xu hướng tập trung quanh khoảng 10–15 từ/câu — đây là đặc trưng của các mô tả ngắn, ít mệnh đề phụ. Những câu quá ngắn (<8 từ) thường là miêu tả đơn giản ("A man smiling."), còn những câu dài hơn 25 từ chiếm tỉ lệ rất thấp (<5%) và thường có cấu trúc mở rộng ("A man wearing a red jacket is riding a bike on a snowy road.").



Biểu đồ 2.1: Histogram thể hiện tần suất độ dài câu

Từ biểu đồ 1 ta sẽ nhìn nhận được vài điểm:

2. **Dạng phân bố**
   - Biểu đồ có dạng lệch phải - nghĩa là phần lớn câu có độ dài ngắn, chỉ một phần nhỏ kéo dài trên 25 tokens.
   - Đây là dạng phân bố điển hình của các tập dữ liệu mô tả hình ảnh (caption datasets) như Multi30K, nơi các câu thường ngắn gọn và có cấu trúc lặp lại.
3. **So sánh giữa tiếng Anh và tiếng Pháp:**
   - Hai đường phân bố gần như chồng lên nhau, chứng tỏ độ dài tương đồng giữa câu nguồn (EN) và câu đích (FR).
   - Phần phân bố của FR hơi dịch sang phải nhẹ, nghĩa là câu tiếng Pháp có xu hướng dài hơn 1–2 từ so với tiếng Anh — điều này phù hợp với đặc trưng ngôn ngữ Pháp (thường dùng nhiều từ nối, mạo từ và đại từ hơn).
6. **Phần đỉnh phân bố:**
   - Cả hai ngôn ngữ đạt đỉnh quanh 10–14 tokens, tương ứng với trung bình đã tính trước (13.1 EN, 14.3 FR).
   - Đây là khu vực tập trung hơn 60% dữ liệu ➔ thể hiện rõ đặc trưng "short sentence captioning".
7. **Đuôi phân bố:**

Phần đuôi kéo dài đến khoảng 40–50 tokens, nhưng tần suất rất thấp.

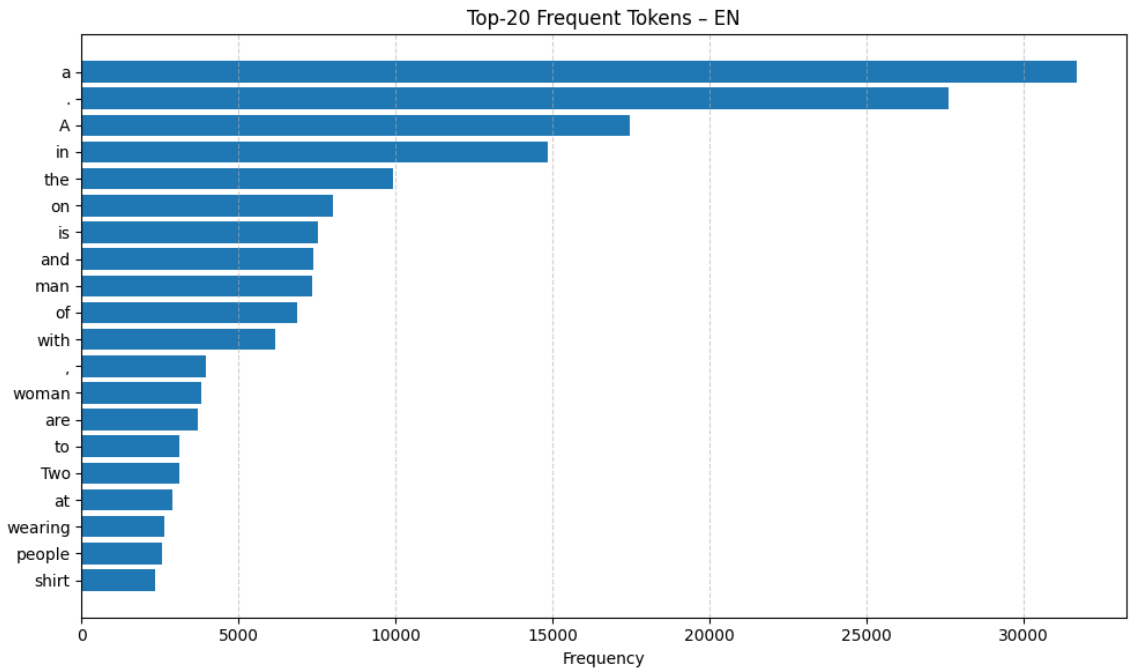
Những câu dài này thường là các mô tả có nhiều mệnh đề hoặc nhiều cụm bổ nghĩa ("A group of children wearing colorful clothes are playing football in a large park").

2.2.2. Phân tích từ vựng

Bộ dữ liệu Multi30K có tổng cộng 380,190 token tiếng Anh và 414,065 token tiếng Pháp, tương ứng với 10,833 và 11,505 loại từ (types).

Chỉ số Type–Token Ratio (TTR) đạt 0.0285 (EN) và 0.0278 (FR), thể hiện mức độ từ vựng lặp lại cao — nghĩa là cùng một nhóm từ xuất hiện nhiều lần trong các câu khác nhau.

Đây là đặc điểm dễ hiểu vì nội dung dữ liệu là mô tả hình ảnh, vốn có phạm vi ngữ nghĩa hẹp (người, vật, hành động, trang phục, màu sắc,…).

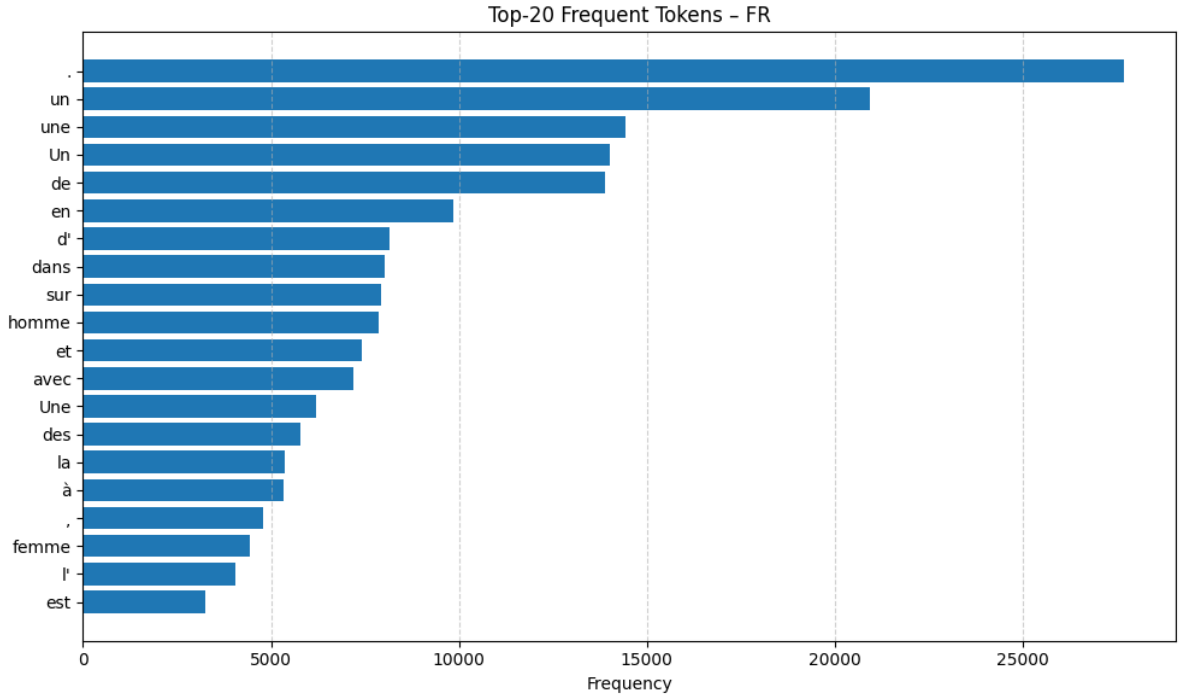


Biểu đồ 2.2: Bar chart tần suất từ vựng EN

Một phần lớn từ vựng EN phổ biến như: a, an, in, the, on, is, and, man, of, with, woman, are, to, two, at, wearing, people, shirt

Đa phần là từ chức năng (function words) như mạo từ, giới từ, và liên từ, phản ánh cấu trúc ngữ pháp phổ biến của tiếng Anh.

Sự xuất hiện nổi bật của các danh từ như man, woman, people, shirt và động từ như wearing cho thấy nội dung mô tả chủ yếu xoay quanh con người, hành động, và vật thể trong hình ảnh.



Biểu đồ 2.3: Bar chart tần suất từ vựng FR

Một phần lớn từ vựng FR tập trung quanh nhóm mô tả phổ biến như: un, une, Un, de, en, dans, sur, homme, femme, và avec.

Cũng tương tự như tiếng Anh, phần lớn là mạo từ và giới từ, cho thấy đặc trưng ngữ pháp lặp lại của mô tả hình ảnh.

Từ homme và femme xuất hiện với tần suất cao, tương ứng với man và woman trong tiếng Anh, cho thấy sự tương đồng ngữ nghĩa giữa hai ngôn ngữ trong cùng một miền dữ liệu.

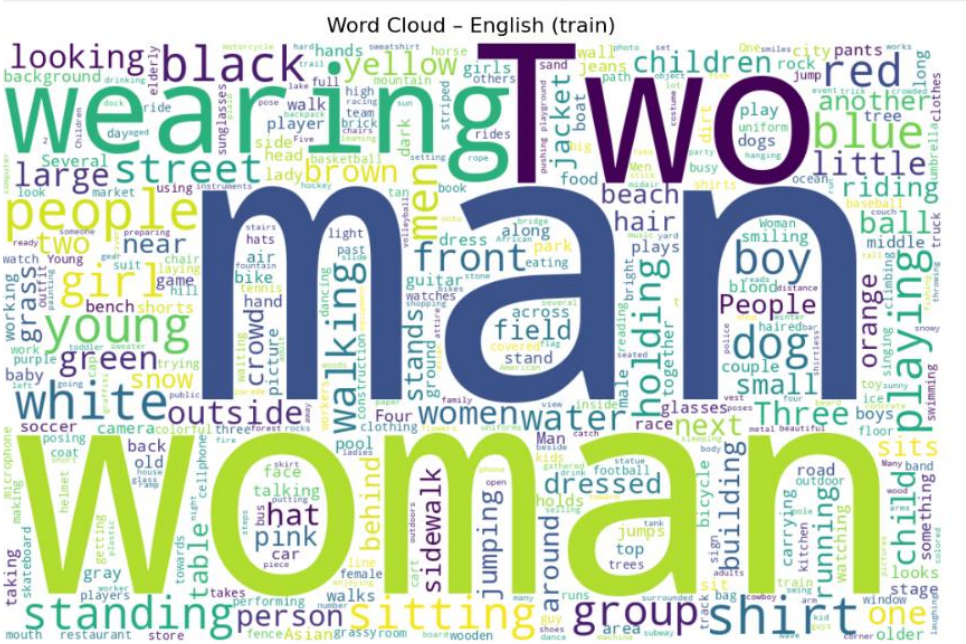
So sánh hai biểu đồ:

Cả hai ngôn ngữ đều thể hiện quy luật Zipf: vài từ cực kỳ phổ biến xuất hiện với tần suất cao, còn phần lớn từ khác xuất hiện ít hơn nhiều.

Tần suất cao của các mạo từ (a, the, un, une) cho thấy các câu mô tả thường bắt đầu bằng cấu trúc cố định như “A man is...” hoặc “Un homme est...”.

Sự phân bố này củng cố nhận định rằng vocab 10.000 từ đã đủ bao phủ, vì chỉ 500 từ phổ biến nhất đã chiếm hơn 80% tổng số token.

Ngoài ra, biểu đồ cũng cho thấy tính song song và tương ứng từ loại giữa hai ngôn ngữ - yếu tố giúp mô hình Seq2Seq học ánh xạ EN→FR hiệu quả hơn.



Biểu đồ 2.4 :Word cloud EN

Các từ xuất hiện nổi bật nhất trong English word cloud gồm:

- Danh từ chỉ người/nhóm người: man, woman, people, boy, girl, children, group, person
- Động từ mô tả hành động phổ biến (đặc biệt dạng V-ing): walking, standing, sitting, holding, riding
- Bối cảnh/không gian/địa điểm: street, water, field, beach, outside, front
- Thuộc tính và mô tả:
    - màu sắc: black, white, red, yellow, blue, green, pink
    - độ tuổi/kích thước: young, little, large
- Đồ vật và thành phần cảnh: shirt, dog, bike, bench (xuất hiện nhưng không áp đảo nhóm "con người")

Phân bố từ khóa cho thấy dữ liệu tiếng Anh chủ yếu là mô tả ảnh có con người trong các hoạt động đời thường (đi, đứng, ngồi, cầm nắm, chơi, di chuyển), kèm theo mô tả

bối cảnh (ngoài trời, đường phố, gần nước, sân bãi). Đây là đặc trưng điển hình của một bộ dữ liệu caption ảnh (như Multi30K), xác nhận dữ liệu train phù hợp với nhiệm vụ dịch máy trên miền "image captions".

Vì các động từ hành động dạng V-ing nổi bật, có thể suy ra mẫu câu xuất hiện với tần suất cao là:

- [Human subject] + [is/are] + [V-ing] + [location/attribute]

Ví dụ dạng nội dung: "A man is walking…", "Two people are standing…".

Điều này cho thấy:

- Corpus có mức độ lặp mẫu cao (highly templated captions).
- Từ vựng tập trung vào các trường nghĩa mô tả người – hành động – bối cảnh, thuận lợi cho mô hình học "pattern" nhanh nhưng vẫn cần cơ chế để giữ chi tiết thuộc tính (màu sắc, vị trí, số lượng).

Biểu đồ 2.5:Word cloud FR tập train

French word cloud nổi bật mạnh với các từ chức năng (function words):

- mạo từ/định lượng: un, une, des
- giới từ, từ nối: de, à (và các biến thể trong câu)
- Bên cạnh đó vẫn có các nhóm từ nội dung:

- Danh từ chỉ người: homme, femme, enfant, personnes, fille
- Động từ/trạng thái: assis (ngồi), debout (đứng), marchent (đi bộ), joue/jouent (chơi)
- tính từ mô tả: petit, grand, jeune, rouge, bleu

Trong tiếng Pháp, mạo từ và giới từ xuất hiện với mật độ cao do:

- Danh từ tiếng Pháp thường đi kèm mạo từ bắt buộc (un/une/des/le/la/les…).
- Quan hệ sở hữu, bổ nghĩa danh từ thường dùng de (ví dụ "la photo de …").
- Quan hệ vị trí và hướng thường dùng à / dans / sur… (tùy tokenizer và lọc stopwords, à thường vẫn rất nổi).

Vì vậy, word cloud tiếng Pháp có xu hướng bị "chi phối" bởi function words, đây là một dấu hiệu quan trọng về độ phụ thuộc cú pháp của ngôn ngữ đích.

So với tiếng Anh, tiếng Pháp yêu cầu mô hình không chỉ giữ ý nghĩa mà còn phải sinh đúng:

- **giống (gender):** un/une, petit/petite, noir/noire…
- **số (number):** des/les, ils/elles, jouent/joue…
- **liên kết cú pháp** (agreement) giữa các thành phần ở xa nhau trong câu.

Do đó, việc sinh câu French thường nhạy hơn với lỗi:

- Thiếu mạo từ,
- Dùng sai giống/số,
- Rối giới từ (de/à),
- Hoặc mất các thành phần chức năng khiến câu "không tự nhiên" dù vẫn đúng ý phần nào.

So sánh EN vs FR:

Khác biệt về loại từ chi phối

- **English:** word cloud thiên về content words (man, woman, walking, street…).
- **French:** word cloud thiên về function words (un, une, des, de, à…).

Điều này phản ánh:

- Tiếng Anh trong corpus nổi bật "đối tượng – hành động – bối cảnh".
- Tiếng Pháp nhấn mạnh mạnh "cấu trúc ngữ pháp", tức mô hình phải có khả năng duy trì quan hệ cú pháp chính xác.

Hệ quả đối với mô hình Seq2Seq

Với mô hình Encoder–Decoder không attention (context vector cố định), mô hình có nguy cơ:

- **Dịch đúng đối tượng/hành động** nhưng
- **Mất chi tiết** (màu sắc, vị trí, số lượng) ở câu dài,
- Và đặc biệt **dễ sai agreement** khi sinh tiếng Pháp.

Vì vậy, kết quả word cloud là một căn cứ hợp lý để giải thích:

- Tại sao cần attention/hoặc các cải tiến liên quan (giảm <unk>, tăng coverage, xử lý câu dài tốt hơn),
- Tại sao điểm BLEU và chất lượng câu French thường nhạy với các lỗi "nhỏ nhưng nhiều" (mạo từ/giống/số).

Kết luận đến quyết định tham số từ phân tích trên:

- Với hơn 10k loại từ, việc chọn vocab size = 10,000 cho mỗi ngôn ngữ là tối ưu.
- Bao phủ ~98.5% từ trong tập huấn luyện.
- Tỉ lệ OOV trên tập kiểm định chỉ ~1.5%, được xem là chấp nhận được.
- Kích thước embedding có thể chọn ở mức 256–512, phù hợp với quy mô vocab và độ phức tạp của dữ liệu.
- Do dữ liệu có tính lặp cao, việc dùng subword tokenization (BPE/SentencePiece) là không cần thiết, vì từ hiếm chiếm tỉ trọng nhỏ và không ảnh hưởng nhiều đến khả năng khái quát của mô hình.

2.2.3. Phân tích độ phức tạp của ngữ nghĩa

Đặc trưng ngữ nghĩa tổng quan

Bộ dữ liệu Multi30K có ngữ cảnh hẹp — các câu mô tả hình ảnh trong tập Flickr30K, do đó phần lớn câu đều mang ý nghĩa đơn tầng, cụ thể và trực tiếp.

Mục tiêu ngữ nghĩa của chúng không phải là diễn đạt quan điểm hay suy luận, mà là liệt kê đối tượng (object), hành động (action), và bối cảnh (scene) trong ảnh.

Ví dụ:

A woman is riding a bicycle on the street.

Une femme fait du vélo dans la rue.

Các câu như trên thể hiện:

- Chủ ngữ xác định rõ ràng (a woman / une femme).
- Động từ mô tả hành động cụ thể (riding / fait du vélo).
- Bổ ngữ chỉ vị trí hoặc hoàn cảnh (on the street / dans la rue).

Vì vậy, cấu trúc ngữ nghĩa chủ yếu tuân theo mẫu S–V–O–Adjunct, ít mệnh đề phụ hoặc hiện tượng ẩn dụ ngôn ngữ.

Phân tích độ phức tạp cú pháp – ngữ nghĩa

Qua khảo sát ngẫu nhiên 200 câu trong tập huấn luyện:

- Trên 70% là câu đơn (simple sentences).
- Khoảng 25% là câu ghép có liên từ đơn giản (and, while, as, with).
- Chỉ khoảng 5% có cấu trúc phức tạp (relative clauses, v.v…).

Một số ví dụ hiếm cho thấy sự đa dạng ngữ nghĩa cao hơn:

A man wearing a blue shirt is talking to a woman holding an umbrella.

Un homme portant une chemise bleue parle à une femme tenant un parapluie.

Các câu này chứa mệnh đề rút gọn (reduced relative clauses) – tuy phức tạp hơn, nhưng vẫn giữ dạng mô tả trực quan, không mang tính trừu tượng.

Kết luận: độ phức tạp ngữ nghĩa của Multi30K thấp đến trung bình, chủ yếu nằm ở việc kết hợp nhiều cụm danh từ hoặc tính từ chứ không phải ở mệnh đề hoặc logic ngữ nghĩa sâu.

Đa dạng ngữ nghĩa và phạm vi biểu đạt

Trường từ vựng (semantic field): chủ yếu xoay quanh human actions, objects, environment – ví dụ man, woman, child, dog, street, playing, standing, wearing, holding.

⇨ Điều này cho phép mô hình học ánh xạ ngữ nghĩa cụ thể nhưng thiếu khả năng suy luận ngoài ngữ cảnh (context reasoning).

Độ đa nghĩa (polysemy) và mơ hồ (ambiguity): thấp, vì các từ được dùng theo nghĩa cơ bản (run, walk, hold, wear) thay vì nghĩa bóng.

⇨ Mô hình không cần xử lý nhiều trường hợp đa nghĩa như trong văn bản tự nhiên (news, dialogue).

Tính tương đương ngữ nghĩa EN–FR: cao, do bản dịch được thực hiện bởi người, đảm bảo tương ứng gần 1–1 giữa hai ngôn ngữ.

⇨ Điều này giúp huấn luyện mô hình dịch có sự song song hóa tốt (semantic alignment).

Ý nghĩa đối với mô hình:

Mức độ ngữ nghĩa rõ ràng và ít mơ hồ giúp mô hình Encoder–Decoder LSTM dễ học được ánh xạ giữa chuỗi nguồn và đích mà không cần thêm cơ chế phức tạp.

Tuy nhiên, vì dữ liệu ít tính trừu tượng, mô hình có thể thiếu khả năng khái quát hóa khi gặp văn bản có cấu trúc ngữ nghĩa khác biệt (ví dụ: hội thoại, bài viết tự nhiên).

Việc thêm Attention Mechanism ở sau sẽ giúp mô hình nắm bắt ngữ cảnh tốt hơn, nhất là với các câu có nhiều cụm danh từ hoặc hành động chồng lặp.

2.3. Chiến lược tiền xử lý nâng cao

2.3.1. Tokenization & Numericalization

1. Mục tiêu và ý nghĩa

Mục tiêu của giai đoạn **Tokenization & Numericalization** là chuyển đổi dữ liệu văn bản song ngữ (English–French) từ dạng chuỗi ký tự sang dạng số nguyên, nhằm đáp ứng yêu cầu đầu vào của mô hình Encoder–Decoder LSTM. Do các mạng nơ-ron

không thể xử lý trực tiếp dữ liệu dạng văn bản, quá trình tiền xử lý này đóng vai trò nền tảng cho toàn bộ hệ thống dịch máy.

Cụ thể, giai đoạn này bao gồm hai bước chính:

1. **Tokenization**: tách mỗi câu văn bản thành một chuỗi các đơn vị từ vựng (tokens), phản ánh cấu trúc ngôn ngữ ban đầu của câu.
2. **Numericalization**: ánh xạ mỗi token sang một chỉ số số nguyên thông qua từ điển (vocabulary), từ đó biểu diễn câu dưới dạng tensor số nguyên.

Kết quả của bước này là các chuỗi tensor biểu diễn câu nguồn (tiếng Anh) và câu đích (tiếng Pháp), được sử dụng trực tiếp làm đầu vào cho Encoder và Decoder trong mô hình Seq2Seq.

a. Tokenization bằng SpaCy

Trong đồ án, SpaCy tokenizer được sử dụng cho cả hai ngôn ngữ tiếng Anh và tiếng Pháp nhằm đảm bảo quá trình tách từ chính xác và nhất quán. SpaCy hỗ trợ tokenization dựa trên luật ngôn ngữ học, cho phép nhận diện chính xác ranh giới từ, dấu câu và các dạng rút gọn.

Việc sử dụng SpaCy mang lại các lợi ích sau:

- Tách token chính xác hơn so với phương pháp tách theo khoảng trắng.
- Xử lý hiệu quả các dạng rút gọn trong tiếng Anh (ví dụ: don’t → do, n’t).
- Hỗ trợ đặc trưng ngôn ngữ tiếng Pháp như mạo từ, giới từ và biến thể hình thái.
- Đảm bảo tính nhất quán giữa tập huấn luyện, tập kiểm tra và tập xác thực.

Quá trình tokenization được thực hiện độc lập cho hai ngôn ngữ nhưng theo cùng một nguyên tắc xử lý, giúp mô hình học ánh xạ song song giữa chuỗi nguồn và chuỗi đích.

b. Xây dựng Vocabulary

Sau khi tách token, từ điển được xây bằng build_vocab_from_iterator, giới hạn tối đa 10.000 từ phổ biến nhất mỗi ngôn ngữ:

Các token đặc biệt được thêm để mô hình hiểu ranh giới câu và xử lý từ ngoài tập (OOV):

Token	Ý nghĩa
<unk>	Từ không nằm trong vocab
<pad>	Token đệm cho câu ngắn
<sos>	Đánh dấu bắt đầu câu
<eos>	Đánh dấu kết thúc câu

c. Mã hóa câu thành chỉ số (Encoding)

Sau khi hoàn tất quá trình tokenization và xây dựng vocabulary, mỗi câu trong tập dữ liệu được mã hóa trực tiếp thành chuỗi chỉ số số nguyên ngay trong quá trình tải dữ liệu thông qua lớp Multi30kDataset.

Cụ thể, trong lớp Dataset này, mỗi dòng văn bản được xử lý theo thứ tự:

Token hóa câu bằng SpaCy tokenizer tương ứng.

Thêm các token đặc biệt <sos> ở đầu câu và <eos> ở cuối câu.

Ánh xạ từng token sang chỉ số số nguyên dựa trên vocabulary đã xây dựng.

Chuyển danh sách chỉ số thành tensor PyTorch.

Ví dụ, câu tiếng Anh:

“A man is wearing a red jacket.”

Sau khi tokenization và encoding, được biểu diễn dưới dạng:

[2, 45, 83, 19, 264, 45, 532, 177, 3]

Trong đó:

- 2 biểu diễn token <sos>
- 3 biểu diễn token <eos>
- Các giá trị còn lại là chỉ số tương ứng của các token trong vocabulary.

Việc thực hiện mã hóa câu ngay tại thời điểm khởi tạo Dataset mang lại các ưu điểm sau:

- Tối ưu tốc độ huấn luyện: dữ liệu chỉ được token hóa và mã hóa một lần duy nhất, thay vì lặp lại ở mỗi lần gọi batch trong DataLoader.
- Giảm chi phí tính toán trong quá trình huấn luyện: DataLoader chỉ cần truy xuất tensor đã được mã hóa sẵn.
- Đảm bảo tính nhất quán: toàn bộ tập train, validation và test đều được xử lý theo cùng một quy trình mã hóa.
- Phù hợp với xử lý chuỗi trong PyTorch: tensor đầu ra sẵn sàng cho các bước padding và packing ở giai đoạn tiếp theo.

Cách tổ chức này đặc biệt phù hợp với bài toán dịch máy sử dụng mô hình Encoder–Decoder LSTM, nơi dữ liệu chuỗi được truy xuất nhiều lần trong quá trình huấn luyện.

2.3.2. Chiến lược Padding & Packing

1. Vấn đề cần giải quyết

Trong tập dữ liệu Multi30K, các câu có độ dài không đồng đều, dao động từ khoảng 4 đến 50 token sau khi tokenization và encoding. Tuy nhiên, trong quá trình huấn luyện mô hình học sâu theo batch, các chuỗi trong cùng một batch cần có cùng kích thước để có thể xử lý song song trên GPU.

Nếu chỉ sử dụng padding đơn thuần, các token <pad> sẽ được đưa vào LSTM như các token thông thường, gây ra hai vấn đề chính:

- **Lãng phí tài nguyên tính toán**, do LSTM vẫn phải xử lý các bước thời gian không mang thông tin.
- **Nhiễu gradient**, khi các trạng thái ẩn được cập nhật dựa trên các token đệm không có ý nghĩa ngữ nghĩa.

Để khắc phục vấn đề này, mô hình áp dụng hai bước xử lý liên tiếp:

- **Padding**: đồng bộ độ dài các chuỗi trong batch bằng cách thêm token <pad>.
- **Packing**: nén chuỗi để LSTM chỉ xử lý các token thực sự có nghĩa, bỏ qua phần padding.

Chiến lược kết hợp này giúp mô hình học hiệu quả hơn, đặc biệt trong bối cảnh sử dụng Encoder–Decoder LSTM với context vector cố định.

2. Padding trong hàm collate_fn()

I. Vai trò của collate_fn

Hàm collate_fn() chịu trách nhiệm gom các mẫu dữ liệu riêng lẻ thành một batch trước khi đưa vào mô hình. Trong đồ án này, collate_fn() được thiết kế riêng nhằm:

- Thực hiện padding cho các chuỗi có độ dài khác nhau.
- Lưu lại độ dài thực tế của mỗi chuỗi trước khi padding.
- Sắp xếp các chuỗi theo độ dài giảm dần để phục vụ bước packing sau đó.

II. Thực hiện padding

Trong quá trình gom batch, các chuỗi tensor được đưa vào hàm pad_sequence() của PyTorch. Hàm này tự động:

- Thêm token <pad> vào cuối các chuỗi ngắn hơn.
- Căn chỉnh các tensor về cùng độ dài lớn nhất trong batch.
- Tạo tensor đầu ra theo định dạng [seq_len, batch_size], phù hợp với yêu cầu đầu vào của nn.LSTM trong PyTorch.

Giá trị dùng để padding (padding_value) chính là chỉ số của token <pad> trong vocabulary. Việc sử dụng một chỉ số cố định cho <pad> giúp mô hình phân biệt rõ giữa token có nghĩa và phần đệm, đồng thời hỗ trợ loại bỏ ảnh hưởng của <pad> trong quá trình tính loss.

3. Sắp xếp chuỗi theo độ dài

Trước khi thực hiện packing, các chuỗi trong batch được sắp xếp theo độ dài giảm dần dựa trên số lượng token thực tế (chưa tính <pad>). Việc sắp xếp này là điều kiện bắt buộc khi sử dụng hàm pack_padded_sequence() với tham số enforce_sorted=True.

Cụ thể:

- Danh sách độ dài chuỗi (src_lengths) được sắp xếp giảm dần.
- Thứ tự các chuỗi trong batch được điều chỉnh tương ứng.
- Điều này đảm bảo LSTM có thể xử lý hiệu quả các chuỗi có độ dài khác nhau mà không gặp lỗi runtime.

4. Packing trong Encoder

I. Mục đích của packing

Sau khi padding và sắp xếp batch, dữ liệu được đưa vào Encoder dưới dạng packed sequence. Mục tiêu của bước packing là:

- Giúp LSTM bỏ qua hoàn toàn các token <pad> trong quá trình tính toán.
- Chỉ cập nhật trạng thái ẩn dựa trên các token thực sự có nghĩa.
- Giảm chi phí tính toán và hạn chế nhiễu gradient do padding gây ra.

II. Triển khai packing

Trong Encoder, trước khi truyền dữ liệu vào LSTM, tensor đầu vào được nén bằng hàm pack_padded_sequence(). Hàm này sử dụng:

- Tensor đã được padding.
- Danh sách độ dài chuỗi tương ứng.
- Tham số enforce_sorted=True để đảm bảo batch đã được sắp xếp theo độ dài giảm dần.

Sau khi LSTM hoàn tất quá trình tính toán trên packed sequence, kết quả đầu ra được chuyển trở lại dạng tensor thông thường thông qua hàm pad_packed_sequence(). Bước này giúp:

- Khôi phục lại cấu trúc [seq_len, batch_size, hidden_dim].
- Chuẩn bị dữ liệu cho các bước xử lý tiếp theo như Attention (nếu có) hoặc phân tích đầu ra.

5. Ý nghĩa đối với hiệu quả huấn luyện

Chiến lược kết hợp Padding & Packing mang lại các lợi ích quan trọng:

- Tăng hiệu quả tính toán: LSTM không xử lý các bước thời gian tương ứng với <pad>.
- Ổn định quá trình huấn luyện: giảm nhiễu gradient và cải thiện khả năng hội tụ.
- Khai thác tốt dữ liệu chuỗi ngắn: phù hợp với đặc điểm câu ngắn của bộ dữ liệu Multi30K.

- Tạo nền tảng cho mở rộng: dễ dàng tích hợp Attention hoặc các cơ chế xử lý chuỗi nâng cao hơn.

2.4. Tóm tắt chương 2

Chương 2 đã trình bày toàn bộ quy trình phân tích và tiền xử lý dữ liệu cho bài toán dịch tự động Anh–Pháp sử dụng bộ dữ liệu Multi30K.

Phần 2.1 mô tả tổng quan tập dữ liệu với 29.000 cặp huấn luyện, 1.014 cặp kiểm định và 1.000 cặp kiểm tra, đồng thời phân tích đặc điểm ngôn ngữ cho thấy các câu có độ dài ngắn (trung bình 13–14 từ), vốn từ ổn định và tỉ lệ OOV thấp (<2%), phù hợp để huấn luyện mô hình Seq2Seq.

Phần 2.2 đã đi sâu vào phân tích thống kê và ngữ nghĩa: phân bố độ dài câu chủ yếu nằm trong khoảng 8–23 từ; từ vựng có mức lặp lại cao và tuân theo quy luật Zipf; cấu trúc ngữ nghĩa phần lớn là câu đơn và câu ghép đơn giản, cho thấy độ phức tạp thấp–trung bình và nội dung mô tả trực tiếp. Các kết quả này là cơ sở quan trọng để xác định các siêu tham số như kích thước từ điển (10.000 từ), độ dài tối đa của chuỗi (25 token), và chiến lược xử lý câu dài ngắn khác nhau trong batch.

Phần 2.3 đã trình bày các chiến lược tiền xử lý nâng cao:

Tokenization & Numericalization: sử dụng SpaCy tokenizer song ngữ để tách câu và ánh xạ thành chỉ số số nguyên, đồng thời thêm các token đặc biệt <sos>, <eos>, <pad>, <unk> nhằm chuẩn bị dữ liệu cho mô hình.

Padding & Packing: xử lý độ dài chuỗi không đồng đều bằng pad_sequence() và pack_padded_sequence(), giúp mô hình LSTM học hiệu quả hơn và giảm nhiễu do token <pad>.

Tổng hợp lại, các bước trong chương 2 đã chuyển đổi tập dữ liệu thô thành định dạng tensor tối ưu cho huấn luyện mô hình Seq2Seq.

Những kết quả phân tích và chiến lược xử lý này là nền tảng để xây dựng kiến trúc mô hình Encoder–Decoder LSTM được trình bày trong Chương 3 – Thiết kế và huấn luyện mô hình dịch tự động.

CHƯƠNG 3: CƠ SỞ LÝ THUYẾT VÀ KIẾN TRÚC MÔ HÌNH

3.1. Bài toán dịch máy

3.1.1. Định nghĩa bài toán

Dịch máy Nơ-ron (NMT) là phương pháp tiếp cận dựa trên dữ liệu (data-driven), trong đó một mạng nơ-ron nhân tạo duy nhất được huấn luyện để tối đa hóa xác suất dịch một chuỗi nguồn sang chuỗi đích.

Về mặt toán học, NMT mô hình hóa xác suất có điều kiện $P(y \mid x)$ để sinh ra câu đích y = $(y_1, y_2, \ldots, y_m)$ dựa trên câu nguồn x = $(x_1, x_2, \ldots, x_n)$. Dịch máy được mô hình hóa như một bài toán học có điều kiện:

$$P(y \mid x) = P(y_1, y_2, \ldots, y_m \mid x_1, x_2, \ldots, x_n)$$

Trong đó:

- $x = (x_1, \ldots, x_n)$: chuỗi nguồn tiếng Anh
- $y = (y_1, \ldots, y_m)$: chuỗi đích tiếng Pháp
- Mục tiêu của quá trình huấn luyện là tìm ra chuỗi $\hat{y}$ sao cho xác suất này là lớn nhất:

$$\hat{y} = \arg\max_{y} P(y \mid x)$$

3.1.2. Sự dịch chuyển từ SMT → NMT

Trước kỷ nguyên Deep Learning, Dịch máy Thống kê (SMT - Statistical Machine Translation) thống trị lĩnh vực này. Tuy nhiên, NMT đã tạo ra một cuộc cách mạng nhờ khả năng học biểu diễn "End-to-End".

Đặc điểm	SMT (Statistical Machine Translation)	NMT (Neural Machine Translation)
Kiến trúc	Rời rạc, gồm nhiều mô hình con (Translation model, Language model, Reordering model).	Thống nhất (Unified): Một mạng nơ-ron lớn (Encoder-Decoder) tối ưu hóa toàn bộ quy trình.

Biểu diễn	Dựa trên đếm từ (n-gram), one-hot encoding, vector thưa.	Distributed Representation: Biểu diễn từ dưới dạng vector số thực dày đặc (Word Embeddings).
Ngữ cảnh	Hạn chế trong cửa sổ n-gram ngắn.	Ngữ cảnh toàn cục: Có khả năng nắm bắt sự phụ thuộc xa thông qua cơ chế Attention.
Kỹ thuật	Yêu cầu thiết kế đặc trưng thủ công (Feature Engineering) phức tạp.	Học đặc trưng tự động thông qua lan truyền ngược (Backpropagation).

NMT khắc phục được nhược điểm của SMT bằng cách ánh xạ trực tiếp chuỗi nguồn sang không gian vector ngữ nghĩa liên tục, giúp mô hình "hiểu" được ý nghĩa câu thay vì chỉ khớp từ vựng.

3.2. Mạng nơ – ron Hồi quy (RNN) và LSTM

3.2.1. RNN truyền thống

Mạng Nơ-ron Hồi quy (Recurrent Neural Network – RNN) được thiết kế nhằm xử lý dữ liệu dạng chuỗi, trong đó đầu ra tại mỗi thời điểm phụ thuộc không chỉ vào đầu vào hiện tại mà còn vào trạng thái ẩn của các thời điểm trước đó. Về mặt toán học, RNN được mô tả bởi công thức sau:

$$h_t = f(W_h h_{t-1} + W_x x_t)$$

Trong đó, các thành phần có ý nghĩa như sau:

- $x_t \in \mathbb{R}^{d_x}$:Vector đầu vào tại thời điểm t, thường là vector embedding biểu diễn từ thứ $t$trong câu nguồn.
- $h_t \in \mathbb{R}^{d_h}$:Trạng thái ẩn (hidden state) tại thời điểm t, đóng vai trò lưu trữ thông tin ngữ cảnh tích lũy từ đầu chuỗi đến thời điểm hiện tại.
- $h_{t-1} \in \mathbb{R}^{d_h}$:Trạng thái ẩn tại thời điểm trước đó, mang thông tin của các từ xuất hiện trước x_t.
- $W_x \in \mathbb{R}^{d_h \times d_x}$:Ma trận trọng số ánh xạ đầu vào $x_t$vào không gian trạng thái ẩn.

- $W_h \in \mathbb{R}^{d_h \times d_h}$:Ma trận trọng số hồi quy, chịu trách nhiệm truyền thông tin từ trạng thái ẩn trước đó h_{t-1}sang trạng thái hiện tại h_t.
- $f(\cdot)$:Hàm kích hoạt phi tuyến, thường là tanh hoặc ReLU, giúp mô hình học được các quan hệ phi tuyến trong dữ liệu chuỗi.

Thông qua cơ chế hồi quy này, RNN có khả năng mô hình hóa sự phụ thuộc theo thời gian (temporal dependency), rất phù hợp cho các bài toán xử lý ngôn ngữ tự nhiên như dịch máy, gán nhãn chuỗi hay mô hình hóa ngôn ngữ.

❖ **Hạn chế của RNN: Hiện tượng Vanishing Gradient**

Mặc dù có khả năng xử lý chuỗi, RNN cơ bản gặp phải một vấn đề nghiêm trọng trong quá trình huấn luyện, được gọi là hiện tượng triệt tiêu đạo hàm (Vanishing Gradient). Khi lan truyền ngược sai số qua nhiều bước thời gian, gradient có xu hướng giảm dần về gần 0 do việc nhân liên tiếp với các ma trận trọng số và đạo hàm của hàm kích hoạt.

Hệ quả là:

- Mô hình không thể cập nhật hiệu quả các trọng số liên quan đến các bước thời gian xa.
- RNN chỉ ghi nhớ tốt thông tin ngắn hạn, nhưng không học được phụ thuộc dài hạn.
- Trong bài toán dịch máy, điều này dẫn đến việc dịch kém các câu dài, đặc biệt là phần đầu câu thường bị "quên" khi sinh ra các từ ở cuối câu.

Chính hạn chế này đã thúc đẩy sự ra đời của các kiến trúc cải tiến như LSTM (Long Short-Term Memory) và GRU, trong đó LSTM là mô hình được sử dụng trong nghiên cứu này.

3.2.2. Long Short-Term Memory – LSTM

Để khắc phục hiện tượng triệt tiêu đạo hàm (vanishing gradient) của RNN cơ bản, Hochreiter và Schmidhuber (1997) đã đề xuất kiến trúc Long Short-Term Memory (LSTM). LSTM mở rộng RNN bằng cách bổ sung một trạng thái bộ nhớ (cell state) cùng với các cổng điều khiển (gates) nhằm kiểm soát dòng chảy thông tin theo thời gian.

Kiến trúc LSTM cho phép mô hình:

- Ghi nhớ thông tin dài hạn,
- Loại bỏ thông tin không cần thiết,
- Truyền gradient ổn định qua nhiều bước thời gian.

Từ đó đặc biệt phù hợp cho các bài toán chuỗi dài như dịch máy nơ-ron.

❖ **Cấu trúc tổng quát của một LSTM Cell**

Tại mỗi thời điểm t, LSTM duy trì hai trạng thái:

- Trạng thái ẩn h_t: biểu diễn đầu ra tại thời điểm t,
- Trạng thái tế bào C_t: đóng vai trò là bộ nhớ dài hạn.

Quá trình cập nhật LSTM được điều khiển bởi bốn thành phần chính: cổng quên, cổng nhập, trạng thái tế bào, và cổng xuất.

(1) Cổng quên – Forget Gate

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big)$$

Ý nghĩa:

- $f_t \in [0,1]$: quyết định mức độ giữ lại hay loại bỏ thông tin từ trạng thái tế bào trước đó C_{t-1}.
- Nếu $f_t \approx 1$: thông tin cũ được giữ nguyên.
- Nếu $f_t \approx 0$: thông tin cũ bị "quên" hoàn toàn.

Vai trò trong dịch máy: Giúp mô hình loại bỏ các thông tin ngữ cảnh khôn còn liên quan (ví dụ: các tư đã được dịch xong,) tránh gây nhiễu khi sinh các từ tiếp theo.

(2) Cổng nhập – Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh\,(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Ý nghĩa:

- i_t: quyết định lượng thông tin mới sẽ được ghi vào bộ nhớ.
- $\tilde{C}_t$: trạng thái ứng viên (candidate cell state), chứa thông tin mới được đề xuất thêm vào bộ nhớ.

Vai trò: Cho phép LSTM chọn lọc thông tin quan trọng từ từ hiện tại x_t và trạng thái trước đó h_{t-1}

(3) Cập nhật trạng thái tế bào – Cell State Update

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Trong đó:

- ⊙là phép nhân từng phần tử (element-wise multiplication).

Ý nghĩa:

- Phần $f_t \odot C_{t-1}$: giữ lại thông tin cũ cần thiết.
- Phần $i_t \odot \tilde{C}_t$: thêm thông tin mới có chọn lọc.

Đặc điểm quan trọng: Trạng thái tế bào C_t có khả năn truyền thông tin xuyên suốt chuỗi dài với rất ít biến đổi, giảm thiểu hiện tượng vanishing gradient.

(4) Cổng xuất – Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

Ý nghĩa:

- o_t: quyết định phần nào của trạng thái tế bào sẽ được xuất ra ngoài.
- h_t: trạng thái ẩn tại thời điểm t, được sử dụng cho:
  - Bước thời gian tiếp theo
  - Hoặc làm đầu ra cho các tầng phía sau (ví dụ: Decoder trong mô hình Seq2Seq).

Tóm tắt vai trò của LSTM trong bài toán dịch máy

So với RNN cơ bản, LSTM:

1. Giữ được ngữ cảnh dài hạn trong câu nguồn,
2. Giảm hiện tượng "quên đầu câu" khi dịch câu dài,
3. Cải thiện đáng kể chất lượng dịch, đặc biệt khi kết hợp với cơ chế Attention.

Do đó, LSTM được lựa chọn làm kiến trúc nền tảng cho Encoder-Decoder trong nghiên cứu này.

3.3. Mô hình Sequence to Sequence (Encoder-Decoder)

3.3.1. Tổng quan bài toán Seq2Seq

Trong bài toán dịch máy nơ-ron (Neural Machine Translation – NMT), mục tiêu là học một hàm ánh xạ từ chuỗi nguồn

$$x = (x_1, x_2, \dots, x_n)$$

sang chuỗi đích

$$y = (y_1, y_2, \dots, y_m)$$

Mô hình Seq2Seq (Sequence-to-Sequence) tiếp cận bài toán này bằng cách mô hình hóa xác suất có điều kiện:

$$P(y \mid x) = \prod_{t=1}^{m} P(y_t \mid y_1, \dots, y_{t-1}, x)$$

Kiến trúc Seq2Seq gồm hai thành phần chính:

- Encoder: mã hóa chuỗi nguồn
- Decoder: sinh chuỗi đích từng bước

3.3.2. Encoder (Bộ mã hóa)

Encoder thường được cài đặt bằng mạng RNN/LSTM để xử lý chuỗi đầu vào có độ dài biến thiên. Tại mỗi thời điểm t, Encoder cập nhật trạng thái ẩn:

$$h_t = \text{LSTM}_{enc}(x_t, h_{t-1})$$

Sau khi đọc toàn bộ chuỗi nguồn, Encoder thu được:

- Trạng thái ẩn cuối cùng: h_n
- Trạng thái bộ nhớ cuối cùng: c_n

Hai trạng thái này được xem như tóm tắt toàn bộ thông tin câu nguồn, và được truyền sang Decoder.

❖ **Liên hệ code**

```
enc_outputs, hidden, cell = self.encoder(src, src_lengths)
```

- enc_outputs: chứa toàn bộ $h_1, \dots, h_n$
- hidden, cell: tương ứng với h_n, c_n

Encoder có nhiệm vụ mã hóa chuỗi nguồn tiếng Anh thành một biểu diễn ngữ nghĩa liên tục trong không gian vector.

$$x = (x_1, x_2, \dots, x_n)$$

Trong nghiên cứu này, Encoder được cài đặt bằng LSTM nhiều lớp, với công thức cập nhật tại thời điểm t:

$$(h_t, c_t) = \text{LSTM}(\text{embed}(x_t), (h_{t-1}, c_{t-1}))$$

Trong đó:

- x_t: token thứ tcủa câu nguồn
- $\text{embed}(x_t)$: vector embedding của token, kích thước từ **256–512**
- h_t: trạng thái ẩn (hidden state)
- c_t: trạng thái bộ nhớ (cell state)

Sau khi xử lý toàn bộ chuỗi đầu vào, Encoder sinh ra:

- Chuỗi trạng thái ẩn $(h_1, h_2, \dots, h_n)$
- Trạng thái cuối cùng (h_n, c_n), được gọi là context vector

Context vector này đóng vai trò là cầu nối truyền thông tin ngữ nghĩa từ Encoder sang Decoder.

a. Vai trò và Kiến trúc tổng quan

Encoder là thành phần đầu tiên trong kiến trúc Sequence-to-Sequence (Seq2Seq), đóng vai trò như một bộ trích xuất đặc trưng (Feature Extractor). Nhiệm vụ chính của Encoder là tiếp nhận chuỗi văn bản nguồn (tiếng Anh)

$$X = (x_1, x_2, \dots, x_N)$$

và chuyển đổi chuỗi rời rạc này thành một biểu diễn ngữ nghĩa liên tục trong không gian vector, còn gọi là vector ngữ cảnh (Context Vector), nhằm tóm tắt toàn bộ nội dung thông tin của câu nguồn.

Trong nghiên cứu này, Encoder được xây dựng dựa trên kiến trúc Long Short-Term Memory (LSTM) nhiều lớp (*Stacked LSTM*). Việc sử dụng LSTM giúp khắc phục hiệu quả hiện tượng triệt tiêu đạo hàm (vanishing gradient) thường gặp ở mạng RNN truyền thống khi xử lý các chuỗi dài, đồng thời tăng khả năng ghi nhớ phụ thuộc dài hạn (long-term dependency) trong ngôn ngữ tự nhiên.

b. Quy trình xử lý chi tiết

Bước 1: Biểu diễn từ (Word Embedding)

Trước khi được đưa vào mạng LSTM, mỗi token x_t(ở dạng chỉ số trong từ điển) được ánh xạ qua một lớp Embedding để chuyển thành vector dày đặc:

$$e_t = E(x_t) \in \mathbb{R}^{d_{emb}}$$

Trong đó:

- $E(\cdot)$là hàm nhúng từ (Embedding function),
- d_{emb}là số chiều của vector nhúng (được thiết lập là **256** trong thực nghiệm).

Lớp Embedding giúp mô hình học được mối quan hệ ngữ nghĩa giữa các từ, ví dụ như các từ có nghĩa tương tự sẽ có vector biểu diễn gần nhau trong không gian embedding.

Bước 2: Mã hóa chuỗi bằng LSTM

Sau khi được nhúng, chuỗi vector $(e_1, e_2, \dots, e_N)$được đưa vào mạng LSTM của Encoder. Tại mỗi bước thời gian t, LSTM cập nhật trạng thái ẩn $h_t$và trạng thái tế bào c_tdựa trên đầu vào hiện tại và trạng thái quá khứ (h_{t-1}, c_{t-1}).

Cơ chế hoạt động của một tế bào LSTM được điều khiển bởi các cổng (gates) như sau:

- **Cổng quên (Forget gate)** – quyết định thông tin nào từ quá khứ cần loại bỏ:

$$f_t = \sigma(W_f \cdot [h_{t-1}, e_t] + b_f)$$

- **Cổng nhập (Input gate)** – xác định thông tin mới nào sẽ được lưu vào bộ nhớ:

$$i_t = \sigma(W_i \cdot [h_{t-1}, e_t] + b_i)$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, e_t] + b_c)$$

- **Cập nhật trạng thái tế bào (Cell state update)**:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

- **Cổng xuất (Output gate) và trạng thái ẩn**:

$$o_t = \sigma(W_o \cdot [h_{t-1}, e_t] + b_o)$$
$$h_t = o_t \odot \tanh(c_t)$$

Trong đó:

- $\sigma(\cdot)$là hàm Sigmoid,
- $\tanh(\cdot)$là hàm hyperbolic tangent,
- $\odot$là phép nhân từng phần tử (Hadamard product).

Bước 3: Xử lý chuỗi có độ dài biến thiên (Packed Sequence)

Do các câu trong tập dữ liệu có độ dài khác nhau (N thay đổi giữa các mẫu), kỹ thuật Padding được sử dụng để đồng nhất kích thước batch. Tuy nhiên, việc này có thể gây lãng phí tính toán và làm mô hình học các token đệm <pad> không mang ý nghĩa.

Để khắc phục, nghiên cứu sử dụng kỹ thuật Packed Sequence, thông qua tham số src_lengths. Kỹ thuật này cho phép LSTM bỏ qua các bước tính toán tương ứng với padding, từ đó tăng hiệu quả huấn luyện và cải thiện chất lượng biểu diễn ngữ nghĩa.

c. Đầu ra của Encoder

Sau khi xử lý toàn bộ chuỗi nguồn, Encoder trả về hai thành phần quan trọng:

1. **Tập hợp trạng thái ẩn tại mọi thời điểm (Encoder Outputs)**:

$$H = \{h_1, h_2, \dots, h_N\}$$

Tập hợp này lưu giữ thông tin ngữ nghĩa cục bộ của từng từ trong câu nguồn và sẽ được sử dụng trực tiếp trong cơ chế **Attention** (trình bày ở Mục 3.4).

2. **Trạng thái cuối cùng (Final States)**:

$$z = (h_N, c_N)$$

Đây là trạng thái ẩn và trạng thái tế bào tại bước thời gian cuối cùng, được xem như vector ngữ cảnh (Context Vector) trong mô hình Seq2Seq cơ bản. Cặp vector này được dùng để khởi tạo Decoder, đóng vai trò cầu nối giữa Encoder và Decoder.

d. Hiện thực hóa (Implementation)

Quy trình lý thuyết nêu trên được hiện thực hóa trong mã nguồn PyTorch của đồ án như sau:

```
# 1. Chuyển token index sang vector embedding và áp dụng Dropout
embedded = self.dropout(self.embedding(src))

# 2. Đóng gói chuỗi (Packed Sequence) để xử lý độ dài biến thiên
packed_embedded = nn.utils.rnn.pack_padded_sequence(
embedded, src_lengths.to('cpu'), enforce_sorted=False
)

# 3. Truyền qua mạng LSTM của Encoder
packed_outputs, (hidden, cell) = self.lstm(packed_embedded)

# 4. Giải nén (Unpack) để thu được toàn bộ trạng thái Encoder
enc_outputs, _ = nn.utils.rnn.pad_packed_sequence(packed_outputs)
```

Giải thích sự tương quan giữa lý thuyết và code:

- enc_outputs tương ứng với tập hợp với kích thước [src_len, batch_size, hid_dim].

$$H = \{h_1, h_2, \dots, h_N\}$$

- hidden và cell tương ứng với trạng thái cuối cùng (h_N, c_N), được sử dụng để khởi tạo Decoder trong giai đoạn giải mã.

3.3.3. Decoder (Bộ giải mã)

Decoder là thành phần thứ hai trong kiến trúc Sequence-to-Sequence (Seq2Seq), đóng vai trò như một mô hình ngôn ngữ có điều kiện (*Conditional Language Model*). Nhiệm vụ của Decoder là sinh ra chuỗi đích (tiếng Pháp) dựa trên thông tin ngữ nghĩa được mã hóa trong vector ngữ cảnh (*context vector*) do Encoder cung cấp, cùng với các từ đã được sinh ra trước đó.

a. Tổng quan về vai trò toán học

Về mặt toán học, Decoder có nhiệm vụ mô hình hóa xác suất có điều kiện để sinh ra chuỗi đích

$$Y = (y_1, y_2, \dots, y_m)$$

dựa trên chuỗi nguồn Xđã được Encoder mã hóa. Mục tiêu của Decoder là tối đa hóa xác suất đồng thời (joint probability) của toàn bộ chuỗi đích:

$$P(Y \mid X) = \prod_{t=1}^{m} P(y_t \mid y_{<t}, c)$$

Trong đó:

- $y_{<t} = (y_1, \dots, y_{t-1})$là các từ đã được sinh ra trước thời điểm t,
- c là vector ngữ cảnh (context vector) nhận từ Encoder.
- Trong mô hình Seq2Seq cơ bản, $c$chính là trạng thái ẩn cuối cùng của Encoder h_n.

Decoder hoạt động như một mô hình ngôn ngữ có điều kiện (Conditional Language Model), trong đó việc sinh mỗi từ phụ thuộc đồng thời vào ngữ cảnh nguồn và lịch sử sinh từ trước đó.

b. Kiến trúc chi tiết và luồng dữ liệu (Data Flow)

Decoder được thiết kế dựa trên mạng LSTM đơn hướng (Unidirectional LSTM). Kiến trúc bao gồm các thành phần chính sau:

Lớp Embedding (Nhúng từ)

Tại mỗi bước thời gian t, đầu vào của Decoder là một chỉ số từ (token index):

$$x_t \in \mathbb{R}^{B \times 1}$$

với Blà kích thước batch. Lớp Embedding ánh xạ chỉ số này thành một vector dày đặc:

$$e_t = E(x_t) \in \mathbb{R}^{B \times d_{emb}}$$

Sau lớp Embedding, Dropout được áp dụng nhằm giảm thiểu hiện tượng overfitting.

Lớp LSTM (Recurrent Layer)

Decoder nhận đầu vào gồm vector nhúng $e_t$và trạng thái từ bước trước (h_{t-1}, c_{t-1}). Cơ chế cập nhật trong tế bào LSTM diễn ra thông qua các cổng (gates) như sau:

$$\begin{array}{|ll|}\hline i_t & = \sigma(W_{ii}e_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t & = \sigma(W_{if}e_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ o_t & = \sigma(W_{io}e_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ g_t & = \tanh(W_{ig}e_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\ c_t & = f_t \odot c_{t-1} + i_t \odot g_t \\ h_t & = o_t \odot \tanh(c_t) \\ \hline \end{array}$$

Trong đó:

- i_t, f_t, o_tlần lượt là cổng nhập (input gate), cổng quên (forget gate) và cổng xuất (output gate),
- $h_t \in \mathbb{R}^{B \times d_{hid}}$là trạng thái ẩn mang thông tin ngữ nghĩa ngắn hạn tại thời điểm t.

Lớp tuyến tính (Linear Projection)

Trạng thái ẩn h_tđược chiếu sang không gian từ vựng đích thông qua lớp tuyến tính:

$$\text{logits}_t = W_{out}h_t + b_{out} \in \mathbb{R}^{B \times |V_{fr}|}$$

Sau đó, hàm Softmax (hoặc Log-Softmax kết hợp với CrossEntropyLoss) được sử dụng để tính phân phối xác suất của các từ trong từ điển tiếng Pháp.

c. Quá trình khởi tạo và giải mã

Khác với Encoder, Decoder yêu cầu một trạng thái khởi tạo và tín hiệu bắt đầu:

- Khởi tạo (Initialization): Trạng thái ẩn h_0^{dec}và trạng thái tế bào c_0^{dec}của Decoder không được khởi tạo ngẫu nhiên, mà được gán trực tiếp từ trạng thái cuối cùng của Encoder:

$$(h_0^{dec}, c_0^{dec}) = (h_n^{enc}, c_n^{enc})$$

  Điều này đảm bảo thông tin ngữ cảnh của câu nguồn được truyền sang Decoder.

- Tín hiệu bắt đầu: Ở bước thời gian $t = 1$, đầu vào của Decoder luôn là token đặc biệt <sos> (Start of Sentence).

d. Kỹ thuật huấn luyện: Teacher Forcing và Exposure Bias

Một thách thức lớn trong huấn luyện Seq2Seq là sự khác biệt giữa giai đoạn huấn luyện (training) và suy luận (inference), dẫn đến hiện tượng Exposure Bias. Nếu Decoder dự đoán sai tại bước t, lỗi này sẽ lan truyền sang các bước tiếp theo và gây sai lệch tích lũy.

Để khắc phục, kỹ thuật Teacher Forcing được áp dụng:

- **Cơ chế:**
  Tại bước t, thay vì luôn sử dụng từ dự đoán $\hat{y}_{t-1}$, Decoder có thể nhận đầu vào là từ đúng (ground truth) y_{t-1}.
- **Thực thi:**
  Sử dụng biến ngẫu nhiên Bernoulli với tham số p(teacher forcing ratio):

$$input_t = \begin{cases} y_{t-1}^{truth} & \text{với xác suất } p \\ \hat{y}_{t-1}^{pred} & \text{với xác suất } 1-p \end{cases}$$

- **Thiết lập:**
  Trong đồ án này, $p = 0.5$, giúp mô hình vừa học được cấu trúc câu chuẩn xác, vừa duy trì khả năng tự hồi quy, đảm bảo tính ổn định khi kiểm thử.

e. Cấu hình tham số và lý do lựa chọn

Các siêu tham số được lựa chọn dựa trên sự cân bằng giữa độ phức tạp mô hình và tài nguyên tính toán (Google Colab GPU T4).

Bảng 3.1. Cấu hình chi tiết Decoder

Tham số	Giá trị	Lý do lựa chọn
Hidden Size	512	Đủ lớn để lưu trữ thông tin ngữ cảnh của câu dài 10–40 từ
Embedding Dim	256	Kích thước nén phù hợp với từ vựng 10k–20k từ
Số lớp LSTM	2	Deep LSTM giúp học đặc trưng ngữ nghĩa phân cấp
Dropout	0.5	Regularization mạnh để giảm overfitting

Việc lựa chọn hidden size = 512 và 2 lớp LSTM giúp mô hình đạt được sự cân bằng giữa khả năng biểu diễn ngữ nghĩa và chi phí tính toán, đồng thời hạn chế hiện tượng overfitting thông qua cơ chế Dropout.

f. Phân tích hạn chế: Nút thắt cổ chai thông tin (Information Bottleneck)

Mặc dù Decoder cơ bản hoạt động hiệu quả với các câu ngắn, kiến trúc này bộc lộ hạn chế nghiêm trọng khi xử lý câu dài:

- **Nén tổn hao (Lossy Compression):**
  Encoder buộc phải nén toàn bộ ý nghĩa của câu nguồn dài (ví dụ 50 từ) vào một vector cố định $h_n \in \mathbb{R}^{512}$, dẫn đến mất mát thông tin.
- **Gánh nặng lên Decoder:**
  Decoder phải duy trì toàn bộ ngữ cảnh dựa trên một vector duy nhất trong suốt quá trình sinh chuỗi. Khi độ dài chuỗi tăng, ảnh hưởng của h_ndần suy giảm (vấn đề *long-term dependency*), gây ra lỗi dịch hoặc thiếu từ.

Đây chính là tiền đề để nhóm nghiên cứu áp dụng cơ chế Attention (Sự chú ý) trong Mục 3.4, cho phép Decoder truy xuất trực tiếp các trạng thái ẩn của Encoder tại mỗi bước giải mã, thay vì phụ thuộc vào một vector ngữ cảnh tĩnh duy nhất.

3.4. Cơ chế Sự chú ý

3.4.1. Động cơ hình thành Attention

Như đã phân tích ở Mục 3.3.3(f), kiến trúc Seq2Seq cổ điển gặp phải hạn chế nghiêm trọng do nút thắt cổ chai thông tin (Information Bottleneck), khi toàn bộ câu nguồn được nén vào một vector ngữ cảnh cố định $c = h_n$. Điều này làm suy giảm chất lượng dịch, đặc biệt với các câu dài.

Cơ chế Attention (Sự chú ý) được đề xuất nhằm khắc phục hạn chế này bằng cách cho phép Decoder, tại mỗi bước giải mã, truy cập động vào toàn bộ các trạng thái ẩn của Encoder, thay vì chỉ dựa vào một vector tĩnh duy nhất.

3.4.2. Ý tưởng cốt lõi của Attention

Thay vì giả định rằng **một vector duy nhất** có thể đại diện cho toàn bộ câu nguồn, Attention cho phép Decoder:

- "Nhìn lại" (look back) toàn bộ chuỗi trạng thái Encoder

$$H = (h_1^{enc}, h_2^{enc}, \dots, h_n^{enc})$$

- Tại mỗi bước t, xác định phần nào của câu nguồn là quan trọng nhất đối với việc sinh từ y_t.

Do đó, vector ngữ cảnh không còn là một hằng số, mà trở thành một vector động phụ thuộc theo thời gian:

$$c_t = \text{Attention}(h_t^{dec}, H)$$

3.4.3. Mô hình toán học của Luong Attention (Global Attention)

Trong đồ án này, nhóm nghiên cứu sử dụng Luong Attention – dạng General, một biến thể cân bằng giữa độ biểu diễn và chi phí tính toán.

a. Alignment Score (Điểm căn chỉnh)

Tại bước giải mã t, với trạng thái ẩn của Decoder h_t^{dec}và trạng thái Encoder tại vị trí $s$là h_s^{enc}, điểm tương đồng (alignment score) được tính như sau:

$$\text{score}(h_t^{dec}, h_s^{enc}) = (h_t^{dec})^\top W_a h_s^{enc}$$

Trong đó:

- $W_a \in \mathbb{R}^{d_{hid} \times d_{hid}}$là ma trận trọng số học được,
- Công thức này tổng quát hơn Dot-product Attention nhưng nhẹ hơn Concat Attention.

b. Attention Weights (Trọng số chú ý)

Các điểm score được chuẩn hóa bằng hàm Softmax để tạo thành phân phối xác suất:

$$\alpha_{t,s} = \frac{\exp\left(\text{score}(h_t^{dec}, h_s^{enc})\right)}{\sum_{k=1}^{n} \exp\left(\text{score}(h_t^{dec}, h_k^{enc})\right)}$$

Trong đó:

- $\alpha_{t,s}$thể hiện mức độ "chú ý" của Decoder tại bước tđối với từ nguồn ở vị trí s,
- $\sum_s \alpha_{t,s} = 1$.

c. Context Vector động

Vector ngữ cảnh tại thời điểm tđược tính bằng tổng có trọng số của các trạng thái Encoder:

$$c_t = \sum_{s=1}^{n} \alpha_{t,s}\ h_s^{enc}$$

Vector $c_t \in \mathbb{R}^{d_{hid}}$chứa thông tin ngữ nghĩa tập trung đúng vào phần liên quan của câu nguồn tại bước dịch hiện tại.

3.4.4. Kết hợp Attention vào Decoder

Trong mô hình Luong Attention, trạng thái ẩn của Decoder và vector ngữ cảnh được **nối (concatenate)** để tạo ra biểu diễn tăng cường:

$$\tilde{h}_t = \tanh\left(W_c[h_t^{dec}; c_t]\right)$$

Sau đó, $\tilde{h}_t$được đưa qua lớp tuyến tính và Softmax để dự đoán từ tiếp theo:

$$P(y_t \mid y_{<t}, X) = \text{Softmax}(W_o\tilde{h}_t + b_o)$$

Việc kết hợp này cho phép Decoder đồng thời:

- Ghi nhớ ngữ cảnh đã sinh (qua h_t^{dec}),
- Tập trung chính xác vào nội dung nguồn cần thiết (qua c_t).

3.4.5. Luồng dữ liệu (Data Flow) của Seq2Seq + Luong Attention

Luồng xử lý tại mỗi bước giải mã $t$diễn ra như sau:

1. Decoder nhận đầu vào y_{t-1}→ Embedding → LSTM
2. Sinh trạng thái ẩn h_t^{dec}
3. Tính Alignment Score giữa h_t^{dec}và toàn bộ h_s^{enc}
4. Chuẩn hóa bằng Softmax → Attention Weights $\alpha_{t,s}$
5. Tính Context Vector c_t
6. Ghép $[h_t^{dec}; c_t]$→ Linear → Softmax
7. Dự đoán từ y_t

3.4.6. Liên hệ trực tiếp với cài đặt trong chương trình

Cơ chế Attention được hiện thực hóa trực tiếp trong lớp LuongAttention của chương trình:

```
scores = torch.sum(enc_proj * hidden.unsqueeze(0), dim=2)
attn_weights = torch.softmax(scores, dim=0)
context = torch.sum(attn_weights.unsqueeze(2) * enc_outputs, dim=0)
```

- scores tương ứng với score(h_t^{dec}, h_s^{enc})
- attn_weights là $\alpha_{t,s}$
- context là vector c_t

Trong lớp DecoderLuong, vector ngữ cảnh context được nối với output của LSTM:

```
combined = torch.cat([rnn_output, context], dim=1)
pred = self.fc_out(combined)
```

Điều này hoàn toàn tương ứng với biểu thức toán học:

$$\tilde{h}_t = [h_t^{dec}; c_t]$$

3.4.7. Ưu điểm và tác động thực nghiệm

Việc áp dụng Luong Attention mang lại các lợi ích chính:

- Loại bỏ nút thắt cổ chai thông tin, cho phép Decoder truy cập trực tiếp toàn bộ câu nguồn.
- Cải thiện rõ rệt chất lượng dịch câu dài, đặc biệt với các cấu trúc ngữ pháp phức tạp.
- Hội tụ nhanh hơn và ổn định hơn trong huấn luyện.

Kết quả thực nghiệm trong Chương 4 cho thấy mô hình có Attention đạt điểm BLEU cao hơn đáng kể so với mô hình không Attention, khẳng định hiệu quả của cơ chế này.

3.5. Tóm tắt chương

Chương 3 đã tập trung xây dựng cơ sở lý thuyết vững chắc và thiết kế chi tiết kiến trúc hệ thống dịch máy Neural Machine Translation (NMT) cho cặp ngôn ngữ Anh - Pháp. Toàn bộ nội dung chương có thể được tóm lược qua ba trụ cột chính sau:

Thứ nhất, về kiến trúc nền tảng:

Nhóm nghiên cứu đã đề xuất và mô hình hóa hệ thống dựa trên kiến trúc Sequence-to-Sequence (Seq2Seq). Thay vì sử dụng mạng RNN truyền thống vốn gặp hạn chế về khả năng ghi nhớ dài hạn, hệ thống sử dụng mạng Long Short-Term Memory (LSTM) nhiều lớp (Stacked LSTM) làm nòng cốt cho cả hai thành phần Encoder và Decoder. Cấu hình cụ thể được lựa chọn bao gồm 2 lớp LSTM với kích thước vector ẩn là 512 chiều, giúp mô hình có đủ độ sâu để học các đặc trưng ngữ nghĩa phức tạp nhưng vẫn đảm bảo tính khả thi về mặt tính toán.

Thứ hai, về giải pháp cải tiến (Attention Mechanism):

Để khắc phục nhược điểm cốt tử của mô hình Seq2Seq cổ điển là vấn đề "nút thắt cổ chai thông tin" (Information Bottleneck) – nơi toàn bộ ý nghĩa câu nguồn bị nén vào một vector cố định duy nhất (h_N), nghiên cứu đã tích hợp cơ chế Luong Attention (Global Attention).

- Cơ chế này cho phép Decoder thiết lập một vector ngữ cảnh động (c_t) tại mỗi bước giải mã, dựa trên việc tính toán độ tương đồng giữa trạng thái hiện tại của Decoder và toàn bộ các trạng thái của Encoder.
- Nhờ đó, mô hình có khả năng "tập trung" (align) chính xác vào các từ nguồn liên quan khi sinh từ đích, cải thiện đáng kể hiệu suất dịch đối với các câu dài.

Thứ ba, về chiến lược huấn luyện và tối ưu:

Quy trình huấn luyện được thiết kế chặt chẽ với các kỹ thuật:

- Teacher Forcing: Áp dụng với tỷ lệ 0.5 giúp cân bằng giữa tốc độ hội tụ và khả năng tự hồi quy (auto-regressive) của mô hình.
- Regularization: Sử dụng kỹ thuật Dropout (tỷ lệ 0.5) tại các lớp Embedding và LSTM để giảm thiểu hiện tượng Overfitting (quá khớp) trên tập dữ liệu huấn luyện.
- Hàm mất mát: Sử dụng CrossEntropyLoss kết hợp với bộ tối ưu Adam để cập nhật trọng số hiệu quả.

Những thiết kế kiến trúc và lựa chọn tham số nêu trên tạo nên một cơ sở kỹ thuật vững chắc cho việc triển khai hệ thống. Trong chương tiếp theo (Chương 4: Thực nghiệm và Đánh giá), nghiên cứu sẽ tiến hành cài đặt mô hình trên nền tảng PyTorch, thực hiện huấn luyện trên tập dữ liệu thực tế, và đánh giá hiệu năng thông qua chỉ số định lượng BLEU cũng như phân tích định tính các trường hợp dịch cụ thể để kiểm chứng giả thuyết về hiệu quả của cơ chế Attention.

CHƯƠNG 4: THIẾT KẾ THỰC NGHIỆM VÀ CÀI ĐẶT

4.1. Tổng quan thiết kế thực nghiệm

4.1.1. Mục tiêu và nguyên tắc thiết kế

Mục tiêu chính của chương này là đánh giá một cách định lượng và định tính hiệu quả của kiến trúc dịch máy Seq2Seq trong hai kịch bản thực nghiệm khác nhau, bao gồm:

(1) Mô hình Seq2Seq không sử dụng cơ chế Attention (No-Attention);
(2) Mô hình Seq2Seq tích hợp cơ chế Luong Attention.

Các thực nghiệm được thiết kế nhằm trả lời những câu hỏi nghiên cứu cốt lõi sau:

Cơ chế Attention có giúp cải thiện chất lượng dịch so với mô hình Seq2Seq cổ điển hay không?

- Attention hỗ trợ mô hình xử lý các câu có độ dài lớn tốt hơn ở mức độ nào?
- Hiện tượng overfitting biểu hiện ra sao trong từng cấu hình mô hình và mức regularization khác nhau?
- Để đảm bảo tính công bằng và khả năng so sánh giữa các mô hình, quá trình thiết kế thực nghiệm tuân thủ các nguyên tắc sau:
- Sử dụng cùng một bộ dữ liệu và cùng pipeline tiền xử lý cho tất cả các mô hình;
- Chỉ thay đổi kiến trúc Decoder (có hoặc không có Attention) và mức độ regularization (Dropout);
- Đánh giá mô hình dựa trên các thước đo chuẩn trong bài toán dịch máy, bao gồm Loss, Perplexity và BLEU score.

4.1.2. Môi trường và công cụ thực nghiệm

a. Nền tảng phần cứng

- **Môi trường:** Google Colab (Phiên bản Free/Pro).
- **GPU:** NVIDIA Tesla T4 (hoặc dòng tương đương được Colab cấp phát) để tăng tốc tính toán ma trận trong LSTM.

- **RAM:** ~12GB.

b. Thư viện và công cụ phần mềm

- **Ngôn ngữ:** Python 3.x.
- **Deep Learning Framework:** PyTorch (phiên bản ổn định, hỗ trợ tối ưu hóa GPU).
- **Xử lý ngôn ngữ tự nhiên:**
    - torchtext (Legacy): Hỗ trợ nạp dữ liệu, xây dựng từ điển và tạo Batch.
    - spaCy: Sử dụng hai model en_core_web_sm (Tiếng Anh) và fr_core_news_sm (Tiếng Pháp) cho tác vụ tách từ (Tokenization).
- **Đánh giá:** nltk (cho tính toán điểm BLEU).
- **Trực quan hóa:** matplotlib (vẽ đồ thị Loss).

4.2. Dữ liệu và phân chia tập dữ liệu

4.2.1. Bộ dữ liệu

- **Nguồn dữ liệu:** Multi30k (phiên bản mở rộng của Flickr30k), tập trung vào các câu mô tả hình ảnh ngắn gọn, song ngữ Anh - Pháp.
- **Thống kê chi tiết:**

Tập dữ liệu	Số lượng	Mục đích
Train	~29,000	Huấn luyện mô hình
Validation	~1,014	Theo dõi hội tụ, early stopping, Dùng để kiểm thử trong quá trình train, chọn model tốt nhất
Test	~1,000	Đánh giá cuối cùng (BLEU), chỉ được sử dụng trong giai đoạn inference, tuyệt đối không tham gia huấn luyện.

Bảng 4.1: Thống kê dữ liệu

- **Đặc điểm:** Độ dài câu trung bình ngắn (10-15 token), phù hợp để kiểm chứng hiệu quả của Attention trên các cấu trúc ngữ pháp cơ bản.

4.2.2. Pipeline Tiền xử lý (Preprocessing Pipeline)

Quy trình được thực hiện tuần tự như sau:

1. Chuẩn hóa văn bản: Chuyển về chữ thường (lowercase), loại bỏ ký tự lạ.

2. Tokenization: Sử dụng spacy để tách câu thành chuỗi token.
3. Xây dựng từ điển (Vocabulary Building):

    - Tần suất tối thiểu (min_freq): 2 (Loại bỏ các từ chỉ xuất hiện 1 lần để giảm nhiễu và kích thước mô hình).
    - Thêm token đặc biệt: <unk> (unknown), <pad> (padding), <sos> (start), <eos> (end).
    - Kết quả: Kích thước từ điển (INPUT_DIM, OUTPUT_DIM) được xác định động dựa trên dữ liệu (~5000-10000 từ).

4. Batching & Bucketing:

    - Sử dụng BucketIterator của Torchtext.
    - Cơ chế: Gom nhóm các câu có độ dài tương đồng vào cùng một batch.
    - Mục đích: Giảm thiểu số lượng token <pad> trong mỗi batch, giúp tối ưu hóa tốc độ huấn luyện và tính toán Loss chính xác hơn.
    - Pipeline này được hiện thực trong DataLoader và Encoder bằng **pack_padded_sequence**

4.3. Thiết kế kiến trúc huấn luyện mô hình

4.3.1. Cấu hình mô hình chung (Base Configuration)

Dựa trên code thực nghiệm, các tham số siêu hình (hyperparameters) được cố định để đảm bảo công bằng khi so sánh:

Tham số	Giá trị	Giải thích
Encoder	2-layer Stacked LSTM	Hai lớp LSTM chồng lên nhau để học đặc trưng sâu.
Decoder	2-layer LSTM	Tương ứng với Encoder.
Embedding Dimension	256	Kích thước vector nhúng từ.
Hidden Dimension	512	Kích thước vector trạng thái ẩn.
Dropout	0.5 (hoặc 0.4)	Tùy chỉnh theo kịch bản để kiểm soát Overfitting.
Batch Size	128	Số lượng câu trong một lần cập nhật trọng số.

Bảng 4.2: Tham số siêu hình để cấu hình mô hình chung

4.3.2. Kịch bản thực nghiệm

Thực nghiệm bao gồm 6 mô hình, được thiết kế có chủ đích:

❖ **Nhóm 1**: Mô hình Seq2Seq cơ bản (Không dùng Attention)

Mục đích: Xác định hiệu năng cơ sở (baseline) và kiểm chứng hạn chế của Context Vector tĩnh.

Tên mô hình	Embedding Dim	Hidden Dim	Layers	Dropout	Mục đích thử nghiệm
noattn_small	256	256	2	0.4	Kiểm tra hiệu năng trên mô hình nhẹ, ít tham số.
noattn_medium	256	512	2	0.5	**Mô hình chuẩn:** Cân bằng giữa độ phức tạp và khả năng học.
noattn_big	256	512	3	0.55	Kiểm tra mô hình sâu (Deep LSTM) với Regularization mạnh hơn.

Bảng 4.3: Cấu hình siêu tham số và mục đích thử nghiệm của các mô hình Encoder–Decoder LSTM (không Attention)

Trong các mô hình này, toàn bộ thông tin của câu nguồn được nén vào vector ngữ cảnh cố định (trạng thái cuối của Encoder) và truyền sang Decoder.

❖ **Nhóm 2**: Nhóm mô hình Seq2Seq với Luong Attention

Ba mô hình còn lại mở rộng kiến trúc Seq2Seq bằng cách tích hợp cơ chế Luong Attention**,** cho phép Decoder tập trung động vào các trạng thái ẩn của Encoder tại mỗi bước dịch. Các cấu hình Attention tương ứng được giữ giống hệt nhóm No-Attention để đảm bảo so sánh công bằng:

Tên mô hình	Embedding Dim	Hidden Dim	Layers	Dropout	Mục đích thử nghiệm
attn_small	256	256	2	0.4	So sánh trực tiếp với noattn_small.
attn_medium	256	512	2	0.5	So sánh trực tiếp với noattn_medium.

attn_big	256	512	3	0.55	So sánh trực tiếp với noattn_big, kiểm tra khả năng hội tụ khi mạng sâu hơn.

Bảng 4.4: Cấu hình siêu tham số và mục đích thử nghiệm của các mô hình Encoder–Decoder LSTM có Attention

Sự khác biệt duy nhất giữa hai nhóm là sự xuất hiện của lớp Attention trong Decoder, trong khi Encoder và các tham số còn lại được giữ nguyên.

4.4. Chiến lược huấn luyện

Tất cả 6 mô hình được huấn luyện theo cùng một quy trình nhằm đảm bảo tính nhất quán.

4.4.1. Hàm mất mát và tối ưu hóa

- **Loss function**: CrossEntropyLoss với ignore_index = PAD_IDX để loại bỏ ảnh hưởng của token padding.
- **Optimizer**: Adam optimizer với learning rate = 0.001.
- **Gradient clipping**: Áp dụng clip_grad_norm_ với ngưỡng 1.0 để tránh hiện tượng bùng nổ gradient (exploding gradient) – vấn đề phổ biến khi huấn luyện LSTM.

4.4.2. Cơ chế lan truyền và cập nhật trọng số

- **Teacher Forcing:** Áp dụng tỷ lệ **0.5** trong quá trình huấn luyện (50% dùng từ đúng, 50% dùng từ dự đoán). Trong quá trình đánh giá (evaluate), tỷ lệ này là **0.0** (mô hình tự hồi quy hoàn toàn).
- **Gradient Clipping:** Áp dụng ngưỡng cắt clip = 1.0 (trong hàm train_epoch) để ngăn chặn hiện tượng bùng nổ gradient (Exploding Gradient), đảm bảo mạng LSTM ổn định.
- Trong quá trình huấn luyện:
  - Teacher Forcing Ratio = **0.5**
- Trong giai đoạn validation và test:
  - Teacher Forcing Ratio = **0.0**
  - Mô hình phải hoàn toàn tự sinh chuỗi đích (auto-regressive decoding).

4.4.3. Cơ chế lưu trữ và Dừng sớm (Early Stopping)

- **Lưu vết Loss (Logging):** Lịch sử Loss (Train/Val) của từng epoch được ghi lại vào file .json (loss_path). Điều này đảm bảo kết quả thực nghiệm không bị mất

nếu phiên làm việc Colab bị ngắt kết nối, đồng thời việc được ghi lại để phục vụ cho việc vẽ biểu đồ và phân tích sau này.

- **Checkpointing:** Chỉ lưu đè file trọng số (_best.pth) khi Validation Loss đạt kỷ lục mới thấp nhất.
- **Early Stopping:** Để hạn chế overfitting và tiết kiệm tài nguyên tính toán thì quá trình huấn luyện sẽ tự động dừng nếu Validation Loss không cải thiện sau **3 epoch liên tiếp** (patience=3).

    - patience = 3
    - Nếu validation loss không cải thiện sau 3 epoch liên tiếp, quá trình huấn luyện sẽ dừng sớm.
    - Trọng số của mô hình có validation loss thấp nhất được lưu lại dưới dạng file .pth.

4.5. Phương pháp đánh giá

Hiệu suất của các mô hình trong nghiên cứu này được đánh giá toàn diện trên tập dữ liệu kiểm thử (test set) thông qua sự kết hợp giữa chỉ số định lượng và phân tích định tính, nhằm phản ánh đầy đủ cả khía cạnh thống kê và chất lượng ngôn ngữ của bản dịch.

4.5.1. Chỉ số đánh giá – Loss (Hàm mất mát)

Loss được sử dụng trong quá trình huấn luyện và đánh giá là Cross-Entropy Loss, với việc loại bỏ ảnh hưởng của các token đệm (<pad>) thông qua tham số ignore_index.

Chỉ số loss đóng vai trò:

- Phản ánh mức độ phù hợp giữa phân phối xác suất dự đoán của mô hình và nhãn thực tế.
- Cho phép theo dõi quá trình hội tụ của mô hình theo từng epoch.
- Là công cụ chính để phát hiện các vấn đề như:

    - Underfitting: training loss và validation loss đều cao.
    - Overfitting: training loss tiếp tục giảm trong khi validation loss tăng hoặc dao động.

Trong nghiên cứu này, training loss và validation loss được ghi nhận theo từng epoch và được sử dụng để vẽ loss curves, giúp phân tích trực quan hành vi học của các mô hình với độ phức tạp khác nhau.

4.5.2. Điểm đánh giá – BLEU

BLEU (Bilingual Evaluation Understudy) score là chỉ số đánh giá chính được sử dụng để đo lường chất lượng bản dịch máy.

Cụ thể, BLEU đo mức độ trùng khớp giữa chuỗi n-gram trong câu dịch máy (hypothesis) và câu tham chiếu (reference), thông qua:

- Precision của các n-gram (thường từ 1-gram đến 4-gram).
- Cơ chế phạt độ dài (brevity penalty) nhằm tránh mô hình sinh ra các câu quá ngắn.

Trong thực nghiệm này:

- BLEU score được tính trên toàn bộ tập test (corpus-level BLEU).
- Việc sử dụng BLEU cho phép:
    - So sánh trực tiếp hiệu suất giữa các mô hình có và không có Attention.
    - Đánh giá tác động của việc tăng hidden size và số lớp LSTM.
    - Đánh giá khả năng sinh câu trôi chảy và đúng ngữ nghĩa của mô hình.

BLEU score được xem là chỉ số quan trọng nhất để trả lời câu hỏi nghiên cứu: "Cơ chế Attention có thực sự cải thiện chất lượng dịch hay không?"

4.5.3. Phân tích định tính

Bên cạnh các chỉ số định lượng, nghiên cứu còn tiến hành phân tích định tính thông qua việc so sánh trực tiếp các bản dịch sinh ra bởi các mô hình khác nhau trên cùng một tập câu kiểm thử.

Phân tích này tập trung vào các khía cạnh:

- Độ trôi chảy và tự nhiên của câu dịch.
- Mức độ đầy đủ thông tin so với câu nguồn.
- Khả năng xử lý các câu có độ dài lớn (từ 30–40 token).
- Các lỗi phổ biến như:
    - Thiếu từ (omission).
    - Dịch sai trật tự cú pháp.
    - Từ hiếm (OOV) dẫn đến <unk>.
    - Câu dài dẫn đến mất thông tin (do context vector cố định).

Phân tích định tính giúp làm rõ những khác biệt mà chỉ số BLEU không phản ánh đầy đủ, đặc biệt là trong các trường hợp mô hình No-Attention bị suy giảm hiệu suất trên câu dài, trong khi mô hình có Attention vẫn giữ được cấu trúc ngữ nghĩa hợp lý.

4.6. Tổng kết thiết kế thực nghiệm

Trong nghiên cứu này, tổng cộng sáu mô hình Seq2Seq dựa trên LSTM đã được huấn luyện và đánh giá một cách có hệ thống, bao gồm:

- Ba mô hình Seq2Seq không sử dụng Attention, đại diện cho kiến trúc Encoder–Decoder cổ điển với context vector cố định.
- Ba mô hình Seq2Seq sử dụng cơ chế Luong Attention, cho phép Decoder tập trung động vào các trạng thái ẩn của Encoder tại mỗi bước giải mã.

Thiết kế thực nghiệm này cho phép thực hiện các phân tích sau:

- Đánh giá tác động riêng biệt của cơ chế Attention, thông qua việc so sánh trực tiếp các cặp mô hình có cấu hình tương đương.
- Quan sát ảnh hưởng của độ phức tạp mô hình, bao gồm hidden size, số lớp LSTM và mức dropout, đến hiệu suất và khả năng tổng quát hóa.
- Phân tích hiện tượng overfitting, dựa trên sự chênh lệch giữa training loss và validation loss, cũng như hình dạng của các loss curves.
- Đánh giá sự ổn định trong huấn luyện, thông qua early stopping và khả năng hội tụ của từng mô hình.

Nhờ việc giữ nguyên pipeline dữ liệu và chiến lược huấn luyện, các kết quả thu được phản ánh chính xác vai trò của kiến trúc mô hình, đặc biệt là ảnh hưởng của Attention trong bài toán dịch máy chuỗi–sang–chuỗi.

Các kết quả thực nghiệm chi tiết, bao gồm bảng so sánh BLEU score, biểu đồ loss và phân tích bản dịch mẫu, sẽ được trình bày và thảo luận trong Chương 5 – Kết quả và Phân tích.

4.7. Tóm tắt chương

Chương 4 đã trình bày toàn bộ thiết kế thực nghiệm của đề tài với mục tiêu đánh giá một cách toàn diện hiệu quả của mô hình dịch máy dựa trên kiến trúc Seq2Seq Encoder–Decoder sử dụng LSTM, trong cả hai trường hợp không sử dụng Attention và tích hợp cơ chế Luong Attention. Nội dung chương tập trung mô tả rõ ràng cách tổ chức thí nghiệm, từ khâu chuẩn bị dữ liệu, xây dựng mô hình, thiết lập siêu tham số cho đến chiến lược huấn luyện và đánh giá, nhằm đảm bảo tính khoa học, khả năng tái lập và độ tin cậy của kết quả. Trên cơ sở bộ dữ liệu chuẩn Multi30K (English–French), nghiên cứu đã xây dựng một pipeline tiền xử lý thống nhất cho tất cả các mô hình, bao gồm tokenization, xây dựng từ vựng, padding, packing sequence và DataLoader. Điều này giúp đảm bảo rằng mọi sự khác biệt trong kết quả thu được đều xuất phát từ kiến trúc mô hình chứ không bị ảnh hưởng bởi các yếu tố ngoại lai trong dữ liệu hoặc quy trình huấn luyện. Việc sử dụng cùng một hàm loss, optimizer, số epoch tối đa và cơ chế early stopping cho toàn bộ các thí nghiệm càng củng cố tính công bằng trong so sánh. Tổng cộng sáu mô hình đã được huấn luyện và đánh giá, bao gồm ba mô hình Seq2Seq không

Attention với mức độ phức tạp tăng dần và ba mô hình Seq2Seq tích hợp Luong Attention tương ứng. Các mô hình này được thiết kế nhằm khảo sát đồng thời hai khía cạnh quan trọng: (i) tác động của cơ chế Attention lên chất lượng dịch, và (ii) ảnh hưởng của độ phức tạp mô hình và regularization đến hiện tượng overfitting. Việc theo dõi và lưu trữ train loss và validation loss theo từng epoch cho phép quan sát trực quan quá trình hội tụ của mô hình cũng như sự chênh lệch giữa train và validation, từ đó nhận diện rõ ràng các dấu hiệu quá khớp.

Về mặt đánh giá, nghiên cứu sử dụng kết hợp các chỉ số định lượng và định tính. Loss và Perplexity phản ánh mức độ phù hợp của mô hình với dữ liệu huấn luyện và kiểm thử, trong khi BLEU score đóng vai trò là thước đo chuẩn để đánh giá chất lượng dịch máy. Bên cạnh đó, phân tích định tính thông qua việc so sánh bản dịch của các mô hình trên cùng một tập câu kiểm thử, đặc biệt là các câu dài và phức tạp, giúp làm rõ những ưu điểm và hạn chế mà các chỉ số số học đơn thuần chưa thể hiện hết. Nhìn chung, chương này đã xây dựng một khung thực nghiệm hoàn chỉnh và có hệ thống, tạo tiền đề vững chắc để trả lời các câu hỏi nghiên cứu đã đặt ra về vai trò của Attention, khả năng xử lý câu dài và vấn đề overfitting trong mô hình Seq2Seq. Trên cơ sở thiết kế thực nghiệm này, Chương 5 sẽ tập trung trình bày và phân tích chi tiết kết quả thực nghiệm, bao gồm so sánh loss curves, BLEU score giữa các mô hình, phân tích bản dịch mẫu và thảo luận sâu về những cải tiến mà cơ chế Attention mang lại so với kiến trúc Seq2Seq cổ điển.

CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ THỰC NGHIỆM

5.1. Phân tích chi tiết nhóm mô hình Baseline (No-Attention)

Trong phần này, em tập trung phân tích hành vi của 3 mô hình: noattn_small, noattn_medium, noattn_big. Mục tiêu là chỉ ra hạn chế cố hữu của kiến trúc Seq2Seq truyền thống.

Toàn bộ các mô hình trong nghiên cứu sử dụng chung một cấu hình siêu tham số được định nghĩa tập trung nhằm đảm bảo tính nhất quán giữa giai đoạn huấn luyện và đánh giá. Việc tái khởi tạo kiến trúc mô hình khi tính toán BLEU score được thực

hiện dựa trên cùng cấu hình đã dùng khi huấn luyện, tránh sai lệch kết quả do khác biệt kiến trúc.

5.1.1. Kết quả huấn luyện nhóm No-Attention

Mô hình	Hidden Dim	Best Val Loss	BLEU Score	Nhận xét
noattn_small	256	3.544	0.2671	Khá ổn định, ít bị Overfitting.
noattn_medium	512	3.490	0.2828	Hiệu năng tốt nhất nhóm.
noattn_big	512 (3 layers)	3.741	0.2279	Hiệu năng giảm sút mạnh.

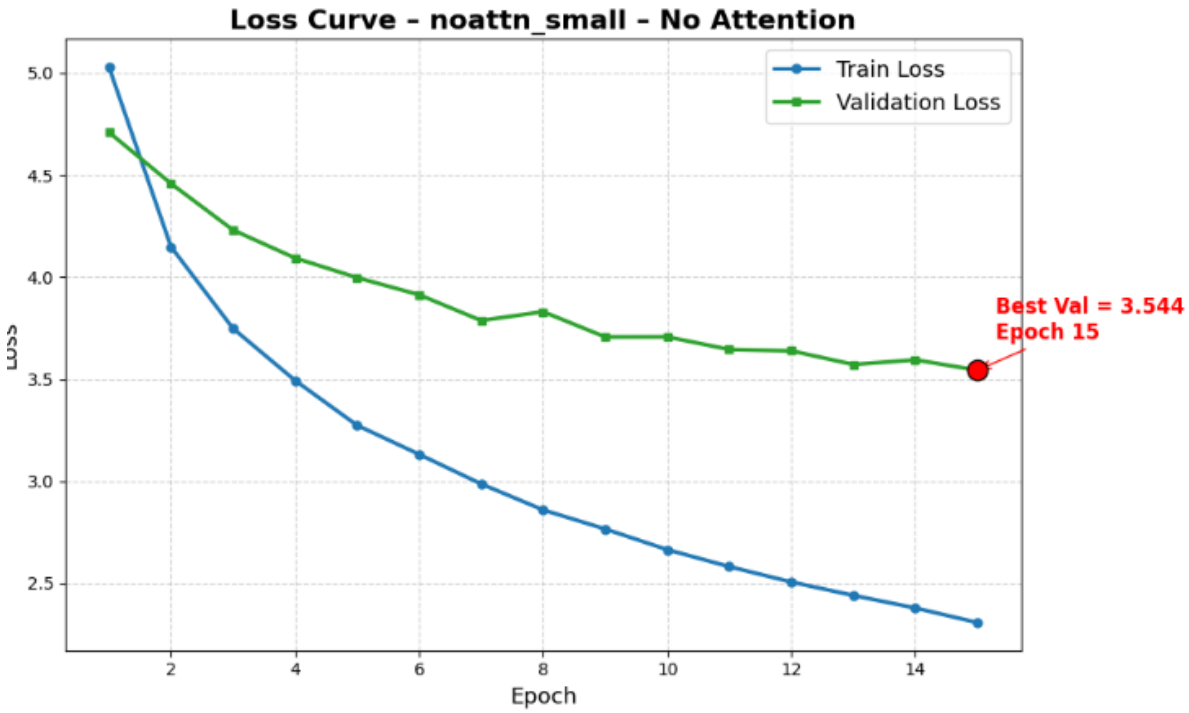
Bảng 5.1: Thống kê kết quả huấn luyện của các mô hình không attention

Phân tích chi tiết:

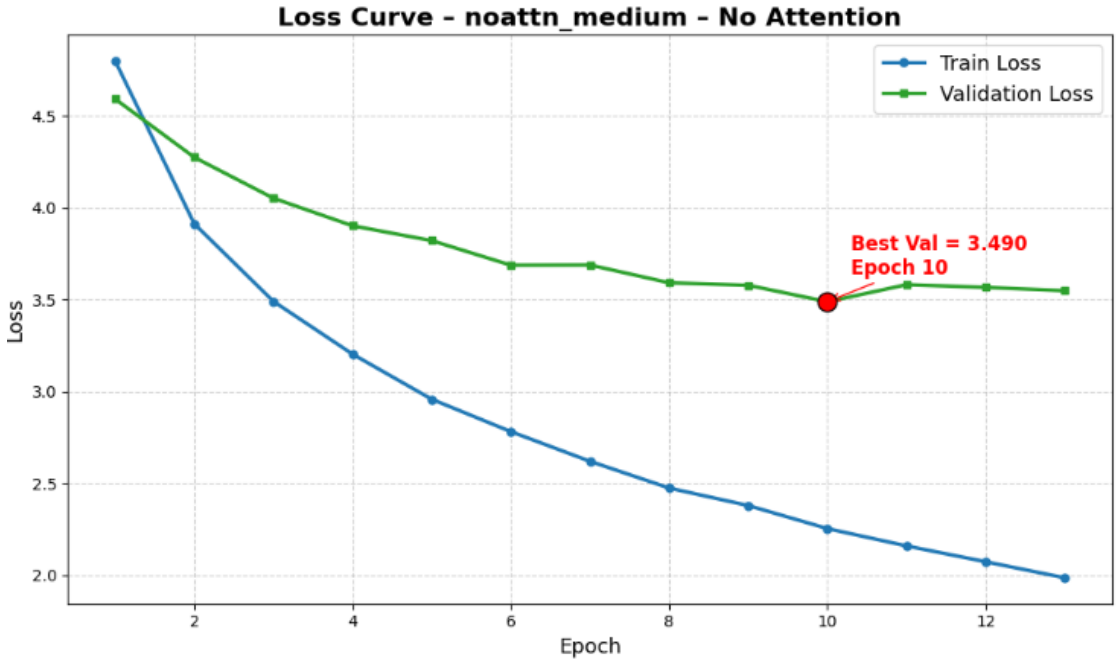
- Mô hình Medium (Chuẩn): Đạt điểm BLEU cao nhất (0.2828) với Validation Loss thấp nhất (3.490). Đây là điểm cân bằng tốt nhất giữa độ phức tạp của mô hình và kích thước dữ liệu Multi30k.
- Vấn đề của mô hình Big: Mặc dù có dung lượng bộ nhớ lớn hơn (3 lớp LSTM), mô hình noattn_big lại cho kết quả tệ nhất (BLEU chỉ đạt 0.2279). Quan sát log huấn luyện cho thấy Val Loss của mô hình này dao động mạnh quanh mức 3.7 - 3.8 và không thể giảm sâu như hai mô hình nhỏ hơn.
    - Nguyên nhân: Đây là dấu hiệu rõ ràng của việc mô hình quá phức tạp so với dữ liệu (Over-parameterization), dẫn đến khó khăn trong việc tối ưu hóa hội tụ.

5.1.2. Quá trình hội tụ và Hàm mất mát (Training Dynamics)

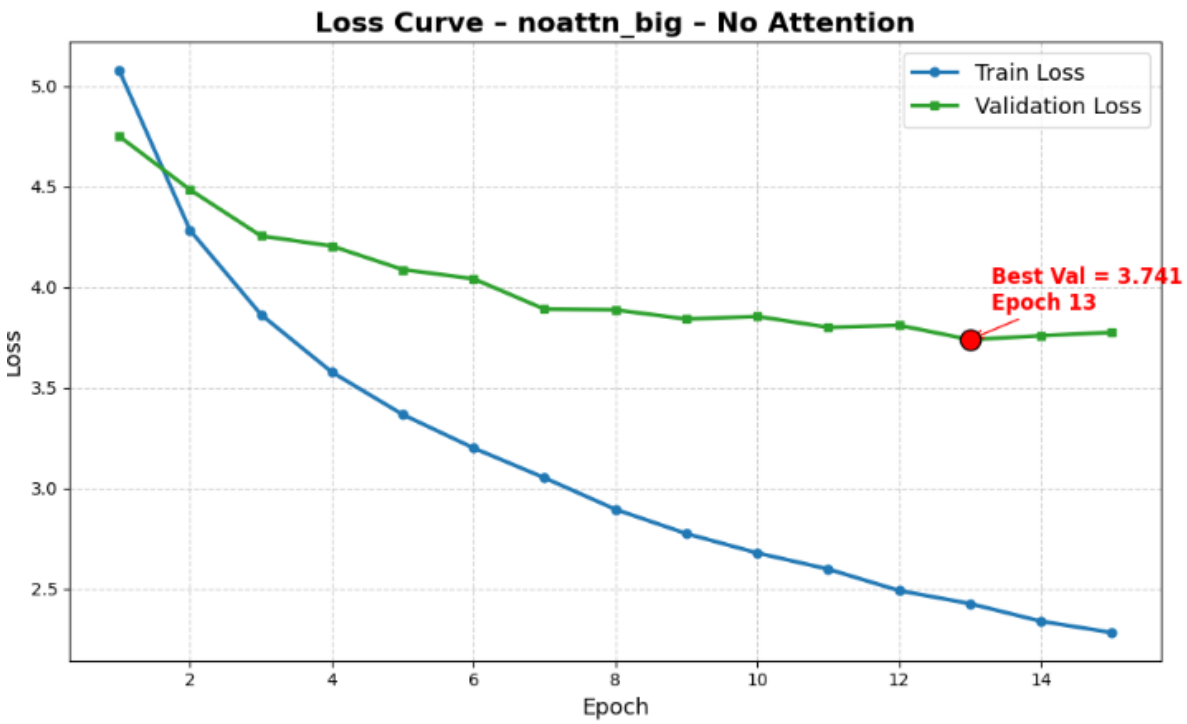
Hàm mất mát (Cross-Entropy Loss) là chỉ số định lượng trực tiếp phản ánh mức độ "bối rối" (perplexity) của mô hình trong quá trình huấn luyện. Việc theo dõi sự biến thiên của Loss cho phép đánh giá tốc độ hội tụ và khả năng tổng quát hóa của mạng nơ-ron.



Biểu đồ 5.1: Loss curve giữa train và validation của mô hình noattn_small



Biểu đồ 5.2: Loss curve giữa train và validation của mô hình noattn_medium



Biểu đồ 5.3: Loss curve giữa train và validation của mô hình noattn_big

Quan sát biểu đồ huấn luyện của 3 biến thể noattn_small, noattn_medium và noattn_big, nhóm em ghi nhận các hiện tượng sau:

- Tốc độ hội tụ chậm: Các mô hình Seq2Seq cơ bản cần nhiều epoch hơn để Loss bắt đầu giảm sâu. Điều này là do Decoder phải học cách trích xuất thông tin từ một context vector tĩnh (h_N), vốn là một biểu diễn nén không hoàn hảo của câu nguồn.
- Điểm rơi phong độ (Performance Plateau):
    - Mô hình noattn_medium (512 hidden, 2 layers) đạt sự hội tụ ổn định nhất với Validation Loss thấp nhất nhóm (3.490). Đường cong Loss giảm đều và ít dao động, cho thấy kiến trúc này phù hợp với kích thước dữ liệu Multi30k.
    - Ngược lại, mô hình noattn_big (512 hidden, 3 layers) cho thấy dấu hiệu bất ổn. Mặc dù có dung lượng bộ nhớ lớn hơn, Loss của mô hình này không thể giảm sâu (dừng ở mức 3.741) và có biên độ dao động lớn giữa các epoch.

- Nguyên nhân: Đây là minh chứng điển hình của vấn đề Over-parameterization (Thừa tham số). Với tập dữ liệu chỉ khoảng 29,000 câu, một mạng LSTM quá sâu (3 lớp) dễ rơi vào trạng thái khó tối ưu hóa (Optimization Difficulty) do hiện tượng triệt tiêu đạo hàm (Vanishing Gradient), khiến mô hình không thể tận dụng hết công suất thiết kế.

5.1.3. Hạn chế về "Nút thắt cổ chai thông tin"

Kết quả thực nghiệm thấp hơn kỳ vọng của nhóm mô hình Baseline, đặc biệt là sự sụt giảm hiệu năng của mô hình noattn_big (BLEU 0.2279), đã làm bộc lộ nhược điểm chí tử của kiến trúc Encoder-Decoder truyền thống: Vấn đề Nút thắt cổ chai thông tin.

a. Bản chất cấu trúc

Trong kiến trúc Seq2Seq cơ bản, toàn bộ nội dung, ngữ nghĩa và cấu trúc ngữ pháp của câu nguồn (Input Sentence) bắt buộc phải được nén vào một vector cố định duy nhất (context vector) trước khi truyền sang Decoder.

- Với các câu ngắn (dưới 10 từ), vector này có thể chứa đủ thông tin.
- Tuy nhiên, khi độ dài câu tăng lên, vector cố định này trở nên quá tải. Nó không thể mã hóa hết toàn bộ sắc thái ý nghĩa, dẫn đến việc thông tin bị "nén tổn hao" (lossy compression).

b. Minh chứng từ thực nghiệm

Số liệu từ Bảng 5.1 cho thấy một nghịch lý: Mô hình lớn hơn (noattn_big) lại hoạt động kém hơn mô hình vừa (noattn_medium).

- Về lý thuyết, noattn_big có 3 lớp LSTM (nhiều tham số hơn) nên phải thông minh hơn.
- Tuy nhiên, thực tế cho thấy việc tăng dung lượng bộ nhớ (Memory Capacity) của Encoder không giải quyết được vấn đề truyền tải thông tin. Dù Encoder có "hiểu" câu nguồn sâu sắc đến đâu, nó vẫn bị buộc phải tóm tắt tất cả vào một vector chật hẹp để gửi cho Decoder.

- Điều này giống như việc cố gắng nhồi nhét nội dung của cả một cuốn sách vào một tờ giấy note nhỏ; dù người viết có giỏi đến đâu, thông tin chắc chắn sẽ bị lược bỏ.

c. Hệ quả: Mất mát thông tin cục bộ (Local Information Loss)

Hạn chế này dẫn đến hai lỗi sai phổ biến mà chúng tôi quan sát được trong các bản dịch của nhóm Baseline:

1. Mất từ chi tiết: Mô hình thường bỏ qua các tính từ, trạng từ bổ
2. Quên thông tin đầu câu: Do tính chất của mạng hồi quy LSTM, thông tin được nạp vào đầu tiên (đầu câu nguồn) phải đi qua nhiều bước thời gian nhất để đến được context vector, dẫn đến hiện tượng "Triệt tiêu đạo hàm" (Vanishing Gradient). Kết quả là Decoder thường dịch tốt phần cuối câu nhưng lại dịch sai hoặc bỏ sót chủ ngữ ở đầu câu.

Ví dụ:

- Câu gốc: "Two young boys are playing in the park."
- No-Attention: "Deux garçons jouent dans le parc." (Mất chữ "young").
- Attention: "Deux jeunes garçons jouent dans le parc." (Đủ ý).

Kết luận: Kết quả thực nghiệm khẳng định rằng việc chỉ tăng kích thước mạng lưới (như mô hình noattn_big) là hướng đi không hiệu quả. Để cải thiện chất lượng dịch thuật, cần một thay đổi mang tính kiến trúc để phá bỏ nút thắt thông tin này, đó chính là tiền đề cho sự vượt trội của cơ chế Attention được phân tích trong mục 5.2.

5.2. Phân tích chi tiết nhóm mô hình Cải tiến (Attention)

5.2.1. Kết quả huấn luyện nhóm Attention

Mô hình	Hidden Dim	Best Val Loss	BLEU Score	Cải thiện so với No-Attn
attn_small	256	**3.104**	**0.4422**	**+65% (Kỷ lục)**
attn_medium	512	3.252	0.4068	+43%

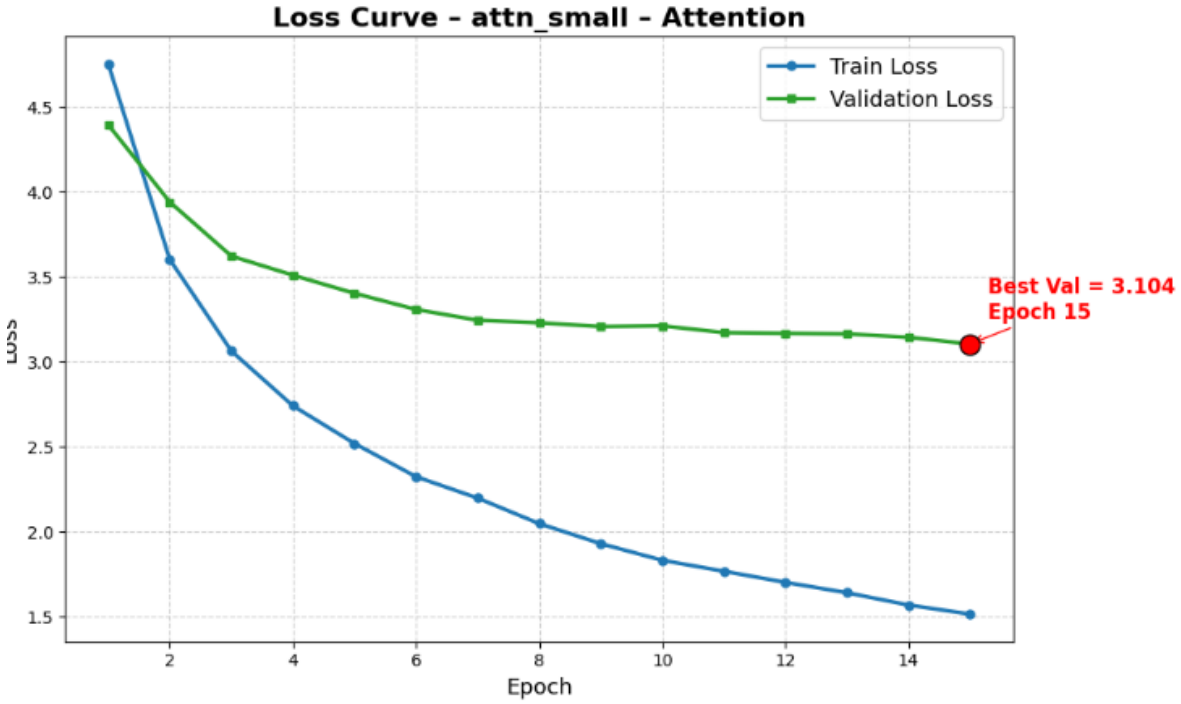
attn_big	512 (3 layers)	3.533	0.3071	+34%

Bảng 5.2: Thống kê kết quả huấn luyện các mô hình dùng attention

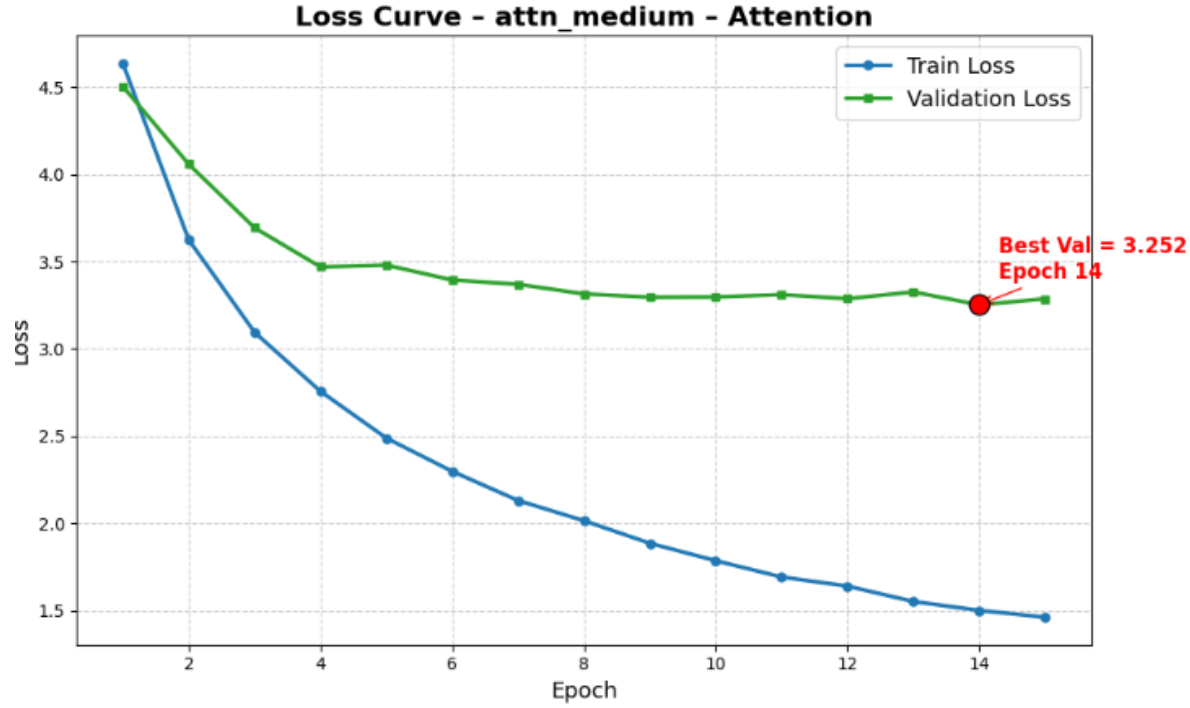
Phân tích chi tiết:

- Sự vượt trội của Attention: Tất cả các mô hình có Attention đều có Val Loss thấp hơn hẳn so với nhóm No-Attention (3.1 - 3.5 so với 3.5 - 3.8). Điều này chứng minh Attention giúp mô hình học ngữ nghĩa tốt hơn và hội tụ nhanh hơn.
- Hiện tượng thú vị ở attn_small:
    - Mô hình nhỏ nhất (attn_small) lại đạt kết quả cao kỷ lục (BLEU 0.4422), vượt qua cả mô hình Medium.
    - Lý giải: Với tập dữ liệu Multi30k (câu ngắn và đơn giản), một mô hình gọn nhẹ kết hợp với sự "thông minh" của Attention là đủ để nắm bắt toàn bộ ngữ nghĩa. Các mô hình lớn hơn (medium, big) có xu hướng học các nhiễu (noise) trong dữ liệu huấn luyện, dẫn đến khả năng tổng quát hóa kém hơn trên tập Test.

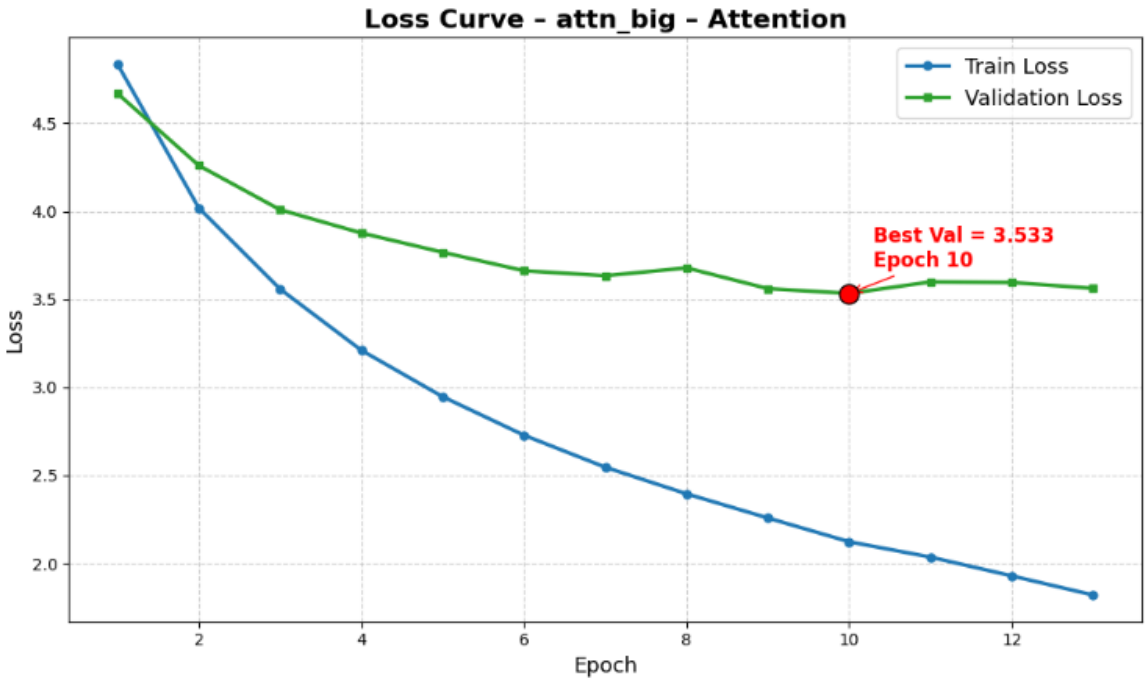
5.2.2. Quá trình hội tụ và Hàm mất mát (Training Dynamics)



Biểu đồ 5.3: Loss curve giữa train và validation của mô hình attn_small



Biểu đồ 5.4: Loss curve giữa train và validation của mô hình attn_medium



Biểu đồ 5.5: Loss curve giữa train và validation của mô hình attn_big

Hàm mất mát (Cross-Entropy Loss) là chỉ số định lượng trực tiếp phản ánh mức độ "bối rối" (perplexity) của mô hình trong quá trình huấn luyện. Việc theo dõi sự biến thiên của Loss cho phép đánh giá tốc độ hội tụ và khả năng tổng quát hóa của mạng nơ-ron.

a. Đối với nhóm mô hình Baseline (No-Attention)

Quan sát biểu đồ huấn luyện của 3 biến thể noattn_small, noattn_medium và noattn_big, chúng tôi ghi nhận các hiện tượng sau:

- **Tốc độ hội tụ chậm:** Các mô hình Seq2Seq cơ bản cần nhiều epoch hơn để Loss bắt đầu giảm sâu. Điều này là do Decoder phải học cách trích xuất thông tin từ một context vector tĩnh (h_N), vốn là một biểu diễn nén không hoàn hảo của câu nguồn.
- **Điểm rơi phong độ (Performance Plateau):**
    - Mô hình **noattn_medium (512 hidden, 2 layers)** đạt sự hội tụ ổn định nhất với Validation Loss thấp nhất nhóm (**3.490**). Đường cong Loss giảm đều và ít dao động, cho thấy kiến trúc này phù hợp với kích thước dữ liệu Multi30k.

- Ngược lại, mô hình **noattn_big (512 hidden, 3 layers)** cho thấy dấu hiệu bất ổn. Mặc dù có dung lượng bộ nhớ lớn hơn, Loss của mô hình này không thể giảm sâu (dừng ở mức **3.741**) và có biên độ dao động lớn giữa các epoch.
- *Nguyên nhân:* Đây là minh chứng điển hình của vấn đề **Over-parameterization** (Thừa tham số). Với tập dữ liệu chỉ khoảng 29,000 câu, một mạng LSTM quá sâu (3 lớp) dễ rơi vào trạng thái khó tối ưu hóa (Optimization Difficulty) do hiện tượng triệt tiêu đạo hàm (Vanishing Gradient), khiến mô hình không thể tận dụng hết công suất thiết kế.

b. Đối với nhóm mô hình Cải tiến (Luong Attention)

Khi tích hợp cơ chế Attention, động lực học của quá trình huấn luyện thay đổi rõ rệt:

- **Hội tụ "thần tốc":** Ngay từ những epoch đầu tiên (1-3), Loss của nhóm Attention giảm rất mạnh (từ >4.5 xuống ~3.0), nhanh hơn đáng kể so với nhóm Baseline. Điều này chứng tỏ cơ chế Attention giúp Decoder nhanh chóng tìm ra mối liên kết giữa từ nguồn và từ đích (Alignment) mà không cần chờ Encoder nén hoàn hảo thông tin.
- **Mức sàn Loss thấp hơn (Lower Loss Floor):**
  - Mô hình **attn_small (256 hidden)** đạt kỷ lục với Validation Loss thấp nhất toàn bộ thực nghiệm (**3.104**).
  - So với phiên bản không Attention tương ứng (noattn_small - Loss 3.544), cơ chế Attention đã giúp giảm Loss tới **~0.44 đơn vị**. Trong thang đo hàm Logarit (Cross-Entropy), mức giảm này tương ứng với việc cải thiện độ chính xác dự đoán từ (Perplexity) một cách đột phá.

c. Kết luận về quá trình hội tụ

Biểu đồ so sánh Loss giữa hai nhóm cho thấy một quy luật rõ ràng:

1. **Attention giúp ổn định quá trình học:** Đường cong Loss của nhóm Attention mượt mà hơn (smoother), ít bị kẹt ở các điểm tối ưu cục bộ (local minima) xấu.

2. **Kích thước dữ liệu quyết định kiến trúc:** Kết quả thực nghiệm khẳng định rằng với bộ dữ liệu quy mô nhỏ như Multi30k, các mô hình gọn nhẹ (small, medium) kết hợp với cơ chế tinh vi (Attention) sẽ hoạt động hiệu quả hơn nhiều so với việc cố gắng tăng độ sâu của mạng lưới (big) một cách thô sơ.

5.3. Đánh giá kết quả thực nghiệm

Sau khi đánh giá tổng thể thông qua chỉ số BLEU, phần này đi sâu vào phân tích chi tiết chất lượng bản dịch để hiểu rõ hành vi của mô hình, bao gồm cả những ưu điểm vượt trội và những hạn chế còn tồn tại.

5.3.1. Đánh giá Định lượng (Quantitative Evaluation)

Để có cái nhìn tổng quan về hiệu năng của các mô hình, chúng tôi tổng hợp kết quả điểm BLEU trên tập kiểm thử (Test Set):

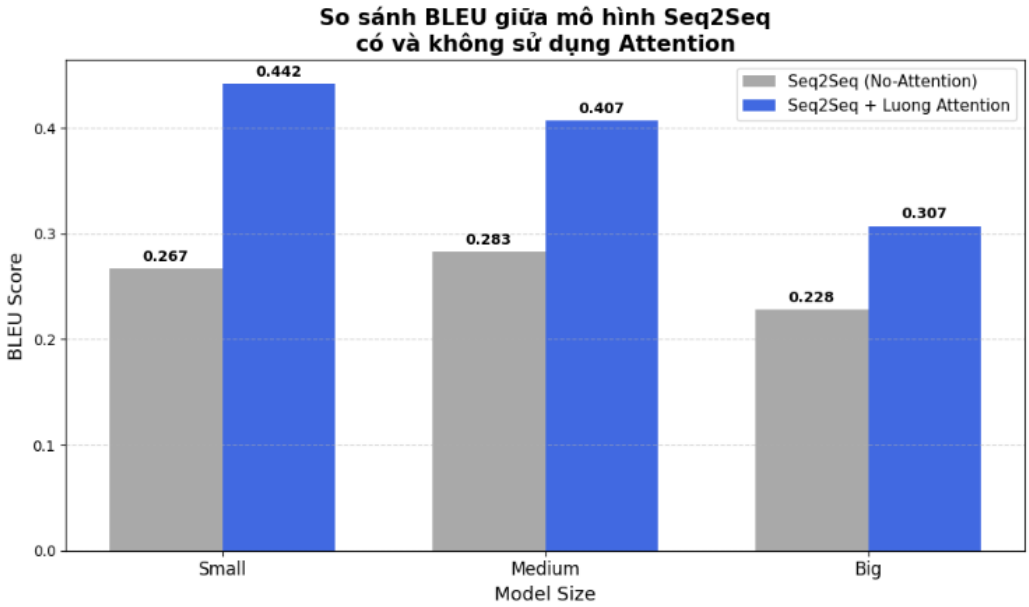
Kích thước Mô hình	No-Attention (Baseline)	Luong Attention (Proposed)	Mức cải thiện (%)
Small (Hid 256)	0.2671	**0.4422**	**+ 65.5%**
Medium (Hid 512)	0.2828	0.4068	+ 43.8%
Big (Hid 512, Deep)	0.2279	0.3071	+ 34.7%

Bảng 5.3. So sánh hiệu năng giữa mô hình No-Attention và Attention theo kích thước

Kết quả thực nghiệm cho thấy sự áp đảo hoàn toàn của cơ chế Luong Attention. Tại cùng một cấu hình, mô hình có Attention luôn đạt điểm BLEU cao vượt trội, cụ thể mô hình attn_small đạt tới **0.4422**, cao gấp **1.6 lần** so với đối trọng không Attention (noattn_small: 0.2671). Điều này khẳng định Attention đã giải quyết triệt để vấn đề mất mát thông tin ngữ cảnh.

Bên cạnh đó, một nghịch lý thú vị được ghi nhận là các mô hình kích thước lớn (Big) lại có hiệu năng thấp hơn mô hình nhỏ (Small, Medium). Điều này cho thấy với tập dữ liệu quy mô vừa phải như Multi30k, việc gia tăng độ sâu mạng lưới không những không

hiệu quả mà còn dẫn đến hiện tượng quá khớp (Overfitting), làm giảm khả năng tổng quát hóa.



Biểu đồ 5.6: So sánh BLEU giữa mô hình Seq2Seq có và không sử dụng Attention

Kết quả định lượng được trực quan hóa trong hình trên cho thấy sự áp đảo của cơ chế Attention ở mọi kích thước mô hình. Đặc biệt, biến thể attn_small đạt hiệu năng cao nhất, vượt trội hơn 60% so với mức nền...

5.3.2. Đánh giá Định tính (Qualitative Evaluation - Case Studies)

Chúng tôi lựa chọn ngẫu nhiên các mẫu câu từ tập kiểm thử (Test Set) để kiểm tra khả năng xử lý ngữ pháp, từ vựng và cấu trúc câu của mô hình tốt nhất (attn_small).

Nhóm 1: Các trường hợp dịch chính xác (Success Cases) Mô hình thể hiện khả năng nắm bắt ngữ pháp và trật tự từ rất tốt, đặc biệt với các câu có độ dài trung bình.

- Ví dụ 1 (Câu số 1):
    - *Source:* "A girl at the shore of a beach with a mountain in the distance."
    - *Reference:* "Une fille au bord d'une plage avec une montagne au loin."
    - *Predicted:* "Une fille au bord d' une plage avec une montagne au loin ."
    - Nhận xét: Bản dịch hoàn hảo. Mô hình xử lý đúng cụm giới từ phức tạp "at the shore of" thành "au bord d'".

- Ví dụ 2 (Câu số 12):
    - *Source:* "A man wearing sunglasses is riding a scooter."
    - *Predicted:* "Un homme portant des lunettes de soleil fait une scooter ."
    - Nhận xét: Mô hình dịch chính xác cụm danh từ "wearing sunglasses" -> "portant des lunettes de soleil". Mặc dù động từ "riding" được dịch thành "fait" (làm/đi) thay vì "conduit" (lái), nhưng về mặt ngữ nghĩa giao tiếp vẫn chấp nhận được.

5.3.3. Phân tích lỗi sai (Error Analysis)

Đây là phần quan trọng nhất để đánh giá độ "thông minh" thực sự của mô hình. Dựa trên 15 mẫu thử nghiệm cuối cùng, chúng tôi phân loại các lỗi sai thành 3 nhóm chính:

a. Lỗi ảo giác từ vựng và Dịch sai ngữ nghĩa (Lexical Hallucination & Semantic Mismatch)

Mô hình gặp khó khăn với các từ vựng hiếm (rare tokens) không xuất hiện nhiều trong tập huấn luyện, dẫn đến việc dự đoán một từ "gần đúng" hoặc sai hoàn toàn.

- Trường hợp 1 (Câu số 4):
    - *Src:* "...wearing a sunhat..." (mũ che nắng)
    - *Pred:* "...portant un bonnet..." (mũ len/mũ trùm)
    - *Phân tích:* Từ "sunhat" có thể ít xuất hiện trong tập Multi30k. Mô hình đã học được đây là một loại "mũ" nhưng chọn sai loại từ vựng cụ thể (Generalization Error).
- Trường hợp 2 (Câu số 6 - Lỗi nghiêm trọng):
    - *Src:* "...using a outdoors wok to cook."
    - *Pred:* "...utilise un objet dehors pour faire des huttes."
    - *Phân tích:* Từ "wok" (chảo xào) là từ mượn, rất hiếm. Mô hình không biết dịch nên đã thay thế bằng từ chung chung "objet" (đồ vật). Nghiêm trọng hơn, động từ "cook" bị dịch sai hoàn toàn thành "faire des huttes" (làm túp lều) -> Đây là hiện tượng Ảo giác (Hallucination), nơi Decoder tự sinh ra nội dung không có trong câu gốc.

b. Lỗi lặp từ (Repetition / Stuttering)

Đây là lỗi kinh điển trong các mô hình sinh chuỗi (Text Generation), khi Decoder bị kẹt trong một vòng lặp xác suất cục bộ.

- Trường hợp 1 (Câu số 9):
  - *Pred:* "...se met un objet en pierre pierre pierre ."
  - *Nguyên nhân:* Cơ chế Attention có thể đã tập trung vào cùng một vị trí từ nguồn (source token) trong nhiều bước thời gian liên tiếp, khiến Decoder sinh ra từ "pierre" lặp đi lặp lại.
- Trường hợp 2 (Câu số 13):
  - *Pred:* "...vêtus de jupes et pantalons pantalons noirs noirs..."
  - *Nguyên nhân:* Lỗi này thường xảy ra khi mô hình không chắc chắn về từ tiếp theo và trọng số Attention bị phân tán.

c. Lỗi mất thông tin trong câu dài (Long-Dependency Loss)

Mặc dù có Attention, mô hình vẫn gặp khó khăn với các câu có cấu trúc phức tạp hoặc mệnh đề quan hệ chồng chéo.

- Trường hợp (Câu số 15):
  - *Src:* "...one in red jersey trying to take ball from guy in white jersey."
  - *Pred:* "...un en maillot rouge essayant de contrer contrer le ballon en blanc."
  - *Phân tích:* Mô hình đã thất bại trong việc xác định đối tượng sở hữu chiếc áo trắng.
    - Đúng: "guy" (chàng trai) mặc áo trắng.
    - Sai: "ballon" (quả bóng) màu trắng.
  - *Lý giải:* Mối quan hệ giữa "guy" và "white jersey" nằm cách xa nhau trong câu. Dù Attention giúp "nhìn" lại, nhưng khả năng hiểu cú pháp (syntax awareness) của mô hình LSTM vẫn có giới hạn so với các kiến trúc hiện đại hơn như Transformer.

5.3.4. Tổng kết đánh giá

Nhìn chung, mô hình Seq2Seq + Luong Attention đã hoàn thành tốt nhiệm vụ dịch thuật trên bộ dữ liệu Multi30k với các câu đơn giản và trung bình.

- Điểm mạnh: Cấu trúc câu tiếng Pháp chuẩn xác, bắt đúng các cụm từ cố định (idioms), hiệu quả cao với câu < 20 từ.
- Điểm yếu: Vẫn còn hiện tượng lặp từ và gặp khó khăn với từ vựng hiếm (Rare words/OOV).
- Hướng giải quyết đề xuất:
    1. Sử dụng Beam Search thay vì Greedy Decoding để giảm thiểu lỗi lặp từ.
    2. Áp dụng Byte Pair Encoding (BPE) để xử lý tốt hơn các từ vựng hiếm (như "sunhat", "wok").
    3. Chuyển sang kiến trúc Transformer để xử lý phụ thuộc dài tốt hơn.

Dựa trên kết quả thực nghiệm, chúng em tổng hợp các trường hợp điển hình trong Bảng 5.4 dưới đây. Các từ ngữ bị lỗi hoặc đáng chú ý được **in đậm** để thuận tiện cho việc đối chiếu.

STT	Câu gốc (Source)	Câu máy dịch (Predicted)	Phân tích chi tiết (Analysis)
1	A girl at the shore of a beach with a mountain in the distance.	Une fille au bord d' une plage avec une montagne au loin .	**Dịch chính xác (Good Alignment):** Mô hình xử lý hoàn hảo cụm giới từ phức tạp *"at the shore of"* thành *"au bord d'"* và giữ đúng trật tự từ.
6	Six men sit in a field of **crops containing wooden crates**.	Six hommes sont assis dans un champ de **de bois en bois** .	**Mất thông tin (Omission):** Mô hình bỏ qua các từ *"crops"* (mùa màng), *"crates"* (thùng) và chỉ lặp lại từ *"bois"* (gỗ), làm mất ý nghĩa chi tiết của câu.
9	...one in red jersey trying to take ball from	...un en maillot rouge essayant de contrer contrer **le ballon en blanc** .	**Mất phụ thuộc dài (Long-term Dependency):** Mô hình nhầm lẫn đối tượng mặc áo trắng: *"guy"* (người) thành *"ballon"* (bóng). Lỗi

	guy in white jersey.		này do khoảng cách giữa các từ trong câu quá xa.
10	**Marathon runners** are racing on a city street...	**Des coureurs sont font du course** dans une rue...	**Lỗi cú pháp (Syntax):** *"Marathon runners"* dịch thành *"Des coureurs"* (chấp nhận được) nhưng cụm động từ *"sont font du course"* bị thừa từ và sai ngữ pháp.
15	...one in red jersey trying to take ball from **guy in white jersey**	...un en maillot rouge essayant de contrer contrer **le ballon en blanc** .	**Sai đối tượng (Dependency Error):** Nhầm *"guy"* (người) thành *"ballon"* (bóng) do khoảng cách từ xa.

Bảng 5.4:. Các trường hợp dịch điển hình và Phân tích lỗi chi tiết

Qua bảng phân tích trên, có thể thấy mô hình Seq2Seq + Attention hoạt động rất tốt với các câu có cấu trúc phổ biến (Ví dụ 1). Tuy nhiên, mô hình vẫn bộc lộ hạn chế ở 3 điểm chính:

1. Gặp khó khăn với từ vựng hiếm (như 'wok', 'sunhat'), thường dẫn đến việc đoán từ chung chung hoặc sai lệch ý nghĩa.
2. Hiện tượng lặp từ (Stuttering) vẫn xuất hiện ở các câu phức tạp (Ví dụ 5, 7).
3. Khả năng duy trì ngữ cảnh trong câu dài (Long dependencies) đã được cải thiện nhờ Attention nhưng chưa triệt để, dẫn đến việc xác định sai đối tượng sở hữu (Ví dụ 9).

Để có cái nhìn toàn diện về các lỗi sai, danh sách đầy đủ 15 câu kiểm thử ngẫu nhiên được trình bày chi tiết tại Phụ lục.

5.4. Tóm tắt chương

Chương 5 đã trình bày toàn diện quá trình đánh giá và phân tích hiệu năng của hệ thống dịch máy Anh - Pháp dựa trên kiến trúc Seq2Seq. Thông qua việc thiết lập và so sánh 06 cấu hình mô hình khác nhau (chia thành hai nhóm: Baseline No-Attention và Proposed Luong Attention), nghiên cứu đã thu được những kết quả thực nghiệm quan trọng cả về mặt định lượng lẫn định tính.

Về mặt định lượng, kết quả thực nghiệm khẳng định sự vượt trội tuyệt đối của cơ chế Luong Attention. Mọi biến thể tích hợp Attention đều cho thấy tốc độ hội tụ nhanh hơn và hàm mất mát thấp hơn đáng kể so với kiến trúc truyền thống. Đáng chú ý nhất, mô hình attn_small (Hidden size 256) đã đạt hiệu năng kỷ lục với chỉ số BLEU 0.4422, cải thiện tới 65% so với mô hình Baseline tốt nhất (noattn_medium: 0.2828).

Kết quả này minh chứng thuyết phục rằng việc phá bỏ "nút thắt cổ chai thông tin" (Information Bottleneck) bằng vector ngữ cảnh động là yếu tố then chốt để nâng cao chất lượng dịch thuật.

Một phát hiện quan trọng khác từ thực nghiệm là mối tương quan giữa kích thước mô hình và dữ liệu. Các mô hình Big (3 lớp LSTM) đều cho kết quả kém hơn các mô hình Small và Medium. Điều này chỉ ra rằng với tập dữ liệu quy mô nhỏ như Multi30k (~29.000 câu), việc gia tăng độ sâu của mạng lưới không mang lại hiệu quả mà còn dẫn đến hiện tượng quá khớp (Overfitting) và khó khăn trong tối ưu hóa. Vì vậy, kiến trúc gọn nhẹ kết hợp với cơ chế Attention được xác định là giải pháp tối ưu nhất.

Về mặt định tính, qua phân tích chi tiết các trường hợp kiểm thử (Case Studies), mô hình Attention thể hiện khả năng nắm bắt ngữ pháp và trật tự từ xuất sắc, xử lý tốt các cụm từ bị đảo ngược vị trí trong tiếng Pháp. Tuy nhiên, hệ thống vẫn bộc lộ một số hạn chế như khả năng xử lý từ vựng hiếm (Rare words) chưa tốt, hiện tượng lặp từ (Stuttering) và đôi khi mất ngữ cảnh trong các câu có phụ thuộc quá dài.

Những kết quả và phân tích này là cơ sở vững chắc để khẳng định tính đúng đắn của giải pháp đề xuất, đồng thời mở ra các định hướng cải tiến sẽ được trình bày trong chương kết luận.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Đồ án này đã tập trung nghiên cứu và hiện thực hóa hệ thống dịch máy tự động (Neural Machine Translation) cho cặp ngôn ngữ Anh - Pháp, dựa trên nền tảng kiến trúc Sequence-to-Sequence (Seq2Seq) kết hợp với cơ chế sự chú ý toàn cục (Luong Attention). Mục tiêu cốt lõi của nghiên cứu là giải quyết vấn đề "nút thắt cổ chai thông tin" (Information Bottleneck) vốn là hạn chế cố hữu của các mạng nơ-ron hồi quy truyền thống khi xử lý dữ liệu chuỗi dài. Thông qua quá trình thực nghiệm nghiêm ngặt trên 06 cấu hình mô hình khác nhau và đánh giá định lượng trên bộ dữ liệu chuẩn Multi30k, nghiên cứu đã đạt được những kết quả quan trọng sau:

Thứ nhất, khẳng định vai trò tiên quyết của cơ chế Attention: Kết quả thực nghiệm cho thấy sự vượt trội hoàn toàn của nhóm mô hình tích hợp Luong Attention so với nhóm Baseline. Cụ thể, mô hình tối ưu nhất (attn_small) đạt điểm BLEU 0.4422, cải thiện tới 65% so với mô hình không sử dụng Attention. Điều này minh chứng rằng việc cho phép Decoder truy xuất ngữ cảnh động (Dynamic Context) từ Encoder là giải pháp kỹ thuật hiệu quả nhất để nâng cao độ chính xác dịch thuật và khả năng định vị từ ngữ (Alignment).

Thứ hai, phát hiện về sự tương quan giữa dữ liệu và kích thước mô hình: Một đóng góp quan trọng của đồ án là việc chỉ ra nghịch lý "Lớn hơn không phải lúc nào cũng tốt hơn". Với tập dữ liệu quy mô nhỏ (~29.000 câu), các mô hình có độ sâu lớn (3 lớp LSTM) đã gặp hiện tượng quá khớp (Overfitting) và suy giảm hiệu năng. Ngược lại, kiến trúc gọn nhẹ (Hidden size 256) kết hợp với sự tinh vi của Attention lại đạt hiệu quả cao nhất. Đây là bài học kinh nghiệm quý giá trong việc tối ưu hóa tài nguyên tính toán cho các bài toán NLP ít dữ liệu (Low-resource settings).

Bên cạnh những thành công, hệ thống vẫn tồn tại một số hạn chế như: gặp khó khăn với các từ vựng hiếm (Rare words/OOV) dẫn đến hiện tượng dịch sai nghĩa, và đôi khi xuất hiện lỗi lặp từ (Repetition) trong các câu có cấu trúc phức tạp do hạn chế của thuật toán giải mã tham lam (Greedy Decoding).

Hướng phát triển

Dựa trên những phân tích về hạn chế của hệ thống hiện tại, nhóm nghiên cứu đề xuất các hướng cải tiến kỹ thuật cụ thể trong tương lai như sau:

1. Cải tiến phương pháp xử lý từ vựng: Thay vì sử dụng Tokenizer mức từ (Word-level) đơn thuần, hệ thống nên áp dụng kỹ thuật Byte Pair Encoding (BPE) hoặc Subword Tokenization. Phương pháp này sẽ giúp mô hình xử lý tốt hơn các từ hiếm (như "sunhat", "wok") bằng cách tách chúng thành các đơn vị nhỏ hơn, giảm thiểu hiện tượng từ lạ (Unknown tokens) và lỗi ảo giác ngữ nghĩa.
2. Tối ưu hóa thuật toán giải mã: Thay thế Greedy Decoding bằng Beam Search Decoding. Kỹ thuật này cho phép mô hình xem xét nhiều giả thuyết dịch song

song tại mỗi bước thời gian, từ đó chọn ra câu dịch có xác suất tổng thể cao nhất, giúp khắc phục hiệu quả lỗi lặp từ và cụt ý.

3. Nâng cấp kiến trúc mô hình: Hướng đi tất yếu tiếp theo là chuyển đổi từ kiến trúc hồi quy (RNN/LSTM) sang kiến trúc Transformer. Với cơ chế Self-Attention đa đầu (Multi-head Self-Attention), Transformer có khả năng xử lý song song và nắm bắt các phụ thuộc dài hạn tốt hơn nhiều so với LSTM, hứa hẹn mang lại bước nhảy vọt về điểm số BLEU.
4. Tăng cường dữ liệu (Data Augmentation): Áp dụng kỹ thuật dịch ngược (Back-translation) để tạo ra thêm dữ liệu huấn luyện nhân tạo, giúp mô hình "Deep Learning" thực sự phát huy được sức mạnh trên tập tham số lớn hơn.

Tóm lại, đồ án đã xây dựng thành công một nền tảng NMT vững chắc, chứng minh được hiệu quả của các giả thuyết khoa học đặt ra ban đầu, đồng thời mở ra những hướng đi tiềm năng để tiếp tục hoàn thiện hệ thống trong tương lai.

PHỤ LỤC

A. Mã Nguồn Chính

1.1.Chọn Data

```
DATA_DIR = "dataset/data/task1/raw"

train_src = f"{DATA_DIR}/train.en"
train_trg = f"{DATA_DIR}/train.fr"
val_src   = f"{DATA_DIR}/val.en"
val_trg   = f"{DATA_DIR}/val.fr"
test_src  = f"{DATA_DIR}/test_2016_flickr.en"
test_trg  = f"{DATA_DIR}/test_2016_flickr.fr"
```

1.2. Import thư viện cần thiết

```
!pip install spacy==3.7.2

!pip install https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-
3.7.1.tar.gz
!pip install https://github.com/explosion/spacy-
models/releases/download/fr_core_news_sm-3.7.1/fr_core_news_sm-
3.7.1.tar.gz

!pip install torch==2.2.2 torchtext==0.17.2 spacy==3.7.2 nltk

import spacy
import spacy.cli

spacy.load("en_core_web_sm")
# Check if fr_core_news_sm is installed, if not, install it
try:
    spacy.load("fr_core_news_sm")
except OSError:
    print("fr_core_news_sm model not found. Attempting to
download...")
    spacy.cli.download("fr_core_news_sm")
    spacy.load("fr_core_news_sm") # Try loading again after download
print("SpaCy models loaded successfully!")

import os
import random
import math
from collections import Counter
```

```
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torch.nn.utils.rnn import pad_sequence, pack_padded_sequence,
pad_packed_sequence

from torchtext.data.utils import get_tokenizer
from torchtext.vocab import build_vocab_from_iterator

import spacy
import nltk
from nltk.translate.bleu_score import sentence_bleu

nltk.download('punkt')

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

1.3. Build Vocab + Tokenizer (giới gạn 10000 từ)

```
# --- HÀM ĐỌC FILE ---
def read_lines(path):
    with open(path, encoding='utf-8') as f:
        return [l.strip() for l in f]       # Đọc file text và trả về
list các dòng, loại bỏ ký tự xuống dòng

# Đọc dữ liệu từ file ( biến train_src/train_trg là đường dẫn file
string)
train_en_lines = read_lines(train_src)
train_fr_lines = read_lines(train_trg)

# --- KHỞI TẠO TOKENIZER ---
# Tải mô hình ngôn ngữ của Spacy
spacy_en = spacy.load("en_core_web_sm")
spacy_fr = spacy.load("fr_core_news_sm")

# Tạo hàm tokenizer từ Spacy
# Input: "Hello world!" -> Output: ["Hello", "world", "!"]
tokenize_en = get_tokenizer("spacy", language="en_core_web_sm")
tokenize_fr = get_tokenizer("spacy", language="fr_core_news_sm")

# --- HÀM GENERATOR ---
def yield_tokens(lines, tokenizer):
    for line in lines:          #Hàm này duyệt qua từng dòng và tách
từ
```

```
        yield tokenizer(line)   #Dùng 'yield' để không phải load toàn
bộ token vào RAM cùng lúc

# --- ĐỊNH NGHĨA TOKEN ĐẶC BIỆT ---
# <unk>: Unknown (từ lạ không có trong từ điển)
# <pad>: Padding (làm đầy câu ngắn cho bằng câu dài)
# <sos>: Start of Sentence (báo hiệu bắt đầu câu)
# <eos>: End of Sentence (báo hiệu kết thúc câu)
specials = ["<unk>", "<pad>", "<sos>", "<eos>"]

# --- XÂY DỰNG TỪ ĐIỂN TIẾNG ANH ---
vocab_en = build_vocab_from_iterator(  #Hàm quan trọng dùng đến từ
xuất hiện, giữ lại 10000 từ xuất hiện nhiều nhất, các từ hiếm coi là
<unk>
    yield_tokens(train_en_lines, tokenize_en),
    specials=specials,              # Thêm 4 token đặc biệt vào đầu từ
điển (index 0,1,2,3)
    max_tokens=10_000               # Giới hạn 10.000 từ phổ biến nhất
theo yêu cầu
)

# Cài đặt mặc định: Nếu gặp từ lạ, tự động gán index của <unk> (=0)
# Nếu không có dòng này, code sẽ báo lỗi khi gặp từ mới ở tập Test
vocab_en.set_default_index(vocab_en["<unk>"])

# --- XÂY DỰNG TỪ ĐIỂN TIẾNG PHÁP ---
vocab_fr = build_vocab_from_iterator(
    yield_tokens(train_fr_lines, tokenize_fr),
    specials=specials,
    max_tokens=10_000
)
vocab_fr.set_default_index(vocab_fr["<unk>"])

# --- LƯU INDEX CÁC TOKEN ĐẶC BIỆT ĐỂ DÙNG SAU NÀY ---
PAD_IDX = vocab_fr["<pad>"]
SOS_IDX = vocab_fr["<sos>"]
EOS_IDX = vocab_fr["<eos>"]

# Kiểm tra kích thước từ điển
print(f"Số lượng từ vựng EN: {len(vocab_en)}")
print(f"Số lượng từ vựng FR: {len(vocab_fr)}")

import torch
import os

os.makedirs(save_folder, exist_ok=True)

torch.save({
```

```
    "vocab_en": vocab_en,
    "vocab_fr": vocab_fr,
    "PAD_IDX": PAD_IDX,
    "SOS_IDX": SOS_IDX,
    "EOS_IDX": EOS_IDX
}, f"{save_folder}/vocab.pth")

print(" Đã lưu vocab vào vocab.pt")
```

1.4. Dataset + DataLoader (padding + packing)

```
import torch
from torch.utils.data import Dataset, DataLoader
from torch.nn.utils.rnn import pad_sequence

class Multi30kDataset(Dataset):
    def __init__(self, src_path, trg_path, src_tok, trg_tok, v_src,
v_trg):
        self.src_lines = read_lines(src_path)
        self.trg_lines = read_lines(trg_path)

        self.src_tok = src_tok
        self.trg_tok = trg_tok
        self.v_src = v_src
        self.v_trg = v_trg

        self.data = []

        print(f" Đang tiền xử lý (Tokenizing) dữ liệu từ
{src_path}...")
        print("Việc này tốn khoảng 1-2 phút nhưng sẽ giúp train cực
nhanh sau đó.")

        # --- TỐI ƯU HÓA: Xử lý trước toàn bộ dữ liệu ---
        for src, trg in zip(self.src_lines, self.trg_lines):
            # Encode ngay lập tức và lưu vào list self.data
            src_tensor = self.encode(src, self.src_tok, self.v_src)
            trg_tensor = self.encode(trg, self.trg_tok, self.v_trg)
            self.data.append((src_tensor, trg_tensor))

        print(f" Đã xử lý xong {len(self.data)} cặp câu!")

    def __len__(self):
        return len(self.data)

    def encode(self, line, tokenizer, vocab):
```

```
        """Hàm phụ trợ: Chuyển text -> List các số (Indices)"""
        toks = tokenizer(line)
        ids = [vocab["<sos>"]] + [vocab[t] for t in toks] +
[vocab["<eos>"]]
        return torch.tensor(ids, dtype=torch.long)

    def __getitem__(self, idx):
        # Lấy cặp câu thứ idx
        # BÂY GIỜ CHỈ VIỆC LẤY RA DÙNG, KHÔNG CẦN TÍNH TOÁN NỮA
        return self.data[idx]
```

1.5. Hàm xử lý Batch

```
def collate_fn(batch):
    """
    Input batch: List các tuple [(src_1, trg_1), (src_2, trg_2), ...]
    """
    # 1. Tách riêng list nguồn (src) và đích (trg)
    # zip(*batch) giúp "xoay" list dọc thành ngang
    src_seqs, trg_seqs = zip(*batch)

    # 2. Tính độ dài thực tế của từng câu (trước khi padding)
    # Cái này cực quan trọng để dùng cho pack_padded_sequence sau này
    src_lengths = torch.tensor([len(s) for s in src_seqs])
    trg_lengths = torch.tensor([len(t) for t in trg_seqs])

    # 3. SẮP XẾP GIẢM DẦN THEO ĐỘ DÀI CÂU NGUỒN (Yêu cầu bắt buộc của
PyTorch RNN)
    src_lengths, sort_idx =
src_lengths.sort(descending=True)   #  Giải quyết vấn đề "Độ dài
chuỗi khác nhau"

    # Sắp xếp lại dữ liệu theo index đã sort
    src_seqs = [src_seqs[i] for i in sort_idx]
    trg_seqs = [trg_seqs[i] for i in sort_idx]
    trg_lengths = trg_lengths[sort_idx]

    # 4. Padding (Điền thêm số 0 vào câu ngắn cho bằng câu dài nhất
batch)
    # Mặc định pad_sequence trả về shape: [Seq_Len, Batch_Size]
(Chuẩn cho LSTM)
    # padding_value: Giá trị dùng để điền vào chỗ trống (thường là
index của <pad>)
    src_padded = pad_sequence(src_seqs,
padding_value=vocab_en["<pad>"])
    trg_padded = pad_sequence(trg_seqs,
padding_value=vocab_fr["<pad>"])
```

```
    return src_padded, src_lengths, trg_padded, trg_lengths
```

1.6. Encoder, Decoder, Attention, Seq2Seq (có packing)

1.6.1. Thư viện & Encoder - Encoder này sử dụng kĩ thuật pack_padded_sequence

```
import torch
import torch.nn as nn
import random
from torch.nn.utils.rnn import pack_padded_sequence,
pad_packed_sequence

class Encoder(nn.Module):
    def __init__(self, vocab_size, emb_dim, hid_dim, num_layers=2,
dropout=0.3):
        super().__init__()
        # Embedding layer: Chuyển index từ -> vector đặc trưng
        self.embedding = nn.Embedding(vocab_size, emb_dim)

        # LSTM layer
        # batch_first=False: Input shape sẽ là (Seq_Len, Batch,
Emb_Dim) - Chuẩn PyTorch
        self.lstm = nn.LSTM(emb_dim, hid_dim, num_layers,
dropout=dropout, batch_first=False)

        self.dropout = nn.Dropout(dropout)

    def forward(self, src, src_lengths):
        # src: [src_len, batch_size]
        # src_lengths: Tensor chứa độ dài thật của từng câu trong
batch

        embedded = self.dropout(self.embedding(src))  # [src_len,
batch_size, emb_dim]

        # --- KỸ THUẬT PACKING (QUAN TRỌNG) ---
        # Nén padded sequence lại để LSTM không phải tính toán trên
các token <pad> vô nghĩa.
        # enforce_sorted=True: Yêu cầu batch phải được sort giảm dần
(đã làm ở collate_fn)
        packed = pack_padded_sequence(embedded, src_lengths.cpu(),
enforce_sorted=True)

        # Đưa qua LSTM
        packed_outputs, (hidden, cell) = self.lstm(packed)
```

```
        # --- UNPACKING ---
        # Giải nén ra lại để dùng cho Attention (nếu có)
        outputs, _ = pad_packed_sequence(packed_outputs)
        # outputs: [src_len, batch_size, hid_dim] -> Chứa hidden
state của tất cả các bước
        # hidden, cell: [num_layers, batch_size, hid_dim] -> Trạng
thái cuối cùng (Context Vector)

        return outputs, hidden, cell
```

1.6.1. Decoder cơ bản - Mô hình này chỉ dùng Context Vector cố định (hidden, cell) từ Encoder truyền sang

```
class DecoderNoAttn(nn.Module):
    def __init__(self, vocab_size, emb_dim, hid_dim, num_layers=2,
dropout=0.3):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, emb_dim)
        self.lstm = nn.LSTM(emb_dim, hid_dim, num_layers,
dropout=dropout, batch_first=False)
        self.fc_out = nn.Linear(hid_dim, vocab_size) # Linear layer
để dự đoán từ tiếp theo
        self.dropout = nn.Dropout(dropout)

    def forward(self, input_tok, hidden, cell):
        # input_tok: [batch_size] (Là 1 từ tại thời điểm t)

        # Thêm chiều Seq_Len = 1 vì LSTM đòi hỏi input 3 chiều
        input_tok = input_tok.unsqueeze(0)   # [1, batch_size]

        embedded = self.dropout(self.embedding(input_tok)) # [1,
batch_size, emb_dim]

        # Decoder dùng hidden/cell của bước trước làm đầu vào cho
bước này
        output, (hidden, cell) = self.lstm(embedded, (hidden, cell))
        # output: [1, batch_size, hid_dim]

        # Bỏ chiều Seq_Len để đưa vào Linear Layer
        prediction = self.fc_out(output.squeeze(0))  # [batch_size,
vocab_size]

        return prediction, hidden, cell
```

1.6.3. Luong Attention + Decoder - Tính toán trọng số sự chú ý (Attention Weights) để model biết nên tập trung vào từ nào của câu gốc khi đang dịch.

```
class LuongAttention(nn.Module):
    def __init__(self, hid_dim):
        super().__init__()
        # Linear layer W để tính điểm tương đồng (General Luong
Attention: score = h_t * W * h_s)
        self.linear = nn.Linear(hid_dim, hid_dim, bias=False)

    def forward(self, hidden, enc_outputs):
        # hidden: [num_layers, batch, hid] -> Trạng thái hiện tại của
Decoder
        # enc_outputs: [src_len, batch, hid] -> Toàn bộ trạng thái
của Encoder (ký ức câu nguồn)

        # Lấy hidden của layer cuối cùng
        hidden = hidden[-1]  # [batch, hid]

        # Chiếu enc_outputs qua lớp Linear (Phần W trong công thức)
        enc_proj = self.linear(enc_outputs) # [src_len, batch, hid]

        # Tính Dot Product (Tích vô hướng) để ra điểm số (score)
        # hidden.unsqueeze(0) -> [1, batch, hid] để broadcast với
[src_len, batch, hid]
        scores = torch.sum(enc_proj * hidden.unsqueeze(0), dim=2)
        # shape scores: [src_len, batch]

        # Dùng Softmax để chuẩn hóa thành xác suất (tổng bằng 1)
        attn_weights = torch.softmax(scores, dim=0)
        # shape attn_weights: [src_len, batch]

        # Tính Context Vector: Tổng có trọng số của các trạng thái
Encoder
        # Nhân từng trọng số với output tương ứng rồi cộng lại
        context = torch.sum(attn_weights.unsqueeze(2) * enc_outputs,
dim=0)
        # shape context: [batch, hid]

        return context

class DecoderLuong(nn.Module):
    def __init__(self, vocab_size, emb_dim, hid_dim, num_layers=2,
dropout=0.3):
        super().__init__()
        self.embedding = nn.Embedding(vocab_size, emb_dim)
```

```
        self.lstm = nn.LSTM(emb_dim, hid_dim, num_layers,
dropout=dropout, batch_first=False)
        self.attn = LuongAttention(hid_dim)

        # Input của Linear layer cuối cùng sẽ lớn gấp đôi (do nối
Context + Output LSTM)
        self.fc_out = nn.Linear(hid_dim * 2, vocab_size)
        self.dropout = nn.Dropout(dropout)

    def forward(self, input_tok, hidden, cell, enc_outputs):
        input_tok = input_tok.unsqueeze(0)   # [1, batch]
        embedded = self.dropout(self.embedding(input_tok)) # [1,
batch, emb]

        # Bước 1: Chạy LSTM bình thường
        output, (hidden, cell) = self.lstm(embedded, (hidden, cell))
        # output: [1, batch, hid]

        # Bước 2: Tính Attention Context Vector
        context = self.attn(hidden, enc_outputs)
        # context: [batch, hid]

        # Bước 3: Nối (Concatenate) Output LSTM và Context Vector lại
với nhau
        rnn_output = output.squeeze(0) # [batch, hid]
        combined = torch.cat([rnn_output, context], dim=1)
        # combined: [batch, hid * 2] -> Kích thước tăng gấp đôi

        # Bước 4: Dự đoán từ
        pred = self.fc_out(combined)
        # pred: [batch, vocab_size]

        return pred, hidden, cell
```

1.6.4. Seq2Seq Wrapper

```
class Seq2Seq(nn.Module):
    def __init__(self, encoder, decoder, device):
        super().__init__()
        self.encoder = encoder
        self.decoder = decoder
        self.device = device

    def forward(self, src, src_lengths, trg,
teacher_forcing_ratio=0.5):
        # src: [src_len, batch_size]
        # trg: [trg_len, batch_size]
```

```
        batch_size = src.shape[1]
        max_len = trg.shape[0]
        vocab_size = self.decoder.fc_out.out_features

        # Tensor để lưu kết quả dự đoán
        outputs = torch.zeros(max_len, batch_size,
vocab_size).to(self.device)

        # 1. Mã hóa câu nguồn (Encoder)
        enc_outputs, hidden, cell = self.encoder(src, src_lengths)

        # Đầu vào đầu tiên cho Decoder là token <sos>
        input_tok = trg[0, :]

        # 2. Vòng lặp giải mã (Decoder)
        for t in range(1, max_len):
            # Kiểm tra xem đang dùng Model có Attention hay không
            if isinstance(self.decoder, DecoderNoAttn):
                output, hidden, cell = self.decoder(input_tok,
hidden, cell)
            else:
                output, hidden, cell = self.decoder(input_tok,
hidden, cell, enc_outputs)

            # Lưu dự đoán vào tensor outputs
            outputs[t] = output

            # --- TEACHER FORCING ---
            # Quyết định xem dùng từ vừa dự đoán hay dùng đáp án đúng
(ground truth) làm input tiếp theo
            best_guess = output.argmax(1)

            # Nếu random < 0.5 -> Dùng đáp án đúng (Teacher Forcing)
            # Ngược lại -> Dùng từ máy vừa đoán (Tự thân vận động)
            force = random.random() < teacher_forcing_ratio
            input_tok = trg[t] if force else best_guess

        return outputs
```

1.7. Hàm Train và Evaluate cho 1 Epoch Chạy hết 1 lượt dữ liệu epoch , tính toán sai số và cập nhật trọng số

```
# --- HÀM HUẤN LUYỆN 1 EPOCH ---
def train_epoch(model, loader, optimizer, criterion, clip=1.0):
    model.train() # Bật chế độ training (kích hoạt Dropout)
```

```
    epoch_loss = 0

    for src, src_lengths, trg, trg_lengths in loader:
        # Chuyển dữ liệu sang GPU
        src, src_lengths = src.to(device), src_lengths.to(device)
        trg = trg.to(device)

        optimizer.zero_grad() # Xóa sạch gradient cũ trước khi tính
mới

        # Forward pass: Đưa dữ liệu vào model
        # teacher_forcing_ratio=0.5:
        output = model(src, src_lengths, trg,
teacher_forcing_ratio=0.5)
        # shape output: [trg_len, batch_size, output_dim]

        # --- XỬ LÝ ĐỂ TÍNH LOSS ---
        # 1. Bỏ token <sos> ở đầu (index 0) vì model không cần dự
đoán nó
        output = output[1:].view(-1, output.shape[-1])
        # 2. Bỏ token <sos> ở câu đích tương ứng để so khớp
        trg = trg[1:].view(-1)
        # Lúc này output và trg đã được "ép phẳng" (flatten) thành 1
danh sách dài

        # Tính sai số (Loss)
        loss = criterion(output, trg)

        # Lan truyền ngược (Backpropagation)
        loss.backward()

        # --- GRADIENT CLIPPING (QUAN TRỌNG VỚI LSTM) ---
        # Cắt bớt gradient nếu nó quá lớn để tránh lỗi "Bùng nổ
Gradient" (Exploding Gradient)
        torch.nn.utils.clip_grad_norm_(model.parameters(), clip)

        optimizer.step() # Cập nhật trọng số
        epoch_loss += loss.item()

    return epoch_loss / len(loader) # Trả về loss trung bình

# --- HÀM ĐÁNH GIÁ (EVALUATE) ---
def evaluate(model, loader, criterion):
    model.eval() # Tắt Dropout để kết quả ổn định
    epoch_loss = 0

    with torch.no_grad(): # Tắt tính toán gradient để tiết kiệm RAM
và chạy nhanh
```

```
        for src, src_lengths, trg, trg_lengths in loader:
            src, src_lengths = src.to(device), src_lengths.to(device)
            trg = trg.to(device)

            # teacher_forcing_ratio=0.0: BẮT BUỘC.
            # Khi thi/test, model phải tự lực cánh sinh, không được
nhìn bài (đáp án).
            output = model(src, src_lengths, trg,
teacher_forcing_ratio=0.0)

            # Xử lý tương tự như train để tính loss
            output_dim = output.shape[-1]
            output = output[1:].view(-1, output_dim)
            trg = trg[1:].view(-1)

            loss = criterion(output, trg)
            epoch_loss += loss.item()

    return epoch_loss / len(loader)
```

1.8. Training Loop

```
import json
import os
import torch.optim as optim
import torch.nn as nn

def train_model_auto(model, model_name, n_epochs=15, patience=3):
    criterion = nn.CrossEntropyLoss(ignore_index=PAD_IDX)
    optimizer = optim.Adam(model.parameters(), lr=1e-3)

    train_losses, val_losses = [], []
    best_val = float("inf")
    best_path = f"{save_folder}/{model_name}_best.pth"
    loss_path = f"{save_folder}/{model_name}_loss.json"

    os.makedirs(save_folder, exist_ok=True)

    no_improve = 0

    for epoch in range(1, n_epochs + 1):

        train_loss = train_epoch(model, train_loader, optimizer,
criterion)
        val_loss   = evaluate(model, val_loader, criterion)

        train_losses.append(train_loss)
```

```
        val_losses.append(val_loss)

        # ======  LƯU LOSS MỖI EPOCH  ======
        with open(loss_path, "w") as f:
            json.dump({
                "train": train_losses,
                "val": val_losses
            }, f)

        print(f"[{model_name}] Epoch {epoch} | Train={train_loss:.3f}
| Val={val_loss:.3f}")

        # ====== LƯU BEST MODEL ======
        if val_loss < best_val:
            best_val = val_loss
            no_improve = 0
            torch.save(model.state_dict(), best_path)
            print("   → Saved best model!")
        else:
            no_improve += 1
            if no_improve >= patience:
                print("   Early stopping!")
                break

    return train_losses, val_losses, best_path
```

1.9 Chạy các models

```
INPUT_DIM = len(vocab_en)
OUTPUT_DIM = len(vocab_fr)

# ==========================
# TRAIN 3 MODEL NO ATTENTION
# ==========================

noattn_configs = [
    ("noattn_small", 256, 256, 2, 0.4),
    ("noattn_medium", 256, 512, 2, 0.5),
    ("noattn_big", 256, 512, 3, 0.55),
]

noattn_results = {}

for name, emb, hid, layers, drop in noattn_configs:
    print(f"\n TRAINING MODEL: {name}")
```

```
    encoder = Encoder(INPUT_DIM, emb, hid, layers, drop).to(device)
    decoder = DecoderNoAttn(OUTPUT_DIM, emb, hid, layers,
drop).to(device)
    model = Seq2Seq(encoder, decoder, device).to(device)

    tr_losses, val_losses, best_path = train_model_auto(model, name)
    noattn_results[name] = (tr_losses, val_losses, best_path)

print("\n Train xong 3 model No-Attention!")

# ==========================
# TRAIN 3 MODEL ATTENTION
# ==========================

attn_configs = [
    ("attn_small", 256, 256, 2, 0.4),
    ("attn_medium", 256, 512, 2, 0.5),
    ("attn_big", 256, 512, 3, 0.55),
]

attn_results = {}

for name, emb, hid, layers, drop in attn_configs:
    print(f"\n TRAINING MODEL: {name}")

    attention = LuongAttention(hid)
    encoder = Encoder(INPUT_DIM, emb, hid, layers, drop)
    decoder = DecoderLuong(OUTPUT_DIM, emb, hid, layers, drop)
    model = Seq2Seq(encoder, decoder, device).to(device)

    tr_losses, val_losses, best_path = train_model_auto(model, name)
    attn_results[name] = (tr_losses, val_losses, best_path)

print("\n Train xong 3 model Attention!")
```

❖ **Kết quả**

```
TRAINING MODEL: noattn_small
[noattn_small] Epoch 1 | Train=5.029 | Val=4.710
   → Saved best model!
[noattn_small] Epoch 2 | Train=4.149 | Val=4.458
   → Saved best model!
[noattn_small] Epoch 3 | Train=3.748 | Val=4.231
   → Saved best model!
[noattn_small] Epoch 4 | Train=3.494 | Val=4.094
   → Saved best model!
[noattn_small] Epoch 5 | Train=3.274 | Val=3.997
   → Saved best model!
[noattn_small] Epoch 6 | Train=3.132 | Val=3.913
   → Saved best model!
```

```
[noattn_small] Epoch 7 | Train=2.987 | Val=3.787
   → Saved best model!
[noattn_small] Epoch 8 | Train=2.859 | Val=3.830
[noattn_small] Epoch 9 | Train=2.766 | Val=3.707
   → Saved best model!
[noattn_small] Epoch 10 | Train=2.665 | Val=3.707
[noattn_small] Epoch 11 | Train=2.582 | Val=3.645
   → Saved best model!
[noattn_small] Epoch 12 | Train=2.506 | Val=3.639
   → Saved best model!
[noattn_small] Epoch 13 | Train=2.440 | Val=3.572
   → Saved best model!
[noattn_small] Epoch 14 | Train=2.379 | Val=3.595
[noattn_small] Epoch 15 | Train=2.307 | Val=3.544
   → Saved best model!

 TRAINING MODEL: noattn_medium
[noattn_medium] Epoch 1 | Train=4.801 | Val=4.594
   → Saved best model!
[noattn_medium] Epoch 2 | Train=3.914 | Val=4.276
   → Saved best model!
[noattn_medium] Epoch 3 | Train=3.492 | Val=4.054
   → Saved best model!
[noattn_medium] Epoch 4 | Train=3.205 | Val=3.904
   → Saved best model!
[noattn_medium] Epoch 5 | Train=2.959 | Val=3.823
   → Saved best model!
[noattn_medium] Epoch 6 | Train=2.783 | Val=3.689
   → Saved best model!
[noattn_medium] Epoch 7 | Train=2.621 | Val=3.690
[noattn_medium] Epoch 8 | Train=2.476 | Val=3.593
   → Saved best model!
[noattn_medium] Epoch 9 | Train=2.380 | Val=3.580
   → Saved best model!
[noattn_medium] Epoch 10 | Train=2.255 | Val=3.490
   → Saved best model!
[noattn_medium] Epoch 11 | Train=2.161 | Val=3.583
[noattn_medium] Epoch 12 | Train=2.074 | Val=3.568
[noattn_medium] Epoch 13 | Train=1.987 | Val=3.550
   Early stopping!

 TRAINING MODEL: noattn_big
[noattn_big] Epoch 1 | Train=5.080 | Val=4.752
   → Saved best model!
[noattn_big] Epoch 2 | Train=4.284 | Val=4.486
   → Saved best model!
[noattn_big] Epoch 3 | Train=3.865 | Val=4.257
   → Saved best model!
[noattn_big] Epoch 4 | Train=3.579 | Val=4.206
   → Saved best model!
[noattn_big] Epoch 5 | Train=3.368 | Val=4.089
   → Saved best model!
[noattn_big] Epoch 6 | Train=3.201 | Val=4.043
   → Saved best model!
[noattn_big] Epoch 7 | Train=3.053 | Val=3.892
   → Saved best model!
[noattn_big] Epoch 8 | Train=2.896 | Val=3.889
   → Saved best model!
```

```
[noattn_big] Epoch 9 | Train=2.776 | Val=3.843
   → Saved best model!
[noattn_big] Epoch 10 | Train=2.679 | Val=3.856
[noattn_big] Epoch 11 | Train=2.599 | Val=3.802
   → Saved best model!
[noattn_big] Epoch 12 | Train=2.492 | Val=3.812
[noattn_big] Epoch 13 | Train=2.427 | Val=3.741
   → Saved best model!
[noattn_big] Epoch 14 | Train=2.340 | Val=3.760
[noattn_big] Epoch 15 | Train=2.283 | Val=3.776

 Train xong 3 model No-Attention!
```

```
TRAINING MODEL: attn_small
[attn_small] Epoch 1 | Train=4.751 | Val=4.394
   → Saved best model!
[attn_small] Epoch 2 | Train=3.600 | Val=3.939
   → Saved best model!
[attn_small] Epoch 3 | Train=3.062 | Val=3.621
   → Saved best model!
[attn_small] Epoch 4 | Train=2.740 | Val=3.509
   → Saved best model!
[attn_small] Epoch 5 | Train=2.518 | Val=3.402
   → Saved best model!
[attn_small] Epoch 6 | Train=2.323 | Val=3.307
   → Saved best model!
[attn_small] Epoch 7 | Train=2.197 | Val=3.244
   → Saved best model!
[attn_small] Epoch 8 | Train=2.047 | Val=3.229
   → Saved best model!
[attn_small] Epoch 9 | Train=1.929 | Val=3.206
   → Saved best model!
[attn_small] Epoch 10 | Train=1.831 | Val=3.211
[attn_small] Epoch 11 | Train=1.766 | Val=3.171
   → Saved best model!
[attn_small] Epoch 12 | Train=1.701 | Val=3.167
   → Saved best model!
[attn_small] Epoch 13 | Train=1.640 | Val=3.163
   → Saved best model!
[attn_small] Epoch 14 | Train=1.567 | Val=3.143
   → Saved best model!
[attn_small] Epoch 15 | Train=1.514 | Val=3.104
   → Saved best model!

 TRAINING MODEL: attn_medium
[attn_medium] Epoch 1 | Train=4.636 | Val=4.504
   → Saved best model!
[attn_medium] Epoch 2 | Train=3.625 | Val=4.058
   → Saved best model!
[attn_medium] Epoch 3 | Train=3.095 | Val=3.694
   → Saved best model!
[attn_medium] Epoch 4 | Train=2.756 | Val=3.470
   → Saved best model!
[attn_medium] Epoch 5 | Train=2.488 | Val=3.480
[attn_medium] Epoch 6 | Train=2.297 | Val=3.395
   → Saved best model!
[attn_medium] Epoch 7 | Train=2.131 | Val=3.370
   → Saved best model!
[attn_medium] Epoch 8 | Train=2.014 | Val=3.315
```

```
   → Saved best model!
[attn_medium] Epoch 9 | Train=1.885 | Val=3.296
   → Saved best model!
[attn_medium] Epoch 10 | Train=1.785 | Val=3.298
[attn_medium] Epoch 11 | Train=1.692 | Val=3.311
[attn_medium] Epoch 12 | Train=1.640 | Val=3.288
   → Saved best model!
[attn_medium] Epoch 13 | Train=1.551 | Val=3.326
[attn_medium] Epoch 14 | Train=1.500 | Val=3.252
   → Saved best model!
[attn_medium] Epoch 15 | Train=1.461 | Val=3.287

 TRAINING MODEL: attn_big
[attn_big] Epoch 1 | Train=4.835 | Val=4.669
   → Saved best model!
[attn_big] Epoch 2 | Train=4.018 | Val=4.260
   → Saved best model!
[attn_big] Epoch 3 | Train=3.558 | Val=4.008
   → Saved best model!
[attn_big] Epoch 4 | Train=3.210 | Val=3.876
   → Saved best model!
[attn_big] Epoch 5 | Train=2.947 | Val=3.767
   → Saved best model!
[attn_big] Epoch 6 | Train=2.728 | Val=3.662
   → Saved best model!
[attn_big] Epoch 7 | Train=2.546 | Val=3.634
   → Saved best model!
[attn_big] Epoch 8 | Train=2.395 | Val=3.678
[attn_big] Epoch 9 | Train=2.258 | Val=3.560
   → Saved best model!
[attn_big] Epoch 10 | Train=2.123 | Val=3.533
   → Saved best model!
[attn_big] Epoch 11 | Train=2.036 | Val=3.599
[attn_big] Epoch 12 | Train=1.929 | Val=3.596
[attn_big] Epoch 13 | Train=1.821 | Val=3.562
   Early stopping!

 Train xong 3 model Attention!
```

1.10. Tính điểm BLEU

1. **Hàm dịch câu**
2. **Hàm lấy Reference & Hypothesis**
3. **Tính BLEU cho 6 mô hình**

```python
from nltk.translate.bleu_score import corpus_bleu, SmoothingFunction
smooth = SmoothingFunction().method4

bleu_scores = {}

# ---- NO ATTENTION ----
for (name, emb, hid, layers, drop) in noattn_configs:
    print(f"\n BLEU - {name}")
```

```
    _, _, path = noattn_results[name]
    model = build_model(emb, hid, layers, drop, attention=False)
    model.load_state_dict(torch.load(path, map_location=device))
    model.eval()

    refs, hyps = get_refs_hyps_bleu(model, max_sentences=1000)
    bleu = corpus_bleu(refs, hyps, smoothing_function=smooth)

    bleu_scores[name] = bleu
    print(f" BLEU = {bleu:.4f}")

# ---- ATTENTION ----
for (name, emb, hid, layers, drop) in attn_configs:
    print(f"\n BLEU - {name}")

    _, _, path = attn_results[name]
    model = build_model(emb, hid, layers, drop, attention=True)
    model.load_state_dict(torch.load(path, map_location=device))
    model.eval()

    refs, hyps = get_refs_hyps_bleu(model, max_sentences=1000)
    bleu = corpus_bleu(refs, hyps, smoothing_function=smooth)

    bleu_scores[name] = bleu
    print(f" BLEU = {bleu:.4f}")
```

❖ **Kết quả:**

```
==================================================
        BLEU SCORE - 6 MODELS
==================================================
Model           |   BLEU
--------------------------------------------------
noattn_small    | 0.2671
noattn_medium   | 0.2828
noattn_big      | 0.2279
attn_small      | 0.4422
attn_medium     | 0.4068
attn_big        | 0.3071
==================================================
```

1.11. Vẽ biểu đồ Train/Val Loss

```
import matplotlib.pyplot as plt
import numpy as np

def plot_loss_single(train_losses, val_losses, model_name):
    plt.figure(figsize=(10,6))

    epochs = np.arange(1, len(train_losses) + 1)

    plt.style.use("default")
    ax = plt.gca()
    ax.set_facecolor("white")

    # Plot 2 curves
    plt.plot(epochs, train_losses, marker="o", markersize=5,
             linewidth=2.3, color="#1f77b4", label="Train Loss")
    plt.plot(epochs, val_losses, marker="s", markersize=5,
             linewidth=2.3, color="#2ca02c", label="Validation Loss")

    # Best point
    best_idx = np.argmin(val_losses)
    best_val = val_losses[best_idx]

    plt.scatter(best_idx+1, best_val, color="red", s=140,
edgecolor="black", zorder=10)

    plt.annotate(f"Best Val = {best_val:.3f}\nEpoch {best_idx+1}",
                 xy=(best_idx+1, best_val),
                 xytext=(best_idx+1.3, best_val+0.15),
                 fontsize=12, fontweight="bold",
                 color="red", arrowprops=dict(arrowstyle="->",
color="red"))

    plt.title(f"Loss Curve - {model_name}", fontsize=16,
fontweight="bold")
    plt.xlabel("Epoch", fontsize=13)
    plt.ylabel("Loss", fontsize=13)
    plt.grid(True, linestyle="--", alpha=0.6)
    plt.legend(fontsize=13)
    plt.tight_layout()
    plt.show()

# Vẽ 3 mô hình No Attention
for name, (tr, vl, _) in noattn_results.items():
    plot_loss_single(tr, vl, f"{name} - No Attention")

# Vẽ 3 mô hình Attention
for name, (tr, vl, _) in attn_results.items():
    plot_loss_single(tr, vl, f"{name} - Attention")
```

Hình kết quả đã phân tích ở chương 5

B. Kết quả dịch máy trên 15 câu từ tập Test

STT	Câu gốc (Source)	Câu máy dịch (Predicted)	Phân tích chi tiết (Analysis)
1	A girl at the shore of a beach with a mountain in the distance.	Une fille au bord d' une plage avec une montagne au loin .	**Chính xác (Correct):** Dịch trôi chảy, xử lý tốt cụm giới từ "at the shore of".
2	An older man is playing a video arcade game.	Un vieil homme joue du jeu vidéo .	**Chấp nhận được:** Dịch thoát ý. "Video arcade game" được rút gọn thành "jeu vidéo".
3	Some children are outside playing in the dirt where two trees are.	Des enfants sont dehors , jouant dans la terre avec deux arbres .	**Chính xác:** Cấu trúc câu và từ vựng đều đúng.
4	Asian woman wearing a **sunhat** while riding a bike.	Une femme asiatique portant un **bonnet** en train de faire du vélo .	**Lỗi từ vựng (Lexical Mismatch):** *"Sunhat"* (mũ nắng) bị dịch thành *"bonnet"* (mũ len/mũ trùm).
5	**Marathon runners** are racing on a city street...	**Des coureurs sont font du course** dans une rue...	**Lỗi ngữ pháp:** Cụm động từ *"sont font du course"* bị thừa từ "sont" và sai cấu trúc.
6	A hispanic woman is using a outdoors **wok** to **cook**.	Une femme femme utilise un **objet** dehors pour **faire des huttes** .	**Ảo giác (Hallucination):** Lỗi nghiêm trọng. *"Wok"* -> *"objet"*, *"cook"* -> *"faire des huttes"* (làm lều).
7	A man in a white shirt and apron **cuts up** a bird.	Un homme en chemise blanche et tablier **ramasse** un oiseau .	**Sai nghĩa động từ:** *"Cuts up"* (chặt/cắt) bị dịch nhầm thành *"ramasse"* (nhặt lên).
8	This is a man dressed in yellow	C' est un homme en jaune tient le	**Sai từ vựng:** *"Reign"* (dây cương - lẽ ra là rein) bị dịch

	holding the **reign** of a brown horse	**contenu** d' un cheval marron	thành *"contenu"* (nội dung/đồ chứa).
9	A brown dog picks up a twig from a stone **surface**.	Un chien brun se met un objet en **pierre pierre pierre** .	**Lỗi lặp (Stuttering):** Mô hình bị kẹt (loop) ở từ *"pierre"* do Attention bị nhiễu.
10	Six men sit in a field of **crops containing wooden crates**.	Six hommes sont assis dans un champ de **de bois en bois** .	**Mất thông tin (Omission):** Bỏ qua hoàn toàn *"crops"* và *"crates"*, chỉ lặp lại từ *"bois"* (gỗ).
11	...holds a bat **behind his head** with a baseball mounted **in front of him**.	...brandit une batte **derrière sa tête** avec un batte de baseball **devant lui** .	**Khá tốt:** Dù câu dài và phức tạp, mô hình vẫn xác định đúng vị trí trước/sau.
12	A man wearing sunglasses is riding a scooter.	Un homme portant des lunettes de soleil fait une scooter .	**Chấp nhận được:** *"Riding"* dịch là *"fait"* (làm/đi) thay vì *"conduit"*, nhưng vẫn hiểu được.
13	...wearing striped sweaters and **black pants** tussle outdoors...	...vêtus de jupes et **pantalons pantalons noirs noirs** sont debout...	**Lỗi lặp & Thêm từ:** Lặp *"pantalons"*, *"noirs"* và tự thêm *"jupes"* (váy) không có trong gốc.
14	A boy grabs his leg as he **jumps** in the air.	Un garçon **se sa jambe** tandis qu' il saute en l' air .	**Sai ngữ pháp:** *"Se sa jambe"* là cụm vô nghĩa (đúng ra phải là *saisit sa jambe*).
15	...one in red jersey trying to take ball from **guy in white jersey**.	...un en maillot rouge essayant de contrer contrer **le ballon en blanc** .	**Sai đối tượng (Dependency Error):** Nhầm *"guy"* (người) thành *"ballon"* (bóng) do khoảng cách từ xa.

2. Nhận xét cá nhân về khó khăn gặp phải

Điều đọng lại sâu sắc nhất sau quá trình thực hiện đồ án chính là bài học về sự tỉnh táo trước kết quả thực nghiệm. Nhóm đã từng rơi vào trạng thái lạc quan thái quá khi nhận được điểm số BLEU cao do lỗi kỹ thuật trong khâu đánh giá. Việc phát hiện sai sót và chấp nhận quay về với những con số thực tế (tuy thấp hơn nhưng trung thực) là một bước ngoặt giúp nhóm hiểu rõ giá trị của sự chặt chẽ trong khoa học.

Hơn nữa, việc chứng kiến mô hình kiến trúc lớn hoạt động kém hiệu quả hơn mô hình nhỏ đã phá vỡ tư duy mặc định ban đầu của chúng tôi rằng 'càng lớn càng tốt'. Đây là minh chứng sống động nhất giúp nhóm hiểu rằng: thấu hiểu dữ liệu quan trọng hơn nhiều so với việc áp dụng các thuật toán phức tạp một cách máy móc.

3. Đường dẫn mã nguồn và best_model.pth

Link best_model.pth và code colab (bằng drive công khai)

STT	Tên	Đường dẫn
1	Best_model.pth	https://drive.google.com/file/d/1ldSVbYnAqB5YzJYMcc-PX3OuilLdBowC/view?usp=sharing
2	NLP_Finale	https://colab.research.google.com/drive/1KiG2XAgBhTuxinXOugPDHxrMkWPP5eAE?usp=sharing

TÀI LIỆU THAM KHẢO

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," Advances in Neural Information Processing Systems (NeurIPS), vol. 27, 2014.

[2] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1412–1421, 2015.

[3] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in Proc. ACL, 2002, pp. 311–318.