

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**PHẠM ĐÌNH VƯƠNG**

**LÊ HOÀNG TRUNG**

**DỊCH MÁY BĂNG MÔ HÌNH HỌC  
LSTM-ATTENTION**

Chuyên ngành: Khoa Học Máy Tính

Mã số chuyên ngành: 60 48 01

**KHÓA LUẬN TỐT NGHIỆP: KHOA HỌC MÁY TÍNH**

NGƯỜI HƯỚNG DẪN KHOA HỌC:

ThS Trần Trung Kiên

Tp. Hồ Chí Minh, Năm 2018

# LỜI CẢM ƠN

Trước tiên, em xin gửi lời tri ân sâu sắc đến Thầy Lê Hoài Bắc. Thầy đã rất tận tâm, nhiệt tình hướng dẫn và chỉ bảo em trong suốt quá trình thực hiện luận văn. Không có sự quan tâm, theo dõi chặt chẽ của Thầy chắc chắn em không thể hoàn thành luận văn này.

Em xin chân thành cảm ơn quý Thầy Cô khoa Công Nghệ Thông Tin - trường đại học Khoa Học Tự Nhiên, những người đã ân cần giảng dạy, xây dựng cho em một nền tảng kiến thức vững chắc.

Con xin cảm ơn ba mẹ đã sinh thành, nuôi dưỡng, và dạy dỗ để con có được thành quả như ngày hôm nay. Ba mẹ luôn là nguồn động viên, nguồn sức mạnh hết sức lớn lao mỗi khi con gặp khó khăn trong cuộc sống.

TP. Hồ Chí Minh, 3/2014

*Trần Trung Kiên*

# MỤC LỤC

<b>LỜI CẢM ƠN</b>	<b>i</b>
<b>MỤC LỤC</b>	<b>ii</b>
<b>DANH MỤC HÌNH ẢNH</b>	<b>iv</b>
<b>DANH MỤC BẢNG</b>	<b>v</b>
<b>Chương 1 Giới thiệu</b>	<b>1</b>
<b>Chương 2 Kiến Thức Nền Tảng</b>	<b>11</b>
2.1 Mạng nơ-ron hồi quy (Recurrent neural network)	11
2.1.1 Huấn luyện mạng nơ-ron hồi quy	15
2.1.2 Thách thức trong việc học các phụ thuộc dài hạn	19
2.2 Long short-term memory	20
2.3 Mô hình ngôn ngữ	23
<b>Chương 3 Dịch máy bằng mô hình học LSTM-Attention</b>	<b>24</b>
3.1 LSTM với kiến trúc Bộ mã hóa-Bộ giải mã	25
3.1.1 Bộ mã hóa	26
3.1.2 Bộ giải mã	27
3.2 Cơ chế Attention	30
3.3 Attention Toàn cục	33
3.4 Attention Cục bộ	35
3.5 Phương pháp Input feeding	39
3.6 Kỹ thuật thay thế từ hiếm	41

<b>Chương 4</b>	<b>Các Kết Quả Thực Nghiệm</b>	<b>44</b>
4.1	Các thiết lập thực nghiệm . . . . .	44
4.2	Kết quả thực nghiệm . . . . .	46
4.2.1	Không sử dụng Attention và có sử dụng Attention . . . . .	46
4.2.2	Giữa các mô hình Attention với nhau . . . . .	47
4.2.3	Phân tích đường cong học . . . . .	48
4.2.4	So sánh với kết quả của bài báo . . . . .	49
4.2.5	Chất lượng giống hàng của mô hình . . . . .	51
<b>Chương 5</b>	<b>Kết Luận Và Hướng Phát Triển</b>	<b>52</b>
5.1	Kết luận . . . . .	52
5.2	Hướng phát triển . . . . .	53
<b>TÀI LIỆU THAM KHẢO</b>		<b>55</b>

# DANH MỤC HÌNH ẢNH

1.1	Lịch sử tóm tắt của dịch máy . . . . .	3
1.2	Ba phương pháp dịch máy dựa trên luật . . . . .	4
1.3	Ví dụ về tập các câu song song trong hai ngôn ngữ . . . . .	7
1.4	Ví dụ về Kiến trúc <i>bộ mã hóa - bộ giải mã</i> trong dịch máy nơ-ron . . .	8
1.5	Kiến trúc bộ mã hóa - bộ giải mã được xây dựng trên mạng nơ-ron hồi quy . . . . .	8
1.6	Cơ chế Attention trong dịch máy nơ-ron . . . . .	9
2.1	Mô hình RNN với kết nối vòng . . . . .	13
2.2	Mô hình RNN dạng dàn trải . . . . .	14
2.3	Một "LSTM cell" . . . . .	21
3.1	Minh họa kiến trúc Bộ mã hóa-Bộ giải mã được sử dụng trong khóa luận. . . . .	26
3.2	Minh họa phương pháp teacher forcing. . . . .	28
3.3	Minh họa cơ chế Attention. . . . .	31
3.4	Minh họa cơ chế Attention Toàn cục. . . . .	34
3.5	Minh họa cơ chế Attention Cục bộ. . . . .	37
3.6	Minh họa cơ chế Attention Cục bộ. . . . .	40
3.7	Minh họa kỹ thuật thay thế từ hiếm. . . . .	43
4.1	Đường cong học của các mô hình . . . . .	50

# DANH MỤC BẢNG

4.1	So sánh giữa mô hình sử dụng cơ chế Attention và mô hình không sử dụng cơ chế Attention. . . . .	47
4.2	So sánh giữa các mô hình Attention. . . . .	47
4.3	Kết quả của các mô hình trên tập dữ liệu WMT'14 English-German. .	48

## Chương 1

# Giới thiệu

Nhờ vào những cải cách trong giao thông và cơ sở hạ tầng viễn thông mà giờ đây toàn cầu hóa đang trở nên gần với chúng ta hơn bao giờ hết. Trong xu hướng đó nhu cầu giao tiếp và thông hiểu giữa những nền văn hóa là không thể thiếu. Tuy nhiên, những nền văn hóa khác nhau thường kèm theo đó là sự khác biệt về ngôn ngữ, là một trong những trở ngại lớn nhất của sự giao tiếp. Một người phải mất rất nhiều thời gian để thành thạo một ngôn ngữ không phải là tiếng mẹ đẻ, và không thể nào học được nhiều ngôn ngữ cùng lúc. Cho nên, việc phát triển một công cụ để giải quyết vấn đề này là tất yếu. Một trong những công cụ như vậy là *Dịch máy*.

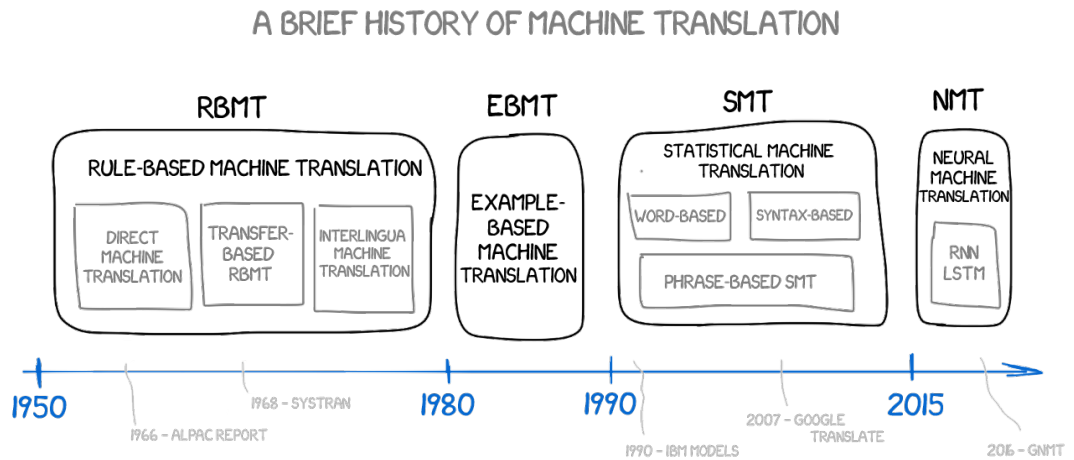
*Dịch máy* là quá trình chuyển đổi văn bản/tiếng nói từ ngôn ngữ này sang dạng tương ứng của nó trong một ngôn ngữ khác, được thực hiện bởi một chương trình máy tính nhằm mục đích cung cấp bản dịch tốt nhất mà không cần sự trợ giúp của con người. Trong khóa luận này, chúng tôi tập trung nghiên cứu dịch máy trên dữ liệu văn bản.

Dịch máy có một quá trình lịch sử lâu dài. Từ thế kỷ XVII, đã có những ý tưởng về việc cơ giới hóa quá trình dịch thuật. Tuy nhiên, đến thế kỷ XX, những nghiên cứu về dịch máy mới thật sự bắt đầu. Vào những năm 1930, Georges Artsrouni người Pháp và Petr Troyanskii người Nga đã nộp bằng sáng chế cho công trình có tên "máy dịch" của riêng họ. Trong số hai người, công trình của Troyanskii có ý nghĩa hơn. Nó đề xuất không chỉ một phương pháp cho bộ từ điển tự động, mà còn là lược đồ cho việc mã hóa các vai trò ngữ pháp song ngữ và một phác thảo về cách phân tích và tổng hợp có thể hoạt động. Tuy nhiên, những ý tưởng của Troyanskii đã không được biết đến cho đến cuối những năm 1950. Trước đó, máy tính đã được phát minh.

Những nỗ lực xây dựng hệ thống dịch máy bắt đầu ngay sau khi máy tính ra đời. Có thể nói, chiến tranh và sự thù địch giữa các quốc gia là động lực lớn nhất cho dịch máy thời bấy giờ. Trong Thế chiến thứ II, máy tính đã được quân đội Anh sử dụng trong việc giải mã các thông điệp được mã hóa của quân Đức. Việc làm này có thể coi là một dạng ẩn dụ của dịch máy khi người ta cố gắng dịch từ tiếng Đức được mã hóa sang tiếng Anh. Trong thời kỳ chiến tranh lạnh, vào tháng 7/1949, Warren Weaver, người được xem là nhà tiên phong trong lĩnh vực dịch máy, đã viết một bản ghi nhớ đưa ra các đề xuất khác nhau của ông trong lĩnh vực này [7]. Những đề xuất đó dựa trên thành công của máy phá mã, sự phát triển của lý thuyết thông tin bởi Claude Shannon và suy đoán về các nguyên tắc phổ quát cơ bản của ngôn ngữ. Trong vòng một năm, một vài nghiên cứu về dịch máy đã bắt đầu tại nhiều trường đại học của Mỹ. Vào ngày 7/1/1954, tại trụ sở chính của IBM ở New York, thử nghiệm Georgetown-IBM được tiến hành. Máy tính IBM 701 đã tự động dịch 49 câu tiếng Nga sang tiếng Anh lần đầu tiên trong lịch sử chỉ sử dụng 250 từ vựng và sáu luật ngữ pháp [7]. Thử nghiệm này được xem như là một thành công và mở ra kỉ nguyên cho những nghiên cứu với kinh phí lớn về dịch máy ở Hoa Kỳ. Ở Liên Xô những thí nghiệm tương tự cũng được thực hiện không lâu sau đó.

Trong một thập kỷ tiếp theo, nhiều nhóm nghiên cứu về dịch máy được thành lập. Một số nhóm chấp nhận phương pháp thử và sai, thường dựa trên thống kê với mục tiêu là một hệ thống dịch máy có thể hoạt động ngay lập tức, tiêu biểu như: nhóm nghiên cứu tại đại học Washington (và sau này là IBM) với hệ thống dịch Nga-Anh cho Không quân Hoa Kỳ, những nghiên cứu tại viện Cơ học Chính xác ở Liên Xô và Phòng thí nghiệm Vật lý Quốc gia ở Anh. Trong khi một số khác hướng đến giải pháp lâu dài với hướng tiếp cận lý thuyết bao gồm cả những vấn đề liên quan đến ngôn ngữ cơ bản như nhóm nghiên cứu tại Trung tâm nghiên cứu lý thuyết tại MIT, Đại học Havard và Đơn vị nghiên cứu ngôn ngữ Đại học Cambridge. Những nghiên cứu trong giai đoạn này có tầm quan trọng và ảnh hưởng lâu dài không chỉ cho Dịch máy mà còn cho nhiều ngành khác như Ngôn ngữ học tính toán, Trí tuệ nhân tạo - cụ thể là việc phát triển các từ điển tự động và kỹ thuật phân tích cú pháp. Nhiều nhóm nghiên cứu đã đóng góp đáng kể cho việc phát triển lý thuyết ngôn ngữ. Tuy nhiên, mục tiêu cơ bản của dịch máy là xây dựng hệ thống có khả năng tạo ra bản dịch tốt lại không đạt được dẫn đến một kết quả là vào năm 1966 bản báo cáo từ Ủy ban tư vấn xử lý





Hình 1.1: Lịch sử tóm tắt của dịch máy, nguồn ảnh: Ilya Pestov trong blog [A history of machine translation from the Cold War to deep learning](#)

ngôn ngữ tự động (Automatic Language Processing Advisory) của Hoa Kỳ, tuyên bố rằng dịch máy là đắt tiền, không chính xác và không mang lại kết quả hứa hẹn [7]. Thay vào đó, họ đề nghị tập trung vào phát triển các từ điển, điều này đã loại bỏ các nhà nghiên cứu Mỹ ra khỏi cuộc đua trong gần một thập kỷ.

Từ đó đến nay, đã có nhiều hướng tiếp cận đã được sử dụng trong dịch máy với mục tiêu tạo ra bản dịch có độ chính xác cao và giảm thiểu công sức của con người. Trong những năm đầu tiên, để tạo ra bản dịch tốt, các phương pháp thời bấy giờ đều hỏi hời những lý thuyết tinh vi về ngôn ngữ học. Hầu hết những hệ thống dịch máy trước những năm 1980 đều là *dịch máy dựa trên luật* (*Rule-based machine translation - RBMT*). Những hệ thống này thường bao gồm:

- Một từ điển song ngữ (ví dụ từ điển Anh - Đức)
- Một tập các luật ngữ pháp (ví dụ trong tiếng Đức, từ kết thúc bằng -heit, -keit, -ung là những từ mang giống cái)

Có ba cách tiếp cận khác nhau theo phương pháp dịch máy dựa trên luật. Bao gồm phương pháp dịch máy trực tiếp, dịch máy chuyển giao và dịch máy ngôn ngữ phổ quát. Mặc dù cả ba đều thuộc về RBMT, tuy nhiên chúng khác nhau về độ sâu của đại



Hình 1.2: Kim tự tháp của Bernard Vauquois thể hiện ba phương pháp dịch máy dựa luật theo độ sâu của đại diện trung gian. Bắt đầu từ dịch máy trực tiếp đến dịch máy chuyển dịch và trên cùng là dịch máy ngôn ngữ phổ quát (Nguồn: [http://en.wikipedia.org/wiki/Machine\\_translation](http://en.wikipedia.org/wiki/Machine_translation))

diện trung gian. Sự khác biệt này được thể hiện qua kim tự tháp Vauquois, minh họa trên hình 1.2

*Dịch máy trực tiếp* (Direct machine translation - DMT): Đây là phương pháp đơn giản nhất của dịch máy. DMT không dùng bất cứ dạng đại diện nào của ngôn ngữ nguồn, nó chia câu thành các từ, dịch chúng bằng một từ điển song ngữ. Sau đó, dựa trên các luật mà những nhà ngôn ngữ học đã xây dựng, nó chỉnh sửa để bản dịch trở nên đúng cú pháp và ít nhiều đúng về mặt phát âm.

*Dịch máy ngôn ngữ phổ quát* (Interlingual machine translation - IMT): Trong phương pháp này, câu nguồn được chuyển thành biểu diễn trung gian và biểu diễn này được thống nhất cho tất cả ngôn ngữ trên thế giới (interlingua). Tiếp theo, dạng đại diện này sẽ được chuyển đổi sang bất kỳ ngôn ngữ đích nào. Một trong những ưu điểm chính của hệ thống này là tính mở rộng của nó khi số lượng ngôn ngữ cần dịch tăng lên. Mặc dù trên lý thuyết, phương pháp này trông rất hoàn hảo. Nhưng trong thực tế, thật khó để tạo được một ngôn ngữ phổ quát như vậy.

*Dịch máy chuyển giao* (Transfer-based machine translation - TMT): dịch máy chuyển giao tương tự như dịch máy ngôn ngữ đại diện ở chỗ, nó cũng tạo ra bản dịch từ biểu diễn trung gian mô phỏng ý nghĩa của câu gốc. Tuy nhiên, không giống như IMT, TMT phụ thuộc một phần vào cặp ngôn ngữ mà nó tham gia vào quá trình dịch. Trên cơ sở sự khác biệt về cấu trúc của ngôn ngữ nguồn và ngôn ngữ đích, một

hệ thống TMT có thể được chia thành ba giai đoạn: i) Phân tích, ii) Chuyển giao, iii) Tạo ra bản dịch. Trong giai đoạn đầu tiên, trình phân tích cú pháp ở ngôn ngữ nguồn được sử dụng để tạo ra biểu diễn cú pháp của câu nguồn. Trong giai đoạn tiếp theo, kết quả của phân tích cú pháp được chuyển đổi thành biểu diễn tương đương trong ngôn ngữ đích. Trong giai đoạn cuối cùng, một bộ phân tích hình thái của ngôn ngữ đích được sử dụng để tạo ra các bản dịch cuối cùng.

Mặc dù đã có một số hệ thống RBMT được đưa vào sử dụng như PROMPT (<http://www.promt.com/>) và Systrans (<http://www.systransoft.com/>). Tuy nhiên, bản dịch của hướng tiếp cận này có chất lượng thấp so với nhu cầu của con người và không sử dụng được trừ một số trường hợp đặc biệt. Ngoài ra chúng còn có một số nhược điểm lớn như:

- Các loại từ điển chất lượng tốt có sẵn là không nhiều và việc xây dựng những bộ từ điển mới là rất tốn kém.
- Hầu hết những luật ngôn ngữ được tạo ra bằng tay bởi các nhà ngôn ngữ học. Việc này gây khó khăn và tốn kém khi hệ thống trở nên lớn hơn.
- Các hệ thống RBMT gặp khó khăn trong việc giải quyết những vấn đề như thành ngữ hay sự nhập nhằng về ngữ nghĩa của các từ.

Từ những năm 1980, dịch máy dựa trên *Ngữ liệu* (Corpus-based machine translation) được đề xuất. Điểm khác biệt lớn nhất và cũng là quan trọng nhất của hướng tiếp cận này so với RBMT là thay vì sử dụng các bộ từ điển song ngữ, nó dùng những tập câu tương đương trong hai ngôn ngữ làm nền tảng cho việc dịch thuật. Tập những câu tương đương này được gọi là ngữ liệu. So với từ điển, việc thu thập ngữ liệu đơn giản hơn rất nhiều. Ví dụ như ta có thể tìm thấy nhiều phiên bản trong các ngôn ngữ khác nhau của những văn bản hành chính hay các trang web đa ngôn ngữ. Trước khi dịch máy nơ-ron ra đời, phương pháp dịch máy dựa trên ngữ liệu hiệu quả nhất chính là dịch máy thống kê.

*Dịch máy thống kê* (Statistical machine translation - SMT): ý tưởng của phương pháp này là thay vì định nghĩa những từ điển và các luật ngữ pháp một cách thủ công, SMT dùng mô hình thống kê để học các từ điển và các luật ngữ pháp này từ ngữ liệu. Những ý tưởng đầu tiên của SMT được giới thiệu đầu tiên bởi Warren Weaver vào năm

1949 bao gồm việc áp dụng lý thuyết thông tin của Claude Shannon vào dịch máy. SMT được giới thiệu lại vào cuối những năm 1980 và đầu những năm 1990 tại trung tâm nghiên cứu Thomas J. Watson của IBM. SMT là phương pháp được nghiên cứu rộng rãi nhất thời bấy giờ và thậm chí đến hiện tại, nó vẫn là một trong những phương pháp được nghiên cứu nhiều nhất về dịch máy.

Để hiểu rõ hơn về dịch máy thống kê, xét một ví dụ: ta cần dịch một câu  $f$  trong tiếng Pháp sang dạng tiếng Anh  $e$  của nó. Có nhiều bản dịch có thể có của  $f$  trong tiếng Anh, việc cần làm là chọn  $e$  sao cho nó là bản dịch "tốt nhất" của  $f$ . Chúng ta có thể mô hình hóa quá trình này bằng một xác suất có điều kiện  $p(e|f)$  với  $e$  là những bản dịch có thể có với câu cho trước  $f$ . Một cách hợp lý để chọn bản dịch "tốt nhất" là chọn  $e$  sao cho nó tối đa xác suất có điều kiện  $p(e|f)$ . Cách tiếp cận quen thuộc là sử dụng định lý Bayes để viết lại  $p(e|f)$ :

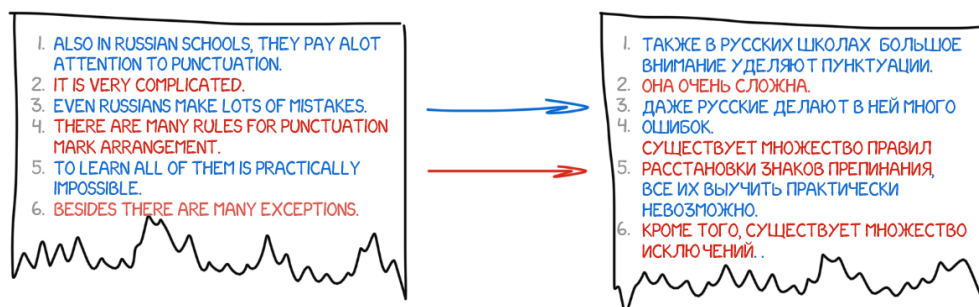
$$p(e|f) = \frac{p(f|e)p(e)}{p(f)} \quad (1.1)$$

Bởi vì  $f$  là cố định, tối đa hóa  $p(e|f)$  tương đương với tìm  $e$  sao cho tối đa hóa  $p(f|e)p(e)$ . Để làm được điều này, chúng ta dựa vào một tập ngữ liệu là những câu song ngữ Anh - Pháp để suy ra các mô hình  $p(f|e)$  và  $p(e)$  và sử dụng những mô hình đó để tìm một bản dịch cụ thể  $\tilde{e}$  sao cho:

$$\tilde{e} = \arg \max_{e \in e^*} p(e|f) = \arg \max_{e \in e^*} p(f|e)p(e) \quad (1.2)$$

Ở đây,  $p(f|e)$  được gọi là *mô hình dịch* (translation model) và  $p(e)$  được gọi là *mô hình ngôn ngữ* (language model). Mô hình dịch  $p(f|e)$  thể hiện khả năng câu  $e$  là một bản dịch của câu  $f$ . Những mô hình dịch ban đầu dựa trên từ (word-based) như các mô hình IBM 1-5 (IBM Models 1-5). Những năm 2000, những mô hình dịch dựa trên cụm từ (phrase based) xuất hiện giúp cải thiện khả năng dịch của SMT. Trong khi đó, mô hình ngôn ngữ  $p(e)$  thể hiện độ trơn tru của câu  $e$ . Ví dụ  $p(\text{"tôi đi học"}) > p(\text{"học tôi đi"})$  vì rõ ràng "tôi đi học" là có lý hơn "học tôi đi". Các mô hình ngôn ngữ cho SMT thường được ước lượng bằng các mô hình n-gram được làm mịn, cách làm này cũng là một nhược điểm của SMT. Mô hình ngôn ngữ là một chủ đề quan trọng và sẽ được chúng tôi đề cập cụ thể hơn trong chương 2.

## PARALLEL CORPUS



Hình 1.3: Ví dụ về tập các câu song song trong hai ngôn ngữ

Mặc dù trên thực tế đã có nhiều hệ thống dịch máy được phát triển dựa trên dịch máy thống kê thời bấy giờ, tuy nhiên nó không hoạt động thực sự tốt bởi một số nguyên nhân. Một là việc những từ hay đoạn được dịch cục bộ và quan hệ của chúng với những từ cách xa trong câu nguồn thường bị bỏ qua. Hai là mô hình ngôn ngữ n-gram hoạt động không thực sự tốt đối với những bản dịch dài và ta phải tốn nhiều bộ nhớ để lưu trữ chúng. Ngoài ra việc sử dụng nhiều thành phần nhỏ được điều chỉnh riêng biệt như mô hình dịch, mô hình ngôn ngữ,.. cũng gây khó khăn cho việc vận hành và phát triển mô hình này.

*Dịch máy nơ-ron* (Neural machine translation) là một hướng tiếp cận mới cho bài toán dịch máy trong những năm gần đây và được đề xuất đầu tiên bởi [9], [16], [1]. Giống như dịch máy thống kê, dịch máy nơ-ron cũng là một phương pháp thuộc hướng tiếp cận dựa trên ngữ liệu, trong khi dịch máy thống kê bao gồm nhiều mô-đun nhỏ được điều chỉnh riêng biệt, dịch máy nơ-ron cố gắng dùng một mạng nơ-ron như là thành phần duy nhất của hệ thống, mọi thiết lập sẽ được thực hiện trên mạng này.

Hầu hết những mô hình dịch máy nơ-ron đều dựa trên kiến trúc *Bộ mã hóa - Bộ giải mã* (Encoder - Decoder) ([16], [1]). Bộ mã hóa thường là một mạng nơ-ron có tác dụng "nén" tất cả thông tin của câu trong ngôn ngữ nguồn vào một vector có kích thước cố định. Bộ giải mã, cũng là một mạng nơ-ron, sẽ tạo bản dịch trong ngôn ngữ đích từ vector có kích thước cố định kia. Toàn bộ hệ thống bao gồm bộ mã hóa và bộ giải mã sẽ được huấn luyện "end-to-end" để tạo ra bản dịch, quá trình này được mô tả như hình 1.4.

Trong thực tế cả bộ mã hóa và giải mã thường dựa trên một mô hình mạng nơ-ron



Hình 1.4: Ví dụ về kiến trúc bộ mã hóa - bộ giải mã trong dịch máy nơ-ron



Hình 1.5: Kiến trúc bộ mã hóa - bộ giải mã được xây dựng trên mạng nơ-ron hồi quy

tên là *mạng nơ-ron hồi quy* là một thiết kế mạng đặc trưng cho việc xử lý dữ liệu chuỗi. Mạng nơ-ron hồi quy cho phép chúng ta mô hình hóa những dữ liệu có độ dài không xác định, rất thích hợp cho bài toán dịch máy. Hình 1.5 mô tả chi tiết hơn về kiến trúc bộ mã hóa - giải mã sử dụng mạng nơ-ron hồi quy. Đầu tiên bộ mã hóa đọc qua toàn bộ câu nguồn và tạo ra một vector đại diện gọi là *vector trạng thái*. Điều này giúp cho toàn bộ những thông tin cần thiết hay quan hệ giữa các từ đều được tập hợp vào một nơi duy nhất. Bộ giải mã, lúc này đóng vai trò như một mô hình ngôn ngữ để tạo ra từng từ trong ngôn ngữ đích và sẽ dừng lại đến khi một ký tự đặc biệt xuất hiện.

Trong hình 1.5, có thể thấy rằng bộ giải mã tạo ra bản dịch chỉ dựa trên trạng thái ẩn cuối cùng, cũng chính là vector có kích thước cố định được tạo ra ở bộ mã hóa. Vector này phải mã hóa mọi thứ chúng ta cần biết về câu nguồn. Giả sử chúng ta có câu nguồn với độ dài là 50 từ, từ đầu tiên ở câu đích có lẽ sẽ có mối tương quan cao với từ đầu tiên ở câu nguồn. Điều này có nghĩa là bộ giải mã phải xem xét thông tin được mã hóa từ 50 "*time step*" trước đó. Mạng nơ-ron hồi quy được chứng minh là gặp khó khăn trong việc mã hóa những chuỗi dài [14]. Để giải quyết vấn đề này, thay vì dùng mạng nơ-ron hồi quy thuần, người ta sử dụng các biến thể của nó như *Long short-term memory (LSTM)*. Trên lý thuyết, LSTM có thể giải quyết vấn đề mất mát thông tin trong chuỗi dài, nhưng trong thực tế vấn đề này vẫn chưa thể được giải quyết



Hình 1.6: Cơ chế Attention trong dịch máy nơ-ron

hoàn toàn. Một số nhà nghiên cứu đã phát hiện ra rằng đảo ngược chuỗi nguồn trước khi đưa vào bộ mã hóa tạo ra kết quả tốt hơn một cách đáng kể [16] bởi nó khiến cho những từ đầu tiên được đưa vào bộ mã hóa sau cùng, và được giải mã thành từ tương ứng ngay sau đó. Cách làm này tuy giúp cho bản dịch hoạt động tốt hơn trong thực tế, nhưng nó không phải là một giải pháp về mặt thuật toán. Hầu hết các đánh giá về dịch máy được thực hiện trên các ngôn ngữ như ngôn ngữ có trật tự câu tương đối giống nhau. Ví dụ trật tự dạng "chủ ngữ - động từ - vị ngữ" như tiếng Anh, Đức, Pháp hay Trung Quốc. Đối với dạng ngôn ngữ có một trật tự khác ví dụ "chủ ngữ - vị ngữ - động từ" như tiếng Nhật, đảo ngược câu nguồn sẽ không hiệu quả.

Năm 2015, trong bài báo "Effective Approaches to Attention-based Neural Machine Translation" [11] được công bố tại hội nghị EMNLP, nhóm tác giả của Đại học Stanford (Minh-Thang Luong, Hieu Pham, Christopher D. Manning) đã đưa ra mô hình dịch máy LSTM-attention, trong đó đưa thêm cơ chế attention vào kiến trúc bộ mã hóa – bộ giải mã (ở đây, bộ mã hóa là một LSTM, bộ giải mã cũng là một LSTM). Attention là cơ chế giải phóng kiến trúc bộ mã hóa - bộ giải mã khỏi nhược điểm chỉ sử dụng một vector có chiều dài cố định làm đại diện cho câu đầu vào. Ý tưởng chính của cơ chế này là ở mỗi thời điểm phát sinh một từ trong bản dịch, bộ giải mã sẽ "chú ý" vào các phần khác nhau của câu nguồn trong quá trình mã hóa. Quan trọng hơn, cơ chế này cho phép mô hình học được cách chọn những phần cần thiết để tập trung vào dựa trên câu nguồn và những gì mà bộ giải mã đã giải mã được.

Cơ chế Attention được miêu tả trong hình 1.6. Ở đây tại mỗi bước tạo ra từ mới, bộ giải mã kết hợp trạng thái ẩn ở bước giải mã trước đó với một vec-tơ gọi là *vec-tơ*

*ngữ cảnh* để sinh ra trạng thái ẩn ở thời điểm hiện tại. Vec-tơ ngữ cảnh này là *tổng có trọng số* của tất cả các vec-tơ trạng thái ẩn trong lúc mã hóa (đang đề cập đến *global attention*). Với cách tiếp cận này, bộ giải mã không chỉ tạo ra bản dịch dựa trên vec-tơ mã hóa của câu nguồn mà còn tận dụng được cả những trạng thái ẩn ở câu nguồn trong quá trình mã hóa. Điều này giúp cho mô hình dịch máy trở nên hiệu quả hơn. Bên cạnh đó các trọng số tạo nên vec-tơ ngữ cảnh cho biết những từ trong câu đích đã "chú ý" đến những từ nào trong câu nguồn. Việc này giúp chúng ta phần nào giải thích được các kết quả của dịch máy nơ-ron - vốn rất khó giải thích vì tính phức tạp của nó.

Với những ưu điểm đã được trình bày ở trên của phương pháp dịch máy nơ-ron so với các phương pháp trước đó, cũng như là với những lợi ích khi sử dụng cơ chế attention trong dịch máy nơ-ron, trong khóa luận này chúng tôi tìm hiểu về mô hình dịch máy LSTM-attention được đề xuất trong bài báo "Effective Approaches to Attention-based Neural Machine Translation" [11] đã nói ở trên. Trong khóa luận, chúng tôi đã thành công xây dựng lại mô hình được đề xuất trong bài báo. Bên cạnh đó chúng tôi cũng thực hiện nhiều những thí nghiệm để kiểm tra độ hiệu quả của mô hình. Những thí nghiệm đó bao gồm: thí nghiệm so sánh giữa mô hình dịch máy nơ-ron sử dụng và không sử dụng attention và các thí nghiệm so sánh giữa các mô hình attention với nhau. Các phần còn lại trong luận văn được trình bày như sau:

- Chương 2 trình bày về những thành nền tảng của kiến trúc bộ mã hóa - giải mã.
- Chương 3 trình bày về cơ chế Attention, đây là phần chính của luận văn. Trong phần này gồm có hai phần nhỏ:
  - *Global attetion*: là cơ chế tập trung vào tất cả các trạng thái ở câu nguồn
  - *Local attetion*: tập trung vào một tập các trạng thái ở câu nguồn tại một thời điểm
- Chương 4 trình bày về các thí nghiệm và các phân tích về kết quả đạt trên hai tập dữ liệu Anh-Đức, Anh-Việt.
- Cuối cùng, kết luận và hướng phát triển được trình bày ở chương 5.



## Chương 2

# Kiến Thức Nền Tảng

*Trong chương này, chúng tôi sẽ trình bày những kiến thức nền tảng trên ba chủ đề bao gồm mạng nơ-ron hồi quy, Long short-term memory và mô hình ngôn ngữ nơ-ron. Mạng nơ-ron hồi quy (RNN) là xương sống của dịch máy nơ-ron. Nó được sử dụng để làm cả bộ mã hóa lẫn bộ giải mã. Ứng với mỗi vai trò, RNN sẽ có một thiết kế riêng. Long short-term memory là phiên bản cải tiến của RNN để giải quyết vấn đề về phụ thuộc dài hạn. Long short-term memory cũng là phiên bản RNN được dùng để xây dựng nên bộ mã hóa - bộ giải trong khóa luận này. Sau đó, dựa trên những kiến thức về mạng nơ-ron hồi quy, chúng tôi nói về khái niệm mô hình ngôn ngữ hồi quy với chức năng tạo ra từ trong bộ giải mã, là bước quan trọng trong dịch máy nơ-ron. Những thành phần này cung cấp kiến thức nền tảng để đi đến mô hình dịch máy nơ-ron theo kiến trúc bộ mã hóa - bộ giải mã mà chúng tôi sẽ trình bày trong chương 3.*

## 2.1 Mạng nơ-ron hồi quy (Recurrent neural network)

Trong tự nhiên, dữ liệu không phải lúc nào cũng được sinh ra một cách ngẫu nhiên. Trong một số trường hợp, chúng được sinh ra theo một thứ tự. Xét trong dữ liệu văn bản, ví dụ ta cần điền vào chỗ trống cho câu sau "*Paris là thủ đô của nước \_\_\_\_*". Để biết được rằng chỉ có duy nhất một từ phù hợp cho chỗ trống này, đó là "*Pháp*". Điều này có nghĩa là mỗi từ trong một câu không được tạo ra ngẫu nhiên mà nó được tạo ra dựa trên một liên hệ với những từ đứng trước nó. Các loại dữ liệu khác như những

khung hình trong một bộ phim hoặc các đoạn âm thanh trong một bản nhạc cũng có tính chất tương tự. Những loại dữ liệu mang thứ tự này được gọi chung là dữ liệu chuỗi (sequential data).

Trong quá khứ, một số mô hình xử lý dữ liệu chuỗi bằng cách giả định rằng đầu vào hiện tại có liên hệ với một số lượng xác định đầu vào trước đó, nhiều mô hình tạo ra một cửa sổ trượt để nối mỗi đầu vào hiện tại với một số lượng đầu vào trước đó nhằm tạo ra sự mô phỏng về tính phụ thuộc. Cách tiếp cận này đã được sử dụng cho mô hình *Deep belief network* trong xử lý tiếng nói [12]. Nhược điểm của những cách làm này là ta phải xác định trước kích thước của cửa sổ. Một mô hình với kích thước cửa sổ với chiều dài bằng 6 không thể nào quyết định được từ tiếp theo trong câu "*Hổ là loài động vật ăn \_\_\_*" sẽ là "*thịt*" hay "*cỏ*". Trong ví dụ này, từ tiếp theo của câu phụ thuộc mật thiết vào từ "*Hổ*" cách nó đúng 6 từ. Trên thực tế, có rất nhiều câu đòi hỏi sự phụ thuộc với nhiều từ xa hơn trước đó. Ta gọi những sự phụ thuộc kiểu như vậy là những *phụ thuộc dài hạn* (long term dependency).

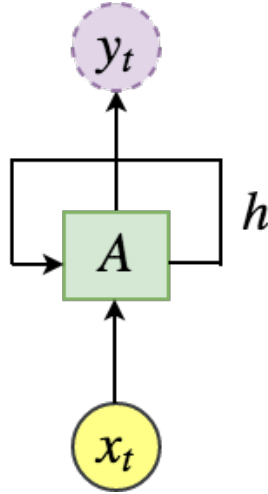
*Mạng nơ-ron hồi quy* (recurrent neural network) [3] gọi tắt là *RNN* là một nhánh của mạng nơ-ron nhân tạo được thiết kế đặc biệt cho việc mô hình hóa dữ liệu chuỗi. Khác với những mô hình đã đề cập giả định sự phụ thuộc chỉ xảy ra trong một vùng có chiều dài cố định. RNN, trên lý thuyết, có khả năng nắm bắt được các phụ thuộc dài hạn với chiều dài bất kỳ. Để làm được điều đó, trong quá trình học, RNN lưu giữ những thông tin cần thiết cho các phụ thuộc dài hạn bằng một vec-tơ được gọi là *trạng thái ẩn*.

Xét một chuỗi đầu vào  $x = x_1, x_2, \dots, x_n$ . Ta gọi  $h_t$  là trạng thái ẩn tại *bước thời gian* (timestep)  $t$ , là lúc một mẫu dữ liệu  $x_t$  được đưa vào RNN để xử lý. Trạng thái ẩn  $h_t$  sẽ được tính toán dựa trên mẫu dữ liệu hiện tại  $x_t$  và trạng thái ẩn trước đó  $h_{t-1}$ . Có thể thể hiện  $h_t$  như một hàm hồi quy với tham số là đầu vào hiện tại và chính nó ở thời điểm trước đó:

$$h_t = f(h_{t-1}, x_t) \quad (2.1)$$

trong đó hàm  $f$  là một ánh xạ phi tuyến. Có thể hình dung  $h_t$  như một đại diện cho những đầu vào mà nó đã xử lý từ thời điểm ban đầu cho đến thời điểm  $t$ . Nói một cách khác, RNN sử dụng trạng thái ẩn như một dạng bộ nhớ để lưu giữ thông tin từ một chuỗi. Hình 2.1 thể hiện định nghĩa hồi quy của RNN.

Thông thường, hàm  $f$  là một hàm phi tuyến như hàm  $\sigma$  hay hàm tanh. Xét một



Hình 2.1: Mô hình RNN đơn giản với kết nối vòng,  $h$  được xem như bộ nhớ được luân chuyển trong RNN. Chú ý rằng đường nét đứt ở đầu ra thể hiện rằng tại một thời điểm  $t$ , RNN có thể có hoặc không có một đầu ra.

RNN với công thức cụ thể như sau:

$$h_t = \phi (W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.2)$$

Trong đó:

- $\phi$  là một hàm kích hoạt (ví dụ: sigmoid, tanh hay ReLU).
- $h_t \in \mathbb{R}^n$  là trạng thái ẩn tại bước thời gian hiện tại.
- $x_t \in \mathbb{R}^m$  là đầu vào hiện tại.
- $h_{t-1} \in \mathbb{R}^n$  là trạng thái ẩn tại bước thời gian trước đó.
- $W_{xh} \in \mathbb{R}^{m \times n}$ ,  $W_{hh} \in \mathbb{R}^{n \times n}$  và  $b_h \in \mathbb{R}^n$  lần lượt là hai ma trận trọng số và vec-tơ "bias".

Ma trận  $W_{xh}$  là làm nhiệm vụ kết nối giữa đầu vào và trạng thái ẩn,  $W_{hh}$  kết nối trạng thái ẩn với chính nó trong các bước thời gian liên tiếp. Vec-tơ  $b_h$  dùng để điều chỉnh giá trị của  $h_t$ . Tại thời điểm bắt đầu, trạng thái ẩn  $h_0$  có thể được khởi tạo bằng 0 hoặc là một vector chứa tri thức có sẵn như trường hợp của bộ giải mã như chúng tôi đã đề cập trong chương 1.



Hình 2.2: Mô hình RNN được dàn trải (unrolled), ví dụ trong 4 bước thời gian.

Tại mỗi bước thời gian  $t$ , tùy vào mục tiêu cụ thể của quá trình học mà RNN có thể có thêm một đầu ra  $y_t$ . Trong ngữ cảnh bài toán dịch máy nơ-ron, đầu ra của RNN trong quá trình giải mã chính là một từ trong ngôn ngữ đích hay nói chung là một đầu ra dạng rời rạc. Với mục tiêu đó, đầu ra dự đoán của RNN  $\hat{y}_t$  sẽ có dạng là một phân phối xác suất trên tập các lớp ở đầu ra. Phân phối này nhằm dự đoán vị trí xuất hiện của  $\hat{y}_t$ .

$$\hat{y}_t = \text{softmax}(W_{hy}h_t + b_y) \quad (2.3)$$

Trong đó:

- softmax là một hàm kích hoạt với  $\text{softmax}(v_j) = \frac{e^{v_j}}{\sum_{k=1}^K e^{v_k}}$ ,  $j = 1, \dots, K$ ,  $K$  là độ dài của vec-tơ  $v$ .
- $h_t \in \mathbb{R}^n$  là trạng thái ẩn tại bước thời gian hiện tại.
- $W_{hy} \in \mathbb{R}^{L \times n}$  và  $b_y \in \mathbb{R}^L$  lần lượt là hai ma trận trọng số và vec-tơ "bias".  $L$  là số lượng lớp cần phân biệt ở đầu ra.

Trong công thức trên, hàm softmax đóng vai trò là một hàm chuẩn hóa để  $\hat{y}_t$  thể hiện một phân phối xác suất trên các lớp ở đầu ra. Ma trận  $W_{hy}$  kết nối đầu ra với trạng thái ẩn,  $b_y$  dùng để điều chỉnh giá trị của kết quả tính toán trước khi đưa qua hàm softmax.

Để ý rằng các ma trận trọng số  $W_{xh}$ ,  $W_{hh}$ ,  $W_{hy}$  và các vector bias  $b_h$ ,  $b_y$  là các tham số học của mô hình và chúng là duy nhất. Có nghĩa là khi những tham số này được

học, bất kỳ một đầu vào nào cũng đều sử dụng chung một bộ tham số. Điều này chính là sự chia sẻ tham số (parameters sharing) trong mạng nơ-ron hồi quy. Chia sẻ tham số khiến cho mô hình học dễ dàng hơn, nó giúp cho RNN có thể xử lý chuỗi đầu vào với độ dài bất kỳ mà không làm tăng độ phức tạp của mô hình. Quan trọng hơn, nó giúp ích cho việc tổng quát hóa. Đây chính là điểm đặc biệt của RNN so với mạng nơ-ron truyền thẳng.

Với một số lượng hữu hạn các bước thời gian, mô hình RNN trên hình 2.1 có thể được dàn trải ra (unrolled). Dạng dàn trải này được miêu tả trực quan như trên hình 2.2. Với cách thể hiện này, RNN có thể được hiểu như là một mạng nơ-ron sâu với mỗi bước thời gian là một mạng nơ-ron một tầng ẩn và các tham số học được chia sẻ giữa các mạng nơ-ron đó. Dạng dàn trải cũng thể hiện rằng RNN có thể được huấn luyện qua nhiều bước thời gian bằng thuật toán lan truyền ngược (backpropagation). Thuật toán này được gọi là "Backpropagation through time" (BPTT) [17]. Thực chất đây là chỉ thuật toán "Backpropagation" khi áp dụng cho RNN dưới dạng dàn trải để tính "gradient" cho các tham số ở từng bước thời gian. Hầu hết cả các mạng nơ-ron hồi quy phổ biến ngày nay đều áp dụng thuật toán này vì tính đơn giản và hiệu quả của nó.

### 2.1.1 Huấn luyện mạng nơ-ron hồi quy

Xét một chuỗi đầu vào  $x = x_1, x_2, \dots, x_n$  với đầu ra tương ứng  $y = y_1, y_2, \dots, y_n$ . Trong quá trình lan truyền tiến, tại mỗi bước thời gian  $t$  ứng mẫu dữ liệu  $(x_t, y_t)$ , công thức tính toán đầu ra dự đoán có dạng:

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.4)$$

$$s_t = W_{hy}h_t + b_y \quad (2.5)$$

$$\hat{y}_t = \text{softmax}(s_t) \quad (2.6)$$

Ta cần định nghĩa hàm độ lỗi giữa đầu ra dự đoán  $\hat{y}_t$  và đầu ra thật sự  $y_t$ . Với  $L$  là số lượng lớp của  $y$ , lúc này có thể thấy  $\hat{y}_t$  là một vec-tơ phân phối xác suất có độ dài  $L$ . Để so sánh với  $\hat{y}_t$ ,  $y_t$  được chuẩn hóa thành một vec-tơ dạng "one hot" có nghĩa là một vec-tơ với độ dài  $V$  có giá trị bằng 0 trừ vị trí ứng với lớp của  $y_t$  có giá trị 1. Như vậy

để so sánh hai phân phối xác suất  $y$  và  $\hat{y}$  ta sử dụng hàm độ lỗi *negative log-likelihood* hay còn gọi là *cross entropy*, gọi  $L_t$  là độ lỗi tại một bước thời gian  $t$ , ta có:

$$L_t = -y_t^T \log(\hat{y}_t) \quad (2.7)$$

Gọi  $\theta$  là một tham số của mô hình, ta biết rằng  $\theta$  được chia sẻ trong quá trình học, tức là ở mọi bước thời gian, chúng đều có giá trị bằng nhau:

$$\theta_t = \theta_k \quad (2.8)$$

Với  $t, k$  là những bước thời gian ( $t \neq k$ ). Tại thời điểm bắt đầu với  $t = k = 0$  mọi  $\theta_i$  đều có giá trị bằng nhau nên trong quá trình học, ta cần:

$$\frac{\partial L}{\partial \theta_t} = \frac{\partial L}{\partial \theta_k} \quad (2.9)$$

Với  $L$  là độ lỗi tổng hợp tại của tất cả các bước thời gian. Như vậy để  $\theta_t = \theta_k, \forall t, k$ , ta chỉ đơn giản xem  $L$  là tổng độ lỗi ở tất cả các bước thời gian. Với cách làm này, tham số luôn bằng nhau sau mỗi lần cập nhật.

$$L = \sum_t L_t = -\sum_t y_t^T \log(\hat{y}_t) \quad (2.10)$$

Mục tiêu của việc học là cực tiểu hóa độ lỗi tổng hợp  $L$ . Thuật toán "backpropagation" với *gradient descent* sẽ được áp dụng để huấn luyện RNN. Trên thực tế, người ta sẽ sử dụng một phiên bản của "gradient descent" là "mini-batch gradient descent" cho việc huấn luyện. Tập dữ liệu ban đầu sẽ được chia thành nhiều "mini-batch", mỗi "mini-batch" là một tập con với số lượng khoảng vài chục đến vài trăm mẫu thuộc tập dữ liệu ban đầu. Với mỗi lần duyệt (iteration), việc tính toán gradient để cập nhật các tham số học của mô hình được thực hiện lần lượt trên tất cả các mini-batch này.

Ta cần tìm bộ tham số  $\theta \in \{W_{hy}, W_{hh}, W_{xh}, b_y, b_h\}$  sao cho cực tiểu hóa hàm độ lỗi  $L$ . Theo thuật toán "gradient descent", bộ tham số được cập nhật theo công thức:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L \quad (2.11)$$

Ở đây,  $\nabla_{\theta} L$  là "gradient" của hàm độ lỗi ứng với tham số  $\theta$ .  $\eta$  được gọi là hệ số học (learning rate) là một siêu tham số quyết định rằng  $\theta$  nên thay đổi nhiều bao nhiêu khi "gradient" ứng với tham số thay đổi. Trong phần dưới đây, chúng tôi sẽ trình bày việc tính toán "gradient" của hàm độ lỗi theo bộ tham số học  $\theta \in \{W_{hy}, W_{hh}, W_{xh}, b_y, b_h\}$ .

Với  $s_t = W_{hy}h_t + b_y$  và  $\hat{y}_t = \text{softmax}(s_t)$ , sử dụng "chain rule" trong tính đạo hàm ta được:

$$\nabla_{\theta} L_t = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial \theta} = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial \theta} \quad (2.12)$$

Và ta có (công thức này được chứng minh trong phần mở rộng):

$$\nabla_{s_t} L_t = \frac{\partial L_t}{\partial s_t} = \hat{y}_t - y \quad (2.13)$$

Ta cũng dễ dàng tính được:

$$\frac{\partial s_t}{\partial h_t} = W_{hy}^T \quad (2.14)$$

Đến đây, ta chỉ cần tìm  $\frac{\partial h_t}{\partial \theta}$  với  $\theta \in \{W_{hy}, W_{hh}, W_{xh}, b_y, b_h\}$ . Ta có thể chia các tham số thành hai nhóm. Nhóm đầu tiên là các tham số liên quan đến quá trình tính toán đầu ra, bao gồm  $\theta^{(1)} \in \{W_{hy}, b_y\}$ . Những tham số này không tham gia vào hàm hồi quy  $h_t$  cho nên "gradient" ứng với chúng được tính một cách dễ dàng:

$$\nabla_{W_{hy}} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial W_{hy}} = (\hat{y}_t - y) h_t^T \quad (2.15)$$

$$\nabla_{b_y} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial b_y} = \hat{y}_t - y \quad (2.16)$$

Nhóm thứ hai gồm những tham số tham gia vào quá trình hồi quy, bao gồm  $\theta^{(2)} \in \{W_{hh}, W_{xh}, b_h\}$ . Để tính được "gradient" ứng với  $\theta^{(2)}$ , ta cần tính được  $\frac{\partial h_t}{\partial \theta^{(2)}}$

Vì  $h_t$  là một hàm hồi quy được xây dựng dựa trên  $\theta^{(2)}$  và  $h_{t-1}$  nên "gradient" của  $h_t$  tương ứng với  $\theta^{(2)}$  được tính dựa trên quy tắc "total derivative". Quy tắc này nói rằng nếu  $f(x, y)$  với  $x, y \in \mathbb{R}^M$ , giả sử  $x, y$  là những hàm số của  $r$  sao cho  $x = x(r); y = y(r)$  thì ta có:

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x} \frac{\partial x}{\partial r} + \frac{\partial f}{\partial y} \frac{\partial y}{\partial r} \quad (2.17)$$

Áp dụng vào trường hợp của  $\frac{\partial h_t}{\partial \theta^{(2)}}$  ta được:

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial \theta^{(2)}} \quad (2.18)$$

Tuy nhiên, ta có thể khai triển công thức trên một lần nữa với cách làm tương tự cho  $\frac{\partial h_{t-1}}{\partial \theta^{(2)}}$ :

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial \theta^{(2)}} \quad (2.19)$$

Khai triển trên sẽ kéo dài cho đến khi gặp  $\frac{\partial h_0}{\partial \theta^{(2)}}$ . Và để ý rằng:

$$\frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial \theta^{(2)}} \quad (2.20)$$

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial h_t} \frac{\partial h_t}{\partial \theta^{(2)}} \quad (2.21)$$

Với cách khai triển như vậy, ta có công thức tổng quát cho  $\frac{\partial h_t}{\partial \theta^{(2)}}$ , đó là:

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \sum_{r=0}^t \frac{\partial h_t}{\partial h_r} \frac{\partial^\dagger h_r}{\partial \theta^{(2)}} \quad (2.22)$$

trong đó  $\frac{\partial^\dagger h_r}{\partial \theta^{(2)}}$  là đạo hàm "lập tức", có nghĩa là khi lấy đạo hàm  $h_r$  theo  $\theta^{(2)}$ ,  $h_{r-1}$  được coi là hằng số.

Lần lượt thay thế  $\theta^{(2)}$  bằng  $W_{hh}, W_{xh}, b_h$  ta có "gradient" ứng với từng tham số là:

$$\nabla_{W_{hh}} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial W_{hh}} = (\hat{y}_t - y) W_{hy}^T \sum_{r=0}^t \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{hh}} \quad (2.23)$$

$$\nabla_{W_{xh}} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial W_{xh}} = (\hat{y}_t - y) W_{hy}^T \sum_{r=0}^t \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{xh}} \quad (2.24)$$

$$\nabla_{b_h} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial b_h} = (\hat{y}_t - y) W_{hy}^T \sum_{r=0}^t \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial b_h} \quad (2.25)$$



Để tính  $\frac{\partial h_t}{\partial h_r}$  ta áp dụng "chain rule" từ bước thời gian thứ  $t$  đến  $r$

$$\frac{\partial h_t}{\partial h_r} = \prod_{i=r}^{t-1} \frac{\partial h_{i+1}}{\partial h_i} \quad (2.26)$$

với  $h_{i+1}, h_i \in \mathbb{R}^n$  nên  $\frac{\partial h_{i+1}}{\partial h_i}$  là một ma trận "Jacobian"

$$\frac{\partial h_{i+1}}{\partial h_i} = \begin{bmatrix} \frac{h_{i+1,1}}{h_{i,1}}, \dots, \frac{h_{i+1,n}}{h_{i,n}} \end{bmatrix} \quad (2.27)$$

$$= \begin{bmatrix} \frac{h_{i+1,1}}{h_{i,1}} & \dots & \frac{h_{i+1,1}}{h_{i,n}} \\ \vdots & \ddots & \vdots \\ \frac{h_{i+1,n}}{h_{i,1}} & \dots & \frac{h_{i+1,n}}{h_{i,n}} \end{bmatrix} = \prod_{i=r}^{t-1} W_{hh}^T \text{diag}(1 - (h_i)^2) \quad (2.28)$$

Như vậy, ta có "gradient" của các tham số qua tất cả các bước thời gian sẽ là:

$$\nabla_{b_y} L = \sum_t \hat{y}_t - y \quad (2.29)$$

$$\nabla_{b_h} L = \sum_t \sum_{r=0}^t (\hat{y}_t - y) W_{hy}^T \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial b_h} \quad (2.30)$$

$$\nabla_{W_{hy}} L = \sum_t (\hat{y}_t - y) h_t^T \quad (2.31)$$

$$\nabla_{W_{hh}} L = \sum_t \sum_{r=0}^t (\hat{y}_t - y) W_{hy}^T \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{hh}} \quad (2.32)$$

$$\nabla_{W_{xh}} L_t = \sum_t \sum_{r=0}^t (\hat{y}_t - y) W_{hy}^T \frac{\partial h_t}{\partial h_r} \frac{\partial h_r}{\partial W_{xh}} \quad (2.33)$$

### 2.1.2 Thách thức trong việc học các phụ thuộc dài hạn

Hãy tưởng tượng rằng dòng thời gian mà chúng ta đang tồn tại là một mạng nơ-ron hồi quy khổng lồ. Tại mỗi thời điểm, trạng thái của thế giới được thay đổi bởi một hàm số vô cùng phức tạp của tự nhiên. Bây giờ, hãy xem một sự thay đổi nhỏ của ngày hôm nay ảnh hưởng đến thế giới sau tương lai như thế nào. Thay đổi ấy có thể lớn như việc

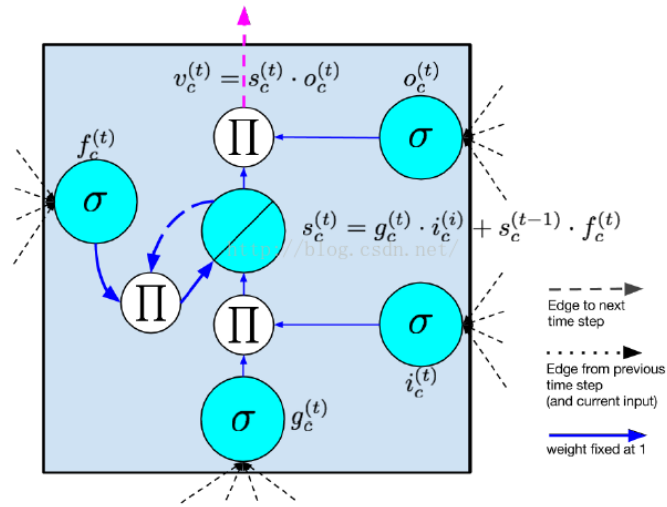
một con bướm đập cánh ở Brazil gây ra một cơn lốc xoáy ở Texas, nhưng nó cũng có thể là một sự kiện to lớn nhưng cuối cùng lại không quan trọng trong nhiều năm nữa. 100 năm sau, sẽ không ai nhắc đến đĩa mềm, thứ được xem là một phát minh lớn trong những năm 1960.

Hai ví dụ trên cũng chính là những vấn đề cơ bản mà mạng nơ-ron hồi quy gặp phải. Sự phát minh ra đĩa mềm hay việc con bướm đập cánh đại diện cho những thay đổi ở những bước thời gian trong quá khứ. Sự thay đổi ấy có thể hoặc là ảnh hưởng to lớn đến đầu ra ở bước thời gian hiện tại hoặc không gây ra ảnh hưởng gì. Quay lại với ví dụ ở đầu chương, ta cần tìm từ tiếp theo trong chuỗi "Hổ là loài động vật ăn \_\_\_\_". Một mạng RNN được huấn luyện để xuất ra từ "thịt" hay "cỏ" hay bất kỳ một từ nào khác trong bước thời gian tiếp theo tùy vào tên của loài động vật ở đầu câu. Trong trường hợp ta thay đổi từ "Hổ" thành "Bò" ta mong rằng "thịt" cũng sẽ trở thành "cỏ". Nhưng nếu RNN của chúng ta hoàn toàn bỏ qua sự thay đổi này (như trong ví dụ về đĩa mềm), ta sẽ không thể có được "cỏ" (vì nó không biết ở đầu câu là con vật gì). Tương tự, nếu RNN quá nhạy cảm với sự thay đổi từ xa này, nó có thể cho ra một từ hoàn toàn khác như "sữa" vì "Bò" đã lấn át hết tất cả những từ còn lại trong câu.

Làm sao để các phụ thuộc dài hạn này không mất đi sự ảnh hưởng cũng như gây ảnh hưởng quá lớn đến hiện tại là một thách thức lớn cho RNN. Về mặt toán học, "gradient" đại diện cho sự thay đổi trong quá khứ có thể bị biến mất (vanish) hoặc bùng nổ (explode). Những vấn đề này được phát hiện và trình bày trong những nghiên cứu [6], [?], [14]. Nếu gradient bùng nổ, mô hình không thể học sai hoặc không học được. Nếu gradient biến mất, việc học những phụ thuộc dài hạn trở nên khó khăn. Trong mục này, chúng tôi sẽ trình bày những vấn đề trên một cách chi tiết và mô tả những giải pháp để vượt qua các vấn đề đó.

## 2.2 Long short-term memory

Hochreiter và Schmidhuber [6] đã giới thiệu mô hình *Long short-term memory* (LSTM) chủ yếu ở để khắc phục vấn đề biến mất gradient trong RNN. Nhớ lại rằng trong RNN, chính việc mô hình hóa phụ thuộc thời gian dựa vào ma trận trọng số  $W_{hh}$  đã gây ra hiện tượng gradient biến mất. Giả sử gọi  $S_t$  là trạng thái liên kết hồi quy (trong RNN, nó chính là trạng thái ẩn). Ý tưởng của LSTM là thay vì tính toán  $S_t$  từ  $S_{t-1}$  với một



Hình 2.3: Mô hình RNN đơn giản với kết nối vòng,  $h$  được xem như bộ nhớ được luân chuyển trong RNN. Chú ý rằng đường nét đứt ở đầu ra thể hiện rằng tại một thời điểm  $t$ , RNN có thể có hoặc không có một đầu ra.

phép nhân ma trận theo sau là hàm kích hoạt phi tuyến, LSTM trực tiếp tính toán một  $\Delta S_t$  sau đó nó được cộng với  $S_{t-1}$  để tạo ra  $S_t$ . Thoạt nhìn, sự khác biệt này có thể không đáng kể khi mà chúng ta đều đạt được  $S_t$  trong cả hai cách. Tuy nhiên, với cách làm này, các "gradient" của LSTM tính toán  $\Delta h_t$  sẽ không bị biến mất.

Thuật ngữ "Long short-term memory" xuất phát từ nhận định sau. Mạng RNN đơn giản có "long term memory" (bộ nhớ dài hạn) dưới dạng các ma trận trọng số. Những ma trận trọng số này thay đổi một cách chậm rãi trong quá trình học nhằm mã hóa kiến thức về dữ liệu. RNN cũng có "short-term memory" (bộ nhớ ngắn hạn) dưới dạng các kích hoạt tạm thời, được truyền từ mỗi bước thời gian sang các bước thời gian sau đó. "Long short-term memory" tạm dịch là "bộ nhớ ngắn hạn dài" cho phép mở rộng bộ nhớ ngắn hạn bằng cách thêm vào một loại lưu trữ trung gian gọi là trạng thái lưu giữ (cell state). Trạng thái lưu giữ này có khả năng lưu giữ các thông tin cần thiết một cách lâu dài dưới dạng một bộ nhớ ngắn hạn. Để làm được điều này, LSTM sử dụng một cơ chế gọi là "cổng", các "cổng" giúp được huấn luyện để chọn lọc thông tin nào là cần thiết để tác động lên trạng thái lưu trữ. Với cách làm này, trạng thái lưu trữ sẽ lưu được nhiều thông tin hơn, vì chỉ những thông tin quan trọng mới tồn tại trong nó.

Về cấu tạo, một LSTM tương tự như một RNN một lớp ẩn, nhưng mỗi "RNN cell"

(ký hiệu "A" trong hình 2.2) được thay thế bằng một "memory cell" (hình 2.3). Giống như "RNN cell", "memory cell" nhận một đầu vào bên ngoài và phát sinh một đầu ra cũng như là truyền đi một trạng thái ẩn sang "memory cell" ở bước thời gian kế tiếp. Tuy nhiên, trong "memory cell" còn có thêm một trạng thái lưu giữ cũng được truyền đi như một trạng thái ẩn. Cấu tạo chi tiết của LSTM sẽ được trình bày trong phần dưới đây, cấu tạo này dựa trên phiên bản LSTM của [4].

- *Nút đầu vào (input node)*: Đơn vị này được ký hiệu là  $g$ , là một mạng nơ-ron một tầng ẩn. Nút đầu vào có nhiệm vụ mô hình hóa đầu vào tại mỗi bước thời gian. Nó nhận tham số là đầu vào tại bước thời gian hiện tại  $x_t$  và trạng thái ẩn tại thời điểm trước đó  $h_{t-1}$ . Cụ thể, tại mỗi bước thời gian nút đầu vào có công thức:

$$g_t = \phi (W_{gx}x_t + W_{gh}h_{t-1} + b_g) \quad (2.34)$$

- *Cổng vào (input gate)*: "Cổng" như đã nói, là một cơ chế đặc biệt của LSTM. Cổng vào cũng được cấu tạo giống như nút đầu vào, nó nhận tham số là  $x_t$  và  $h_{t-1}$ . Sau đó được đưa qua hàm kích hoạt sigmoid để tạo ra giá trị trong khoảng  $(0, 1)$ . Sở dĩ đơn vị này được gọi là "cổng vào" vì giá trị của nó sẽ được sử dụng để nhân với giá trị của nút đầu vào. Giá trị của nó thể hiện lượng thông tin mà nút đầu vào được phép truyền đi. Nếu cổng vào bằng 0, nút đầu vào sẽ truyền đi với giá trị 0. Nếu cổng vào bằng 1, nút đầu vào sẽ truyền đi với giá trị ban đầu. Cụ thể hơn, ta có công thức của cổng vào, được ký hiệu là  $i$ , tại bước thời gian  $t$ :

$$i_t = \sigma (W_{ix}x_t + W_{ih}h_{t-1} + b_i) \quad (2.35)$$

- *Trạng thái lưu giữ (cell state)*: Trái tim của LSTM chính là trạng thái lưu trữ, là một mạng nơ-ron với hàm kích hoạt tuyến tính. Khá giống với trạng thái ẩn trong RNN, trạng thái lưu trữ  $s_t$  cũng có một kết nối hồi quy với trạng thái lưu trữ trước đó  $s_{t-1}$ . Tuy nhiên, trọng số kết nối hồi quy luôn có giá trị cố định là 1. Bởi vì kết nối hồi quy này qua nhiều bước đều có trọng số không đổi nên khi tính toán, "gradient" của độ lỗi không bị bùng nổ hay biến mất. Tại mỗi bước thời gian, trạng thái lưu trữ được tính như sau:

$$s_t = s_{t-1} + g_t \odot i_t \quad (2.36)$$

- *Cổng quên (forget gate)*: Cổng quên là một đề xuất của [4] so với bài báo LSTM gốc. Thay vì kiểm soát lượng thông tin để đưa vào trạng thái lưu giữ như cổng vào, cổng quên cung cấp khả năng tẩy đi một lượng thông tin trong trạng thái lưu giữ. Cụ thể, cổng quên với giá trị thuộc khoảng  $(0, 1)$  sẽ được nhân với  $s_{t-1}$  trong công thức 2.36. Tại mỗi bước thời gian, giá trị của cổng quên  $f_t$  được tính như sau:

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \quad (2.37)$$

Công thức của trạng thái lưu trữ được sửa lại khi có cổng quên:

$$s_t = s_{t-1} \odot f_t + g_t \odot i_t \quad (2.38)$$

- *Cổng ra (output gate)*: Giá trị đầu ra  $v_t$  của "memory cell" tại mỗi bước thời gian chính là tích giá trị của trạng thái lưu trữ  $s_t$  với giá trị của cổng ra  $o_t$ . Trong một số phiên bản của LSTM, trạng thái lưu trữ trước đó được cho qua một hàm kích hoạt như tanh. Tại mỗi bước thời gian, giá trị của cổng ra  $o_t$  được tính như sau:

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (2.39)$$

Đầu ra của "memory cell" cũng chính là trạng thái ẩn  $h_t$  có giá trị:

$$h_t = s_t \odot o_t \quad (2.40)$$

## 2.3 Mô hình ngôn ngữ

Như chúng tôi đã đề cập trong chương 1, *mô hình ngôn ngữ* (language model) là một bộ phận quan trọng trong cả dịch máy thống kê và dịch máy nơ-ron. Cụ thể hơn, mô hình ngôn ngữ là một phân phối xác suất trên một chuỗi các từ. Cho trước một chuỗi  $w_1, w_2, \dots, w_m$  (ký hiệu  $w_{1:n}$ ), mô hình ngôn ngữ gán cho nó một xác suất  $P(w_{1:n})$  đại diện cho độ "trơn tru" của chuỗi đó. Sử dụng quy tắc dây chuyền (chain rule) trong xác suất, ta có:

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2})P(w_4|w_{1:3})\dots P(w_n|w_{1:n-1}) \quad (2.41)$$

## Chương 3

# Dịch máy bằng mô hình học LSTM-Attention

*Chương này trình bày về việc tổ chức các LSTM theo kiến trúc Bộ mã hóa-Bộ giải mã cùng với sử dụng cơ chế Attention. Ở đây, chúng tôi tập trung tìm hiểu về LSTM trong kiến trúc Bộ mã hóa-Bộ giải mã và các phiên bản của cơ chế Attention, sau đó giải thích chúng dựa trên cơ sở Toán học. Cụ thể, về LSTM, chúng tôi sử dụng bi-LSTM và uni-LSTM, về Attention, chúng tôi tìm hiểu về hai phiên bản Toàn cục (Global) và Cục bộ (Local):*

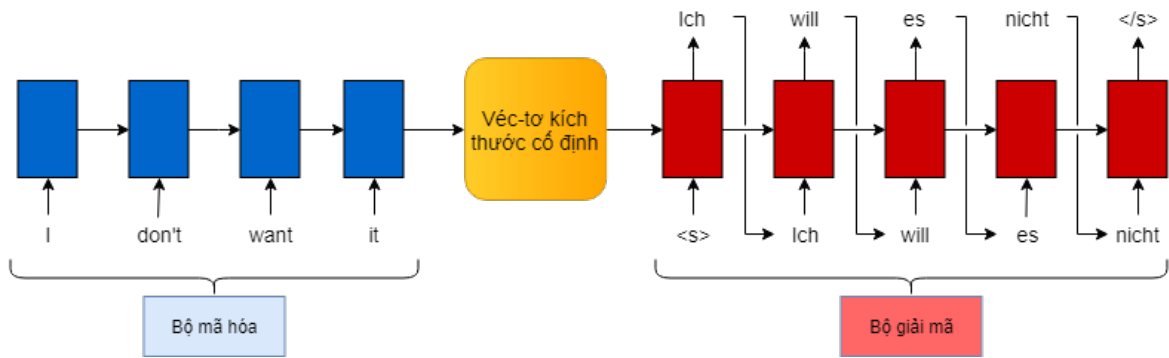
- *LSTM: chúng tôi tổ chức các bi-LSTM và uni-LSTM theo kiến trúc Bộ mã hóa-Bộ giải mã để tránh được hạn chế về số lượng từ ở ngôn ngữ nguồn và ngôn ngữ đích.*
- *Attention: chúng tôi tìm hiểu được sự hạn chế hiện có của kiến trúc Bộ mã hóa-Bộ giải mã khi thực hiện dịch những câu dài. Do vậy, chúng tôi sử dụng cơ chế Attention để giải quyết hạn chế đó bằng cách sử dụng thêm các trạng thái ẩn của bộ mã hóa trong quá trình giải mã.*
  - *Toàn cục: phiên bản này sử dụng tất cả trạng thái ẩn trong bộ mã hóa trong quá trình giải mã.*
  - *Cục bộ: phiên bản này chỉ sử dụng một số trạng thái ẩn trong bộ mã hóa trong quá trình giải mã.*

### 3.1 LSTM với kiến trúc Bộ mã hóa-Bộ giải mã

Với những kiến thức về mô hình LSTM mà đã được trình bày ở chương trước, dùng một mô hình LSTM hoàn toàn có thể áp dụng cho việc giải quyết bài toán Dịch máy. Với đầu vào tại mỗi bước thời gian là một từ ở trong câu nguồn và đầu ra là một từ tương ứng trong ngôn ngữ đích. Tuy nhiên, do bản thân của mô hình LSTM, khi áp dụng cho bài toán Dịch máy thì phải chịu một số hạn chế.

Một trong những hạn chế rất lớn của việc sử dụng một mô hình LSTM khi áp dụng vào trong bài toán Dịch máy là ràng buộc chiều dài của câu đầu vào (ngôn ngữ nguồn) phải bằng chiều dài của câu đầu ra (ngôn ngữ đích). Bởi vì tại mỗi thời điểm, LSTM nhận đầu vào là một từ ở ngôn ngữ nguồn, sau đó thực hiện dự đoán một từ ở ngôn ngữ đích. Với một mô hình LSTM thì chỉ có thể dịch ổn trên những cặp ngôn ngữ rất giống nhau (ví dụ như Trung-Việt) nhưng chất lượng dịch vẫn rất khó đảm bảo vì thậm chí giữa cặp ngôn ngữ này dù rất giống nhau nhưng vẫn có cách diễn đạt, sử dụng từ ngữ khác nhau. // TODO: Đưa ra ví dụ. Để nhận thấy rằng đây là một giả định phi thực tế, dấu là con người dịch thì cũng rất khó và phải là người am hiểu về cả 2 ngôn ngữ đó thì mới có thể dịch sang ngôn ngữ đích mà vẫn truyền đạt được đầy đủ ý nghĩa ở ngôn ngữ nguồn. Đối với ràng buộc chiều dài ở 2 câu phải bằng nhau thì chỉ phù hợp với khi dịch thơ có luật ràng buộc về số từ như thơ lục bát, thơ Đường v.v... Tuy nhiên, dù đã thỏa ràng buộc về số lượng từ nhưng vẫn khó đảm bảo về mặt ngữ nghĩa. Trong thơ văn, người ta sử dụng rất nhiều biện pháp tu từ, văn phong và ngữ pháp thì rất đa dạng, cách hành văn rất khác. Hơn nữa, một câu có thể biểu diễn bởi rất nhiều cách khác nhau. Minh chứng là một bài thơ tiếng Trung có thể có rất nhiều bản dịch khác nhau. Từ những lí do kể trên, sử dụng một mô hình LSTM để giải quyết bài toán Dịch máy là không phù hợp với thực tế.

Để giải quyết những hạn chế của đó, nhóm tác giả Sutskever, 2014 [16] đã đề xuất một kiến trúc tên gọi là Chuỗi tới Chuỗi (Sequence to Sequence - Seq2Seq). Tuy nhiên, mọi người thường gọi kiến trúc này là kiến trúc Bộ mã hóa-Bộ giải mã (Encoder-Decoder). Trong khóa luận này, chúng tôi sử dụng kiến trúc này để giải quyết bài toán Dịch máy và gọi nó là Bộ mã hóa-Bộ giải mã. Ý tưởng của kiến trúc này rất đơn giản. Kiến trúc này tận dụng các đặc tính của các mô hình LSTM và tổ chức chúng một cách hợp lí sao cho phù hợp với các yêu cầu của bài toán Dịch máy.



Hình 3.1: Minh họa kiến trúc Bộ mã hóa-Bộ giải mã được sử dụng trong khóa luận. Kiến trúc có 2 bộ phận. Trong đó, bộ mã hóa là một bi-LSTM có nhiệm vụ nhận câu nguồn làm đầu vào và xuất ra một véc-tơ có kích thước cố định chứa thông tin cần thiết để dịch của toàn bộ câu nguồn. Bộ giải mã là một uni-LSTM có đầu vào là véc-tơ có kích thước cố định của bộ mã hóa và đầu ra là câu đích.

Kiến trúc Bộ mã hóa-Bộ giải mã như tên gọi, gồm có 2 bộ phận chính là bộ mã hóa và bộ giải mã. Mỗi một bộ phận sẽ là một mô hình học được lựa chọn sao cho phù hợp với bài toán và có thể hoạt động tốt với nhau. Đối với bài toán Dịch máy và trong phạm vi khóa luận này, mỗi bộ phận sẽ là một mạng nơ-ron hồi qui, cụ thể là các LSTM.

Hình 3.1 minh họa kiến trúc Bộ mã hóa-Bộ giải mã được sử dụng trong khóa luận để giải quyết bài toán Dịch máy. Chúng tôi sẽ trình bày cụ thể về cách hoạt động của 2 bộ phận này trong các phần tiếp theo.

### 3.1.1 Bộ mã hóa

Bộ mã hóa là một bi-LSTM (LSTM 2 chiều). Bi-LSTM này nhận đầu vào là một câu ở ngôn ngữ nguồn  $x = \{x_0, \dots, x_{S-1}\}$ . Đầu ra là một véc-tơ có kích thước cố định. Véc-tơ này chứa các thông tin cần thiết để tiến hành dịch sang câu ở ngôn ngữ đích. Nhiệm vụ của bộ mã hóa (bi-LSTM) là thực hiện mã hóa toàn bộ câu nguồn thành véc-tơ có chứa các thông tin hữu ích. Véc-tơ kích thước cố định này rất quan trọng, nó là tiền đề để việc dịch sang câu đích đạt chất lượng tốt.

Ở đây, chúng tôi sử dụng véc-tơ trạng thái ẩn  $h_t$  làm véc-tơ có kích thước cố định. Lí do bởi vì trong bi-LSTM, trạng thái ẩn của một từ thứ  $t$  (tại bước thời gian  $t$ ) chứa những thông tin, mối quan hệ giữa từ này và những từ lân cận. Hơn nữa, nó còn chứa thông tin của toàn bộ những từ ở đầu câu cho tới từ thứ  $t$ . Vậy nên trạng thái ẩn của từ



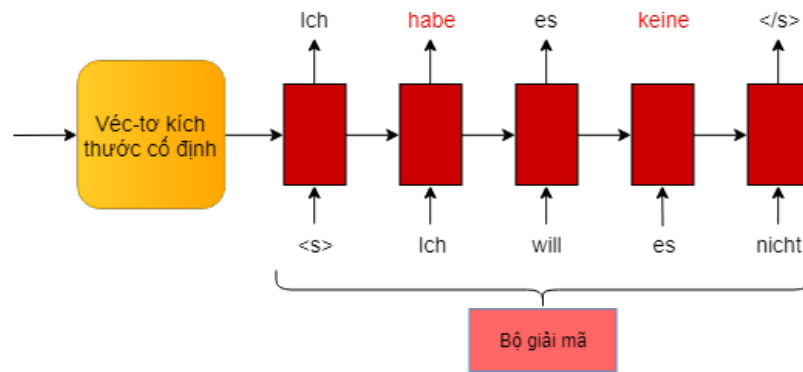
cuối cùng của câu sẽ chứa đựng thông tin của toàn bộ những từ trong câu. Và điều đó phù hợp với ý tưởng của véc-tơ có kích thước cố định của bộ mã hóa.

Một cách tổng quát, bộ mã hóa là một mô hình mà nó mã hóa toàn bộ một đầu vào bất kì thành một véc-tơ bất kì mà véc-tơ đó chứa đầy đủ những thông tin cần thiết để thực hiện các công việc sau này. Trong một số công trình, họ sử dụng thêm một số phép biến đổi trên trạng thái ẩn cuối cùng của bi-LSTM để tạo ra một véc-tơ đầu ra phù hợp với mục đích của họ. Đối với bài toán như Phát sinh câu miêu tả cho ảnh, đầu ra của bộ mã hóa thường là một véc-tơ kích thước cố định chứa đặc trưng hữu ích của toàn bộ bức ảnh. Còn đối với bài toán Tóm tắt văn bản, đầu ra của bộ mã hóa là một véc-tơ chứa toàn bộ thông tin cần thiết dùng cho việc tóm tắt của văn bản đầu vào.

### 3.1.2 Bộ giải mã

Bộ giải mã là một uni-LSTM (LSTM 1 chiều). Uni-LSTM này nhận đầu vào là một véc-tơ kích thước cố định chứa thông tin cần thiết của câu cần dịch. Véc-tơ này là véc-tơ kích thước cố định từ đầu ra của bộ mã hóa. Đầu ra của bộ giải mã là câu đã được dịch sang ngôn ngữ đích  $\mathbf{y} = \{y_0, \dots, y_{S-1}\}$ . Lí do chúng tôi sử dụng uni-LSTM mà không phải là bi-LSTM bởi vì trong quá trình giải mã (dịch) mô hình chỉ biết có được những từ đã được dịch ở trước đó (thông tin trong quá khứ) mà không biết được những từ đằng sau (thông tin trong tương lai) (vì những từ phía sau chưa được dịch). Do đó, việc sử dụng lợi thế của bi-LSTM trong bộ giải mã là không thể và có thể gây ảnh hưởng tới chất lượng dịch của mô hình.

Một điều đáng lưu ý trong việc huấn luyện uni-LSTM trong bộ giải mã đó là đầu vào của uni-LSTM trong mỗi bước thời gian  $t$ . Hình 3.1 minh họa bộ giải mã trong quá trình kiểm thử. Từ được dự đoán ở bước thời gian  $t - 1$  sẽ là đầu vào của uni-LSTM ở bước thời gian  $t$ . Tuy nhiên, việc như vậy sẽ là hạn chế lớn nếu sử dụng nó trong quá trình huấn luyện. Lí do bởi vì tại mỗi bước thời gian  $t$ , khi thực hiện dự đoán từ đích tiếp theo, mô hình sẽ dựa vào những từ ở phía trước để dự đoán, rồi sau đó sẽ tính độ lỗi dựa trên từ đúng được cung cấp trong dữ liệu huấn luyện. Do vậy, có sự phụ thuộc giữa từ hiện tại và các từ phía trước nó trong quá trình dịch. Việc dự đoán tại bước thời gian  $t$  có tốt hay không còn cần phải xem những từ đã được dự đoán ở phía trước có tốt hay không nữa. Nếu những từ ban đầu dự đoán không tốt, tất cả những được dự đoán ở phía sau sẽ càng kém hơn. Để mô hình có thể học được cách dịch một



Hình 3.2: Minh họa phương pháp teacher forcing. Đầu vào tại các bước thời gian là các từ đúng dù ở bước thời gian trước mô hình có đưa ra dự đoán sai.

câu dài với quá trình huấn luyện như trên thì phải tốn nhiều thời gian và công sức tinh chỉnh mô hình. Vì vậy, chúng tôi sử dụng phương pháp huấn luyện mạng nơ-ron hồi quy hiệu quả là *teacher forcing* (tạm dịch là hướng dẫn của giáo viên). Phương pháp này đơn giản là tại mỗi bước thời gian  $t$ , từ đúng tại  $t - 1$  sẽ làm đầu vào của mạng tại bước thời gian  $t$ . Tức là mô hình tại mỗi bước thời gian  $t$  đều nhận được đầu vào đúng bất kể kết quả dự đoán của bước thời gian trước  $t - 1$ . Hình 3.2 minh họa phương pháp teacher forcing. Với phương pháp này, mô hình hội tụ nhanh hơn so với việc sử dụng cách huấn luyện ở trên. Bởi vì khi dịch những câu dài, lúc dự đoán một từ thì không phải phụ thuộc hoàn toàn vào những từ đã được dự đoán trước đó mà dựa vào những từ đúng đã được cung cấp. Khi thực hiện quá trình kiểm thử thì hiển nhiên sẽ phải dùng kết quả của những lần dự đoán phía trước để dự đoán từ tiếp theo hiện tại.

Trong quá trình giải mã, bộ giải mã thực hiện dự đoán từ tiếp theo dựa trên những từ đã biết trước đó:  $p(y_t | y < t, x)$ . Việc quyết định xem chọn từ nào sẽ là từ tiếp theo dựa trên những xác suất  $p_t$  trên tất cả các từ có trong bộ từ vựng  $V$  mà đã tính được là một chiến lược quan trọng trong quá trình giải mã. Với chiến lược đúng đắn, mô hình sẽ cải thiện được chất lượng của đầu ra mà không phải thay đổi bộ tham số của mô hình thông qua các cách như huấn luyện mô hình.

Cách đơn giản nhất để quyết định xem nên chọn từ nào từ phân phối xác suất trên bộ từ vựng  $P(y | y < t, x)$  là chọn ra từ nào có xác suất cao nhất:

$$\hat{y} = \operatorname{argmax}_{y_t \in V} (p(y_t | y < t, x)) \quad (3.1)$$

Cách mà lựa chọn những từ theo tiêu chí có xác suất cao nhất này được gọi là *thuật toán tìm kiếm tham lam (greedy search algorithm)*. Tuy nhiên, cách này trong thực tế cho ra kết quả không thực sự tốt. Để làm rõ hơn về sự hạn chế của thuật toán tìm kiếm tham lam trong thực tế, chúng tôi lấy một ví dụ trong quá trình dịch từ một câu tiếng Đức "Ich schwimme am nächsten Wochenende" sang tiếng Anh như sau: "I am swimming at the next weekend" hoặc "I am going to be swimming playing at the next weekend". Dễ thấy, câu dịch đầu tiên "I am swimming at the next weekend" là câu dịch tốt hơn vì nó ngắn mà vẫn đảm bảo đầy đủ ý của câu. Câu dịch thứ 2 "I am going to be swimming playing at the next weekend" mặc dù vẫn đảm bảo ý nghĩa của câu nhưng lại dài hơn. Vấn đề của tìm kiếm tham lam phát sinh từ đây. Giả sử, mô hình thực hiện quá trình giải mã và đã dịch được 2 từ đầu tiên "I am". Để đơn giản, chúng tôi giả định mô hình đang trong quá trình kiểm thử nên không sử dụng teacher forcing. Khi đó, xác suất của từ tiếp theo khi cho trước 2 từ "I am":

$$\hat{y}_t = \operatorname{argmax}_{y_t \in V} p(y_t | I, am)$$

Trong tiếng Anh, dễ nhận thấy rằng tần số xuất hiện của "I am going" lớn hơn của "I am swimming", do đó rất có khả năng bộ giải mã sẽ tính toán ra được rằng  $p(\text{going} | I, am) > p(\text{swimming} | I, am)$  và dẫn tới câu kết quả thứ 2 "I am going to be swimming at the next weekend" mà không phải là câu dịch tốt hơn. Ví dụ chúng tôi vừa trình bày thể hiện rằng với thuật toán tìm kiếm tham lam, kết quả mà bộ giải mã đưa ra sẽ bị ảnh hưởng rất nhiều bởi những từ có tần số xuất hiện lớn trong tập dữ liệu huấn luyện. Do đó, mô hình sẽ thường có xu hướng cho ra những câu dịch mà trong đó những từ phổ biến như the, a, an, and, of, to, v.v... và tạo ra những câu dịch không thực tế, không phù hợp ngữ pháp, gây giảm chất lượng dịch của mô hình đi đáng kể. Thực chất, vấn đề của tìm kiếm tham lam là chỉ lựa chọn một từ duy nhất mà có xác suất có điều kiện  $p(y_t | y < t, x)$  lớn nhất và chỉ xét một *câu giả thuyết (hypothesis)* duy nhất trong quá trình giải mã. Khi bộ giải mã dịch một câu, sẽ có nhiều câu ứng viên, mỗi câu ứng viên này chúng tôi gọi là một câu giả thuyết.

Khi thực hiện công việc dịch, mọi người nhìn vào câu nguồn và dịch ra từng từ ở câu đích. Tuy nhiên, ngôn ngữ tự nhiên rất phức tạp, mỗi một ngôn ngữ sẽ có ngữ pháp, hành văn khác nhau cũng như mối quan hệ giữa các từ trong câu cũng sẽ có trật

tự riêng nên khó có thể dịch nhanh chóng trong một lần. Do đó, sau khi dịch được một số lượng từ nhất định hay hết một ý, mọi người cần phải nhìn lại những từ đã dịch và chỉnh sửa lại sao cho câu văn trôi chảy, mạch lạc, ý nghĩa hoàn thiện hơn.

Từ hạn chế của tìm kiếm tham lam và quá trình dịch trong thực tế, để dịch câu tối ưu hơn, mô hình sẽ dùng sử dụng tất cả tổ hợp của các từ để xây dựng thành các câu giả thuyết và đánh giá xem câu nào là tốt nhất bằng xác suất của các câu giả thuyết đó thông qua mô hình ngôn ngữ, câu tối ưu  $\hat{y}$  được lựa chọn dựa trên biểu thức:

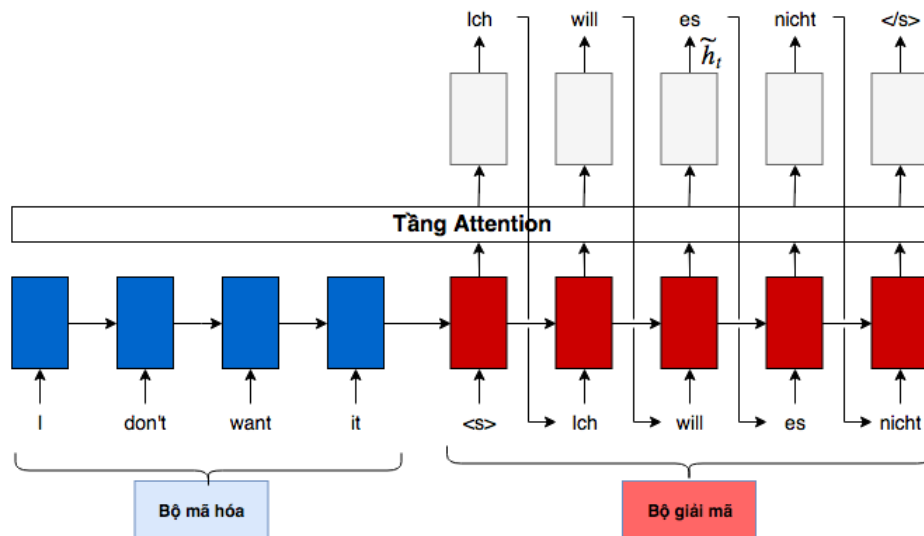
$$\hat{y} = \underset{y}{\operatorname{argmax}} p(y)$$

Tuy nhiên, với số lượng từ vựng lớn, ví dụ bộ từ vựng  $V = 10000$  từ thì số lượng tổ hợp câu giả thuyết có thể có là  $V^S$ , với  $S$  là số từ trong câu. Với cách này, mô hình có thể đảm bảo được rằng câu dịch ra là tối ưu dựa trên trạng thái mô hình hiện có nhưng chi phí tính toán rất lớn và lưu trữ rất lớn. Do vậy, việc mà có thể làm được hiện tại là tìm kiếm gần đúng trên không gian tổ hợp các câu giả thuyết đó. Thuật toán tìm kiếm gần đúng này được gọi là beam search (tạm dịch: tìm kiếm theo chùm).

## 3.2 Cơ chế Attention

Ở phần trước, chúng tôi đã trình bày về kiến trúc Bộ mã hóa-Bộ giải mã cùng với những điểm mạnh của nó trong việc giải quyết bài toán Dịch máy. Tuy nhiên, kiến trúc này vẫn còn tồn tại hạn chế về việc dịch những câu dài do những thông tin được mã hóa của câu nguồn bị mất dần theo các bước thời gian về sau. Lí do mà vấn đề này tồn tại thực chất là bởi vì các mô hình LSTM được sử dụng trong Bộ mã hóa và Bộ giải mã. Bản thân mô hình LSTM chưa thật sự giải quyết hoàn toàn vấn đề "sự phụ thuộc dài hạn". Để có thể vẫn tận dụng được các mô hình LSTM mà vẫn nâng cao được chất lượng dịch, chúng tôi sử dụng cơ chế Attention.

Trước khi đi vào cách hoạt động của cơ chế Attention, chúng tôi đi qua một chút về nguồn cảm hứng và lịch sử của cơ chế này. Cơ chế Attention được lấy cảm hứng trên cơ chế đặt sự chú ý khi quan sát sự vật, hiện tượng của thị giác con người. Khi con người quan sát một sự vật, hiện tượng nào đó bằng mắt, con người chỉ có thể tập trung vào một vùng nhất định trên sự vật, hiện tượng được quan sát để ghi nhận



Hình 3.3: Minh họa cơ chế Attention. Một tầng Attention được đặt ở trước bước dự đoán đầu ra của bộ giải mã.

thông tin. Sau đó, khi cần ghi nhận thêm thông tin khác, con người sẽ di chuyển vùng tập trung lên vật thể của mắt sang vị trí khác. Những vùng lân cận xung quanh vùng tập trung sẽ bị "mờ" hơn so với vùng tập trung. Cơ chế Attention đã được ứng dụng trong lĩnh vực Thị giác máy tính từ khá lâu [?] [2]. Vào những năm gần đây, cơ chế Attention được sử dụng cho các kiến trúc mạng nơ-ron hồi quy trên bài toán Dịch máy và đã đạt được những kết quả ấn tượng.

Cơ chế Attention được sử dụng trong đề tài này là một cơ chế sử dụng thông tin trong các trạng thái ẩn của RNN trong bộ mã hóa khi thực hiện quá trình giải mã. Cụ thể là:

- Trong quá trình giải mã, trước khi dự đoán đầu ra, bộ giải mã nhìn vào các thông tin nằm trong các trạng thái ẩn của RNN ở bộ mã hóa.
- Ở mỗi phần tử đầu ra tại bước thời gian  $t$ , bộ giải mã dựa vào trạng thái ẩn tại bước thời gian  $t$  hiện tại và quyết định sử dụng các thông tin trong trạng thái ẩn ở bộ mã hóa như thế nào.

2 phiên bản Toàn cục và Cục bộ mà trong khóa luận này chúng tôi trình bày là 2 cách mà cơ chế Attention sử dụng các trạng thái ẩn của RNN trong bộ mã hóa. Để làm rõ hơn về ý tưởng của cơ chế Attention, dưới đây chúng tôi sẽ trình bày chi tiết về nền tảng Toán học của nó. Attention sử dụng thêm một số đại lượng:

- $a_t$ : trọng số giống hàng,  $a_t$  được tính theo công thức dưới đây:

$$a_t = \text{align}(h_t, \bar{h}_s) = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'} \exp(\text{score}(h_t, \bar{h}_{s'}))} \quad (3.2)$$

$a_t$  là một véc-tơ chứa các điểm số giữa trạng thái ẩn ở bước thời gian  $t$   $h_t$  và các trạng thái ẩn ở câu nguồn  $\bar{h}_s$ . Hàm điểm số  $\text{score}$  mà chúng tôi sử dụng là gồm 2 hàm:

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} h_t^T \bar{h}_s & \text{dot} \\ h_t^T W_a \bar{h}_s & \text{general} \end{cases} \quad (3.3)$$

Đối với hàm  $\text{score}$  là hàm *dot*, mô hình chỉ đơn giản là thực hiện tính tích vô hướng giữa 2 trạng thái ẩn. Giá trị của hàm  $\text{score}$  này đạt cao nhất khi 2 véc-tơ trạng thái ẩn hoàn toàn giống nhau. Ưu điểm của hàm *dot* này là chi phí tính toán thấp nên thời gian huấn luyện và dự đoán nhanh. Đối với hàm  $\text{score}$  là hàm *general*, hàm này có sự tinh tế hơn hàm *dot*. Hàm *dot* thực hiện tính sự tương đồng lên tất cả cặp phần tử trong 2 véc-tơ, trong khi đó hàm *general* sử dụng thêm một bộ trọng số  $W_a$ , do đó những thông tin giữa hai trạng thái ẩn sẽ được tính một cách chọn lọc hơn. Tuy nhiên, đổi lại thì hàm này sẽ có thời gian thực thi chậm hơn hàm *dot* một chút. Trong thực tế, không có minh chứng rõ ràng nào cho thấy rằng hàm nào sẽ tốt hơn, do vậy cần phải thực nghiệm cẩn thận để có được sự lựa chọn chính xác nhất.

- $c_t$ : véc-tơ ngữ cảnh tại bước thời gian  $t$ , là trung bình có trọng số của các trạng thái ẩn ở câu nguồn:

$$c_t = \sum_s a_{ts} h_s \quad (3.4)$$

Véc-tơ  $c_t$  cho mô hình biết thông tin rằng với trạng thái ẩn hiện tại (chứa thông tin của quá trình dịch trước đó) thì ngữ cảnh hiện tại của bước thời gian  $t$  là gì. Ngữ cảnh đó được thể hiện thông qua những thông tin của các trạng thái ẩn  $h_s$  của câu nguồn mà được lựa chọn một cách có chọn lọc (có trọng số). Véc-tơ ngữ cảnh  $c_t$  là một cách biểu diễn ngữ cảnh của ngôn ngữ đích bằng ngữ cảnh của ngôn ngữ nguồn. Trong quá trình dịch, bộ giải mã cần phải dự đoán từ tiếp theo của câu dịch. Để dự đoán được chính xác, mô hình cần phải biết được ngữ cảnh hiện tại của câu là gì. Để đảm bảo ngữ cảnh mà mô hình nhận được chính

xác, mô hình không thể chỉ dựa vào các trạng thái ẩn của bộ giải mã ở các bước thời gian trước đó. Do vậy, mô hình sử dụng thêm các trạng thái ẩn của các từ ở câu nguồn để thể hiện ngữ cảnh một cách chính xác hơn.

- $\tilde{h}_t$ , véc-tơ attention tại bước thời gian  $t$ , được tính như sau:

$$\tilde{h}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t]) \quad (3.5)$$

Véc-tơ attention chứa thông tin giống hàng và trạng thái ẩn của bước thời gian  $t$  hiện tại. Nhờ đó, mô hình nắm giữ được nhiều thông tin hơn để có thể dự đoán tốt hơn.

Bước dự đoán đầu ra không thay đổi ngoài trạng thái ẩn  $\mathbf{h}_t$  được thay thế bởi véc-tơ attention  $\tilde{h}_t$ .  $\tilde{h}_t$  được đưa qua tầng softmax để cho ra phân bố xác suất dự đoán trên các từ:

$$p(y_t | y_{<t}, x) = \text{softmax}(\mathbf{W}_s \tilde{\mathbf{h}}) \quad (3.6)$$

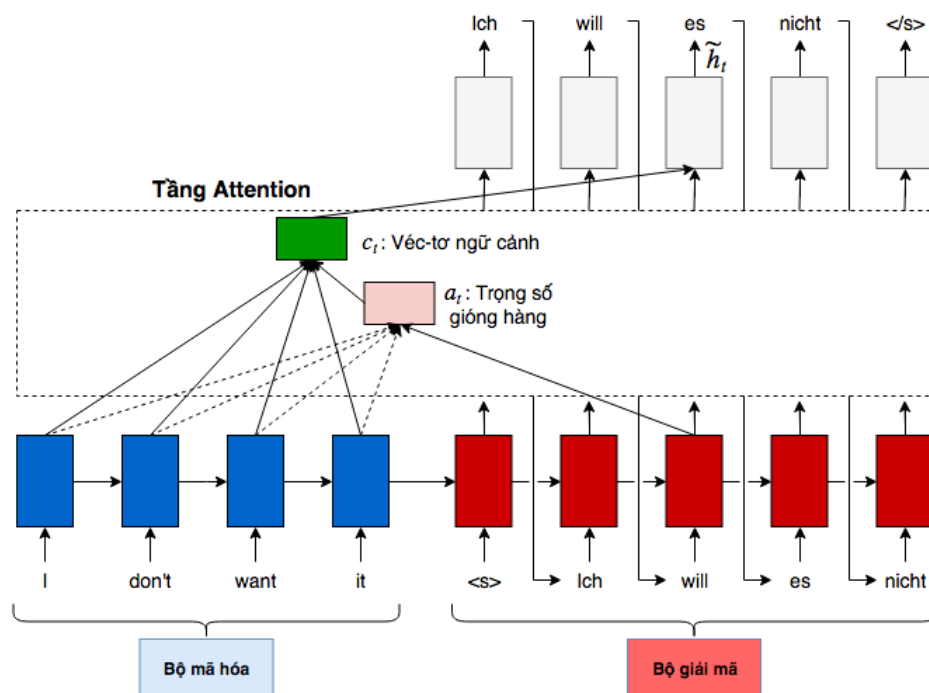
Nói một cách đơn giản, mục tiêu của cơ chế Attention là xoay quanh việc tìm véc-tơ ngữ cảnh  $c_t$  một cách hiệu quả. Tiếp theo, chúng tôi trình bày chi tiết hơn về 2 phiên bản Toàn cục và Cục bộ. 2 phiên bản này chỉ khác nhau về cách suy ra véc-tơ ngữ cảnh  $\mathbf{c}_t$ , còn các bước còn lại giống nhau. Quy trình tính toán của cơ chế Attention:  $h_t -> a_t -> c_t -> \tilde{h}_t$

### 3.3 Attention Toàn cục

Ý tưởng của Attention toàn cục là nhìn vào toàn bộ các vị trí nguồn (các trạng thái ẩn của RNN ở bộ mã hóa) khi thực hiện giải mã. Khi đó trọng số giống hàng  $a_t$  là một véc-tơ có kích thước thay đổi và bằng số trạng thái ẩn (số từ) ở câu nguồn:  $\text{len}(a_t) = S$ .

$$a_t = \text{align}(h_t, \bar{h}_s) = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'=1}^S \exp(\text{score}(h_t, \bar{h}_{s'}))} \quad (3.7)$$

Hình 3.4 minh họa cơ chế Attention Toàn cục cho thấy rằng tại bước thời gian  $t$ , trước khi thực hiện dự đoán từ tiếp theo, mô hình đặt "sự chú ý" lên toàn bộ các trạng thái ẩn ở câu nguồn hay bộ mã hóa.



Hình 3.4: Minh họa cơ chế Attention Toàn cục. Tại bước thời gian  $t$ , bộ giải mã nhìn vào toàn bộ trạng thái ẩn ở các vị trí nguồn. Trọng số giống hàng và véc-tơ ngữ cảnh được tính dựa trên những trạng thái ẩn được "nhìn" bởi bộ giải mã. Sau đó véc-tơ ngữ cảnh sẽ được nối với trạng thái ẩn ở bộ giải mã ở bước thời gian hiện tại để tạo thành véc-tơ attention. Sau đó mô hình sẽ dự đoán từ tiếp theo dựa trên véc-tơ attention này.



Ưu điểm của phương pháp này là ý tưởng đơn giản, dễ cài đặt nhưng vẫn đạt được hiệu quả tốt (sẽ được trình bày ở phần thực nghiệm). Tuy nhiên, ý tưởng này vẫn còn chưa thực sự tự nhiên và còn hạn chế. Khi dịch một từ thì không cần phải đặt "sự chú ý" lên toàn bộ câu nguồn, chỉ cần đặt "sự chú ý" lên một số từ cần thiết. Mặc dù khi mô hình Attention Toàn cục được huấn luyện tốt thì hoàn toàn có thể chỉ đặt "sự chú ý" lên một số từ thật sự cần thiết. Trong thực tế, để đạt được độ chính xác như thế thì phải tiêu tốn nhiều tài nguyên cho việc huấn luyện mô hình như tài nguyên về tập dữ liệu đủ lớn, đủ tốt hay thời gian huấn luyện phải đủ lâu. Nhưng dễ thấy rằng dù bản thân mô hình đã học được cách đặt "sự chú ý" thật tốt nhưng vẫn phải tiêu tốn chi phí cho việc tính toán trọng số giống hàng  $a_t$  cho những vị trí không cần thiết. Để giải quyết hạn chế trên của Attention Toàn cục, chúng tôi đã tìm hiểu và sử dụng phiên bản tinh tế hơn, đó là mô hình Attention Cục bộ. Ở phần tiếp theo, chúng tôi sẽ trình bày về mô hình này.

### 3.4 Attention Cục bộ

Như đã nêu ở phần trước, Attention Toàn cục có một hạn chế là đặt "sự chú ý" lên toàn bộ các từ ở câu nguồn khi dịch từng từ ở câu đích. Điều này gây tiêu tốn chi phí tính toán và có thể tạo ra những câu dịch không thực tế khi dịch những câu dài như trong các đoạn văn hay trong một tài liệu. Attention Cục bộ ra đời để giải quyết hạn chế này. Lưu ý, ý tưởng của cơ chế Attention Cục bộ này dựa vào đặc điểm của 2 cặp ngôn ngữ tiếng Anh và tiếng Đức. Do đó, sự hiệu quả của cơ chế này không đảm bảo cho các cặp ngôn ngữ khác.

Khi dịch mỗi từ ở câu đích, Attention Cục bộ chỉ đặt "sự chú ý" lên một số từ gần nhau ở câu nguồn. Mô hình này lấy cảm hứng từ sự đánh đổi giữa 2 mô hình "soft attention" và "hard attention" được đề xuất trong công trình Show, Attend and Tell [18] để giải quyết bài toán Phát sinh câu miêu tả cho ảnh (Image Captioning). Trong công trình [18], Attention Toàn cục tương ứng với "soft attention", "sự chú ý" được đặt trên toàn bộ bức ảnh. Còn "hard attention" thì đặt "sự chú ý" lên một số phần của bức ảnh.

Dễ thấy, với cách hoạt động chỉ tập trung một số các từ gần nhau ở câu nguồn, mô hình hoạt động gần với cách con người tập trung vào một sự vật, hiện tượng nào đó.

Chi phí cho huấn luyện và dự đoán sẽ được giảm bớt bởi vì chúng ta chỉ thực hiện tính véc-tơ trọng số giống hàng  $a_t$  cho những từ mà mô hình đặt "sự chú ý" lên.

Để làm rõ hơn về cách thức hoạt động của mô hình Attention Cục bộ, chúng tôi sẽ trình bày cụ thể hơn về nền tảng Toán học của mô hình này. Bên cạnh những đại lượng đã có ở mô hình Attention Toàn cục, Attention Cục bộ có thêm và thay đổi một số đại lượng như sau:

- $p_t$ : vị trí đã được giống hàng. Tại mỗi bước thời gian  $t$ , mô hình sẽ phát sinh một số thực  $p_t$ . Số thực này có giá trị nằm trong đoạn  $[0, S]$  với ý nghĩa rằng đây là vị trí đã được giống hàng với từ ở câu nguồn với từ ở câu đích tại bước thời gian  $t$  hiện tại. Hay nói cách khác, "sự chú ý" được đặt trên từ có vị trí  $p_t$  này. Để ý thấy rằng có sự không tự nhiên khi  $p_t$  là một số thực, do vậy  $p_t$  không thể cho biết được chính xác từ nào sẽ được đặt "sự chú ý" lên, do vậy ta làm tròn xuống giá trị  $p_t$  này để dễ chọn vị trí trung tâm của sổ  $D$ . Để làm rõ hơn về vấn đề này, chúng tôi sẽ trình bày rõ ràng hơn ở sau.
- Đối quá trình tính véc-tơ ngữ cảnh  $c_t$  có sự thay đổi rằng mô hình xét các vị trí ở câu nguồn mà nằm xung quanh vị trí  $p_t$  một đoạn  $D$ .  $D$  là một đại lượng với miền số nguyên lớn hơn 0 và được gọi là kích thước cửa sổ. Cụ thể:

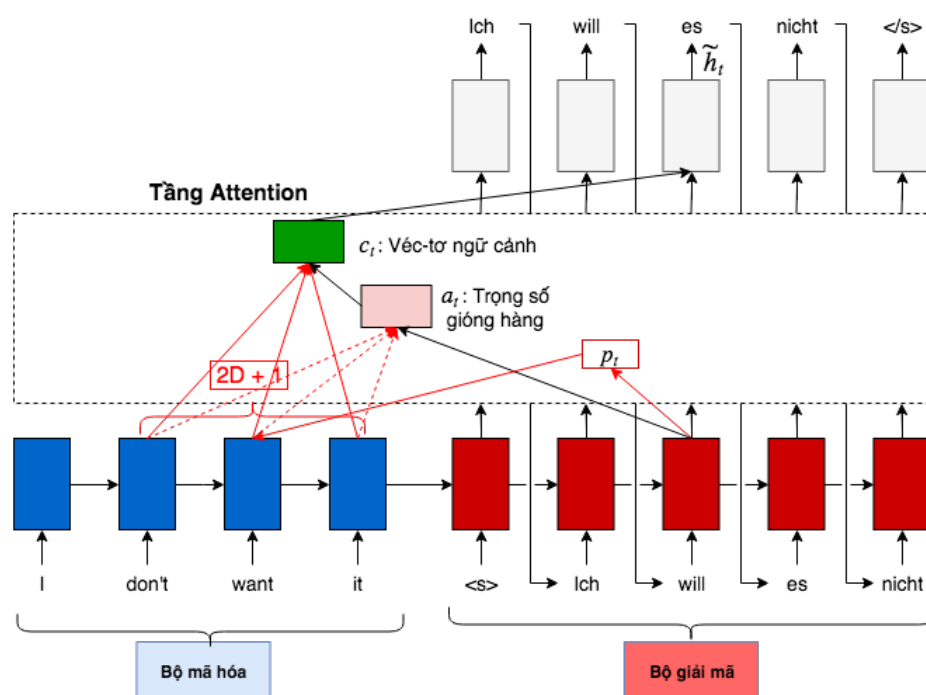
$$c_t = \sum_{x \in [p_t - D, p_t + D]} a_{tx} \tilde{h}_x \quad (3.8)$$

$D$  là một siêu tham số của mô hình. Việc lựa chọn giá trị của  $D$  là dựa vào thực nghiệm. Theo đề xuất của [11], chúng tôi lựa chọn  $D = 10$ .

Hình 3.5 minh họa cơ chế Attention Cục bộ cho thấy cách hoạt động của Attention Cục bộ cùng với sự khác biệt giữa Attention Cục bộ và Attention Toàn cục.

Mô hình Attention Cục bộ có 2 biến thể:

- Giống hàng đều (monotonic alignment - local-m): vị trí được giống hàng được phát sinh một cách đơn giản bằng cách cho  $p_t = t$  tại mỗi bước thời gian  $t$ . Ta giả định rằng các từ ở câu nguồn và các từ ở câu đích được giống hàng đều nhau theo từng từ.



Hình 3.5: Minh họa cơ chế Attention Cục bộ. Tại bước thời gian  $t$ , bộ giải mã nhìn vào một số trạng thái ẩn ở các vị trí nguồn nằm trong phạm vi của cửa sổ có kích thước  $2D + 1$ . Trọng số giống hàng và véc-tơ ngữ cảnh được tính dựa trên những trạng thái ẩn được bộ giải mã đặt sự chú ý.

- Giống hàng dự đoán (predictive alignment - local-p): giả định rằng tất cả từ ở câu nguồn và câu đích đều được giống hàng đều nhau không thực tế vì giữa 2 ngôn ngữ có ngữ pháp riêng và trật tự từ khác nhau. Do vậy, mô hình sẽ phát sinh vị trí được giống hàng  $p_t$  một cách tự nhiên hơn cho phù hợp đặc điểm của ngôn ngữ. Cụ thể mô hình sẽ phát sinh vị trí  $p_t$  tại mỗi bước thời gian  $t$  như sau:

$$p_t = S \cdot \text{sigmoid}(v_p^T \tanh(W_p h_t)) \quad (3.9)$$

Trong đó,  $v_p$  và  $W_p$  là 2 tham số mới của mô hình dùng cho việc dự đoán vị trí  $p_t$ . Mô hình cần học 2 tham số này để có thể dự đoán vị trí  $p_t$  được chính xác. Miền giá trị của  $p_t \in [0, S]$ . Để "ưu tiên" các vị trí được giống hàng  $p_t$ , mô hình thêm vào trọng số giống hàng của những từ lân cận đó một lượng có giá trị bằng giá trị của phân phối chuẩn (Gauss) mà đã được đơn giản hóa (chưa được chuẩn hóa) với giá trị kì vọng  $p_t$  và độ lệch chuẩn  $\sigma = \frac{D}{2}$ :

$$p_t = \text{align}(h_t, \bar{h}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \quad (3.10)$$

Mô hình sử dụng hàm giống hàng như các phiên bản trước.  $s$  là giá trị số nguyên thể hiện các vị trí nằm xung quanh  $p_t$  mà nằm trong cửa sổ  $D$ .

Đối với những vị trí  $s$  nằm ngoài câu (cửa sổ  $D$  vượt qua các biên của câu) thì mô hình sẽ bỏ qua những vị trí  $s$  nằm ngoài và chỉ xem xét những vị trí  $s$  nằm trong biên của câu.

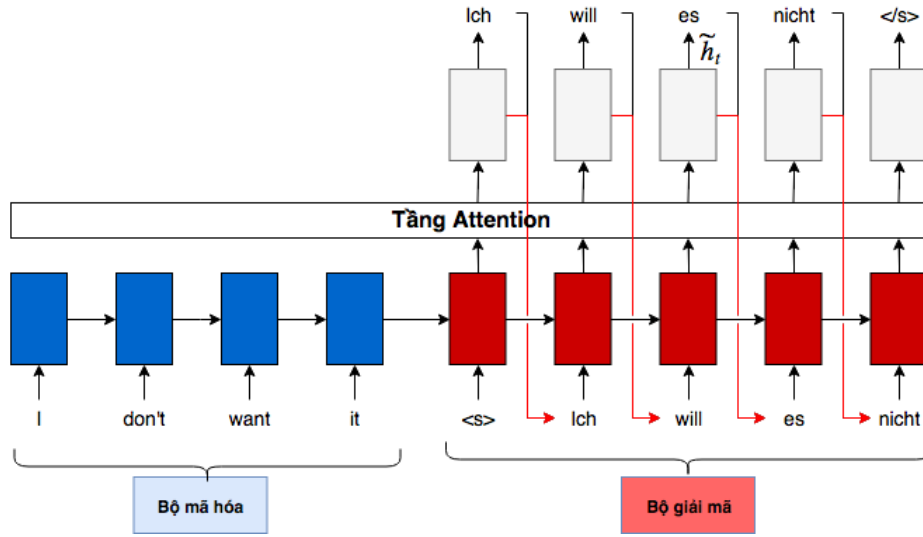
Véc-tơ trọng số giống hàng  $a_t$  ở Attention Cục bộ có kích thước cố định  $\in \mathbb{R}^{2D+1}$  và thường ngắn hơn  $a_t$  ở Attention Toàn cục. Local-p và local-m giống nhau chỉ khác rằng local-p tính vị trí  $p_t$  một cách linh hoạt và sử dụng một phân phối chuẩn đã được đơn giản hóa để điều chỉnh các trọng số giống hàng  $\text{align}(h_t, \bar{h}_s)$ . Việc sử dụng thêm phân phối chuẩn để khuyến khích mô hình đặt "sự chú ý" lên vị trí  $p_t$  và phân chia dần cho các vị trí lân cận. Nếu không có việc sử dụng phân phối chuẩn này, mô hình có thể sẽ đặt "sự chú ý" hoàn toàn lên các từ lân cận xung quanh  $p_t$  mà không phải là vị trí  $p_t$ . Điều này không phù hợp với ý tưởng ban đầu của việc phát sinh vị trí  $p_t$  do vị trí này thể hiện ý nghĩa rằng bộ giải mã đang tập trung vào những từ ở vị trí  $p_t$ . Với cơ chế được trình bày cụ thể như trên, mô hình Attention Cục bộ hoạt động tự nhiên

hơn, phù hợp với ý tưởng về cách con người đặt "sự chú ý" khi quan sát sự vật, hiện tượng. Bên cạnh đó, Attention Cục bộ giảm chi phí tính toán của mô hình khi chỉ thực hiện tính trên những từ được chú ý nằm trong phạm vi cửa sổ nhất định khi câu nguồn dài.

### 3.5 Phương pháp Input feeding

Trong quá trình dịch, các mô hình được đề cập ở trên như Attention Toàn cục hay Cục bộ, đều vẫn còn một hạn chế về cách đặt "sự chú ý" hay giống hàng lên các vị trí nguồn. Ở mỗi bước thời gian  $t$  khi dịch một từ ở câu đích, việc đặt "sự chú ý" độc lập hoàn toàn với việc đặt "sự chú ý" của các bước thời gian trước đó. Việc quyết định giống hàng như thế nào (giá trị của véc-tơ  $a_t$ ) hoàn toàn phụ thuộc vào điểm số (giá trị của hàm score) giữa trạng thái ẩn  $h_t$  hiện tại và các trạng thái ẩn  $\bar{h}_s$  ở câu nguồn. Trong thực tế, khi dịch, một từ ở câu nguồn chỉ tương ứng với một vài từ ở câu đích. Do vậy, mô hình cần phải theo dõi xem là những từ nào ở câu nguồn đã được dịch trước đó thì hạn chế đặt "sự chú ý" lên lại những từ đó. Việc không có cơ chế kiểm soát những từ nào đã được dịch sẽ khiến cho mô hình sẽ rơi vào 2 trường hợp "được dịch quá nhiều" (over-translated) hoặc "được dịch quá ít" (under-translated). Tức là có một số từ ở câu nguồn sẽ được đặt "sự chú ý" lên quá nhiều lần dẫn tới bỏ qua những từ quan trọng khác hoặc là một số từ quan trọng được đặt "sự chú ý" lên quá ít dẫn tới việc bỏ qua thông tin của từ đó trong quá trình dịch. Dù là trường hợp nào thì cũng gây giảm chất lượng dịch của mô hình và cho ra những câu dịch không thực tế.

Trong Dịch máy Thống kê, Koehn et al. 2003 [10] đã đề xuất một mô hình dịch dựa trên cụm từ (phrase-based) mà có cơ chế để giải quyết vấn đề trên. Cơ chế này rất đơn giản và trực quan. Trong quá trình dịch, bộ giải mã duy trì một véc-tơ bao phủ (coverage vector) để chỉ ra rằng từ ở câu nguồn nào đã được dịch hoặc chưa được dịch. Quá trình dịch được hoàn thành khi toàn bộ từ ở câu nguồn được "bao phủ" hay đã được dịch. Trong khi đó, các mô hình Dịch máy nơ-ron hiện nay chỉ kết thúc quá trình dịch khi và chỉ khi gặp kí tự kết thúc câu hoặc vượt quá số lượng từ cho trước. Việc này dễ dẫn đến trường hợp "được dịch quá nhiều" khi kí hiệu kết thúc câu xuất hiện trễ hay ngược lại dẫn đến trường hợp "được dịch quá ít" khi kí hiệu kết thúc câu xuất hiện sớm. Ngoài ra còn bị ảnh hưởng bởi số lượng từ quy định khi dịch.



Hình 3.6: Minh họa phương pháp Input feeding. Tại bước thời gian  $t$ , bộ giải mã nhận đầu vào gồm véc-tơ attention ở bước thời gian trước đó  $t - 1$  và từ hiện tại  $x_t$ .

Công trình [11] đề xuất một cơ chế góp phần giải quyết vấn đề ở trên: (tạm dịch là "cho đầu vào ăn" // TODO: dịch khác). Ý tưởng và cách thực hiện của Input feeding rất đơn giản. Nhận thấy véc-tơ attention  $\tilde{h}_{t-1}$  lưu giữ thông tin giống hệt của bước thời gian  $t - 1$  trước đó, mô hình thực hiện nối véc-tơ  $\tilde{h}_{t-1}$  với đầu vào  $x_t$  của bước thời gian  $t$  hiện tại. Bằng cách như vậy, mô hình có thể nắm được thông tin giống hệt trước đó từ  $\tilde{h}_{t-1}$ .

$$x'_t = [x_t, \tilde{h}_t] \quad (3.11)$$

Tuy nhiên, phương pháp này chưa thực sự giải quyết triệt để vấn đề "được dịch quá nhiều" hay "được dịch quá ít". Vì mô hình chỉ nhận được thông tin giống hệt từ các bước thời gian trước đó nhưng lại không được hướng dẫn hay có ràng buộc cụ thể nào mà để giải quyết vấn đề này. Việc giải quyết vấn đề trên hoàn toàn phụ thuộc vào quyết định của mô hình. Mặc dù chưa thực sự giải quyết triệt để, nhưng lại cho mô hình tăng thêm tính mềm dẻo trong việc sử dụng thông tin giống hệt trong quá khứ. Thực tế, phương pháp này đã cải thiện chất lượng dịch lên đáng kể và đơn giản về mặt cài đặt.

Ngoài ra, phương pháp này giúp cho mô hình phức tạp hơn nhờ vào việc tăng kích thước đầu vào của mô hình, do đó làm tăng khả năng học của mô hình.

### 3.6 Kỹ thuật thay thế từ hiếm

Trong quá trình dịch thuật, có rất nhiều hạn chế gây ảnh hưởng tới chất lượng của bản dịch. Trong phần này, chúng tôi đề cập tới một vấn đề quan trọng mà dù là con người hay máy tính đều gặp phải và rất khó giải quyết. Đó là vấn đề về những "từ hiếm" (unknown words).

Mỗi ngôn ngữ có muôn hình vạn trạng các từ ngữ khác nhau. Số lượng từ ngữ trong một ngôn ngữ là không có định. Trong quá trình hình thành và phát triển ngôn ngữ, theo thời gian số lượng từ ngữ sẽ tăng lên hoặc mất đi (bị lãng quên hay không dùng nữa) tùy thuộc vào hoàn cảnh, môi trường sử dụng của ngôn ngữ đó. Nhưng thường đối với những ngôn ngữ phổ biến hiện nay thì số lượng từ ngữ tăng lên lớn hơn nhiều so với số lượng từ ngữ mất đi. Khi xã hội phát triển, nhu cầu giao tiếp giữa các dân tộc, quốc gia, nền văn hóa khác nhau cũng tăng theo. Mỗi nơi lại có cách sử dụng ngôn ngữ khác nhau, do đó bộ từ vựng của mỗi ngôn ngữ cũng phải thay đổi sao cho phù hợp với nhu cầu giao tiếp. Khoa học kỹ thuật phát triển kèm theo đó là những khám phá về thế giới tự nhiên. Những sự vật, hiện tượng mới được phát hiện ngày càng nhiều. Và không phải sự vật, hiện tượng nào cũng có thể được mô tả, thể hiện bằng những vốn từ vựng vốn có của một số ngôn ngữ. Ngoài ra còn có nhiều lí do làm cho bộ từ vựng của các ngôn ngữ thay đổi theo thời gian.

Với tốc độ phát triển của ngôn ngữ là như vậy nhưng khả năng của con người là hữu hạn. Một người dù có thông thạo một ngôn ngữ tới đâu thì cũng không thể nào biết được hết tất cả từ vựng của ngôn ngữ đó. Theo thống kê, số lượng từ ngữ cần để giao tiếp hàng ngày trong tiếng Anh chỉ khoảng từ 2000-3000 từ, đối với lĩnh vực chuyên ngành thì khoảng 5000-6000 từ. Nhưng theo kích thước của một số bộ từ điển thịnh hành trong tiếng Anh thì số lượng từ vựng của những bộ từ điển đó khoảng 60000 từ. Tức là đa số mọi người chưa biết hết được 10% từ vựng của tiếng Anh. Do vậy khi thực hiện việc dịch thuật giữa các ngôn ngữ với nhau, mọi người chỉ có thể dịch tốt khi văn bản, hội thoại cần dịch thuộc về chủ đề mà họ quen thuộc. Mọi người sẽ gặp khó khăn khi gặp những từ nằm ngoài bộ từ vựng của bản thân (out-of-vocabulary words - OOV words) vì không biết phải dịch như thế nào.

Khi huấn luyện một mô hình Dịch máy thì cần phải có một bộ từ vựng cố định cho mô hình đó trong suốt quá trình huấn luyện và dự đoán. Kích thước của bộ từ vựng

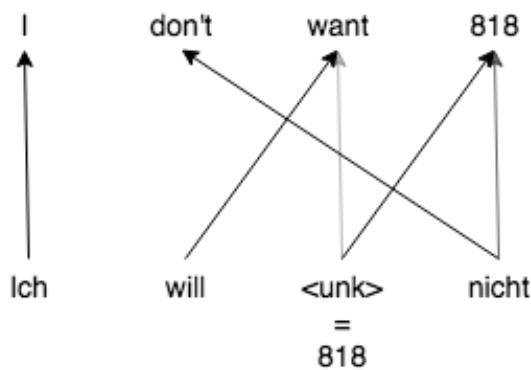
này bị hạn chế với số lượng nhất định. Sự hạn chế về kích thước này xuất phát từ nhiều lí do như giới hạn về dữ liệu huấn luyện, khả năng học của mô hình, tài nguyên tính toán (phần cứng), v.v... Do vậy việc quyết định xem những từ nào sẽ được đưa vào bộ từ vựng của mô hình cũng rất quan trọng. Thông thường có 2 chiến thuật để xây dựng bộ từ vựng này. Cách đầu tiên phù hợp cho việc phát triển các ứng dụng là lấy các từ vựng có trong dữ liệu huấn luyện làm bộ từ vựng và lọc ra những từ nào có tần số xuất hiện trong dữ liệu huấn luyện thấp hơn một ngưỡng nhất định (ví dụ: lọc ra những từ vựng nào có tần số xuất hiện ít hơn 10). Cách thứ 2 thường phù hợp cho việc nghiên cứu, đó là lựa chọn số lượng từ vựng nhất định mà có tần số xuất hiện cao nhất (ví dụ: lấy 50000 từ có tần số xuất hiện cao nhất). Do đó có những từ xuất hiện trong dữ liệu huấn luyện nhưng vì có tần số xuất hiện thấp nên bị coi là từ nằm ngoài bộ từ vựng (OOV). Đó là lí do chúng tôi gọi đây là vấn đề "từ hiếm".

Có nhiều cách để giải quyết vấn đề này, cách mà mọi người hay sử dụng nhất là thêm từ mới đó vào bộ từ vựng. Cách thứ 2 là giữ nguyên từ đó và đưa nó vào vị trí thích hợp trong câu ở ngôn ngữ đích. Trong khóa luận này chúng tôi sẽ sử dụng cách thứ 2 để giải quyết vấn đề các từ nằm ngoài bộ từ vựng.

Kĩ thuật thay thế từ hiếm mà chúng tôi trình bày sau đây là một phương pháp dựa trên kết quả của cơ chế Attention. Do vậy, hiệu quả của phương pháp này phụ thuộc lớn vào độ chính xác của cơ chế Attention. Kĩ thuật này chúng tôi sử dụng từ bài báo của Jean et al., 2015 [8] về sử dụng cơ chế Attention trong mô hình Dịch máy nơ-ron. Nếu mô hình không sử dụng cơ chế Attention thì cũng không sử dụng được phương pháp thay thế từ hiếm này. Kĩ thuật này chỉ được sử dụng trong quá trình dự đoán, trong quá trình huấn luyện thì không sử dụng. Cách hoạt động của phương pháp này rất đơn giản. Sau khi mô hình đã dự đoán (dịch) xong một câu, mô hình sẽ thực hiện xử lý những từ nào mà được dự đoán là từ hiếm (unknown words) trong câu đã được dự đoán (những từ hiếm được ký hiệu là  $\langle unk \rangle$ ). Đối với mỗi từ hiếm, mô hình sẽ thực hiện dịch lại từ đó bằng cách chọn một từ phù hợp trong câu nguồn rồi thực hiện sao chép từ được chọn vào từ hiếm hiện tại. Cách mà mô hình lựa chọn từ phù hợp là dựa vào véc-tơ trọng số giống hàng  $a_t$ . Mô hình sẽ lựa chọn từ nào có trọng số cao nhất.

Trong thực tế, kĩ thuật này giúp cho mô hình có thể dịch được chính xác những câu có chữ số, số, tên riêng, tên địa danh v.v... Bởi vì những từ này rất ít xuất hiện





Hình 3.7: Minh họa phương pháp thay thế từ hiếm. Khi gặp một từ hiếm (được kí hiệu là <unk>) trong kết quả dự đoán, mô hình sẽ tìm một từ ở câu nguồn có trọng số giống hàng từ kết quả cơ chế Attention cao nhất và thực hiện sao chép từ đó thay cho từ hiếm hiện tại. (Mũi tên càng đậm thì trọng số giống hàng càng cao). Kết quả dự đoán được cập nhật với từ hiếm đã được thay thế.

trong tập dữ liệu so với những từ khác. Hơn nữa, những loại từ như thế này rất đa dạng (số lượng số có thể có rất lớn hay tên riêng, tên địa danh có rất nhiều). Điều đặc biệt rằng những từ này thường không cần phải dịch, chúng ta chỉ cần sao chép chính xác chúng lại qua câu ở ngôn ngữ đích với một vị trí phù hợp do những từ này dù ở ngôn ngữ nào thì cũng đều có một cách biểu diễn

Với kĩ thuật đơn giản là tận dụng ý nghĩa của kết quả của cơ chế Attention, kĩ thuật này đã cải thiện kết quả dịch lên một cách rõ rệt (sẽ được trình bày ở trong phần thực nghiệm).

## Chương 4

# Các Kết Quả Thực Nghiệm

*Trong chương này, chúng tôi trình bày các kết quả thực nghiệm để đánh giá các mô hình, phương pháp và kỹ thuật được tìm hiểu mà đã trình bày ở chương trước. Bộ dữ liệu được dùng để tiến hành các thực nghiệm là bộ WMT'13, 14 English-German (bộ dữ liệu tiếng Anh-tiếng Đức của cuộc thi Dịch máy WMT năm 2013 và 2014). Các kết quả thực nghiệm cho thấy khi huấn luyện mô hình mà không sử dụng cơ chế Attention thì kết quả đạt được rất thấp. Các mô hình Attention Toàn cục, Attention Cục bộ, phương pháp Input feeding và kỹ thuật thay thế từ hiếm cho kết quả được cải thiện một cách vượt trội. Chúng tôi đánh giá các mô hình trên các phương diện: độ lỗi, độ đo perplexity, độ đo BLEU, đường cong học và chất lượng giống hàng.*

### 4.1 Các thiết lập thực nghiệm

Chúng tôi tiến hành các thực nghiệm trên bộ dữ liệu WMT' 14 English-German được cung cấp trên trang chủ của Nhóm Xử lý Ngôn ngữ Tự nhiên Đại học Stanford [5]. Bộ dữ liệu này gồm các cặp câu được viết dưới dạng ngôn ngữ tự nhiên ở 2 ngôn ngữ là tiếng Anh và tiếng Đức. Tất cả mô hình sẽ được huấn luyện trên tập dữ liệu này. Tập dữ liệu có khoảng 4.5 triệu cặp câu (trong đó có khoảng 116 triệu từ tiếng Anh và khoảng 110 triệu từ tiếng Đức). Chúng tôi thực hiện thiết lập thực nghiệm giống với các thiết lập của bài báo của Luong, 2015 [11].

Dữ liệu được tiến hành tiền xử lý bằng cách thực hiện tách từ đối với mỗi câu. Bộ

từ vựng cho mỗi ngôn ngữ được sử dụng cho các mô hình là bộ từ vựng có 50,000 từ xuất hiện nhiều nhất (có tần số lớn nhất) trong dữ liệu huấn luyện của mỗi ngôn ngữ đó. Những từ nào không nằm trong bộ từ vựng sẽ được gán cho kí hiệu  $\langle unk \rangle$  (từ hiếm).

Trong quá trình huấn luyện, chúng tôi lọc bỏ những cặp câu mà một trong 2 câu thuộc cặp đó có chiều dài hơn 50 từ. Chúng tôi thực hiện sắp xếp tất cả câu theo chiều dài của câu giảm dần (những câu nào có chiều dài lớn nhất thì đứng đầu), sau đó lấy ngẫu nhiên các mini-batches từ những câu đã được sắp xếp. Với việc sắp xếp như vậy, tốc độ huấn luyện của mô hình được cải thiện do được tối ưu về chi phí tính toán và còn giúp mô hình học được tốt hơn.

Chúng tôi sử dụng các mô hình LSTM với mỗi LSTM có 4 tầng. Mỗi tầng LSTM có kích thước trạng thái ẩn là 1000 (đối với bộ mã hóa sử dụng bi-LSTM nên mỗi chiều của bi-LSTM sẽ có kích thước trạng thái ẩn là 500) và số chiều của word embedding là 1000. Các tham số của mô hình được khởi tạo ngẫu nhiên với phân phối đều trong đoạn  $[-0, 1; 0, 1]$ . Thuật toán để cực tiểu hóa hàm chi phí là Stochastic Gradient Descent (SGD) với kích thước của một mini-batch là 128 mẫu huấn luyện. Cách lập lịch cho hệ số học: huấn luyện 12 epochs; hệ số học ban đầu là 1.0; sau 8 epochs, hệ số học sẽ giảm đi 1 nửa sau mỗi epoch tiếp theo (epoch thứ 9 có hệ số học là 0.5, thứ 10 là 0.25, ...). Gradient của các tham số sẽ được chuẩn hóa nếu norm của chúng vượt quá 5.0. Mô hình còn sử dụng cơ chế Dropout với xác suất tắt các nơ-ron  $p = 0.2$ . Mỗi câu ở ngôn ngữ nguồn khi được đưa vào mô hình thì sẽ được đảo ngược trật tự. Đối với các mô hình Attention Cục bộ, kích thước cửa sổ  $D = 10$ .

Chúng tôi sử dụng ngôn ngữ lập trình Python và framework PyTorch dành cho Học sâu [15]. PyTorch hỗ trợ việc cài đặt các thuật toán Học sâu một cách thân thiện, tự nhiên giống như ngôn ngữ lập trình Python và còn hỗ trợ xử lý tính toán song song trên GPU (Graphical Processing Units) rất mạnh mẽ. GPU mà chúng tôi sử dụng để thực hiện các thực nghiệm là NVIDIA GeForce GTX 1080 Ti. Để có thể huấn luyện một mô hình, cần đến 3-5 ngày.

Để đánh giá chất lượng dịch của các mô hình đã được huấn luyện, chúng tôi sử dụng tập dữ liệu kiểm thử *newstest\_2014.en* và *newstest\_2014.de* của cuộc thi WMT'14 và độ đo được sử dụng để đánh giá là BLEU (BiLingual Evaluation Understudy) [13] cùng với Perplexity. Dữ liệu validation được sử dụng là tập dữ liệu kiểm

thử *newstest\_2013.en* và *newstest\_2013.de* của cuộc thi WMT'13.

Để đánh giá độ hiệu quả của cơ chế Attention, chúng tôi tiến hành huấn luyện một mô hình cơ bản (Baseline) mà không sử dụng cơ chế Attention (chỉ dùng kiến trúc Bộ mã hóa-Bộ giải mã với các LSTM). Các mô hình có sử dụng cơ chế Attention sẽ được so sánh với mô hình cơ bản này. Chi tiết của mô hình cơ bản này sẽ được trình bày trong phần tiếp theo.

## 4.2 Kết quả thực nghiệm

### 4.2.1 Không sử dụng Attention và có sử dụng Attention

Chúng tôi thực hiện đánh giá độ hiệu quả của cơ chế Attention bằng cách so sánh với mô hình không sử dụng cơ chế Attention và các mô hình có sử dụng cơ chế Attention.

Đối với mô hình không sử dụng cơ chế Attention, chúng tôi sử dụng:

- Kiến trúc Bộ mã hóa-Bộ giải mã với bộ mã hóa là bi-LSTM và bộ giải mã là uni-LSTM.
- Đảo ngược trật tự từ trong câu.
- Dropout.

Chúng tôi gọi đó là mô hình cơ bản. Đối với mô hình sử dụng cơ chế Attention, chúng tôi thiết lập mô hình như mô hình căn bản cộng với sử dụng cơ chế Attention Toàn cục với hàm tính điểm là hàm *dot*.

Bảng 4.1 cho thấy kết quả giữa 2 mô hình. Với cơ chế Attention, chất lượng dịch của mô hình được cải thiện rất lớn. Điểm BLEU tăng từ 15.04 đến 19.02 (+3.98). Đây là bước tiến lớn trong Dịch máy nơ-ron khi Dịch máy Thống kê đang dần chạm tới giới hạn. Kết quả này chứng minh được rằng cơ chế Attention rất hiệu quả trong việc giải quyết các hạn chế của kiến trúc Bộ mã hóa-Bộ giải mã với LSTM đã được trình bày ở ban đầu và cũng rất phù hợp với bài toán Dịch máy.

Mô hình	Perplexity	BLEU
Cơ bản		15.04
Cơ bản + global (dot)		19.02 (+3.98)

Bảng 4.1: So sánh giữa mô hình sử dụng cơ chế Attention và mô hình không sử dụng cơ chế Attention.

Mô hình	Perplexity	BLEU	
		Trước khi	Sau khi dùng thay thế từ hiếm
Global (dot)		19.02	
Global (dot) + input feed		19.78	22.35 (+2.57)
Local-p + input feed		20.37	22.75 (+2.38)

Bảng 4.2: So sánh giữa các mô hình Attention.

## 4.2.2 Giữa các mô hình Attention với nhau

Để đánh giá độ hiệu quả của các mô hình Attention với nhau, chúng tôi thực hiện huấn luyện các mô hình Attention và đánh giá chúng trên độ đo Perplexity và BLEU.

Kết quả của các mô hình Attention được thể hiện trong bảng 4.2. Từ bảng kết quả cho thấy mô hình Attention cục bộ với Giống hàng dự đoán (Local-p) có chất lượng dịch tốt nhất với điểm BLEU là 20.37 (22.75 khi dùng thay thế từ hiếm). Mô hình Attention Toàn cục với hàm tính điểm dot cộng với phương pháp Input feeding (Global (dot) + input feed) có độ tăng điểm BLEU cao nhất khi dùng kỹ thuật thay thế từ hiếm. Kết quả này chứng minh được rằng mô hình Local-p có ý tưởng phù hợp cho cặp ngôn ngữ Anh-Đức và hoạt động hiệu quả như mong đợi trong thực tế.

Kết quả cho thấy rằng với 2 phương pháp Input feeding và thay thế từ hiếm mà chúng tôi đã trình bày ở phần trước đều góp phần tăng hiệu quả của mô hình lên rất đáng kể. Cụ thể, phương pháp Input feeding tăng BLEU lên trung bình khoảng 1.0. Kỹ thuật thay thế từ hiếm tăng trung bình hơn 2.25 BLEU. Các kết quả này chứng minh được rằng những ý tưởng, lý thuyết chúng tôi trình bày ở phần trước đều hoạt động tốt như mong đợi. Những phương pháp, kỹ thuật này vừa rõ ràng về mặt lý thuyết, vừa hiệu quả trong thực tế.

Về phương pháp Input feeding, có một chút hạn chế của phương pháp này về chi phí tính toán. Input feeding làm kích thước đầu vào ở các bước thời gian của uni-LSTM của bộ giải mã tăng lên theo kích thước của véc-tơ attention. Hơn nữa, về mặt

Mô hình	Perplexity	BLEU	
		Trước khi	Sau khi dùng thay thế từ hiếm
Global (dot)		19.02	
Global (dot) + input feed		19.78	22.35 (+2.57)
Global (general)		19.99	22.23 (2.24)
Local-p + input feed		20.37	22.75 (+2.38)

Bảng 4.3: Kết quả của các mô hình trên tập dữ liệu WMT'14 English-German.

cài đặt, chúng ta không tận dụng được LSTM mà đã được hỗ trợ cài đặt bởi NVIDIA. Do vậy, tốc độ huấn luyện và trong kiểm thử của mô hình khi có sử dụng cơ chế Input feeding sẽ giảm đi đáng kể, bên cạnh đó kích thước của mô hình cũng tăng lên theo.

Về kĩ thuật thay thế từ hiếm, từ kết quả cho thấy kĩ thuật này rất mạnh mẽ. Các mô hình Attention được sử dụng cùng với kĩ thuật này đều được tăng điểm BLEU hơn 2. Mô hình này chỉ có hạn chế nhỏ là phải phụ thuộc vào kết quả của cơ chế Attention có tốt hay không. Còn lại kĩ thuật này rất hiệu quả và tiện lợi. Thay thế từ hiếm không ảnh hưởng tới quá trình huấn luyện, do vậy mô hình không phải tốn chi phí tính toán cho kĩ thuật này trong quá trình huấn luyện và tốc độ huấn luyện không bị ảnh hưởng.

Nhìn chung, kết quả cho thấy kĩ thuật thay thế từ hiếm cải thiện kết quả tốt hơn phương pháp Input feeding, nhưng Input feeding giúp tạo ra một mô hình Attention đủ tốt để tạo điều kiện cho kĩ thuật thay thế từ hiếm phát huy sự hiệu quả của nó.

Các kết quả trên cho thấy những cơ chế mà chúng tôi tìm hiểu và sử dụng trong khóa luận này thực sự hiệu quả. Các cơ chế này vừa rõ ràng về lý thuyết vừa có hiệu suất tốt trong thực tế. Đặc biệt với cơ chế thay thế từ hiếm dựa vào kết quả của cơ chế Attention đã tăng kết quả của mô hình lên rất đáng kể. Điều này cũng cho thấy tiềm năng của cơ chế Attention trong việc giải quyết bài toán Dịch máy. Không chỉ bản thân của cơ chế này nâng cao chất lượng dịch mà còn là nền tảng để những cơ chế khác sử dụng và tiếp tục nâng cao chất lượng dịch. Phương pháp này như mở ra thêm 1 hướng nghiên cứu cho bài toán Dịch máy nơ-ron.

### 4.2.3 Phân tích đường cong học

Phần này, chúng tôi tập trung vào so sánh các tất cả các mô hình đã thực hiện ở phương diện quá trình học bằng cách quan sát độ lỗi của mô hình bằng độ đo Perplexity trên cả tập huấn luyện và tập validation (newstest\_2013) trong quá hình huấn luyện mô

hình. Điều này giúp chúng tôi biết được mô hình nào sẽ hội tụ và tổng quát hóa tốt hơn.

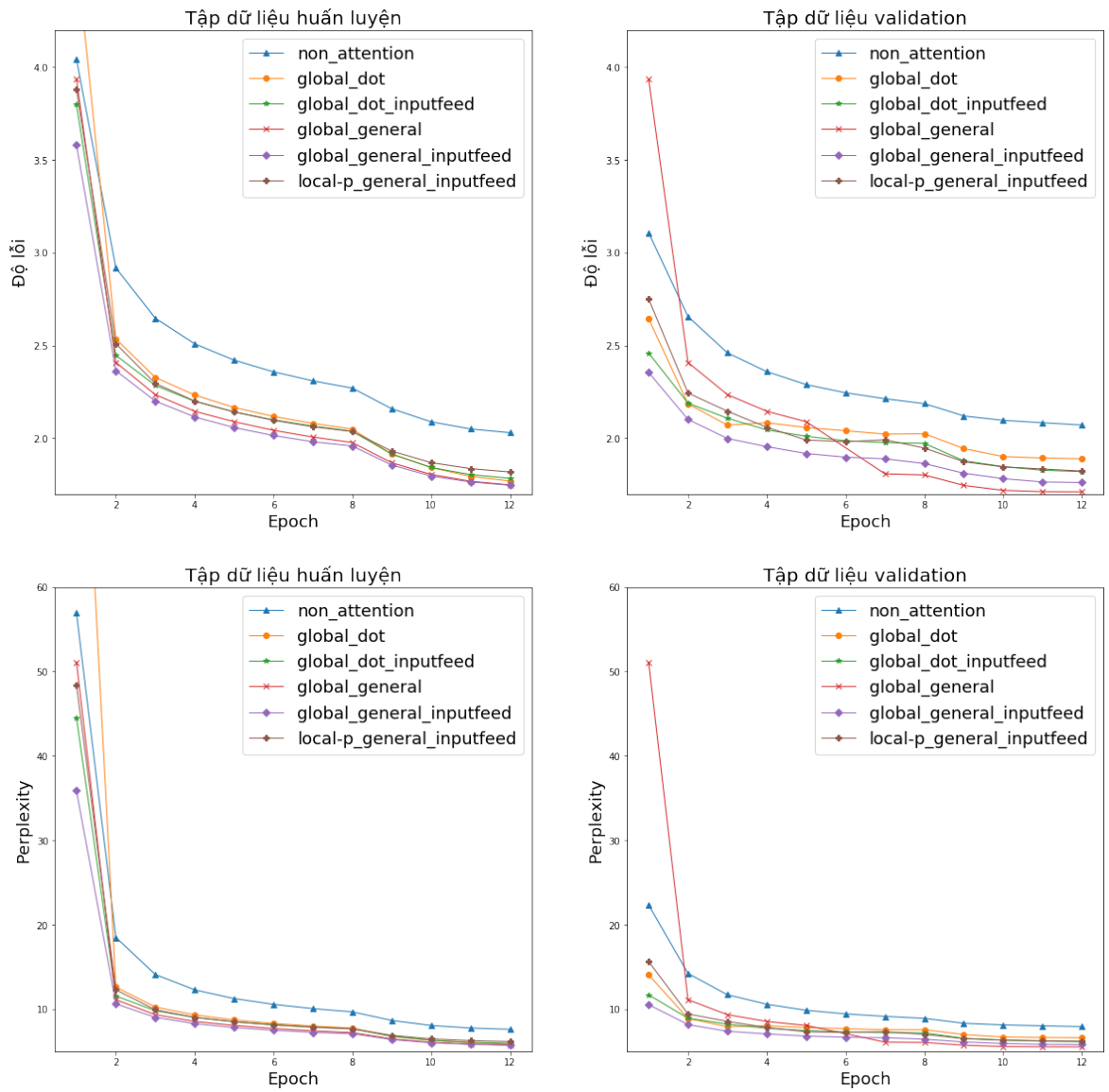
Từ đồ thị 4.1 thấy được rằng mô hình Global Attention với general product cho tốc độ hội tụ nhanh nhất (perplexity trên tập huấn luyện giảm nhanh nhất qua các epoch), cùng với đó khả năng tổng quát hóa cũng đáng kể (perplexity trên tập validation có độ giảm tương đương). Mô hình Local Attention với general product cho khả năng tổng quát hóa tốt nhất (perplexity trên tập huấn luyện thấp hơn các mô hình khác nhưng kết quả điểm BLEU trên tập kiểm thử là tốt nhất). Trong các epoch đầu, hầu hết các mô hình có tốc độ giảm perplexity nhanh như nhau. Các mô hình Attention không sử dụng phương pháp Input feeding ở epoch đầu có perplexity cao hơn so với các mô hình sử dụng Input feeding, nhưng khi ở các epoch về sau thì tốc độ giảm perplexity giống nhau. Các mô hình Local Attention ở các epoch đầu đều có hành vi giống Global Attention, chỉ có khác nhau ở các epoch cuối cùng. Các mô hình Local Attention này ở các epoch cuối có tốc độ giảm perplexity chậm đi rõ rệt.

#### 4.2.4 So sánh với kết quả của bài báo

Phần này chúng tôi thực hiện so sánh kết quả của chúng tôi với kết quả của bài báo *Effective Approaches to Attention-based Neural Machine Translation* để đánh giá rằng mô hình của chúng tôi hoạt động có thực sự đúng không. // TODO: thêm bảng kết quả

Từ bảng kết quả trên cho thấy mô hình của chúng tôi hoạt động tốt. Kết quả của chúng tôi có khác với bài báo rằng hầu hết mô hình của chúng tôi đều hơn kết quả của bài báo từ 1.0 đến gần 2.0 điểm BLEU. Đặc biệt riêng với mô hình Global Attention với general product, kết quả có sự chênh lệch vô cùng lớn.

Lí do của sự khác biệt lớn này là về sự khác nhau giữa 2 ngôn ngữ và framework - Matlab và PyTorch. Matlab là một ngôn ngữ lâu đời và tại thời điểm năm 2015 mà tác giả thực hiện bài báo này thì Matlab vẫn chưa có hỗ trợ nhiều cho Học sâu và chưa tối ưu cho lập trình song song trên GPU. Do vậy, hầu hết các bước tính toán quan trọng của mô hình đều là do tác giả tự cài đặt, bao gồm cả các bước thực hiện tính đạo hàm để lan truyền ngược. Trong khi đó, PyTorch là một framework được thiết kế dành riêng cho việc hỗ trợ cài đặt các mô hình Python dựa trên ngôn ngữ Python được Facebook cùng với các trường đại học, công ty v.v... hỗ trợ phát triển. Do vậy, về mặt ý tưởng và được cài đặt khá giống nhau trên mã nguồn Matlab và Python nhưng các



Hình 4.1: Đường cong học của các mô hình phân tích trên độ lỗi, điểm BLEU trên tập dữ liệu huấn luyện và validation.



phần lõi của mô hình được cài đặt lại khác nhau.

#### 4.2.5 Chất lượng giống hàng của mô hình

Trong phần này chúng tôi trình bày về một trong những lợi ích quan trọng của các mô hình Attention là khả năng trực quan hóa những gì mô hình đã học được. Từ khi Học sâu ra đời, bên cạnh sự thành công rực rỡ trong việc giải quyết các bài toán phổ biến thì mạng Học sâu có hạn chế rất lớn về khả năng diễn dịch kết quả. Do khả năng tự học các đặc trưng, khi mạng Học sâu xuất ra các kết quả, mọi người hầu như không thể giải thích được lí do vì sao mô hình cho ra kết quả như vậy. Vì thế, mạng Học sâu không được sử dụng cho các lĩnh vực, ứng dụng mà cần sự rõ ràng khi đưa ra quyết định như trong lĩnh vực y tế, kinh tế, v.v... Hơn nữa còn gây cản trở việc nghiên cứu và phát triển các mô hình Học sâu do khó để kiểm nghiệm được rằng các ý tưởng, phương pháp sử dụng trên mạng Học sâu có hoạt động như mong muốn hay không. Do vậy, khả năng diễn dịch của cơ chế Attention vô cùng hữu ích cho những mạng Học sâu mà sử dụng cơ chế Attention.

Giữa các từ ở câu nguồn và câu đích sẽ có trọng số giống hàng (điểm attention)  $a_t$  tương ứng. Sử dụng những trọng số giống hàng đó giúp mọi người có thể biết được rằng mô hình đã học được những gì và có hoạt động đúng như mong đợi hay không. Chúng tôi vẽ các ma trận trọng số giữa một số cặp câu mẫu đã được dịch bằng cơ chế Attention để cho thấy sự hiệu quả của cơ chế này.

## Chương 5

# Kết Luận Và Hướng Phát Triển

### 5.1 Kết luận

Trong khóa luận này, chúng tôi nghiên cứu bài toán Dịch máy nơ-ron bằng mô hình Attention-LSTM. Mô hình Attention-LSTM học được cách dịch giữa 2 ngôn ngữ (trong khóa luận này là Anh-Đức) như các mô hình với kiến trúc Bộ mã hóa-Bộ giải mã với bộ mã hóa và bộ giải mã là các LSTM. Tuy nhiên cơ chế Attention đem lại nhiều lợi thế mà những mô hình không sử dụng Attention không có được:

- Các mô hình sử dụng Attention tận dụng được các trạng thái ẩn trên bộ mã hóa để hạn chế vấn đề "sự phụ thuộc dài" của các mô hình RNNs. Trong quá trình dự đoán, bộ giải mã sử dụng các trạng thái ẩn của bộ mã hóa bằng cách đặt "sự chú ý" lên một số trạng thái ẩn cần thiết và sử dụng nó để suy ra ngữ cảnh hiện tại của câu, sau đó mô hình dự đoán từ tiếp theo dựa vào ngữ cảnh đó.
- Cơ chế có cách dịch giống với ý tưởng về cách con người dịch nhìn sự vật, hiện tượng. Con người thường chỉ tập trung vào những phần quan trọng mà cung cấp những thông tin cần thiết, phù hợp với mục đích quan sát của sự vật, hiện tượng.
- Có thể sử dụng kết quả của cơ chế Attention để phát triển thêm các phương pháp, kỹ thuật khác:
  - Phương pháp Input feeding: giúp mô hình có thể biết được thông tin giống hàng trong những thời điểm trước đó thông qua véc-tơ attention  $\tilde{h}_t$ . Từ đó giúp mô hình có thể hạn chế vấn đề "được dịch quá nhiều" hoặc "được

dịch quá ít", tránh được những câu dịch không thực tế như những câu dịch lặp lại một từ nhiều lần.

- Kỹ thuật thay thế từ hiếm: giúp mô hình giải quyết được vấn đề hạn chế về kích thước của bộ từ vựng. Do bộ từ vựng không thể chứa hết tất cả các từ có thể có trong quá trình dịch, vì vậy chất lượng dịch của mô hình bị giảm đáng kể nếu gặp những từ hiếm đó. Đặc biệt là khi mô hình gặp những câu có chứa các số, tên riêng, tên các địa danh, v.v...

Các kết quả thực nghiệm trên bộ dữ liệu WMT'14 English-German cho thấy rằng:

- Cơ chế Attention cải thiện chất lượng dịch của mô hình rất cao so với những mô hình không sử dụng cơ chế Attention.
- Những phương pháp, kỹ thuật sử dụng kết quả của cơ chế Attention để giải quyết những vấn đề còn tồn tại khi sử dụng mô hình dịch máy nơ-ron cũng cải thiện chất lượng dịch của mô hình lên đáng kể.

Cơ chế Attention đã mở ra một không gian rộng lớn để phát triển cho việc cải tiến mô hình các dịch máy nơ-ron.

// TODO Chúng tôi đạt được trong khóa luận này:

- 

## 5.2 Hướng phát triển

Trong khóa luận này chúng tôi mới chỉ tìm hiểu một phần nhỏ của kiến trúc Bộ mã hóa-Bộ giải mã cũng như cơ chế Attention. Trong những năm gần đây, đã xuất hiện nhiều cách tận dụng kiến trúc Bộ mã hóa-Bộ giải mã hơn để giải quyết một số bài toán như Phát sinh câu mô tả cho ảnh, Trả lời câu hỏi tự động, Tóm tắt văn bản, Phát sinh văn bản từ giọng nói, Phát sinh giọng nói từ văn bản và nhiều ứng dụng khác nữa. Ở mỗi bài toán sẽ có cách sử dụng kiến trúc Bộ mã hóa-Bộ giải mã khác nhau như sử dụng các kiến trúc mạng nơ-ron khác nhau cho các bộ mã hóa, bộ giải mã cũng như trao đổi thông tin giữa 2 bộ phận với nhau. Ngoài ra, cơ chế Attention ngày càng phức tạp hơn để phù hợp hơn với những bài toán phức tạp. Định hướng phát triển của khóa luận:

- Tìm hiểu thêm những mô hình khác ngoài LSTM để áp dụng vào kiến trúc Bộ mã hóa-Bộ giải mã tiêu biểu như Convolutional Neural Networks và những cải tiến nhỏ để thiện chất lượng mã hóa.
- Tìm hiểu thêm những cải tiến khác trên cơ chế Attention như cấu trúc phân tầng cho Attention, tìm hiểu thêm những hàm tính điểm khác nhằm giúp cho việc tính toán trọng số giống hàng và véc-tơ ngữ cảnh chính xác và hữu ích hơn.

# TÀI LIỆU THAM KHẢO

- [1] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, pp. 1724–1734. [Online]. Available: <http://www.aclweb.org/anthology/D14-1179> 7
- [2] M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas, “Learning where to attend with deep architectures for image tracking,” *CoRR*, vol. abs/1109.3737, 2011. [Online]. Available: <http://arxiv.org/abs/1109.3737> 31
- [3] J. L. Elman, “Finding structure in time,” *Cognitive Science*, vol. 14, no. 2, pp. 179 – 211, 1990. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/036402139090002E> 12
- [4] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: continual prediction with lstm,” in *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, vol. 2, 1999, pp. 850–855 vol.2. 22, 23
- [5] T. S. N. L. P. Group. (2015) Neural machine translation. [Online]. Available: <https://nlp.stanford.edu/projects/nmt/> 44
- [6] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: <http://dx.doi.org/10.1162/neco.1997.9.8.1735> 20
- [7] W. J. Hutchins, “Machine translation: A concise history,” 2007. 2, 3

- [8] S. Jean, K. Cho, R. Memisevic, and Y. Bengio, “On using very large target vocabulary for neural machine translation,” *CoRR*, vol. abs/1412.2007, 2014. [Online]. Available: <http://arxiv.org/abs/1412.2007> 42
- [9] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, October 2013, pp. 1700–1709. [Online]. Available: <http://www.aclweb.org/anthology/D13-1176> 7
- [10] P. Koehn, F. J. Och, and D. Marcu, “Statistical phrase-based translation,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL ’03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 48–54. [Online]. Available: <https://doi.org/10.3115/1073445.1073462> 39
- [11] M. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: <http://arxiv.org/abs/1508.04025> 9, 10, 36, 40, 44
- [12] A. Maas, Q. V. Le, T. M. O’Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, “Recurrent neural networks for noise reduction in robust asr,” in *INTERSPEECH*, 2012. 12
- [13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: A method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL ’02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 311–318. [Online]. Available: <https://doi.org/10.3115/1073083.1073135> 45
- [14] R. Pascanu, T. Mikolov, and Y. Bengio, “Understanding the exploding gradient problem,” *CoRR*, vol. abs/1211.5063, 2012. [Online]. Available: <http://arxiv.org/abs/1211.5063> 8, 20

- [15] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017. [45](#)
- [16] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: <http://arxiv.org/abs/1409.3215> [7](#), [9](#), [25](#)
- [17] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct 1990. [15](#)
- [18] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” *CoRR*, vol. abs/1502.03044, 2015. [Online]. Available: <http://arxiv.org/abs/1502.03044> [35](#)