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

PHẠM ĐÌNH VƯƠNG

LÊ HOÀNG TRUNG

DỊCH MÁY BẰNG 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

Lời đầu tiên, chúng em xin được gửi lời tri ân vô cùng sâu sắc đến thầy hướng dẫn khóa luận tốt nghiệp của chúng em - Thầy Trần Trung Kiên. Thầy đã rất tận tâm, nhiệt tình hướng dẫn và động viên chúng em trong suốt quá trình làm khóa luận. Nếu không có sự quan tâm, theo dõi chặt chẽ của Thầy chắc chắn chúng em không thể hoàn thành khóa luận này. Thầy còn giúp chúng em hình thành được một nền tảng rất vững chắc và củng cố lại những kiến thức trong suốt quãng đường Đại học.

Em xin chân thành cảm ơn quý Thầy Cô khoa Công nghệ Thông tin - trường Đại học Khoa học Tự nhiên - ĐHQG-HCM, những người đã ân cần giảng dạy cho chúng em những kiến thức vô cùng cần thiết, bổ ích để chúng em có thể vững bước trên con đường sau này.

Con xin cảm ơn ba mẹ đã sinh thành, nuôi dưỡng và dạy dỗ để chúng 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 chúng con gặp khó khăn trong cuộc sống. Có bố mẹ ủng hộ, chúng con luôn luôn cảm thấy an tâm để tiếp tục tiến lên phía trước.

Chúng em cũng xin gửi lời cảm ơn chân thành và sâu sắc tới anh Trần Duy Quang cùng với anh chị, bạn bè đồng nghiệp trong công ty AIOZ đã hỗ trợ, giúp đỡ chúng em về mặt phần cứng để có thể thuận lợi hoàn thành khóa luận này.

Tp.Hồ Chí Minh, 07/2018

Phạm Đình Vương

Lê Hoàng Trung

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

Chương 1

Giới thiệu

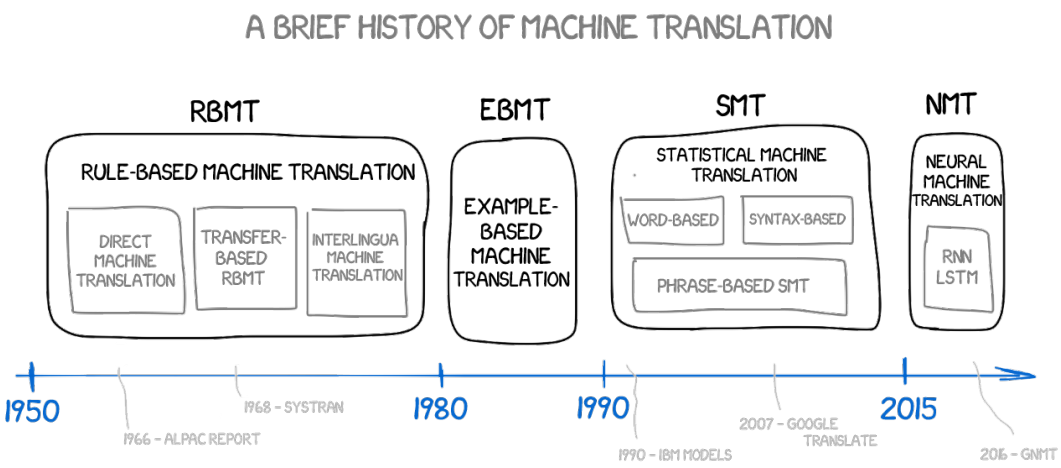
Nhờ vào những cải cách trong giao thông và cơ sở hạ tầng viễn thông mà giờ đây toàn cầu hóa đang trở nên gần với chúng ta hơn bao giờ hết. Trong xu hướng đó nhu cầu giao tiếp và thông hiểu giữa những nền văn hóa là không thể thiếu. Tuy nhiên, những nền văn hóa khác nhau thường kèm theo đó là sự khác 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 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 (memorandum) đưa ra các đề xuất khác nhau của ông trong lĩnh vực này [14]. 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 [14]. Thí 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



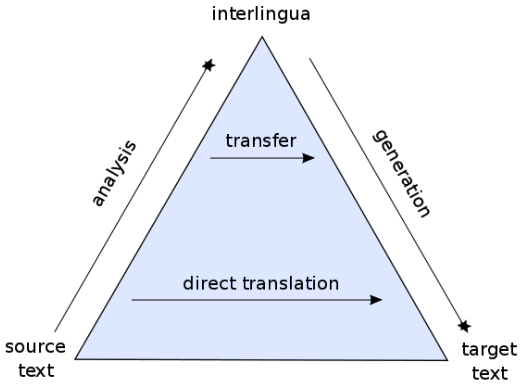
Hình 1.1: Lịch sử tóm tắt của dịch máy, nguồn ảnh: Ilya Pestov trong blog "A history of machine translation from the Cold War to deep learning"

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 [14]. 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 những năm 1950 đến nay, đã có nhiều hướng tiếp cập 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. Hình 1.1 thể hiện các phương pháp dịch máy đã được đề xuất trong quá trình hình thành và phát triển của dịch máy từ những năm 1950. Trong những năm đầu tiên, để tạo ra bản dịch tốt, các phương pháp thời bấy giờ đều đòi hỏi 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)*. 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: dịch máy trực tiếp, dịch máy chuyển giao và dịch máy ngôn ngữ phổ quát. Mặc dù cả



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)

ba đều thuộc về dịch máy dựa trên luật, tuy nhiên chúng khác nhau về độ sâu của đại 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): Đây là phương pháp đơn giản nhất của dịch máy. Dịch máy trực tiếp 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): 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): dịch máy chuyển giao tương tự như dịch máy ngôn ngữ phổ quát 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 dịch máy ngôn ngữ phổ quát, dịch máy chuyển giao 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 dịch máy chuyển giao 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 dịch máy dựa trên luật 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 dịch máy dựa trên luật 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 dịch máy dựa trên luật 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 (hình 1.3 mô tả tập ngữ liệu Anh - Nga). 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): ý tưởng của phương pháp này là thay vì định nghĩa những từ điển và các luật ngữ pháp một cách thủ công, dịch máy thống kê 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 dịch máy thống kê được giới thiệu đầu tiên bởi Waren Weaver vào năm 1949 bap gồm việc áp dụng lý thuyết thông tin của Claude Shannon vào dịch máy. Dịch máy thống kê được giới thiệu lại vào cuối những năm 1980 và đầu những năm 1990 tại trung tâm nghiên cứu Thomas J. Watson của IBM. Dịch máy thống kê 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.

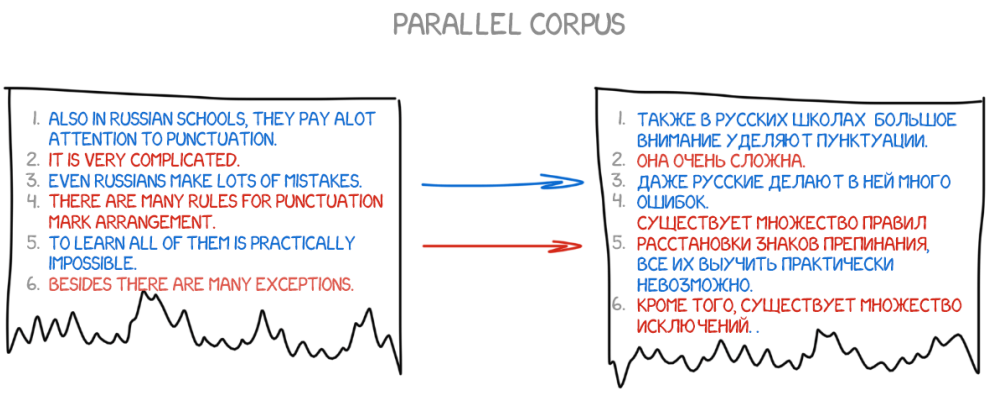
Để hiểu rõ hơn về dịch máy thống kê, xét một ví dụ: ta cần dịch một câu f trong tiếng Pháp sang dạng tiếng Anh e của nó. Có nhiều bản dịch có thể có cuả f trong tiếng Anh, việc cần làm là chọn e sao cho nó là bản dịch "tốt nhất" của f. Chúng ta có thể mô hình hóa quá trình này bằng một xác suất có điều 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 $p(e|f)$:

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

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:

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

trong đó e^* là tập các bản dịch ứng viên. 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 $p(\text{"I am at home"}) > p(\text{"I am at house"})$ vì mặc dù "home" và "house" đề mang nghĩa là ngôi nhà nhưng không ai nói "I am at house" cả. Ngoài



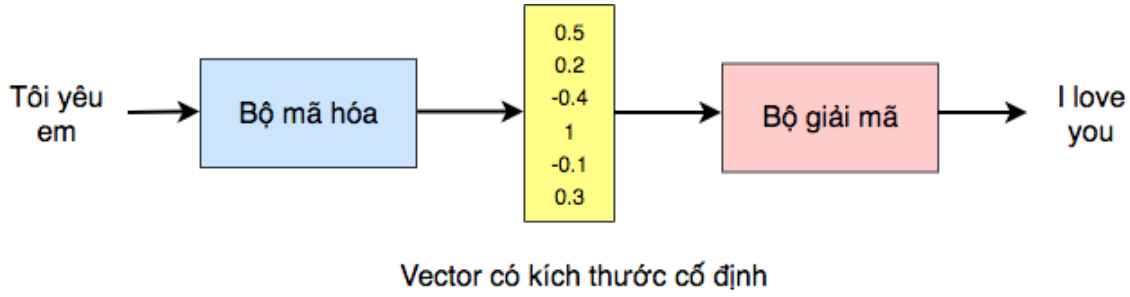
Hình 1.3: Ví dụ về tập các câu song song trong hai ngôn ngữ: tiếng Anh - tiếng Nga, nguồn ảnh: Ilya Pestov trong blog "A history of machine translation from the Cold War to deep learning"

ra, các mô hình ngôn ngữ cho dịch máy thống kê thường được ước lượng bằng các mô hình *n*-gram được làm mịn, cách làm này cũng là một nhược điểm của dịch máy thống kê. Mô hình ngôn ngữ là một chủ đề quan trọng và sẽ được chúng tôi đề cập cụ thể hơn trong chương 2.

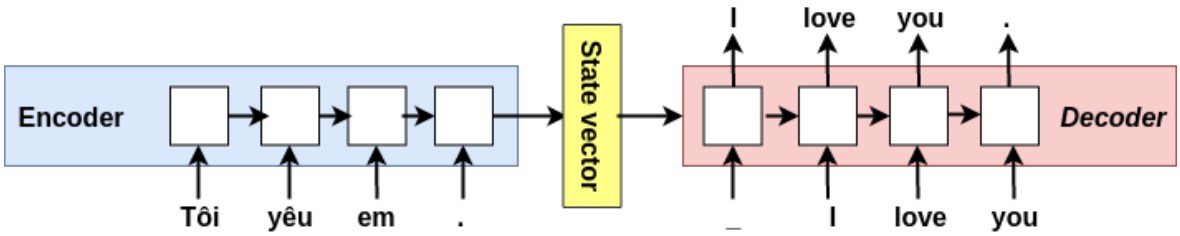
Mặc dù trên thực tế đã có nhiều hệ thống dịch máy được phát triển dựa trên dịch máy thống kê thời bấy giờ, tuy nhiên nó không hoạt động thực sự tốt bởi một số nguyên nhân. Một là việc những từ hay đoạn được dịch cục bộ và quan hệ của chúng với những từ cách xa trong câu nguồn thường bị bỏ qua. Hai là 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 [19], [32], [5]. 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) ([32], [5]). 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 véc-tơ có kích



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

thước cố định. Bộ giải mã, cũng là một mạng nơ-ron, sẽ tạo bản dịch trong ngôn ngữ đích từ véc-tơ 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"*, 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 tên là *mạng nơ-ron hồi quy*; đây 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 - Bộ giải mã sử dụng mạng nơ-ron hồi quy. Đầu tiên mạng nơ ron hồi quy mà đóng vai trò bộ mã hóa sẽ duyệt qua từng từ trong câu nguồn, duyệt đến đâu sẽ cập nhật véc-tơ trạng thái ẩn (véc-tơ lưu thông tin cần thiết) của mạng đến đó, véc-tơ trạng thái ẩn cuối cùng được xem là chứa toàn bộ thông tin của câu nguồn được gọi là véc-tơ mã hóa của câu nguồn (véc-tơ này có kích thước cố định với các câu nguồn có chiều dài khác nhau). Véc-tơ mã hóa này sẽ được đưa vào một mạng nơ-ron hồi quy khác mà đóng vai trò là bộ giải mã. Tại một thời điểm, bộ giải mã sẽ phát sinh ra (hay giải mã ra) một từ trong câu đích dựa vào véc-tơ mã hóa câu nguồn và các từ đã sinh ra trước đó của câu đích.

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 véc-tơ có kích thước cố định được tạo ra ở bộ mã hóa.

Véc-tơ 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 thời điểm trước đó. Mạng nơ-ron hồi quy được chứng minh là gặp khó khăn trong việc mã hóa những chuỗi dài [28]. 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 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 [32] 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 Apporoaches to Attention-based Neural Machine Translation" [22] 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). Cơ chế attention dựa trên quan sát rằng: tại một thời điểm trong quá trình giải mã, để quyết định từ tiếp theo của câu đích là gì thì không cần hết tất cả thông tin của câu nguồn mà chỉ cần thông tin của một số từ trong câu nguồn, nên chỉ cần tập trung (attend) vào một số từ trong câu nguồn. Như vậy, thay vì cố gắng mã hóa tất cả thông tin của câu nguồn vào trong véc-tơ mã hóa có kích thước cố định (điều mà khó thực hiện khi câu nguồn dài) và luôn dùng véc-tơ mã hóa này ở tất cả các thời điểm trong quá trình giải mã (điều mà không cần thiết), cơ chế attention cho phép ở mỗi thời điểm trong quá trình giải mã chỉ tập trung vào một vùng cần thiết trong câu nguồn (ở đây, một vùng nghĩa là một vài trạng thái ẩn trong tập tất cả các trạng thái ẩn của bộ mã hóa).

Với những ưu điểm đã được trình bày 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" [22] đã nói ở trên. Trong khóa luận này, chúng tôi đã thành công xây dựng lại mô hình được đề xuất trong bài báo. Bên cạnh đó chúng tôi cũng thực 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 khóa luận được trình bày như sau:

- Chương 2 trình bày về những thành nền tảng của kiến trúc Bộ mã hóa - Bộ 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: mô hình ngôn ngữ, mạng nơ-ron hồi quy và Long short-term memory. Mô hình ngôn ngữ cho phép chúng ta dự đoán từ tiếp theo trong một chuỗi từ cho trước; trong dịch máy, nó giúp tạo ra những bản dịch lưu loát. Mạng nơ-ron hồi quy (RNN) là xương sống của dịch máy nơ-ron. RNN được dùng làm bộ mã hóa lẫn bộ giải mã. Ứng với mỗi vai trò, RNN sẽ có một thiết kế riêng. Sau đó, dựa trên những kiến thức về RNN, chúng tôi nói về khái niệm mô hình ngôn ngữ dựa trên mạng nơ-ron hồi quy với chức năng tạo ra các từ trong bộ giải mã. Long short-term memory (LSTM) là phiên bản cải tiến của RNN nhằm giải quyết vấn đề về phụ thuộc dài hạn. LSTM cũng là phiên bản RNN được dùng để xây dựng nên Bộ mã hóa - Bộ giải mã trong khóa luận này. Những thành phần nói trên 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 Mô hình ngôn ngữ

Như đã nói trong chương 1, mô hình ngôn ngữ là thành phần không thể thiếu trong dịch máy; nó đảm bảo rằng hệ thống tạo ra những bản dịch "trơn tru" (fluent). Trong dịch máy nơ-ron, người ta thường sử dụng mô hình ngôn ngữ dựa trên mạng nơ-ron hồi quy (trong khóa luận này nó là một LSTM). Trước khi đi đến mô hình ngôn ngữ

dựa trên mạng nơ-ron hồi quy, chúng tôi sẽ nhắc lại một vài mô hình ngôn ngữ đã được sử dụng trong quá khứ, cũng như những điểm mạnh, yếu của từng mô hình. Những đặc điểm ấy nói lên rằng vì sao mô hình ngôn ngữ dựa trên mạng nơ-ron hồi quy là hướng tiếp cận nổi bật và thích hợp cho bài toán dịch máy nơ-ron.

Mô hình ngôn ngữ là một phân phối xác suất trên một chuỗi các từ nhằm đánh giá độ trơn tru của chuỗi ấy so với những chuỗi khác. Cho trước một chuỗi $w_1, w_2, ..., w_n$, mô hình ngôn ngữ gán cho nó một xác suất $p(w_1, w_2, ..., w_n)$ đại diện cho độ trơn tru của chuỗi đó, ví dụ $p(\text{"How tall are you ?"}) > p(\text{"How high are you ?"})$ vì từ "high" trong thực tế không dùng để hỏi về chiều cao của con người. Theo công thức xác suất có điều kiện, ta có:

$$\begin{aligned} p(w_{1:n}) &= p(w_1)p(w_2|w_1)p(w_3|w_1, w_2)...p(w_n|w_1, w_2, ..., w_{n-1}) \\ &= \prod_{t=1}^{n} p(w_t|w_1, w_2, ..., w_{t-1}) \end{aligned} \tag{2.1}$$

Trong công thức trên, mỗi thành phần $p(w_i|w_1, w_2, ..., w_{i-1})$ là xác suất có điều kiện của từ w_i biết những từ trước đó $w_1, w_2, ..., w_{i-1}$, những từ này còn được gọi là ngữ cảnh đối với từ đang xét. Ta thấy rằng để tính được xác suất $p(w_i|w_1, w_2, ..., w_{i-1})$ ta phải xét đến tất cả những từ đứng trước nó. Điều này làm cho chi phí tính toán trở nên rất lớn. Để giảm chi phí tính toán, người ta sử dụng giả định *markov* (markov-assumption); giả định này nói rằng các từ tiếp theo chỉ liên quan đến từ hiện tại và độc lập với các từ trước đó. Chính xác hơn, một giả định markov bậc k nói rằng từ tiếp theo trong một chuỗi chỉ phụ thuộc vào k từ cuối cùng trong chuỗi.

$$p(w_{i+1}|w_1, w_2, ..., w_i) \approx p(w_{i+1}|w_{i-k}, ..w_i) \tag{2.2}$$

Công thức 2.3 theo giả định markov trở thành:

$$p(w_{1:n}) = \prod_{t=1}^{n} \approx p(w_{i-k}|w_{i-k}, ..w_{i-1}) \tag{2.3}$$

Mặc dù giả định markov bậc k rõ ràng là sai với k bất kỳ (một câu có thể có phụ thuộc dài hạn với chiều dài bất kỳ như câu bắt đầu bằng từ *what* và kết thúc bằng dấu *?*), tuy nhiên nó vẫn tạo ra những mô hình ngôn ngữ mạnh mẽ với với các giá trị tương

đối nhỏ của k ($k = 3$ hoặc $k = 5$), và là phương pháp chủ yếu cho bài toán mô hình hóa ngôn ngữ trong hàng thập kỷ.

Do mô hình ngôn ngữ là một mô hình học không giám sát, một cách thông thường để đánh giá một mô hình học không giám sát là áp dụng nó lên một bài toán học có giám sát rồi đánh giá thông qua kết quả của bài toán đó, ví dụ: đo lường sự cải thiện chất lượng dịch khi chuyển đổi từ mô hình ngôn ngữ A sang mô hình ngôn ngữ B trong một hệ thống dịch máy. Cách đánh giá này được gọi là đánh giá bên ngoài (extrinsic evaluation). Tuy nhiên, đánh giá bên ngoài rất tốn kém đối với những bài toán học có giám sát lớn và phức tạp. Vì vậy, một độ đo được thiết kế để đánh giá mô hình ngôn ngữ một cách nội tại (intrinsic evaluation) đó là "perplexity". Perplexity là một độ đo để đánh giá một mô hình xác suất tốt như thế nào khi dự đoán một mẫu chưa nhìn thấy (unseen sample). Cho trước một tập ngữ liệu $W = w_1, w_2, ..., w_N$ (N là một số rất lớn) và một mô hình ngôn ngữ LM, perplexity của W ứng với mô hình ngôn ngữ LM là:

$$\begin{aligned} Perplexity(W) &= \sqrt[N]{\frac{1}{LM(p(w_{1:N}))}} \\ &= \sqrt[N]{\prod_{i=1}^{N} \frac{1}{LM(p(w_i|w_{1:i-1}))}} \\ &= 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 LM(p(w_i|w_{1:i-1}))} \end{aligned} \tag{2.4}$$

Một mô hình ngôn ngữ tốt sẽ gán xác suất lớn cho những chuỗi trong tập ngữ liệu (phản ánh đúng ngôn ngữ thật sự) sẽ dẫn đến một giá trị nhỏ của perplexity và ngược lại. Perplexity là một độ đo tốt và thuận tiện để đánh giá các mô hình ngôn ngữ với nhau, nhưng việc perplexity được cải thiện khi ta thay đổi mô hình ngôn ngữ không đảm bảo rằng đánh giá bên ngoài cũng được cải thiện. Tuy vậy, perplexity vẫn thường được sử dụng như là một cách kiểm tra nhanh một mô hình ngôn ngữ. Nhưng để có thể đánh giá một cách chính xác trước khi kết luận về hiệu suất của một mô hình ngôn ngữ, ta luôn luôn phải kiểm tra lại mô hình ngôn ngữ này bằng cách đánh giá ngoài dựa trên một ứng dụng cụ thể.

Cách tiếp cận truyền thống cho bài toán mô hình hóa ngôn ngữ là sử dụng mô hình ngôn ngữ n-gram. Sở dĩ nó có tên như vậy là vì nó sử dụng giả định markov bậc n -

1 để ước lượng xác suất $p\left(w_{i+1}=m|w_1,..,w_i\right) \approx p\left(w_{i+1}=m|w_{i-(n-1)},...,w_i\right)$. Theo mô hình ngôn ngữ n-gram, xác suất để từ m theo sau chuỗi các từ $w_1,...,w_i$ là:

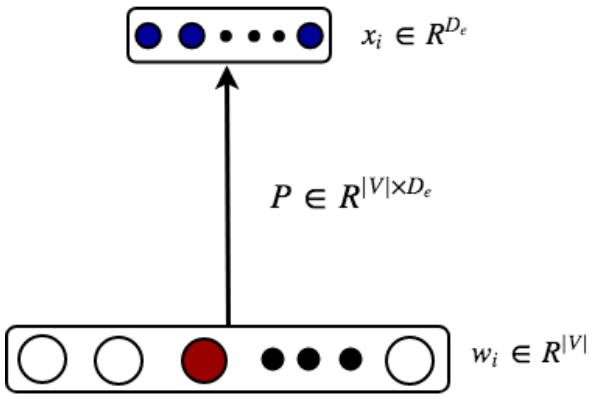
$$p\left(w_{i+1}=m|w_{i-(n-1):i}\right)=\frac{\#\left(w_{i-(n-1)},...,w_{i+1}\right)}{\#\left(w_{i-(n-1)},...,w_i\right)} \tag{2.5}$$

trong đó $\#(w_{i:j})$ là số lần xuất hiện của chuỗi $w_i,...,w_j$ trong tập ngữ liệu. Các mô hình n-gram thông thường sử dụng $n=2$ và $n=1$ được gọi lần lượt là mô hình *trigram* và mô hình *bigram*. Trong trường hợp mô hình không sử dụng ngữ cảnh để ước lược lượng xác suất của một từ ($n=0$), thì mô hình này được gọi là mô hình *unigram*.

Mô hình ngôn ngữ n-gram là một mô hình phi tham số, nó có một ưu điểm là tốc độ tính toán rất nhanh (do các xác suất của các n-gram đã được tính toán trước và lưu trữ lại) và dễ dàng mở rộng cho nhiều lĩnh vực (ta chỉ cần tập ngữ liệu thuộc lĩnh vực ấy). Mặc dù có một vài mô hình ngôn ngữ có kết quả tốt hơn so với mô hình n-gram, tuy nhiên các mô hình đó thường kèm theo độ phức tạp tính toán lớn và kết quả cải thiện không đáng kể. Do đó, mô hình n-gram vẫn được sử dụng phổ biến cho đến ngày nay.

Thuy nhiên, mô hình n-gram vẫn còn nhiều hạn chế. Đầu tiên, mặc dù tốc độ tính toán của mô hình n-gram là nhanh, nhưng kèm theo đó là sự đánh đổi về mặt lưu trữ. Nếu tập ngữ liệu là lớn thì chúng ta phải lưu những bảng xác suất khổng lồ, điều này gây ra khó khăn khi ta muốn sử dụng mô hình này trên các thiết bị có bộ nhớ nhỏ như thiết bị di động hay cảm biến. Thứ hai, điểm yếu của mô hình này đến từ việc sử dụng giả định markov; giả định này khiến cho mô hình n-gram không thể nắm bắt được các thông tin ngữ cảnh dài hạn. Ví dụ cho trước ngữ cảnh *"Columbus is the man who discovered __"*, ta muốn từ tiếp theo sẽ là *"America"*; một mô hình 5-gram không thể dự đoán được từ tiếp theo là *"America"* vì độ dài ngữ cảnh tối đa của nó chỉ đạt đến từ *"is"*.

Thứ ba, ta thấy rằng trong mô hình n-gram, nếu một chuỗi $w_{i:j} \subseteq w_1,...,w_n$ chưa từng xuất hiện trong tập ngữ liệu, tức $\#\left(w_i,...,w_j\right)=0$ thì xác suất ước lượng $p(w_{j+1}|w_i,...,w_j)$ sẽ có giá trị bằng 0. Điều này dẫn đến xác suất của toàn bộ chuỗi $w_1,...,w_n$ cũng bằng 0 do phép nhân trong công thức tính của nó (công thức 2.3). Việc xác suất bằng 0 xảy ra khá thường xuyên do sự giới hạn của tập ngữ liệu. Một cách để tránh việc xảy ra các sự kiện xác suất bằng không là sử dụng những kỹ thuật *làm mịn* (smoothing



Hình 2.1: Minh họa cách biểu diễn từ bằng "word embedding": một từ w_i được biểu diễn dưới dạng véc-tơ "one-hot", w_i sau đó được biến đổi thành x_i dựa trên một phép biến đổi tuyến tính với ma trận P của phép biến đổi. $|V|$ là số lượng từ vựng, D_e là số chiều của "word embedding" được xác định trước.

techniques). Làm mịn bảo đảm tất cả mọi chuỗi đều có một xác suất xuất hiện (mặc dù nhỏ). Kỹ thuật làm mịn đơn giản nhất là làm mịn thêm α (add-α smoothing) [11]; nó bảo đảm bất kỳ chuỗi nào cũng xuất hiện ít nhất α lần. Với làm mịn thêm α ta được:

$$p_{add-\alpha}\left(w_{i+1} = m | w_{i-k}, \ldots, w_i\right) = \frac{\#\left(w_{i-(n-1)}, \ldots, w_{i+1}\right) + \alpha}{\#\left(w_{i-(n-1)}, \ldots, w_i\right) + \alpha |V|} \tag{2.6}$$

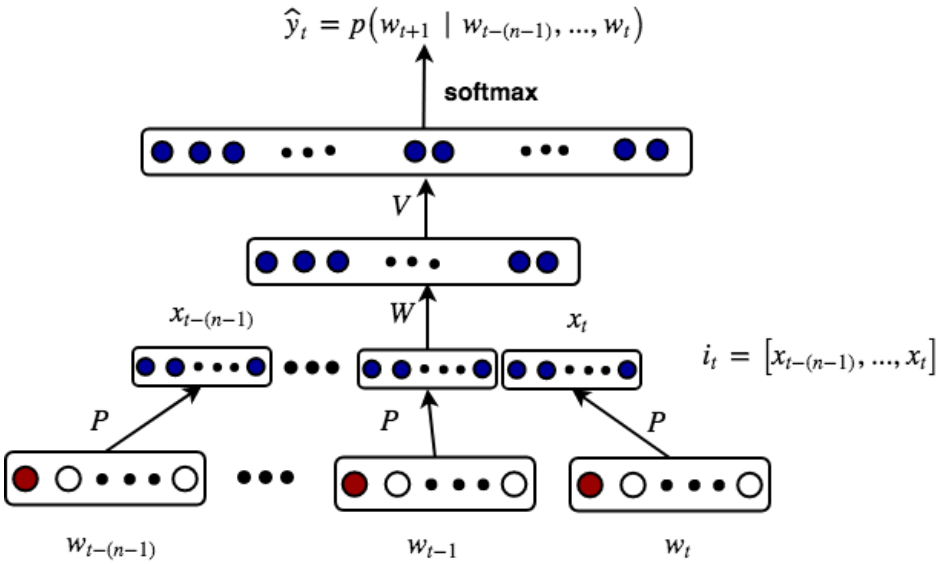
trong đó $|V|$ là số lượng từ vựng trong tập ngữ liệu. Hiện nay phương pháp làm mịn Kneser-Key là phương pháp phổ biến và đạt được kết quả tốt nhất [18].

Ngoài ra còn một vấn đề với mô hình n-gram là **tính tổng quát hóa không cao** [3]. Điều này có thể được giải thích bằng việc mô hình này coi các cụm từ là rời rạc khiến nó không có khả năng nắm bắt được tính tương tự về ngữ nghĩa giữa các từ [3]. Xét hai câu sau: $s_1 =$ *"She is a good cook"* và $s_2 =$ *"She cooks very cook"*, ta thấy rằng hai câu này về mặt ngữ nghĩa là tương đương nhau nên ta muốn xác suất của chúng cũng xấp xỉ nhau $p(s_2) \approx p(s_1)$. Tuy vậy, do một lý do nào đó mà câu s_2 xuất hiện ít hơn rất nhiều so với câu s_1 nên $p(s_2) << p(s_1)$. Do đó, có thể thấy rằng việc không nắm bắt tính tương tự về ngữ nghĩa làm cho mô hình ngôn ngữ n-gram có tính tổng quát hóa không cao.

Trong bài báo "A Neural Probabilistic Language Model" [3], giáo sư Yoshua Bengio và các cộng sự đã đề xuất một mô hình ngôn ngữ dựa trên mạng nơ-ron truyền thẳng. Nó giải quyết hai vấn đề mà mô hình n-gram gặp phải: một là mô hình ngôn

ngữ dựa trên mạng nơ-ron truyền thẳng là một mô hình tham số, nghĩa là đầu ra sẽ được tính dựa trên các tham số của mô hình sau khi được huấn luyện), điều này giúp giảm bộ nhớ lưu trữ - đặc biệt hơn những tham số này sẽ được chia sẻ cho toàn bộ mô hình, điều này giúp tăng tính tổng quát hóa [3]; hai là nó sử dụng biểu diễn véc-tơ để biểu diễn các từ gọi là "word embedding", điều này giúp mô hình có khả năng nắm bắt tính tương tự về ngữ nghĩa giữa các từ. Hình 2.1 mô tả cách tạo ra các véc-tơ "embedding". "Word embedding" sẽ được nhắc lại một lần nữa trong mục 2.2: Biểu diễn từ trong mạng học sâu.

Tuy nhiên, mô hình này vẫn dựa trên giả định markov. Tại mỗi thời điểm, đầu vào của mạng là một nối dài của các véc-tơ "embedding" tại những thời điểm trước đó $i_t = \left[x_{t-(n-1)}, ..., x_t\right]$ trong đó x_i là véc-tơ "embedding" của từ w_i, n là bậc của giả định markov (hình 2.2 mô tả cách hoạt động của mô hình). Đây là một khuyết điểm lớn của mô hình khi nó chỉ nắm bắt được thông tin trong một vùng $n-1$ từ trước đó. Do những khuyết điểm này, một mô hình ngôn ngữ dựa trên *mạng nơ-ron hồi quy* được đề xuất trong [25]. Mô hình này kế thừa những ưu điểm của mô hình ngôn ngữ dựa trên mạng nơ-ron truyền thẳng và có khả năng nắm bắt được thông tin trên một vùng có chiều dài bất kỳ. Mô hình này sẽ được nói đến trong phần kiến thức cơ bản về mạng nơ-ron hồi quy và được giải thích chi tiết trong chương 3 trong mục bộ giải mã.



Hình 2.2: Minh họa mô hình ngôn ngữ dựa trên mạng nơ-ron truyền thẳng: mô hình nhận đầu vào là các véc-tơ ngữ cảnh $w_{t-(n-1)}, ..., w_t$ dưới dạng "one-hot", sau đó những véc-tơ này được biểu diễn lại dưới dạng các véc-tơ "embedding" $x_{t-(n-1)}, ..., x_t$. Những véc-tơ "embedding" được nối dài tạo thành véc-tơ đầu vào i_t, i_t lần lượt được đưa qua tầng ẩn tanh và sau đó là softmax. Tại đây mô hình dự đoán đầu ra: là một véc-tơ "one-hot" dựa trên véc-tơ đầu ra ở tầng softmax.

2.2 Biểu diễn từ trong mạng học sâu

Trong phần mô hình ngôn ngữ, chúng tôi có nhắc đến "word embedding", trong mục này, chúng tôi sẽ nói thêm về nó. Để máy tính có thể hiểu được và xử lí được những thông tin đến từ thế giới thực, những thông tin đó phải được biểu diễn dưới dạng các con số. Trong các lĩnh vực như thị giác máy tính, xử lý tiếng nói, ... dữ liệu có thể được biểu diễn một cách dễ dàng thông qua dạng ban đầu của chúng vì hầu như các loại dữ liệu này là các số thực (điểm ảnh, histogram của âm thanh). Lĩnh vực xử lí ngôn ngữ tự nhiên gặp khó khăn trong việc biểu diễn các từ ngữ trên máy tính. Ví dụ: những từ như "tôi", "bạn", "học", ... được viết dưới dạng ngôn ngữ tự nhiên mà máy tính không thể nào hiểu được. Làm thế nào để biểu diễn ngôn ngữ tự nhiên trong máy tính là một trong những bài toán rất quan trọng trong lĩnh vực xử lí ngôn ngữ tự nhiên mà cần phải được giải quyết. Biểu diễn dạng con số của các từ trong máy tính được gọi là *biểu diễn từ* (word representation).

Phương pháp đơn giản nhất để biểu diễn các từ là đánh số thứ tự cho các từ đó. Để làm được việc này, đầu tiên, ta cần phải có một bộ từ vựng mà được sử dụng để giải

quyết một bài toán nào đó. Bộ từ vựng này sẽ được giữ một kích thước cố định trong suốt quá trình giải quyết bài toán (ví dụ như kích thước bộ từ vựng $V = 50000$ từ). Sau đó, ta thực hiện đánh số thứ tự cho tất cả các từ trong bộ từ vựng đó như: "anh" = 0, "ăn" = 1, ..., "yêu" = 49999. Khi ta đọc một câu ở ngôn ngữ tự nhiên vào máy tính, các từ trong câu sẽ được biểu diễn bằng các chỉ số đã được gán. Ví dụ:

Hôm nay tôi đi học.
23 66 150 31 54

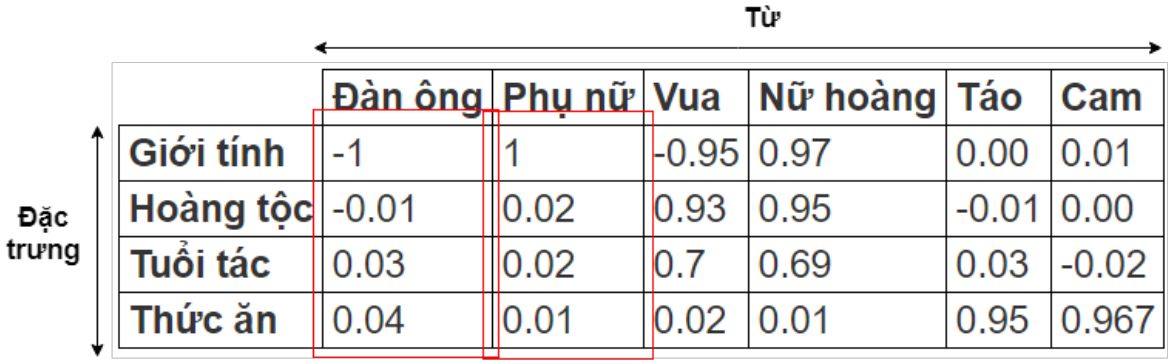
Tuy nhiên, cách này hoàn toàn không biểu diễn được ý nghĩa của từ. Thậm chí nó còn làm cho máy hiểu sai vì cách biểu diễn này. Những từ có giá trị lớn hơn (chỉ số index lớn hơn) hoặc là quan trọng hơn, hoặc là không quan trọng. Cách hiểu này là một thảm họa cho những mô hình sử dụng cách biểu diễn như thế. Để tránh cho máy hiểu sai ý nghĩa của từ như trên, phương pháp *one-hot encoding* được sử dụng. Phương pháp này biểu diễn các từ thành các véc-tơ có kích thước V chiều (véc-tơ từ). Trong đó tất cả các phần tử trong véc-tơ đều bằng 0, chỉ trừ phần tử tại vị trí có chỉ số (số thứ tự) giống với chỉ số của từ đó trong bộ từ vựng. Ví dụ: từ "tôi" có chỉ số trong bộ từ vựng V là 150, vì vậy, phần tử thứ 150 trong véc-tơ biểu diễn từ "tôi" sẽ có giá trị là 1, còn tất cả phần tử còn lại bằng 0. // TODO: Vẽ 1 hình cho 2 ví dụ về index và one-hot encoding.

Với cách biểu diễn bằng one-hot encoding thì ý nghĩa của từ sẽ không bị máy tính hiểu sai như trên tuy nhiên thì các véc-tơ từ này lại không thể hiện được bất kì mối quan hệ nào giữa các từ. Như từ "tôi" và từ "đi" trong hình trên (// TODO: ref), chúng không thể hiện được mối quan hệ về ngữ nghĩa của nó với nhau. Dễ thấy rằng dù với việc sử dụng bất kì độ đo tương đồng nào thì đều cho ra kết quả rằng hai véc-tơ từ không có liên quan với nhau (giá trị của độ đo bằng 0). Dù là như thế, nhưng phương pháp one-hot encoding này vẫn có được một số các lợi thế. Đó là các từ được biểu diễn dưới dạng các véc-tơ có số chiều bằng nhau. Có rất nhiều thuật toán thao tác trên các dữ liệu mà các điểm dữ liệu có số chiều bằng nhau, do đó thuận lợi hơn cho việc xử lí dữ liệu. Ngoài việc không biểu diễn được các mối quan hệ giữa các từ, one-hot encoding còn gặp hạn chế về kích thước của véc-tơ từ. Khi kích thước bộ dữ liệu V lớn như 50000, 100000, 200000, v.v... thì dẫn tới kích thước của các véc-tơ từ cũng lớn theo. Nếu cần xử lí một câu dài hay một đoạn văn thì chi phí tính toán rất lớn, tăng tuyến tính theo số từ cần xử lí là NV (N là số từ).

Phương pháp one-hot encoding được trình bày ở trên được gọi là biểu diễn *cục bộ (localist representation)*. Biểu diễn cục bộ là cách biểu diễn mà đối với mỗi một đối tượng (từ) cần được biểu diễn thì chỉ có một giá trị được kích hoạt (giá trị bằng 1) cho đối tượng đó. Đặc điểm chung của biểu diễn cục bộ là dễ hiểu cho người lẫn máy (chỉ có hạn chế là không biểu diễn được mối quan hệ giữa các từ), dễ cài đặt, dễ học và dễ gắn với các cách biểu diễn khác. Nhược điểm thì như đã trình bày ở trên, biểu diễn cục bộ cần có véc-tơ có kích thước tương ứng với số lượng đối tượng mà nó muốn biểu diễn. Ngoài ra, đối với những đối tượng nào mà có mối quan hệ cấu thành từ các đối tượng khác hay là góp phần cấu thành các đối tượng khác (ví dụ như bánh xe gồm có trục bánh xe, khung bánh xe và bánh xe góp phần tạo nên xe hơi) thì biểu diễn cục bộ biểu diễn không hiệu quả. Do vậy, cần có một cách biểu diễn khác mà giải quyết được các hạn chế này.

Biểu diễn phân tán (distributed representation) là một cách biểu diễn mà có thể giải quyết các hạn chế của biểu diễn cục bộ. Một trong những công trình kinh điển trong thời kì đầu là của GS Hinton [31]. Tên gọi của biểu diễn cục bộ và biểu diễn phân tán đều là được đề xuất bởi GS Hinton. Biểu diễn phân tán này có hai tính chất quan trọng: giảm số chiều không gian và tính tương đồng ngữ nghĩa. Cụ thể, biểu diễn phân tán không còn dùng một phần tử trong véc-tơ để thể hiện một đối tượng nhất định mà các phần tử trong véc-tơ thể hiện một số đặc trưng, tính chất có khả năng phân biệt được các đối tượng. Hình 2.3 minh họa phương pháp biểu diễn phân tán. Các véc-tơ từ sẽ có kích thước nhất định. Mỗi phần tử trong véc-tơ thể hiện một đặc trưng nhất định của một từ. Tất cả các từ được biểu diễn bởi một tập các đặc trưng chung. Các từ sẽ được phân biệt với nhau thông qua các giá trị của các đặc trưng. Phương pháp biểu diễn này có thể biểu diễn được các mối quan hệ giữa các từ với nhau. Ví dụ, để xác định xem từ "Đàn ông" và từ "Phụ nữ" có tính tương đồng về ngữ nghĩa như thế nào: $\cos(\textit{"Đàn ông", "Phụ nữ"}) = -0.98$. Giá trị này cho thấy rằng từ "Đàn ông" và từ "Phụ nữ" có ý nghĩa trái ngược nhau. Nhìn vào các giá trị đặc trưng của véc-tơ từ của hai từ này, có thể xác định được rằng sự trái ngược về ngữ nghĩa này là về mặt giới tính.

Ví dụ trên đã làm rõ hơn về hai tính chất của biểu diễn phân tán là giảm số chiều không gian và tính tương đồng ngữ nghĩa. Phương pháp này không cần phải sử dụng kích thước véc-tơ lớn tương ứng với số lượng từ mà cần biểu diễn (bộ từ vựng V), chỉ



	Đàn ông	Phụ nữ	Vua	Nữ hoàng	Táo	Cam
Giới tính	-1	1	-0.95	0.97	0.00	0.01
Hoàng tộc	-0.01	0.02	0.93	0.95	-0.01	0.00
Tuổi tác	0.03	0.02	0.7	0.69	0.03	-0.02
Thức ăn	0.04	0.01	0.02	0.01	0.95	0.967

Hình 2.3: Minh họa biểu diễn phân tán. Để biểu diễn được các từ, phương pháp này sẽ xác định một số đặc trưng, tính chất chung mà dựa vào đó có thể phân biệt được tất cả các từ cần biểu diễn. Mỗi cột là một từ cần được biểu diễn, mỗi dòng là một đặc trưng được dùng để biểu diễn các từ đó.

cần sử dụng một số lượng đặc trưng nhất định E (E nhỏ hơn rất nhiều so với bộ từ vựng V) là có thể biểu diễn được. Những đặc trưng mà véc-tơ từ thể hiện góp phần thể hiện được các phương diện ngữ nghĩa nhất định giữa các từ. Ngoài ra, từ cách biểu diễn các từ bằng các véc-tơ như vậy, biểu diễn phân tán mang lại một số kết quả thú vị khi thực các phép tính số học đối với các véc-tơ từ đó. Sử dụng lại ví dụ được trình bày ở 2.3, giả sử cần tìm một từ nào mà thỏa mãn mối quan hệ "Đàn ông" -> "Phụ nữ" = "Vua" -> ?. Gọi véc-tơ từ cần tìm là e_k, có thể biểu diễn mối quan hệ trên lại là: "Đàn ông" - "Phụ nữ" = "Vua" - e_k $\Leftrightarrow$ e_k = "Vua" - "Đàn ông" + "Phụ nữ". Thực hiện tính toán, có được véc-tơ từ $e_k = [1.05, 0.94, 0.69, -0.06]^T$. Trong các từ thì $e_k \approx$ "Nữ hoàng" (có khoảng cách tới từ "Nữ hoàng" nhỏ nhất), do đó $e_k =$ "Nữ hoàng". Vậy mối quan hệ tìm được là "Đàn ông" -> "Phụ nữ" = "Vua" -> "Nữ hoàng". Đây là tính chất rất hữu ích của biểu diễn phân tán, dựa vào tính chất này, các thuật toán học có thể tận dụng nó để làm tăng độ hiệu quả của mô hình và trực quan hóa các véc-tơ từ bằng cách vẽ đồ thị nhằm có cái nhìn trực quan hơn về các mối quan hệ giữa các từ.

Tuy biểu diễn phân tán có rất nhiều ưu điểm nhưng đi cùng với nó là một nhược điểm lớn và đó cũng là thách thức khó khăn. Làm sao để có thể xây dựng được các véc-tơ từ như vậy? Không thể xây dựng nó bằng cách thủ công như tự định nghĩa các đặc trưng và gán các giá trị cho từng từ trong bộ từ vựng được bởi vì công sức làm việc đó là rất lớn khi kích thước bộ từ vựng lớn lên. Các đặc trưng tự định nghĩa cũng rất khó thể hiện chính xác hoàn toàn được những ngữ nghĩa của các từ theo mong đợi. Do đó, các thuật toán, mô hình học biểu diễn phân tán một cách tự động được ra đời.

Các thuật toán, mô hình như thế được gọi chung là *word embedding* (tạm dịch là *phép nhúng từ*).

Từ word embedding được tạo ra bởi GS Bengio và cộng sự vào năm 2003 [4] khi họ đã huấn luyện một mô hình ngôn ngữ nơ-ron đồng thời học biểu diễn từ thông qua các tham số của mô hình. Tuy nhiên công trình này vẫn chưa thực sự thành công, cho tới năm 2008, Collobert và Weston [6] đã đề xuất một kiến trúc mạng nơ-ron sâu tổng quát. Hiện nay, có hai cách chính để học word embedding.

- Học theo chiến lược học không giám sát: sử dụng các thuật toán học đặc biệt để học biểu diễn từ dựa trên một tập dữ liệu. Thường là những tập dữ liệu thu thập được từ các nguồn khác nhau về mà chưa có được xử lí, đánh nhãn để giải quyết cho một bài toán cụ thể nào. Các tập dữ liệu này thường là dùng để huấn luyện cho mô hình ngôn ngữ.

- Học theo chiến lược có giám sát: các véc-tơ biểu diễn từ là một phần của một mô hình học có giám sát, cụ thể là các tham số trong mô hình. Mô hình sẽ học đồng thời cách giải quyết bài toán và các véc-tơ biểu diễn từ. Tùy thuộc vào bài toán mà mô hình này giải quyết như Dịch máy, Phân tích ý kiến của người dùng, các véc-tơ biểu diễn từ học được sẽ phù hợp duy nhất cho bài toán đó.

Hiện nay học theo chiến lược học không giám sát đang có được những kết quả ấn tượng. Cách học này giúp word embedding học được thường tốt hơn do tập dữ liệu không cần nhãn, vì thế tập dữ liệu dành cho việc học word embedding là rất lớn (có thể lên tới 1 tỉ từ) do dễ thu thập. Hơn nữa, word embedding học được còn có khả năng tổng quát hóa tốt (biểu diễn từ học được tốt trong nhiều ngữ cảnh). Đại diện cho cách học này là gồm Word2Vec [26] (sử dụng một mạng nơ-ron và ý tưởng đơn giản để học) và GloVe [30] (sử dụng thống kê tần số của các từ để học). Đây là hai word embedding mà được mọi người tin tưởng và sử dụng rộng rãi để giải quyết các bài toán khác nhau.

Học theo chiến lược có giám sát thì ít được biết đến hơn bởi vì word embedding học được theo cách này thì chỉ có hữu dụng với bài toán mà mô hình đang giải quyết. Nếu như mô hình đang giải quyết bài toán Dịch máy thì word embedding sẽ tập trung vào biểu diễn các từ sao cho phù hợp nhất với bài toán Dịch máy và không thể mang word embedding này để giải quyết bài toán khác.

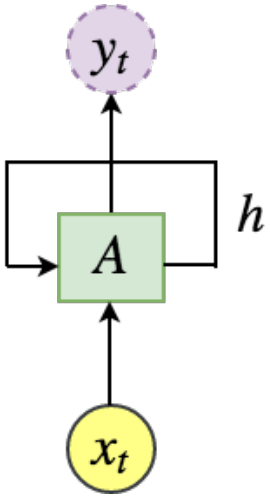
2.3 Mạng nơ-ron hồi quy (Recurrent neural network)

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 __"*. Dễ 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 [23]. 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) [9] gọi tắt là *RNN* là một nhánh của nạng 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 véc-tơ được gọi là *trạng thái ẩn*.

Xét một chuỗi đầu vào $x = x_1, x_2, ..., x_n$. 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



Hình 2.4: 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.

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:

$$h_t = f(h_{t-1}, x_t) \tag{2.7}$$

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 2.4 thể hiện định nghĩa hồi quy của RNN.

Thông thường, hàm f là một hàm phi tuyến như hàm **sigmoid** hay hàm **tanh**. Xét một RNN với công thức cụ thể như sau:

$$h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{2.8}$$

Trong đó:

- ϕ là một hàm kích hoạt (ví dụ: sigmoid, tanh hay ReLU).
- $h_t \in \mathbb{R}^{D_h}$ là trạng thái ẩn tại bước thời gian hiện tại có chiều dài D_h.
- $x_t \in \mathbb{R}^{D_x}$ là véc-tơ đầu vào hiện tại có chiều dài D_x.
- $h_{t-1} \in \mathbb{R}^{D_h}$ là trạng thái ẩn tại bước thời gian trước đó.

- $W_{xh} \in \mathbb{R}^{D_x \times D_h}, W_{hh} \in \mathbb{R}^{D_h \times D_h}$ và $b_h \in \mathbb{R}^{D_h}$ lần lượt là hai ma trận trọng số và véc-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 liền kề. Véc-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 véc-tơ chứa tri thức có sẵn như trường hợp của bộ giải mã như chúng tôi đã đề cập trong chương 1.

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 phần phối xác suất trên tập các các lớp ở đầu ra. Phân phối này nhằm dự đoán vị trí xuất hiện của $\hat{y}_t$.

$$\hat{y}_t = \text{softmax}(W_{hy} h_t + b_y) \tag{2.9}$$

Trong đó:

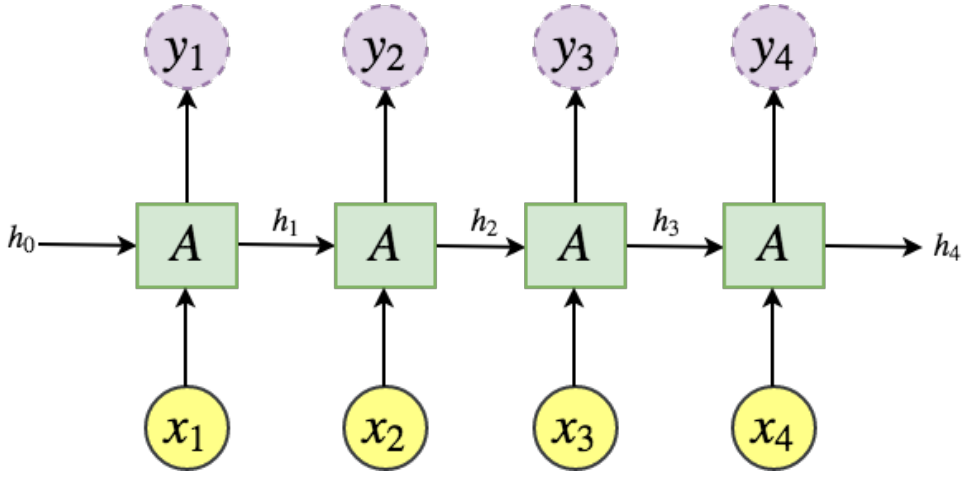
- softmax là một hàm kích hoạt với $\text{softmax}(v) = \frac{e^{v_j}}{\sum_{k=1}^{K} e^{v_k}}$, $j = 1, ..., K$. K là độ dài của véc-tơ v.

- $h_t \in \mathbb{R}^{D_h}$ là trạng thái ẩn tại bước thời gian hiện tại.

- $W_{hy} \in \mathbb{R}^{L \times D_h}$ và $b_y \in \mathbb{R}^{L}$ lần lượt là hai ma trận trọng số và véc-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 véc-tơ 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



Hình 2.5: Mô hình RNN được dàn trải (unrolled), ví dụ trong 4 bước thời gian.

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 (feedforward neural network).

Với một số lượng hữu hạn các bước thời gian, mô hình RNN trên hình 2.4 có thể được dàn trải ra (unrolled). Dạng dàn trải này được miêu tả trực quan như trên hình 2.5. 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) [33]. 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.

2.3.1 Huấn luyện mạng nơ-ron hồi quy

Xét một chuỗi đầu vào $x = x_1, x_2, ..., x_n$ với đầu ra tương ứng $y = y_1, y_2, ..., y_n$. 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:

$$h_t = \phi\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right) \tag{2.10}$$

$$s_t = W_{hy} h_t + b_y \tag{2.11}$$

$$\hat{y}_t = \text{softmax}(s_t) \tag{2.12}$$

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 véc-tơ phân phối xác suất có độ dài L. Để so sánh với $\hat{y}_t$, y_t được chuẩn hóa thành một véc-tơ dạng "one hot" có nghĩa là một véc-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:

$$L_t = -y_t^T \log(\hat{y}_t) \tag{2.13}$$

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

$$\theta_t = \theta_k \tag{2.14}$$

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

$$\frac{\partial L}{\partial \theta_t} = \frac{\partial L}{\partial \theta_k} \tag{2.15}$$

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.

$$L = \sum_t L_t = -\sum_t y_t^T \log(\hat{y}_t) \tag{2.16}$$

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:

$$\theta \leftarrow \theta - \eta \nabla_\theta L \tag{2.17}$$

Ở đây, $\nabla_\theta L$ là "gradient" của hàm độ lỗi ứng với tham số θ. η được gọi là hệ số học (learning rate) là một siêu tham số quyết định rằng θ sẽ thay đổi nhiều bao nhiêu theo giá trị gradient tính được. 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:

$$\nabla_\theta L_t = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial \theta} = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial h_t} \frac{\partial h_t}{\partial \theta} \tag{2.18}$$

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

$$\nabla_{s_t} L_t = \frac{\partial L_t}{\partial s_t} = \hat{y}_t - y \tag{2.19}$$

Ta cũng dễ dàng tính được:

$$\frac{\partial s_t}{\partial h_t} = W_{hy}^T \tag{2.20}$$

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:

$$\nabla_{W_{hy}} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial W_{hy}} = (\hat{y}_t - y) h_t^T \tag{2.21}$$

$$\nabla_{b_y} L_t = \frac{\partial L_t}{\partial s_t} \frac{\partial s_t}{\partial b_y} = \hat{y}_t - y \tag{2.22}$$

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" cuả 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:

$$\frac{\partial f}{\partial r} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} \tag{2.23}$$

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

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial \theta^{(2)}} \tag{2.24}$$

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

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial \theta^{(2)}} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial \theta^{(2)}} \tag{2.25}$$

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

$$\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial \theta^{(2)}} \tag{2.26}$$

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \frac{\partial h_t}{\partial h_t}\frac{\partial h_t}{\partial \theta^{(2)}} \tag{2.27}$$

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

$$\frac{\partial h_t}{\partial \theta^{(2)}} = \sum_{r=0}^{t} \frac{\partial h_t}{\partial h_r}\frac{\partial^{\dagger} h_r}{\partial \theta^{(2)}} \tag{2.28}$$

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:

$$\nabla_{W_{hh}} L_t = \frac{\partial L_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}\frac{\partial h_t}{\partial W_{hh}} = (\hat{y}_t - y) W_{hy}^T \sum_{r=0}^{t} \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial W_{hh}} \tag{2.29}$$

$$\nabla_{W_{xh}} L_t = \frac{\partial L_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}\frac{\partial h_t}{\partial W_{xh}} = (\hat{y}_t - y) W_{hy}^T \sum_{r=0}^{t} \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial W_{xh}} \tag{2.30}$$

$$\nabla_{b_h} L_t = \frac{\partial L_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}\frac{\partial h_t}{\partial b_h} = (\hat{y}_t - y)W_{hy}^T \sum_{r=0}^{t} \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial b_h} \tag{2.31}$$

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

$$\frac{\partial h_t}{\partial h_r} = \prod_{i=r}^{t} \frac{\partial h_i}{\partial h_{i-1}} \tag{2.32}$$

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

$$\begin{aligned}
\frac{\partial h_i}{\partial h_{i-1}} &= \left[\frac{h_i}{h_{i-1,1}}, \ldots, \frac{h_i}{h_{i-1,n}}\right] \\
&= \begin{bmatrix} \frac{h_{i,1}}{h_{i-1,1}} & \cdots & \frac{h_{i,1}}{h_{i-1,n}} \\ \vdots & \ddots & \vdots \end{bmatrix} \\
&= W_{hh}^T \text{diag}\left(\phi'(h_i)\right)
\end{aligned} \tag{2.33}$$

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

$$\nabla_{b_y} L = \sum_t \hat{y}_t - y \tag{2.34}$$

$$\nabla_{b_h} L = \sum_t \sum_{r=0}^{t} (\hat{y}_t - y)W_{hy}^T \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial b_h} \tag{2.35}$$

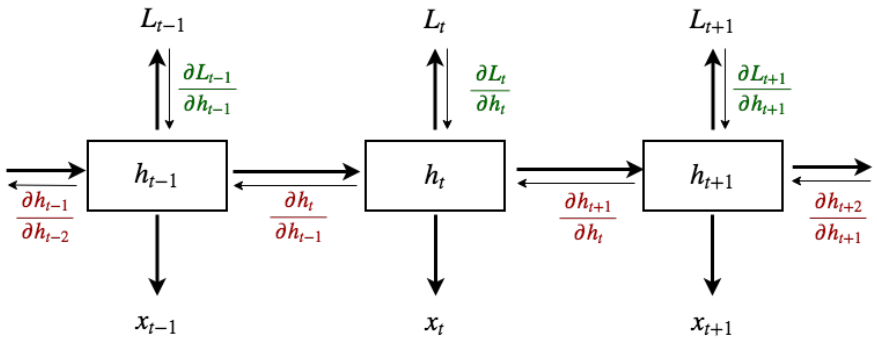
$$\nabla_{W_{hy}} L = \sum_t (\hat{y}_t - y)h_t^T \tag{2.36}$$

$$\nabla_{W_{hh}} L = \sum_t \sum_{r=0}^{t} (\hat{y}_t - y)W_{hy}^T \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial W_{hh}} \tag{2.37}$$

$$\nabla_{W_{xh}} L_t = \sum_t \sum_{r=0}^{t} (\hat{y}_t - y)W_{hy}^T \frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial W_{xh}} \tag{2.38}$$

2.3.2 Thách thức trong việc học các phụ thuộc dài hạn

Theo thuật toán BPTT, ta biết rằng gradient theo một tham số hồi quy ở mỗi bước thời gian $\nabla_{W_{hh}} L_t$ là tổng gradient của L_t theo tham số đó ở tất cả trạng thái ẩn tại các bước



Hình 2.6: Tại mỗi bước thời gian t, đội lỗi không chỉ được lan truyền qua các tầng ở bước thời gian hiện tại mà còn phải lan truyền thông tin độ lỗi qua tất cả thời điểm trước đó. Chính sự lan truyền qua thời gian này gây ra hiện tượng bùng nổ hoặc biến mất gradient.

thời gian trước đó $1,\ldots,t$ (công thức 2.38). Chính bởi cơ chế tính toán gradient như vậy, trên lý thuyết RNN có khả năng học được những phụ thuộc dài hạn với độ dài bất kỳ. Tuy nhiên, trong thực tế, việc học các phụ thuộc dài hạn là một vấn đề lớn của RNN được gây ra bởi hai nguyên nhân: sự biến mất gradient (vanishing gradient) và sự bùng nổ gradient (exploding gradient). Nói một cách đơn giản, gradient của một hàm mục tiêu ứng với một tham số nói lên rằng hàm số kia sẽ thay đổi bao nhiêu khi tham số thay đổi. Sự biến mất gradient xảy ra khi gradient trở nên cực kỳ nhỏ; nó khiến cho hay sự thay đổi của tham số không ảnh hưởng đến sự thay đổi của hàm mục tiêu. Ngược lại sự bùng nổ gradient khiến gradient trở nên lớn một cách đột ngột; điều này khiến cho một sự thay đổi nhỏ trong tham số cũng ảnh hưởng mạnh đến hàm mục tiêu, hoặc đơn giản là gradient lớn đến mức không thể tính toán được. Sự biến mất và bùng nổ gradient được phát hiện và trình bày trong những nghiên cứu [13], [1], [28]. Trong mục này, chúng tôi sẽ trình bày lại nguyên nhân và mô tả một số giải pháp cho hai vấn đề này. Những chứng minh của chúng tôi dựa trên chứng minh trong nghiên cứu [28].

Nhắc lại rằng trong RNN, gradient của hàm lỗi tại bước thời gian t theo tham số hồi quy $\theta \in \{W_{hh}, W_{xh}, b_h\}$ là:

$$\nabla_\theta L = \frac{\partial L_t}{\partial s_t}\frac{\partial s_t}{\partial h_t}\sum_{r=0}^{t}\frac{\partial h_t}{\partial h_r}\frac{\partial h_r}{\partial \theta} \tag{2.39}$$

và

$$\frac{\partial h_t}{\partial h_r} = \prod_{i=r}^{t} W_{hh}^T \text{diag}\left(\phi'(h_i)\right) \tag{2.40}$$

Đầu tiên, để cho đơn giản, ta hãy xem ϕ là hàm một hàm đồng nhất (identity function) với $\phi(x) = x$ như vậy theo công thức trên, ta có:

$$\frac{\partial h_t}{\partial h_r} = \prod_{i=r}^{t} W_{hh}^T = \left(W_{hh}^T\right)^{t-r} \tag{2.41}$$

Với W_{hh} là một ma trận vuông, nó có thể được phân tích thành dạng:

$$W_{hh} = \Sigma \text{diag}\,(\lambda)\,\Sigma^{-1} \tag{2.42}$$

Với Σ, λ lần lượt là ma trận véc-tơ riêng và véc-tơ trị riêng của ma trận W_{hh}. Ta biết rằng lũy thừa của W_{hh} cũng chính là lũy thừa của véc-tơ trị riêng của nó λ. Khi số mũ của phép lũy thừa lớn, những véc-tơ riêng ứng với trị riêng $\lambda_i < 1$ sẽ giảm theo hàm mũ, những véc-tơ riêng ứng với trị riêng $\lambda_i > 1$ sẽ tăng theo hàm mũ. Nói cách khác, *điều kiện đủ* để xảy ra "gradient" biến mất là trị riêng lớn nhất của ma trận kết nối hồi quy λ_{max} có giá trị nhỏ hơn 1. Để xảy ra "gradient" bùng nổ, *điều kiện cần* là $\lambda_{max} > 1$.

Trong trường hợp tổng quát với một hàm kích hoạt ϕ bất kỳ, với ϕ' bị chặn trên bởi $\gamma \in \mathbb{R}$ và do đó $||diag(\phi'(h_k))|| \leq \gamma$. Với $\lambda_{max} < \frac{1}{\gamma}$, hiện tượng "gradient" biến mất sẽ xảy ra. Theo công thức 2.33:

$$\forall i, \left\| \frac{\partial h_{i+1}}{\partial h_i} \right\| \leq \left\| W_{hh}^T \right\| \left\| diag(\phi'(h_k)) \right\| < \frac{1}{\gamma}\gamma < 1 \tag{2.43}$$

Đặt $\beta \in \mathbb{R}$ sao cho $\forall i, \left\| \frac{\partial h_{i+1}}{\partial h_i} \right\| \leq \beta < 1$, như vậy ta có thể thấy được rằng:

$$\lim_{t-r\to\infty} \frac{\partial h_t}{\partial h_r} = \prod_{i=r}^{t} W_{hh}^T \text{diag}\left(\phi'(h_i)\right) \leq \beta^{t-r} = 0 \tag{2.44}$$

Bằng cách đảo ngược chứng minh này ta được *điều kiện cần* để xảy ra "gradient" bùng nổ là trị riêng lớn nhất λ_{max} lớn hơn $\frac{1}{\gamma}$.

Chứng minh trên cho thấy rằng, khi $t - r$ lớn, tức là khoảng cách giữa từ đang xét và một từ trong quá khứ là lớn thì $\nabla_{\theta_r} L_t$ sẽ hoặc rất bé nếu $\lambda_{max} < \frac{1}{\gamma}$ hoặc có khả năng trở nên rất lớn nếu $\lambda_{max} > \frac{1}{\gamma}$. Trong thực tế, với ϕ là hàm **tanh** ta có $\gamma = 1$ trong khi với ϕ là hàm sigmoid, ta có $\gamma = 1/4$. Nếu ta khởi tạo tham số θ nhỏ hoặc dùng hàm

kích hoạt có γ nhỏ như hàm **sigmoid** thì sự biến mất gradient sẽ dễ rất dễ xảy ra; nó khiến cho mô hình chỉ học được những phụ thuộc cục bộ. Ngược lại, nếu θ được khởi tạo với giá trị lớn, gradient tại các bước thời gian ở xa sẽ bùng nổ và kết quả là mô hình không thể học được [28].

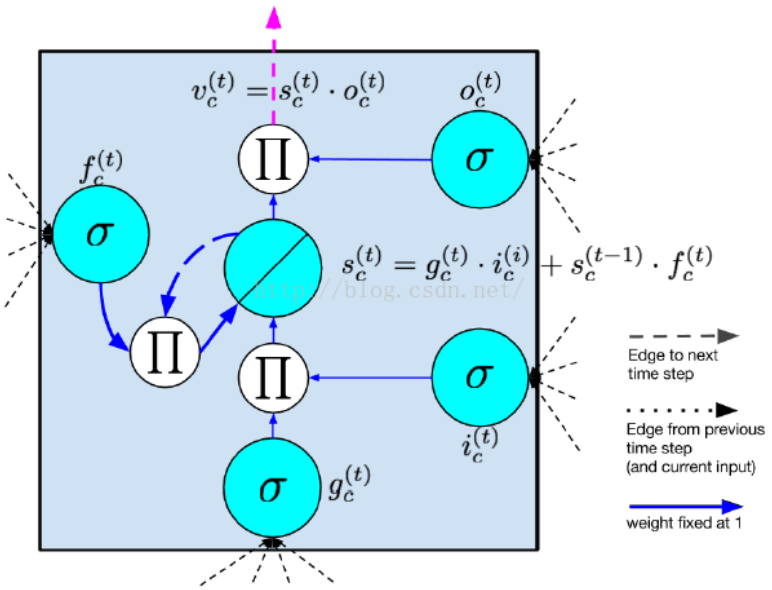
Một kỹ thuật để đối phó với sự bùng nổ gradient được đề là chuẩn hóa gradient về một giá trị nếu nó vượt quá một ngưỡng nào đó. Kỹ thuật này gọi là "gradient norm clipping" [28]. Cụ thể, ta đặt một ngưỡng là chặn trên cho gradient, tại mỗi bước thời gian t trong lúc "backpropagation", nếu độ lớn của gradient lớn hơn ngưỡng này ta sẽ chuẩn hóa trở về một giá trị nhỏ hơn. Giải pháp này được áp dụng trong thực tế để ngăn chặn các giá trị **NaN** (Not a Number) trong gradient và cho phép quá trình huấn luyện tiếp tục. "Gradient clipping" được trình bày trong thuật toán 2.1.

Thuật toán 2.1 Gradient clipping

Đầu vào: Gradient ứng với tham số θ
Đầu ra: Gradient ứng với tham số θ được chuẩn hóa về một ngưỡng *threshold*
Thao tác:

1: $\hat{g} \leftarrow \frac{\partial E}{\partial \theta}$
2: **if** $\|\hat{g}\| \geq threshold$ **then**
3: $\quad \hat{g} \leftarrow \frac{threshold}{\|\hat{g}\|}\hat{g}$
4: **end if**

Mặt khác, vấn đề biến mất gradient là khó giải quyết hơn. Đã có nhiều cách tiếp cận được đề xuất để giải quyết vấn đề này như "Hessian Free Optimization" [24], "Echo State Network" [15], Long short-term memory [13] và một số phương pháp khác được liệt kê trong [2]. Trong số những phương pháp đó, "Long short-term memory" (LSTM) là phương pháp thường được sử dụng nhất. LSTM là phiên bản nâng cấp của RNN với các chê chế để giải quyết hiệu quả vấn đề biến mất gradient, hơn thế nữa, với các cơ chế của mình, LSTM còn có thể giải quyết vấn đề bùng nổ gradient và giúp mạng nơ-ron hồi quy học tốt hơn. Nó cũng chính là phiên bản mạng nơ-ron hồi quy mà chúng tôi sử dụng trong khóa luận này. Trong phần tiếp theo, chúng tôi sẽ trình bày về LSTM và giải thích vì sao LSTM hiệu quả.



Hình 2.7: Một LSTM cell với cổng quên, ô với ký hiệu σ đại diện cho những nút; các ô với ký hiệu $\prod$ đại diện cho các cổng. Ô tròn ở trung tâm đại diện cho trạng thái lưu giữ.

2.4 Long short-term memory (LSTM)

Hochreiter và Schmidhuber [13] 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 phép nhân ma trận theo sau là hàm kích hoạt phi tuyến, LSTM trực tiếp tính toán một ΔS_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 ΔS_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 2.5) được thay thế bằng một "memory cell" (hình 2.7). 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 [10].

- *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:

$$g_t = \phi\left(W_{gx}x_t + W_{gh}h_{t-1} + b_g\right) \tag{2.45}$$

- *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:

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \tag{2.46}$$

- *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:

$$s_t = s_{t-1} + g_t \odot i_t \tag{2.47}$$

- *Cổng quên (forget gate)*: Cổng quên là một đề xuất của [10] 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 2.47. Tại mỗi bước thời gian, giá trị của cổng quên f_t được tính như sau:

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \tag{2.48}$$

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

$$s_t = s_{t-1} \odot f_t + g_t \odot i_t \tag{2.49}$$

- *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, hàm kích hoạt ϕ có thể là hàm **tanh** hoặc **sigmoid** hoặc không sử dụng hàm kích hoạt nào. Tại mỗi bước thời gian, giá trị của cổng ra o_t được tính như sau:

$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right) \tag{2.50}$$

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

$$h_t = \phi(s_t) \odot o_t \tag{2.51}$$

2.4.1 Mạng nơ-ron hồi quy hai chiều

Chương 3

Dịch máy bằng mô hình học LSTM-Attention

Chương này trình bày về mô hình học LSTM-Attention cho bài toán Dịch máy dựa trên bài báo "Effective Approaches to Attention-based Neural Machine Translation" [22] được công bố tại hội nghị EMNLP của nhóm tác giả thuộc Đại học Stanford gồm Minh-Thang Luong, Hieu Pham, Christopher D. Manning. Chúng tôi trình bày về việc tổ chức các LSTM theo kiến trúc Bộ mã hóa - Bộ giải mã cùng với sử dụng các phiên bản của cơ chế Attention, đồng thời giải thích chúng dựa trên cơ sở Toán học. Cụ thể:

- *LSTM: chúng tôi tổ chức các theo kiến trúc Bộ mã hóa - Bộ giải mã để tránh được ràng buộc về số lượng từ ở ngôn ngữ nguồn và ngôn ngữ đích khi dùng một LSTM để dịch.*
- *Attention: chúng tôi tìm hiểu sự hạn chế hiện có của kiến trúc Bộ mã hóa - Bộ giải mã khi thực hiện dịch những câu dài. Từ đó, cơ chế Attention được sử dụng 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 Mô hình học LSTM cho bài toán Dịch máy

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 sử dụng một mô hình LSTM áp dụng cho bài toán Dịch máy thì phải chịu một số hạn chế. Một trong những hạn chế lớn nhất chính là ràng buộc chiều dài của câu đầu vào (câu ở ngôn ngữ nguồn) phải bằng chiều dài của câu đầu ra (câu ở ngôn ngữ đích). Hơn nữa, đối với những ngôn ngữ có trật tự từ khác nhau như tiếng Anh (chủ ngữ-vị ngữ-tân ngữ) và tiếng Nhật (chủ ngữ-tân ngữ-vị ngữ) thì với một LSTM mô hình không thể dịch tốt được.

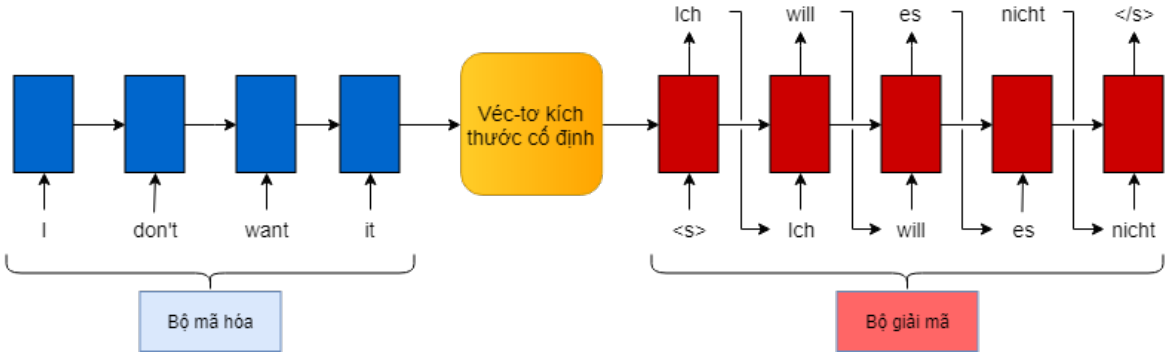
Để giải quyết những hạn chế của đó, nhóm tác giả Sutskever, 2014 [32] đã đề xuất một kiến trúc tên gọi là *Chuỗi tới Chuỗi (Sequence to Sequence - Seq2Seq)*. Tuy nhiên, mọi người thường gọi kiến trúc này là kiến trúc Bộ mã hóa - Bộ giải mã (Encoder-Decoder). Trong khóa luận này, chúng tôi tìm hiểu và dùng kiến trúc này để giải quyết bài toán Dịch máy và gọi nó là Bộ mã hóa - Bộ giải mã.

Kiến trúc Bộ mã hóa -Bộ giải mã như tên gọi, gồm có hai 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 một LSTM. Các LSTM trong kiến trúc sẽ được sắp xếp một cách hợp lí dựa trên những đặc tính của chúng sao cho phù hợp với bài toán Dịch máy.

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 hai bộ phận này trong các phần tiếp theo. Để thuận lợi hơn cho việc trình bày về mô hình học, đối với các véc-tơ và các ma trận, chúng tôi sẽ in đậm các kí hiệu đó.

3.1.1 Bộ mã hóa

Bộ mã hóa là một bi-LSTM (LSTM hai chiều). Bi-LSTM này nhận đầu vào là một câu ở ngôn ngữ nguồn $\boldsymbol{x} = \{x_1, ..., x_S\}$ với S là chiều dài, $x_1, ...x_S$ là các từ trong câu. Đầu ra là một véc-tơ có kích thước cố định. Véc-tơ này chứa các thông tin cần thiết để



Hình 3.1: Minh họa kiến trúc Bộ mã hóa - Bộ giải mã được sử dụng trong khóa luận. Kiến trúc có hai 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.

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, véc-tơ trạng thái ẩn $\boldsymbol{h}_S$ (trạng thái ẩn cuối cùng trong bi-LSTM) được sử dụng 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, mỗi 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. Điểm đặc biệt của bi-LSTM trong bộ mã hóa này so với các LSTM thông thường khác là do chúng tôi chỉ sử dụng véc-tơ trạng thái ẩn cuối cùng làm đầu ra của bộ mã hóa, vì vậy bi-LSTM này tại tất cả bước thời gian đều không xuất ra bất kỳ đầu ra nào. Hình 3.1 minh họa loại bi-LSTM được sử dụng cho bộ mã hóa trong khóa luận (bên trái). Tại mỗi bước thời gian, bi-LSTM nhận một từ ở câu ngôn ngữ nguồn và thực hiện cập nhật trạng thái ẩn cho đến hết câu.

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 một 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 $\boldsymbol{y} = \{y_1, ..., y_{S'}\}$ với S' là chiều dài của câu đích, $y_1, ..., y_{S'}$ là các từ trong câu đích. 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. Hình 3.1 minh họa loại uni-LSTM được sử dụng cho bộ giải mã trong khóa luận này (bên phải). Ở bước thời gian đầu tiên, uni-LSTM nhận đầu vào là kí tự bắt đầu câu <*s*> và dự đoán từ đầu tiên của câu đích. Tại mỗi bước thời gian tiếp theo, bi-LSTM này nhận đầu vào là một từ đã dự đoán ở bước thời gian trước và dự đoán từ tiếp theo cho đến khi bi-LSTM dự đoán ra kí hiệu kết thúc câu </*s*>.

3.1.3 Huấn luyện mô hình

Phần này chúng tôi trình bày quá trình huấn luyện của mô hình Dịch máy được nêu ở trên trong một vòng lặp:

- Ở bộ mã hóa: nhận đầu vào là một câu nguồn, sau đó bi-LSTM trong bộ mã hóa thực hiện xử lí câu nguồn này. Trạng thái ẩn cuối cùng của bi-LSTM được sử dụng để làm véc-tơ có kích thước cố định.
- Ở bộ giải mã: nhận đầu vào là véc-tơ có kích thước cố định lấy ở bộ mã hóa, sau đó uni-LSTM trong bộ giải mã thực hiện dự đoán những từ trong câu đích

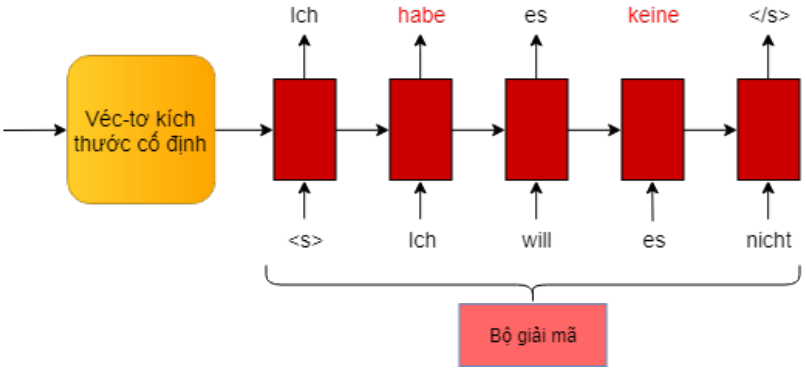
cho đến khi dự đoán ra kí tự kết thúc câu </s> hoặc vượt quá độ dài câu đích đã được qui định.

- Các từ đã được dự đoán ở bộ giải mã được so sánh với từ đúng cho trước trong bộ dữ liệu huấn luyện thông qua hàm lỗi. Dựa vào kết quả của hàm lỗi, mô hình thực hiện cập nhật các tham số của mô hình.

Sau khi mô hình đã được huấn luyện, để có thể phát sinh câu đích khi nhận một câu nguồn, mô hình thực quá trình suy diễn. Quá trình suy diễn này giống như quá trình huấn luyện, chỉ khác là không có bước tính độ lỗi và cập nhật các tham số. Mô hình nhận đầu vào là câu nguồn, sau đó thực hiện dự đoán các từ ở câu đích.

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 suy diễn (là quá trình mà sau khi mô hình đã huấn luyện xong được mang đi dịch). 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 câu dài với quá trình huấn luyện như trên thì phải tốn nhiều thời gian và công sức tinh chỉnh mô hình.

Vì hạn chế đó, chúng tôi đã tìm hiểu và 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 suy diễn thì



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.

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: $p(y_t|y_{<t}, \boldsymbol{x})$. 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 $p(\boldsymbol{y}|y_{<t}, \boldsymbol{x}\})$ là chọn ra từ nào có xác suất cao nhất:

$$\hat{y} = \underset{y_t \in V}{\operatorname{argmax}}(p(y_t|y_{<t}, \boldsymbol{x})) \tag{3.1}$$

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 hai "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 hai 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 hai từ "I am":

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

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 $p(going|I,am) > p(swimming|I,am)$ và dẫn tới câu kết quả thứ hai "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 $p(y_t|y_{<t}, \boldsymbol{x})$ lớn nhất và duy trì xét một *câu giả thuyết (hypothesis)* duy nhất trong quá trình giải mã. Khi bộ giải mã dự đoán một câu, sẽ có nhiều tổ hợp khác nhau của các từ tạo thành các câu đích, mỗi câu đích được tạo bởi quá trình dự đoán của bộ giải mã chúng tôi gọi đó là 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 $\hat{\boldsymbol{y}}$ được lựa chọn dựa trên biểu thức:

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

Tuy nhiên, với số lượng từ vựng lớn, ví dụ bộ từ vựng $V = 10000$ từ thì số lượng tổ hợp câu gải 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 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*).

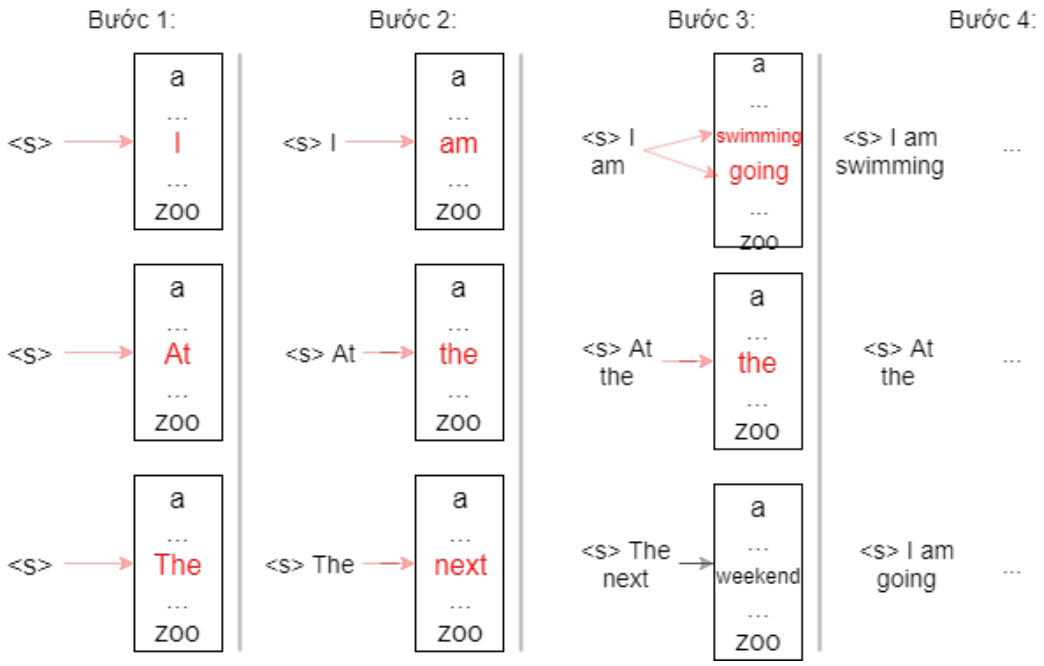
Beam search về cơ bản có cách hoạt động giống tìm kiếm tham lam. Khác biệt ở đây là khi dự đoán, thay vì chỉ dùng một câu giả thuyết có xác suất lớn nhất thì beam search chọn B câu giả thuyết có xác suất lớn nhất. B là một siêu tham số chỉ kích thước của beam (chùm). Do beam search dùng B câu giả thuyết, do đó để lựa chọn B câu giả thuyết có xác suất lớn nhất thì cần phải tính xác suất của từng câu. Cách tính xác suất cho một câu giống như mô hình ngôn ngữ:

$$p(\boldsymbol{y}) = p(y_1, y_2, ..., y_S|\boldsymbol{x}) = p(y_1|\boldsymbol{x})p(y_2|p_1, \boldsymbol{x})...p(y_S|y_1, y_2, ..., y_{S-1}, \boldsymbol{x}) \tag{3.2}$$

Để làm rõ hơn cách hoạt động của beam search, chúng tôi xét lại ví dụ dịch câu tiếng Đức "Ich schwimme am nächsten Wochenende" sang tiếng Anh. Trong ví dụ này chúng tôi lựa chọn $B = 3$. Bộ giải mã khởi tạo $B = 3$ câu giả thuyết rỗng (không có từ nào). Mô hình có bộ từ vựng $V = \{a, an, ..., zoo\}$. Tại bước thời gian đầu, bộ giải mã nhận đầu vào là kí tự bắt đầu câu <s>. Sau đó, đối với mỗi câu giả thuyết, bộ giải mã thực hiện dự đoán từ tiếp theo bằng cách tính xác suất có điều kiện dựa trên câu giả thuyết hiện tại, câu đầu vào và bộ từ vựng V: $p(y_1| < s >, \boldsymbol{x}),\ y_1 \in V$. Xét tất cả các xác suất có điều kiện $p(y_1| < s >, \boldsymbol{x})$ vừa tính được trên cả 3 câu giả thuyết, mô hình lựa chọn ra $B = 3$ câu có xác suất $p(y_1, < s > |\boldsymbol{x})$ cao nhất.

$$p(y_1, < s > |\boldsymbol{x}) = p(y_1| < s >, \boldsymbol{x}),\ y_1 \in V$$

Giả sử, các từ y_1 "I", "At", "The" làm xác suất $p(y_1, < s > |\boldsymbol{x})$ cao nhất và được mô hình lựa chọn làm 3 câu giả thuyết. Tiếp theo, mô hình tiếp tục dự đoán từ tiếp theo và tính các xác suất câu $p(y_2| < s >, "I")$, $p(y_2| < s >, "At")$, $p(y_2| < s >, "The")$. Tiếp tục giả định rằng mô hình có 3 câu có xác suất cao nhất là "I am", "At the", "The next" (để đơn giản, chúng tôi không ghi lại kí tự bắt đầu câu <s>) và mô hình chọn 3 câu này thành 3 câu giả thuyết. Mô hình lại tiếp tục dự đoán từ tiếp theo cho 3 câu



Hình 3.3: Minh họa thuật toán beam search.

giả thuyết hiện có và tính xác suất câu để tìm ra câu có xác suất cao nhất. Lần này xác suất 3 câu "I am swimming", "I am going", "The next weekend" là lớn nhất, tức là câu "The next" không còn được chọn là cách dịch phù hợp nữa và bị loại bỏ. Cứ lặp lại như vậy cho đến khi cả 3 câu đều đã dự đoán ra kí tự kết thúc câu </s> hoặc đã đạt tới giới hạn chiều dài do người dùng đặt thì dừng. Bây giờ mô hình có được 3 câu giả thuyết "I am swimming at the next weekend", "I am going to be swimming at the next weekend" và "At the next weekend, I am swimming". Cuối cùng, mô hình tính được câu giả thuyết "I am swimming at the next weekend" có xác suất cao nhất. Vậy mô hình chọn "I am swimming at the next weekend" làm kết quả cuối cùng. Hình 3.3 minh họa thuật toán beam search cho ví dụ vừa trình bày. Về chi phí tính toán, trong ví dụ này, giả sử $V = 10000$, thì mô hình phải tính $V \times B = 10000 \times 3 = 30000$ xác suất có điều kiện $p(y_t|y_{<t}, \boldsymbol{x})$ và phải tính 30000 xác suất câu $p(\boldsymbol{y}|\boldsymbol{x})$ cho mỗi trường hợp vừa dự đoán nữa. Tổng cộng là 60000. Tóm lại, chi phí tính toán của beam search so với tìm kiếm tham lam là gấp $2B$ lần.

Với ví dụ chúng tôi đã trình bày ở trên thể hiện rằng beam search giải quyết được hạn chế của thuật toán tìm kiếm tham lam và dịch ra được những câu tối ưu hơn. Tuy nhiên, vì beam search là thuật toán tìm kiếm gần đúng nên không thể đảm bảo rằng kết quả luôn là tối ưu như các thuật toán tìm kiếm chính xác Depth First Search, Bread

First Search. Ưu điểm lớn nhất của beam search chính là nó khả thi trong thực tế nhờ tốc độ rất nhanh mà vẫn đảm bảo được đầu ra tốt. Việc lựa chọn kích thước của beam B cũng rất quan trọng. Nếu chúng ta chọn $B = 1$, beam search trở thành thuật toán tìm kiếm tham lam. Với B càng lớn thì có xu hướng cho ra kết quả tốt hơn, nhưng bù lại tốc độ và chi phí không gian lưu trữ tăng cao. Ngược lại, với B nhỏ thì tốc độ nhanh, tiết kiệm chi phí không gian lưu trữ nhưng kết quả có xu hướng kém đi. Việc lựa chọn B như thế nào phù thuộc vào từng hoàn cảnh. Trong Dịch máy, thường mọi người chỉ chọn $B = 5, 10, 15$. Trong một số ứng dụng, thậm chí $B = 1000$ tới 3000. Cần lưu ý rằng không phải lúc nào kích thước beam B càng lớn thì chất lượng dịch sẽ càng tốt mà còn phụ thuộc vào chất lượng của mô hình nữa. Cho nên, để có thể có được những câu dịch trôi chảy, chất lượng thì còn cần phải thực hiện phân tích kĩ càng hơn để xem là vấn đề nằm ở beam search hay là nằm ở mô hình.

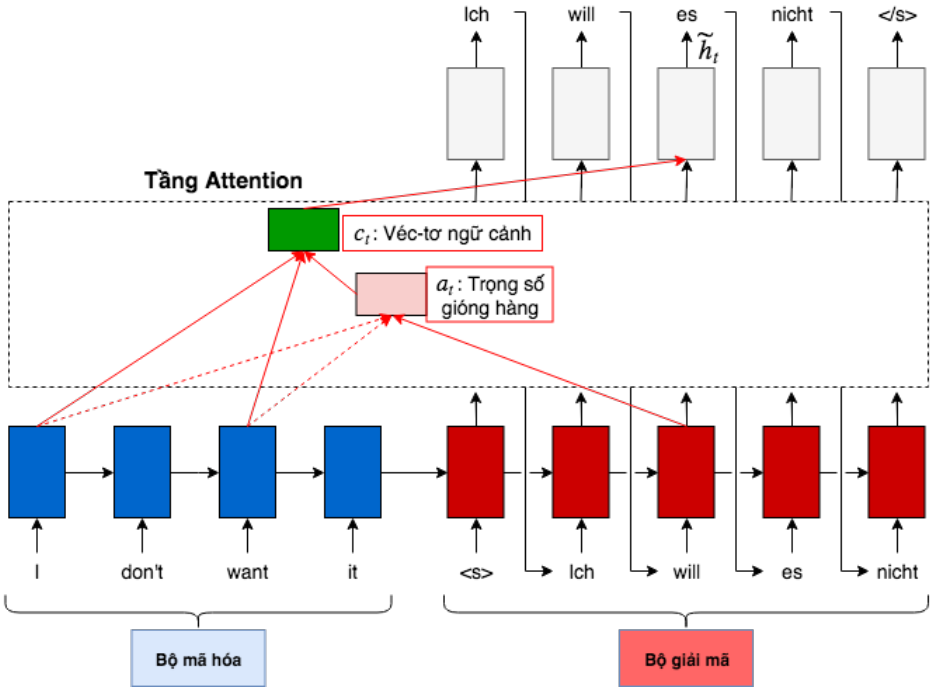
Trong bài toán Dịch máy, beam search thường chỉ được sử dụng trong quá trình suy diễn. Bởi vì trong quá trình huấn luyện, mô hình sử dụng phương pháp teacher forcing có hiệu quả lớn hơn về mặt chi phí tính toán, tốc độ hội tụ cũng như dễ cài đặt.

3.2 Mô hình học LSTM-Attention cho bài toán Dịch máy

3.2.1 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, cơ chế Attention được sử dụng.

Trước khi đi vào cách hoạt động của cơ chế Attention, chúng tôi điểm qua một chút về nguồn cảm hứng và lịch sử của cơ chế này. Cơ chế Attention được lấy cảm hứng trên cách đặt sự chú ý khi quan sát sự vật, hiện tượng của thị giác con người.



Hình 3.4: Minh họa cơ chế Attention. Một tầng Attention được đặt ở trước bước dự đoán đầu ra của bộ giải mã. Bộ giải mã có thêm một số bước tính toán trước khi dự đoán từ tiếp theo ở câu đích. Bộ giải mã tính trọng số giống hàng $\boldsymbol{a}_t$ cho một số trạng thái ẩn ở bộ mã hóa được lựa chọn. Sau đó tính véc-tơ ngữ cảnh $\boldsymbol{c}_t$ và nối véc-tơ này với trạng thái ẩn hiện tại của bộ giải mã thành véc-tơ attention $\tilde{\boldsymbol{h}}_t$. Cuối cùng mô hình dự đoán từ tiếp theo dựa trên véc-tơ attention vừa tính được.

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 thông tin. Sau đó, khi cần ghi nhận thêm thông tin khác, con người sẽ di chuyển vùng tập trung của mắt sang vị trí khác trên vật thể, hiện tượng. 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 [21] [7]. 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 LSTM 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 LSTM 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.

Hai phiên bản Toàn cục và Cục bộ mà trong khóa luận này chúng tôi trình bày là hai cách mà cơ chế Attention sử dụng các trạng thái ẩn của LSTM 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 công thức tính toán của nó. Attention sử dụng thêm một số đại lượng:

- $\boldsymbol{a_t}$: véc-tơ trọng số gióng hàng. Gồm có các phần tử $a_{ts},\ s \in \{1,...,S\}$, $\boldsymbol{a_t}$ được tính theo công thức dưới đây:

$$a_{ts} = \text{align}(\boldsymbol{h_t}, \boldsymbol{h_s}) = \frac{\exp\left(\text{score}(\boldsymbol{h_t}, \boldsymbol{h_s})\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h_t}, \boldsymbol{h_{s'}})\right)} \tag{3.3}$$

  a_{ts} là một số thực nằm trong đoạn $[0;1]$ thể hiện mức độ mà trạng thái ẩn ở bước thời gian t $\boldsymbol{h_t}$ tập trung vào một trạng thái ẩn ở câu nguồn $\boldsymbol{h_s}$. Hàm điểm số score mà chúng tôi sử dụng là gồm hai hàm:

$$\text{score}(\boldsymbol{h_t}, \boldsymbol{h_s}) = \begin{cases} \boldsymbol{h_t}^T \boldsymbol{h_s} & dot \\ \boldsymbol{h_t}^T \boldsymbol{W}_a \boldsymbol{h_s} & general \end{cases} \tag{3.4}$$

  Đối với hàm score là hàm *dot*, mô hình chỉ đơn giản là thực hiện tính tích vô hướng giữa hai trạng thái ẩn. Ưu điểm của hàm *dot* này là chi phí tính toán thấp nên thời gian huấn luyện và suy diễn nhanh. Đối với hàm score là hàm *general*, hàm này có sự tinh tế hơn hàm *dot*. Hàm *dot* chỉ thực hiện nhân hai phần tử tương ứng giữa hai véc-tơ với nhau, trong khi đó hàm *general* sử dụng thêm một bộ tham số $\boldsymbol{W}_a$, do đó những thông tin giữa hai trạng thái ẩn sẽ được tính một cách chọn lọc hơn và được biến đổi bởi một phép biến đổi tuyến tính $\boldsymbol{W}_a$. Tuy nhiên, đổi lại thì hàm này sẽ có thời gian thực thi chậm hơn hàm *dot* một chút do cần phải nhân với bộ tham số. 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.

- $\boldsymbol{c_t}$: véc-tơ ngữ cảnh tại bước thời gian t, là trung bình có trọng số của các trạng

thái ẩn ở câu nguồn:

$$\boldsymbol{c}_t = \sum_s a_{ts} \boldsymbol{h}_s \tag{3.5}$$

Véc-tơ $\boldsymbol{c}_t$ 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 $\boldsymbol{h}_s$ của câu nguồn mà được lựa chọn một cách có chọn lọc (có trọng số). Véc-tơ ngữ cảnh $\boldsymbol{c}_t$ 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 đích. Để dự đoán được chính xác, mô hình cần phải biết được ngữ cảnh hiện tại của câu như thế nào. 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.

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

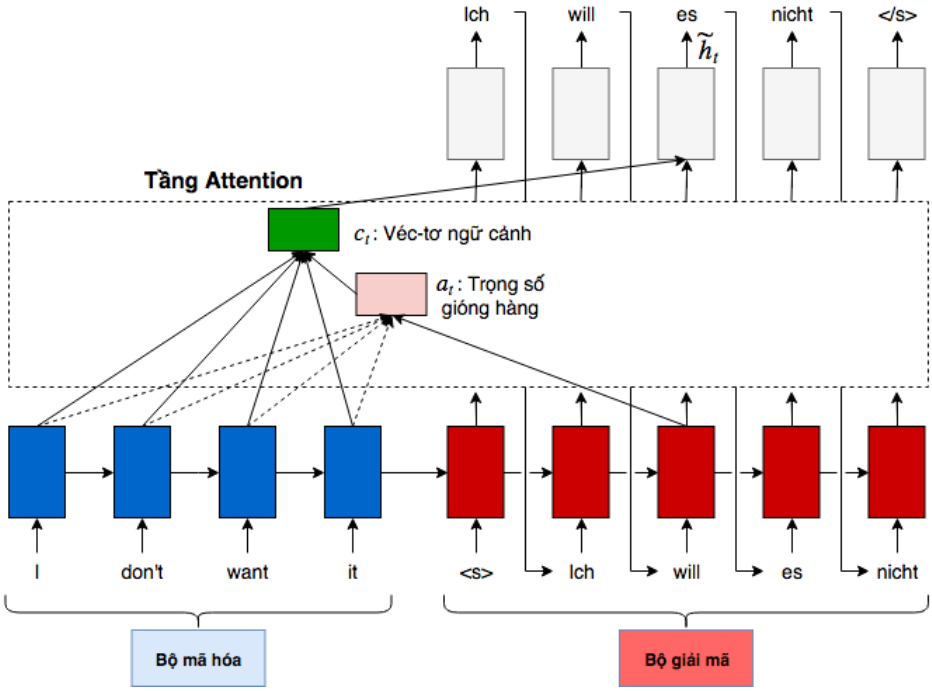
$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{3.6}$$

Véc-tơ attention chứa thông tin gióng 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 $\boldsymbol{h}_t$ được thay thế bởi véc-tơ attention $\tilde{\boldsymbol{h}}_t$. Véc-tơ này được đưa qua tầng softmax để cho ra phân bố xác suất dự đoán trên các từ:

$$p(y_t | y_{<t}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s \tilde{\boldsymbol{h}}) \tag{3.7}$$

Hình 3.4 minh họa cơ chế Attention. Trước khi bộ giải mã thực hiện dự đoán từ tiếp theo, mô hình tính toán thêm một số đại lượng $\boldsymbol{a}_t$, $\boldsymbol{c}_t$. Một cách tổng quát, mục tiêu cuối cùng của cơ chế Attention là xoay quanh việc tìm véc-tơ ngữ cảnh $\boldsymbol{c}_t$ một cách hiệu quả. Tiếp theo, chúng tôi trình bày chi tiết hơn về hai phiên bản Toàn cục và Cục bộ. Hai phiên bản này chỉ khác nhau về cách suy ra véc-tơ ngữ cảnh $\boldsymbol{c}_t$, còn các bước còn lại giống nhau. Quy trình tính toán của cơ chế Attention: $\boldsymbol{h}_t \rightarrow \boldsymbol{a}_t \rightarrow \boldsymbol{c}_t \rightarrow \tilde{\boldsymbol{h}}_t$



Hình 3.5: 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 gióng 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.

3.2.2 Attention Toàn cục

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

$$\boldsymbol{a}_t = \text{align}(\boldsymbol{h}_t, \boldsymbol{h}_s) = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \boldsymbol{h}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \boldsymbol{h}_{s'})\right)} \tag{3.8}$$

Hình 3.5 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 "nhìn" lên toàn bộ các trạng thái ẩn ở câu nguồn hay bộ mã hóa. Sau đó mô hình dựa vào trạng thái ẩn hiện tại ở bộ giải mã và toàn bộ các trạng thái ẩn ở bộ mã hóa để quyết định xem nên tập trung vào những trạng thái ẩn nào ở bộ mã hóa dựa trên véc-tơ trọng số gióng hàng $\boldsymbol{a}_t$.

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 "nhìn" lên toàn bộ câu nguồn, chỉ cần đặt "nhìn" lên một số từ cần thiết rồi sau đó tập trung lên những từ quan trọng. Việc giảm phạm vi "nhìn" trước khi tập trung lên những từ quan trọng giúp giảm chi phí tính toán $\boldsymbol{a}_t$ 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.

3.2.3 Attention Cục bộ

Như đã nêu ở phần trước, Attention Toàn cục có một hạn chế là "nhìn" lên toàn bộ các từ ở câu nguồn khi dự đoán các từ ở câu đích. Điều này gây tiêu tốn chi phí tính toán và gây cản trở khi dịch những câu dài như trong các đoạn văn hay trong một tài liệu. Attention Cục bộ là phiên bản cải thiện của Attention Toàn cục để giải quyết hạn chế này. Lưu ý, ý tưởng của cơ chế Attention Cục bộ này dựa vào đặc điểm của hai 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 dự đoán mỗi từ ở câu đích, Attention Cục bộ chỉ "nhìn" lên một số từ gần nhau ở câu nguồn. Mô hình này lấy cảm hứng từ sự đánh đổi giữa hai mô hình "soft attention" và "hard attention" được đề xuất trong công trình Show, Attend and Tell [34] để giải quyết bài toán Phát sinh câu miêu tả cho ảnh (Image Captioning). Trong công trình [34], Attention Toàn cục tương ứng với "soft attention", mô hình "nhìn" vào toàn bộ bức ảnh. Còn "hard attention" thì mô hình "nhìn" vào một số phần của bức ảnh.

Dễ thấy, với cách hoạt động chỉ "nhìn" một số các từ gần nhau ở câu nguồn, mô hình hoạt động gần với cách con người "nhìn" 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 gióng hàng $\boldsymbol{a}_t$ cho những từ mà mô hình "nhìn" vào.

Để làm rõ hơn về cách thức hoạt động của mô hình Attention Cục bộ, chúng tôi sẽ trình bày cụ thể hơn về công thức tính toán 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 gióng 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 mà từ hiện tại thứ t ở câu đích được gióng hàng với từ thứ p_t ở câu nguồn. 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 cửa 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 $\boldsymbol{c}_t$, Attention Toàn cục có sự thay đổi. 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. Cửa sổ này xác định vùng nào trong câu nguồn sẽ được mô hình "nhìn" vào. Cụ thể:

$$\boldsymbol{c}_t = \sum_{x \in [p_t - D, p_t + D]} a_{tx} h_x \tag{3.9}$$

  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 bài báo [22], chúng tôi lựa chọn $D = 10$.

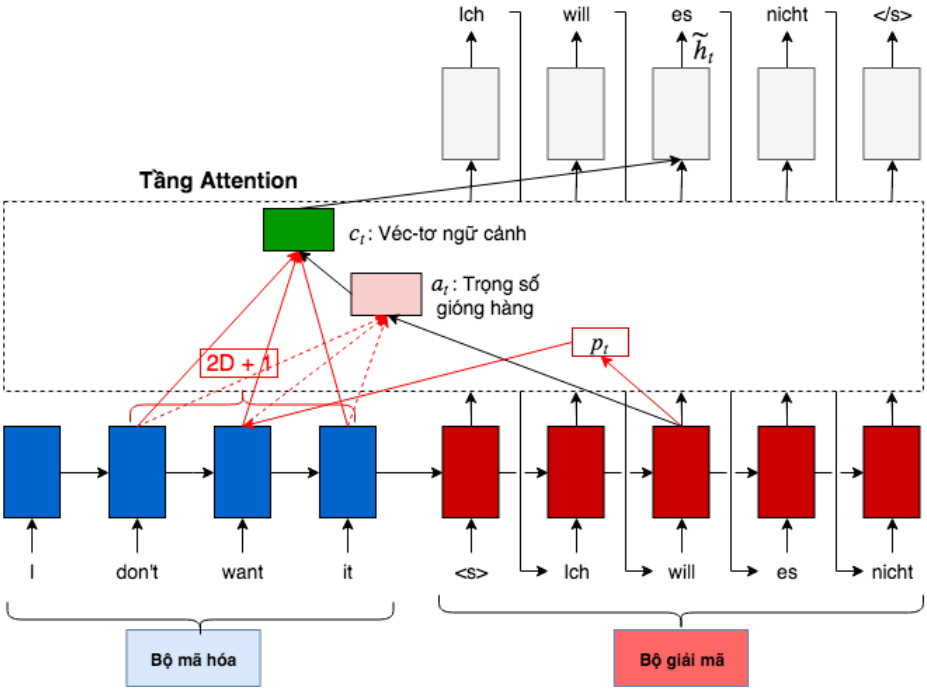
Hình 3.6 minh họa cơ chế Attention Cục bộ cho thấy cách hoạt động của Attention Cục bộ cùng với sự khác biệt giữa Attention Cục bộ và Attention Toàn cục.

Mô hình Attention Cục bộ có hai biến thể:

- Gióng hàng đều (monotonic alignment - local-m): vị trí được gióng 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 gióng hàng đều nhau theo từng từ.

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

$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^T \tanh(\boldsymbol{W}_p \boldsymbol{h}_t)) \tag{3.10}$$



Hình 3.6: 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 gióng hàng và véc-tơ ngữ cảnh được tính dựa trên những trạng thái ẩn được bộ giải mã đặt sự chú ý.

Trong đó, $\boldsymbol{v}_p$ và $\boldsymbol{W}_p$ là hai tham số mới của mô hình dùng cho việc dự đoán vị trí p_t. Mô hình cần học hai tham số này để có thể dự đoán vị trí p_t được chính xác. Miền giá trị của $p_t \in [0, S]$. Lúc này, vị trí p_t dựa vào trạng thái ẩn hiên tại của bộ giải mã. Để "ưu tiên" các vị trí gióng hàng p_t, mô hình thêm vào các trọng số gióng hàng $\boldsymbol{a}_t$ 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 (không chuẩn hóa) với giá trị kì vọng p_t và độ lệch chuẩn $\sigma = \frac{D}{2}$:

$$\boldsymbol{a}_t = \text{align}(\boldsymbol{h}_t, \boldsymbol{h}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right) \tag{3.11}$$

Mô hình sử dụng hàm gióng 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 có kích thước D.

Một điều cần lưu ý là nếu cửa sổ vượt ra ngoài câu nguồn (p_t - D < 0 và p_t + D > S) thì mô hình sẽ bỏ qua phần nằm vượt ra ngoài đó. Chỉ xét phần nằm trong câu.

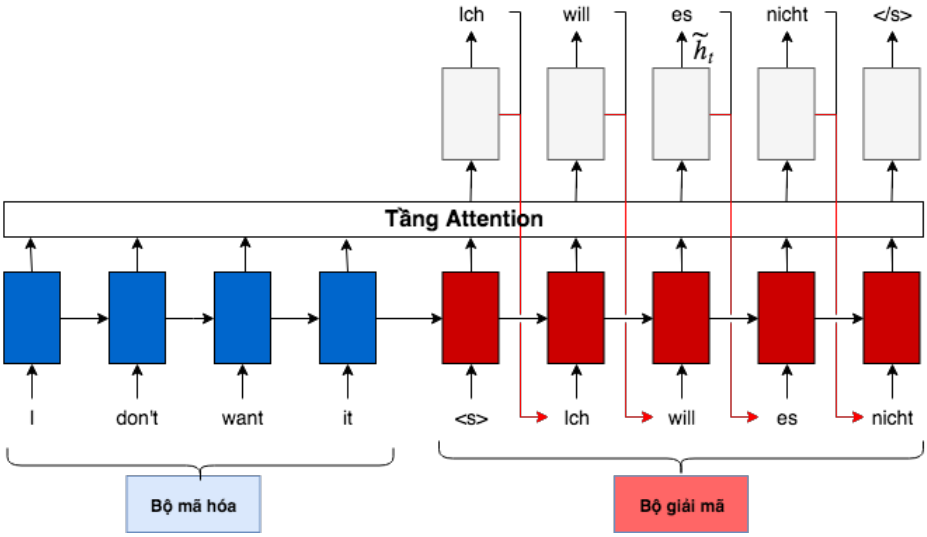
Véc-tơ trọng số gióng hàng $\boldsymbol{a}_t$ ở Attention Cục bộ có kích thước cố định $\in \mathbb{R}^{2D+1}$

và thường ngắn hơn $\boldsymbol{a}_t$ 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 dựa vào trạng thái ẩn hiện tại của bộ giải mã và sử dụng một phân phối chuẩn đã được đơn giản hóa để điều chỉnh các trọng số gióng hàng $\boldsymbol{a}_t$. Việc sử dụng thêm phân phối chuẩn để khuyến khích mô hình đặt tập trung 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 tập trung 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 gần 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 "nhìn" 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.

3.2.4 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" (tập trung) hay gióng hàng lên các vị trí nguồn. Ở mỗi bước thời gian t khi dự đoán 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 gióng hàng như thế nào (giá trị của véc-tơ $\boldsymbol{a}_t$) hoàn toàn phụ thuộc vào điểm số (giá trị của hàm score) giữa trạng thái ẩn $\boldsymbol{h}_t$ hiện tại và các trạng thái ẩn $\boldsymbol{h}_s$ 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 hai 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 [20] đã đề xuất một mô hình dịch



Hình 3.7: 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.

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 nào ở câu nguồn 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 (câu dài hơn cần thiết) hay ngược lại dẫn đến trường hợp "được dịch quá ít" (câu ngắn hơn cần thiết) khi kí hiệu kết thúc câu xuất hiện sớm. Ngoài ra còn bị ảnh hưởng bởi số lượng từ đích được quy định khi dịch.

Công trình [22] đề xuất một cơ chế góp phần giải quyết vấn đề ở trên: *input feeding* (tạm dịch là "*tăng cường thông tin cho đầu vào*"). Ý tưởng và cách thực hiện của input feeding rất đơn giản. Nhận thấy véc-tơ attention $\tilde{\boldsymbol{h}}_{t-1}$ lưu giữ thông tin gióng hàng của bước thời gian $t-1$ trước đó, mô hình thực hiện nối véc-tơ $\tilde{\boldsymbol{h}}_{t-1}$ với đầu vào x_t của bước thời gian t hiện tại. Bằng cách như vậy, mô hình có thể nắm được thông tin gióng hàng trước đó từ $\tilde{\boldsymbol{h}}_{t-1}$.

$$x_t^{'} = [x_t, \tilde{\boldsymbol{h}}_t] \tag{3.12}$$

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 gióng hàng 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 gióng hàng 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 có thể làm tăng khả năng học của mô hình.

3.2.5 Kĩ 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 biển 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 kĩ 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.

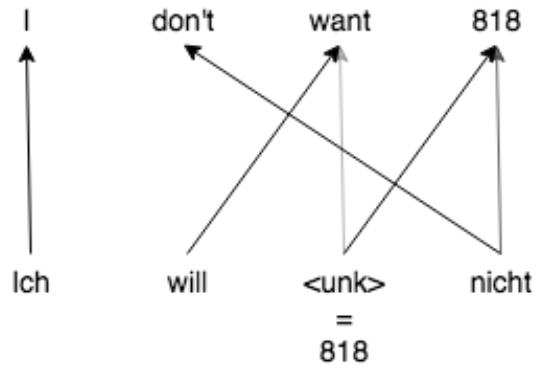
Với tốc độ phát triển của ngôn ngữ là như vậy nhưng khả năng của con người là hữu hạn. Một người dù có thông thạo một ngôn ngữ tới đâu thì cũng không thể nào biết được hết tất cả từ vựng của ngôn ngữ đó. Theo thống kê, số lượng từ ngữ cần để giao tiếp hàng ngày trong tiếng Anh chỉ khoảng từ 3000 từ và 3000 từ này chiếm đến 95% các đoạn văn bản, cuộc nói chuyện phổ biển hàng ngày (tin tức, sách, phim,

v.v...), đối với lĩnh vực chuyên ngành thì khoảng 4000-10000 từ. Đối với những người bản xứ thì họ biết khoảng 10000-30000 từ [8]. Còn đối những người không phải bản xứ, trung bình bộ từ vựng của họ khoảng 4500 từ. Nhưng theo kích thước của bộ từ điển lớn nhất trong tiếng Anh là Oxford thì tổng số lượng từ vựng của tiếng Anh là 171476 từ [17]. 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 hai 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 hai 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 hai 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 tìm hiểu và dùng cách thứ hai để giải quyết vấn đề các từ nằm ngoái 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 [16] 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



Hình 3.8: 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 gióng 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 gióng hàng càng cao). Kết quả dự đoán được cập nhật với từ hiếm đã được thay thế.

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 <*unk*>). 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 gióng 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 trong tập dữ liệu so với những từ khác. Hơn nữa, những loại từ như thế này rất đa dạng (số lượng số từ 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 kĩ 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 và 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 kĩ 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 perlexity, 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 [12]. 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 [22].

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 50000 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 $< unk >$ (từ hiếm).

Trong quá trình huấn luyện, chúng tôi lọc bỏ những cặp câu mà một trong hai 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-batch 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 [29]. 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 lí 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 đành giá là BLEU (BiLingual Evaluation Understudy) [27] 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 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 *general* và có sử dụng phương pháp Input feeding, kĩ thuật thay thế từ hiếm.

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 22.87 (+7.83). Đây là cải thiện rất 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 trong việc áp dụng vào cho bài toán Dịch máy.

Mô hình	Perplexity	BLEU
Cơ bản	8.03	15.04
Cơ bản + global (general) + input feed + unk rpl	**5.22 (-2.81)**	**22.87 (+7.83)**

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 unk rpl	Sau unk rpl
Cơ bản	8.03	15.08	
Cơ bản + global (dot)	6.1	18.62	18.81 (+0.19)
Cơ bản + global (general)	4.98	19.99	22.33 (+2.24)
Cơ bản + global (dot) + input feed	5.43	19.78	22.24 (+2.46)
Cơ bản + global (general) + input feed	5.22	**20.41**	**22.87 (+2.46)**
Cơ bản + local-p (general) + input feed	5.43	20.37	22.75 (+2.38)

Bảng 4.2: So sánh kết quả giữa các mô hình Attention với nhau.

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 Toàn cục với general product mà có sử dụng phương pháp Input feeding (global general + input feed)) có chất lượng dịch tốt nhất với điểm BLEU là 20.41 (22.87 khi dùng kĩ thuật thay thế từ hiếm). Mô hình Attention Toàn cục với hàm tính điểm *dot, general* cộng với phương pháp Input feeding (global (dot) + input feed) có độ tăng điểm BLEU cao nhất khi dùng kĩ thuật thay thế từ hiếm. Kết quả này chứng minh được rằng mô hình global (general) có ý tưởng phù hợp cho cặp ngôn ngữ Anh - Đức và hoạt động hiệu quả như mong đợi trong thực tế. Mô hình có độ đo Perplexity thấp nhất là mô hình global (general) với 4.98. Mặc dù Perplexity hơn mô hình (global (general) + input feed) là 5.43, nhưng (global (general) + input feed) có điểm BLEU tốt nhất là 20.41, hơn 0.42 điểm. Điều này cho thấy là mô hình global (general) có khả năng tổng quát hóa chưa tốt so với các mô hình khác.

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 giúp mô hình global (dot) tăng điểm BLEU từ 18.62 lên 19.78 (+1.16), giúp mô hình global (general) tăng từ 19.99 lên 20.41 (+0.42). Kĩ thuật thay thế từ hiếm tăng trung bình hơn 2.0 BLEU (trừ mô hình global (dot) chỉ tăng 0.19). 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, kĩ 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 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.

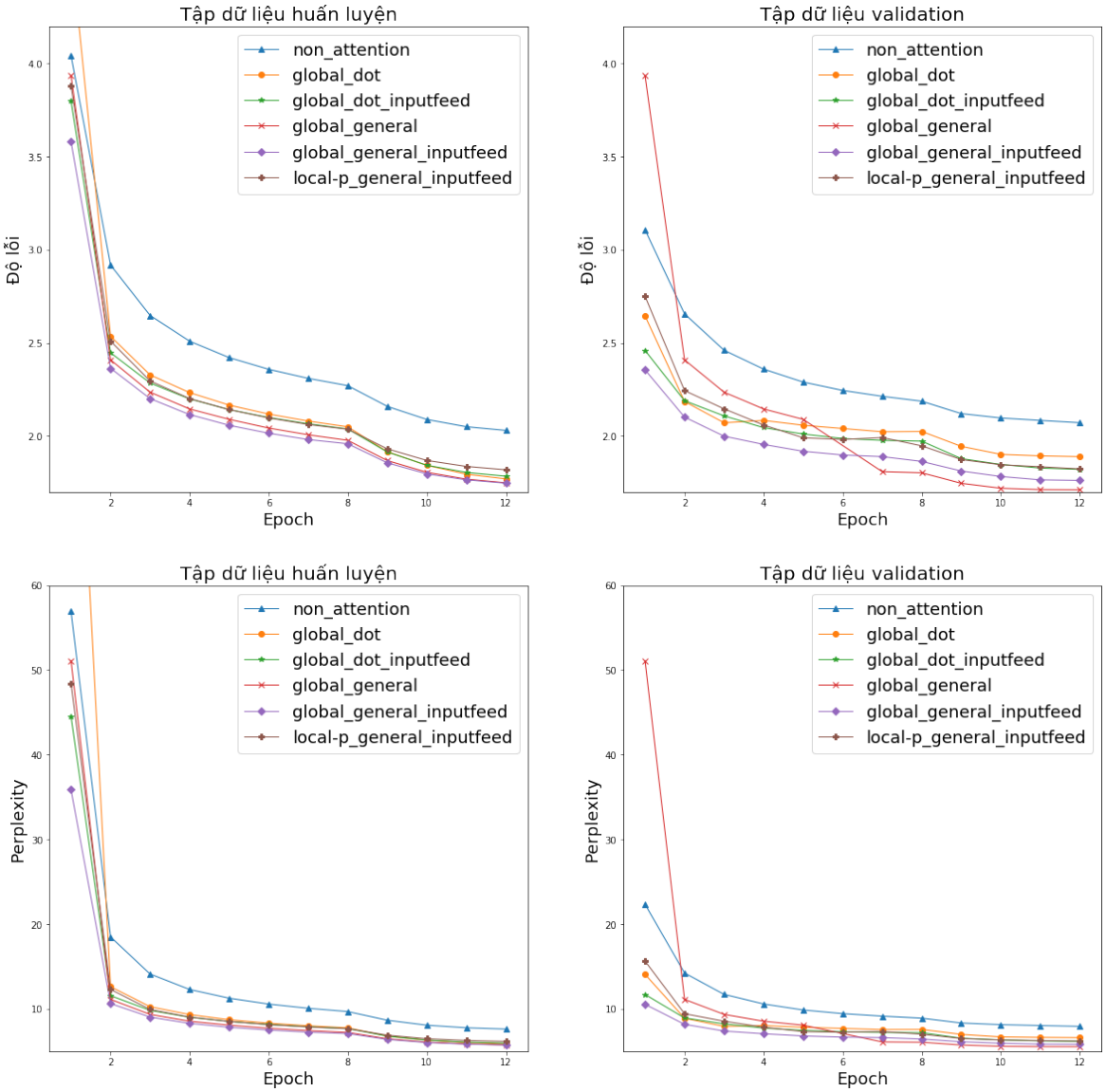
Về kĩ thuật thay thế từ hiếm, từ kết quả cho thấy kĩ thuật này rất mạnh mẽ. Các mô hình Attention được sử dụng cùng với kĩ thuật này đều được tăng điểm BLEU hơn 2. Mô hình này chỉ có hạn chế là phải phụ thuộc vào kết quả của cơ chế Attention có tốt hay không. Kết quả cũng cho thấy rằng chất lượng mô hình Attention không tốt sẽ dẫn tới độ hiệu quả của kĩ thuật thay thế từ hiếm kém đi (mô hình global (dot)). Còn lại thì 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.

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 (general) cho tốc độ hôi tụ nhanh nhất (độ lỗi trên tập huấn luyện giảm nhanh nhất qua các epoch). Trong các epoch



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.

Mô hình	Perplexity		BLEU	
	KL	Bài báo	KL	Bài báo
Cơ bản	8.03	8.1	15.08	14.0
Cơ bản + global (dot) + input feed	5.43	6.1	19.78	18.6
Cơ bản + global (dot) + input feed + unk rpl			22.24	20.5
Cơ bản + global (general) + input feed	5.22	6.1	**20.41**	17.3
Cơ bản + global (general) + input feed + unk rpl			**22.87**	19.1
Cơ bản + local-p (general) + input feed	5.43	5.9	20.37	**19.0**
Cơ bản + local-p (general) + input feed + unk rpl			22.75	**20.9**

Bảng 4.3: So sánh kết quả của khóa luận và bài báo.

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. Mô hình local-p general này ở các epoch cuối có tốc độ giảm perplexity chậm đi rõ rệt. Mô hình global (general) là mô hình tốt nhất trong tất cả mô hình khi có perplexity và độ lỗi luôn thấp nhất (trừ trường hợp độ lỗi trên tập validation). Đối với mô hình không sử dụng Attention (non-attention) thì bị bỏ xa bởi các mô hình sử dụng Attention.

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* [22] để đánh giá rằng mô hình của chúng tôi hoạt động có thực sự đúng không.

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 chúng tôi cho rằng đó 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 phần lõi của mô hình được cài đặt lại khác nhau.

4.2.5 Chất lượng gióng 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 gióng hàng (điểm attention) $\boldsymbol{a}_t$ tương ứng. Sử dụng những trọng số gióng 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 hai 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 RNN. Trong quá trình dự đoán, bộ giải mã sử dụng các trạng thái ẩn của bộ mã hóa bằng cách "nhìn" vào các 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 hay nhìn sự vật, hiện tượng. Mọi người thường chỉ nhìn rồi 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, kĩ thuật khác:

  – Phương pháp Input feeding: giúp mô hình có thể biết được thông tin gióng hàng trong những thời điểm trước đó thông qua véc-tơ attention $\tilde{\boldsymbol{h}}_t$. Từ đó

giúp mô hình có thể hạn chế vấn đề "đươc 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.

- Kĩ 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, mô hình sẽ khó khăn nếu gặp những từ hiếm đó, vì vậy chất lượng dịch của mô hình bị giảm đáng kể. Đặc biệt là khi mô hình hay gặp những câu có chứa các từ hiếm như 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 mô hình không sử dụng cơ chế Attention.
- Những phương pháp, kĩ 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.

Trong khóa luận, chúng tôi đạt được:

- Tìm hiểu và cài đặt được mô hình Dịch máy với các LSTM.
- Tìm hiểu và cài đặt được các mô hình sử dụng cơ chế Attention và các phiên bản Toàn cục, Cục bộ của nó.
- Tìm hiểu và cài đặt được phương pháp Input feeding và kĩ thuật thay thế từ hiếm nhằm nâng cao chất lượng dịch của mô hình dựa trên kết quả của cơ chế Attention.
- Có được sự hiểu biết sâu hơn về những thuận lợi, khó khăn của bài toán Dịch máy và những ưu, khuyết điểm của các mô hình Dịch máy hiện nay.

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 hai 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 gióng hàng và véc-tơ ngữ cảnh chính xác và hữu ích hơn.

TÀI LIỆU THAM KHẢO

[1] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994. 30

[2] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," *CoRR*, vol. abs/1212.0901, 2012. 32

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. 15, 16

[4] ——, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. 21

[5] K. Cho, B. van Merrienboer, 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. 7

[6] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 160–167. 21

[7] 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 47

[8] J. E. K. Edward B. Fry. (2006) The reading teacher's book of lists, 5th edition. 57

[9] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179 – 211, 1990. 22

[10] 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. 34, 35

[11] J. Goodman, "A bit of progress in language modeling," *CoRR*, vol. cs.CL/0108005, 2001. 15

[12] T. S. N. L. P. Group. (2015) Neural machine translation. [Online]. Available: https://nlp.stanford.edu/projects/nmt/ 59

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. 30, 32, 33

[14] W. J. Hutchins, "Machine translation: A concise history," 2007. 2, 3

[15] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Networks*, vol. 20, no. 3, pp. 335 – 352, 2007, echo State Networks and Liquid State Machines. 32

[16] 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 57

[17] Johnson. (2013) Vocabulary size lexical facts. 57

[18] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech*

Recognition, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000. 15

[19] 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. 7

[20] 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 54

[21] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 1243–1251. [Online]. Available: http://papers.nips.cc/paper/4089-learning-to-combine-foveal-glimpses-with-a-third-order-boltzmann-machine.pdf 47

[22] 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, 37, 52, 55, 59, 65

[23] 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. 22

[24] J. Martens and I. Sutskever, "Learning recurrent neural networks with hessian-free optimization," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. USA: Omnipress, 2011, pp. 1033–1040. 32

[25] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur, "Recurrent neural network based language model." in *INTERSPEECH*, T. Kobayashi, K. Hirose, and S. Nakamura, Eds. ISCA, 2010, pp. 1045–1048. 16

[26] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. USA: Curran Associates Inc., 2013, pp. 3111–3119. 21

[27] 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 60

[28] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *CoRR*, vol. abs/1211.5063, 2012. 9, 30, 32

[29] 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. 60

[30] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. 21

[31] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Neurocomputing: Foundations of research," J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, ch. Learning Representations by Back-propagating Errors, pp. 696–699. 19

[32] 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, 38

[33] 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. 25

[34] 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 51