TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PHÂN TÍCH XÁC SUẤT VÀ GIẢI THUẬT NGẪU NHIÊN**

**TIẾP CẬN DỊCH MÁY BẰNG**

**PHƯƠNG PHÁP ENCODER-DECODER**

*Người hướng dẫn*: **Thầy Nguyễn Chí Thiện**

*Người thực hiện*: **Nguyễn Duy Hàn Lâm (MSHV: 196005004)**

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**PHÂN TÍCH XÁC SUẤT VÀ GIẢI THUẬT NGẪU NHIÊN**

**TIẾP CẬN DỊCH MÁY BẰNG**

**PHƯƠNG PHÁP ENCODER-DECODER**

*Người hướng dẫn*: **Thầy Nguyễn Chí Thiện**

*Người thực hiện*: **Nguyễn Duy Hàn Lâm (MSHV: 196005004)**

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

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

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

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

*TP. Hồ Chí Minh, ngày 05 tháng 04 năm 2020*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Duy Hàn Lâm*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

MỤC LỤC

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN ii](#_Toc35716930)

[MỤC LỤC 1](#_Toc35716931)

[CHƯƠNG 1: GIỚI THIỆU VỀ MẠNG NEURAL 4](#_Toc35716932)

[1.1 Perceptron 5](#_Toc35716933)

[1.2 Sigmoid neuron 8](#_Toc35716934)

[1.3 Cho mô hình học bằng kĩ thuật gradient descent 10](#_Toc35716935)

[1.3.1 Training dataset, testing dataset 10](#_Toc35716936)

[1.3.2 Gradient descent 11](#_Toc35716937)

[1.3.3 Áp dụng gradient descent vào mạng neural 13](#_Toc35716938)

[1.3.4 Cải tiến gradient descent 14](#_Toc35716939)

[1.4 Tóm tắt chương 1 15](#_Toc35716940)

[CHƯƠNG 2: GIẢI THUẬT BACKPROPAGATION 16](#_Toc35716941)

[2.1 Matrix-based algorithm 16](#_Toc35716942)

[2.2 Phép nhân Hadamard 17](#_Toc35716943)

[2.3 Bốn yếu tố cơ bản đằng sau giải thuật backpropagation 17](#_Toc35716944)

[2.3.1 Yếu tố thứ 1: Biểu thức tính mức độ sai số ở output layer (lớp cuối cùng) 18](#_Toc35716945)

[2.3.2 Yếu tố thứ 2: Biểu thức tính mức độ sai số ở các lớp ẩn 19](#_Toc35716946)

[2.3.3 Yếu tố thứ 3: Biểu thức tính mức độ sai số của phần bù đại số 19](#_Toc35716947)

[2.3.4 Yếu tố thứ 4: Biểu thức tính mức độ sai số của các trọng số 20](#_Toc35716948)

[2.4 Một số phép tính thay thế 20](#_Toc35716949)

[2.5 Cách chứng minh các biểu thức (27), (30), (31), (32) 21](#_Toc35716950)

[2.5.1 Chứng minh biểu thức (27) 21](#_Toc35716951)

[2.5.2 Chứng minh biểu thức (30) 21](#_Toc35716952)

[2.5.3 Chứng minh biểu thức (31) 22](#_Toc35716953)

[2.5.4 Chứng minh biểu thức (32) 23](#_Toc35716954)

[2.6 Giải thuật backpropagation 25](#_Toc35716955)

[2.7 Một góc nhìn khác 26](#_Toc35716956)

[2.8 Tổng kết chương 2 26](#_Toc35716957)

[CHƯƠNG 3: THỰC HÀNH LẬP TRÌNH 27](#_Toc35716958)

[3.1 Khởi tạo mạng neural 27](#_Toc35716959)

[3.2 Stochastic gradient descent 28](#_Toc35716960)

[3.3 Chạy mạng neural 32](#_Toc35716961)

[3.4 Tổng kết 34](#_Toc35716962)

[TÀI LIỆU THAM KHẢO 35](#_Toc35716963)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1. Cấu trúc cơ bản của 1 neural [1] 4](#_Toc35716964)

[Hình 1.2. Cấu trúc chi tiết 1 tầng của perceptron [2] 5](#_Toc35716965)

[Hình 1.3. Cấu trúc thu gọn của perceptron 1 tầng [3] 6](#_Toc35716966)

[Hình 1.4. Cấu trúc thu gọn của perceptron nhiều tầng [3] 7](#_Toc35716967)

[Hình 1.5. Minh họa về việc thay đổi [3] 8](#_Toc35716968)

[Hình 1.6. Đồ thị của hàm sigmoid [3] 9](#_Toc35716969)

[Hình 3.1. Hàm khởi tạo cho đối tượng mạng neural 27](#_Toc35716970)

[Hình 3.2. Khởi tạo đối tượng network 27](#_Toc35716971)

[Hình 3.3. Hàm tính sigmoid và đạo hàm của sigmoid kèm theo 28](#_Toc35716972)

[Hình 3.4. Hàm thực thi chiều thuận của mạng neural 28](#_Toc35716973)

[Hình 3.5. Hàm thực hiện huấn luyện cho mạng neural 29](#_Toc35716974)

[Hình 3.6. Hàm update các giá trị trọng số, bias 29](#_Toc35716975)

[Hình 3.7. Hàm giải thuật backpropagation 30](#_Toc35716976)

[Hình 3.8. Hàm tính đạo hàm của loss function 31](#_Toc35716977)

[Hình 3.9. Hàm tính giá trị mất mát 31](#_Toc35716978)

[Hình 3.10. Đồ thị minh họa cho tập dữ liệu đầu vào 32](#_Toc35716979)

[Hình 3.11. Khởi tạo đối tượng mạng neural 32](#_Toc35716980)

[Hình 3.12. Chuẩn hóa và phân tập tập dữ liệu thành tập train và tập test 33](#_Toc35716981)

[Hình 3.13. Chạy và huấn luyện mạng neural 33](#_Toc35716982)

[Hình 3.14. Kết quả RMSE sau khi chạy chương trình 33](#_Toc35716983)

[Hình 3.15. Đồ thị biểu diễn xu hướng dự đoán 34](#_Toc35716984)

CHƯƠNG 1: DỊCH MÁY NEURAL

Dịch máy là công cụ giúp chúng ta thực hiện việc dịch một cách tự động từ một ngữ này sang một ngôn ngữ khác.

Dịch máy dựa trên mạng neural (dịch máy neural) là một mô hình dịch máy mà trong đó có sử dụng các hệ thống mạng neural. Các mô hình dịch máy neural có cấu trúc gồm 2 phần chính: phần thứ nhất là bộ mã hóa (encoder), phần thứ hai là bộ giải mã (decoder). Kiến trúc này ta gọi là encoder-decoder hay seq2seq.

Bộ mã hóa (encoder) là một mô hình mạng neural mà trong đó các neural của nó được tính toán hay thực hiện dựa trên một trong các giải thuật deep learning (RNN, LSTM,…) dùng để biểu diễn tập dữ liệu đầu vào (tập dữ liệu ngôn ngữ gốc) thành 1 tập dữ liệu số đặc biệt. Trong tài liệu này các neural của encoder sẽ thực hiện trên Recurrent Neural Network (RNN).

Bộ giải mã (decoder) tương tự cũng là một một mô hình mạng neural và chức năng của nó từ chuyển hóa từ tập dữ liệu số đặc biệt của encoder thành tập dữ liệu đầu ra mà ta mong muốn (tập dữ liệu ngôn ngữ mong muốn).

Ví dụ khi ta muốn dịch từ tiếng Anh sang tiếng Việt, encoder sẽ chuyển tập dữ liệu tiếng Anh thành một dạng tập dữ liệu số đặc biệt và decoder sẽ có nhiệm vụ chuyển tập dữ liệu số đặc biệt này thành tập dữ liệu tiếng Việt tương ứng ở đầu ra.

Tài liệu này sẽ tập trung vào việc phân tích mô hình dịch máy neural bằng cách sử dụng hai phương pháp. Phương pháp thứ nhất là RNN sẽ được sử dụng cho encoder và decoder. Và phương pháp thứ hai là Convolutional Neural Network (CNN) sẽ được dụng cho encoder.

* 1. Giới thiệu về dịch máy neural

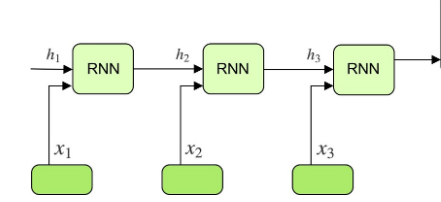
Ý tưởng dịch máy dựa trên hệ thống mạng neural được xuất phát từ ý tưởng ban đầu của Kalchbrenner và Blunsom vào năm 2013. Được lấy cảm hứng bằng việc áp dụng các kĩ thuật deep learning vào trong dịch máy. Tất cả các giải thuật đều được giới thiệu trong các tài liệu của Kalchbrenner, Blunsom vào năm 2013, của Sutskever vào năm 2014,… Trong tất cả các tài liệu này đều biểu diễn mô hình dịch thành 2 phần cơ bản là encoder và decoder.

Trong đó, encoder có chức năng chuyển hóa các tập dữ liệu ngôn ngữ gốc riêng biệt thành 1 tập dữ liệu số chung đặc biệt. Từ đó decoder có nhiệm vụ giải mã tập dữ liệu số đặc biệt này thành các tập dữ liệu ngôn ngữ muốn dịch tương ứng.

Cụ thể tài liệu này sẽ trình bày cách thực hiện làm sao ta có thể dịch từ tiếng anh sang tiếng việt.

* 1. Recurrent Neural Network trong mô hình

Ta có tập dữ liệu đầu vào:



1. Sơ lược cấu tạo hoạt động của RNN [1]

Tại mỗi bước (lớp) t, neural sẽ tính toán ra được một lượng giá trị gọi là ht. Giá trị ht này sẽ được đem đi tính chung với dữ liệu xt+1 tại bước t + 1. Giá trị ht tại mỗi bước t được tính như sau:

Trong đó:

* là kết quả của neural tại bước (lớp) t.
* là hàm activation.
* dữ liệu từ tập dữ liệu gốc

RNN là một kĩ thuật huấn luyện hiệu quả để tìm phân phối xác suất cho toàn tập bằng cách tính phân phối xác suất cho lớp tiếp theo . Trong trường hợp tập dữ liệu đầu vào là một chuỗi các vector từ 1 đến K thì phân phối xác suất của toàn tập ở đầu ra có thể được tính như sau:

Trong đó:

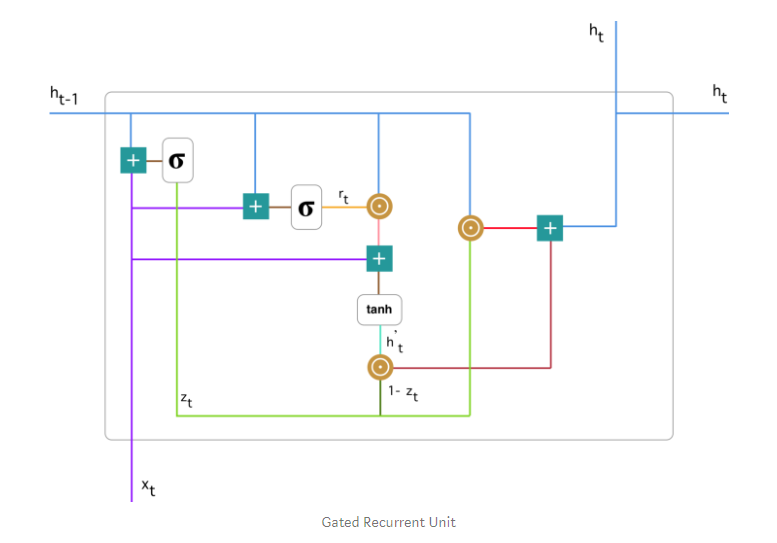
* là vector dữ liệu của neural thứ j ở bước thứ t.
* là trọng số tương ứng của neural thứ j.

Từ đó ta tính được phân bố xác suất cho toàn tập:

Trên hình 1.1 ta có thể thấy ở mỗi neural đều có 2 nguồn dữ liệu đầu vào (input). Input thứ nhất đó là dữ liệu gốc của tập dữ liệu và input thứ 2 là kết quả h tính được ở lớp phía trước (t – 1).

Dựa trên ý tưởng của LSTM, có một phương pháp tính mới cho mỗi neural được gọi là Gated Recurrent Unit (GRU). Tài liệu này sẽ trình bày dựa trên phương pháp tính toán này.

Trong GRU, ngoài 2 input đầu vào, ta sẽ có thêm 2 thành phần mới đó là cổng update z (update gate z) và cổng reset (reset gate r).



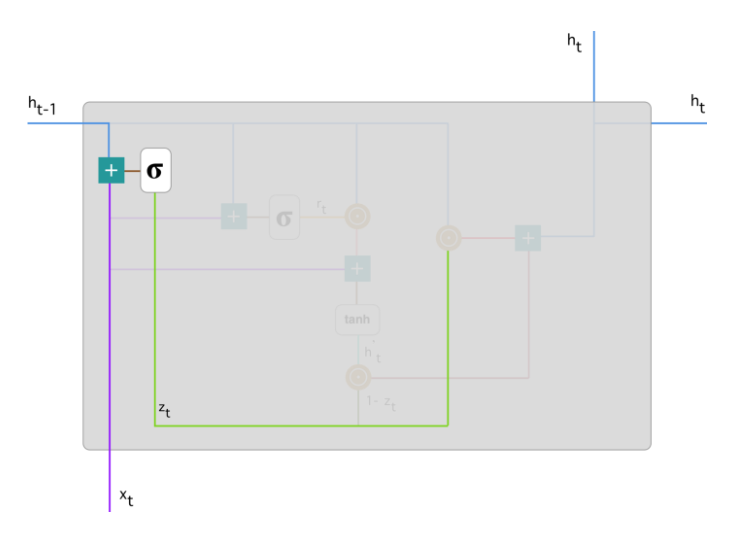
1. Cấu trúc neural theo GRU [2]
   * 1. Cổng update

Khi mới vào neural sẽ thực hiện việc tính toán các cổng trước.

Đầu tiên ta sẽ đi tìm hiểu về cổng update z. Cổng z được tính như sau:

Trong đó:

* trọng số ứng với
* trọng số ứng với



1. Mô tả cách tính của cổng update [2]

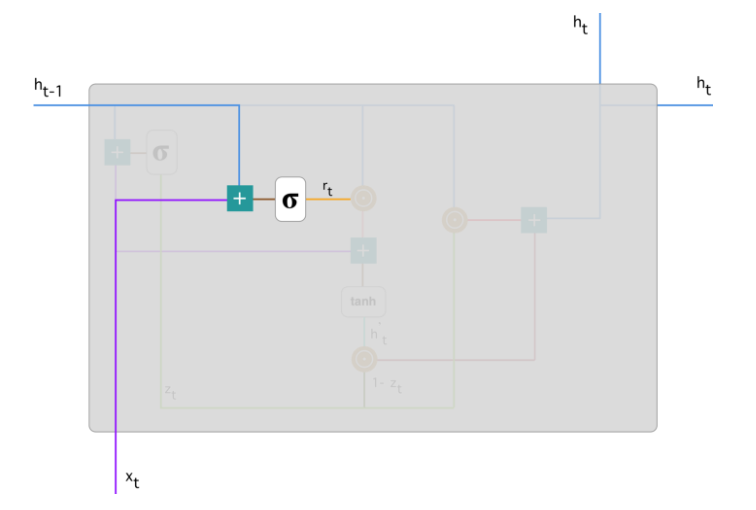
Cổng update tượng trưng cho việc tại mỗi neural, neural sẽ quyết định nên giữ lại bao nhiêu phần từ input để truyền lại cho neural ở các lớp tiếp theo. Điều này giúp neural chỉ giữ lại những phần cần thiết nhằm tránh gây ra hiện tượng vanishing gradient về sau.

* + 1. Cổng reset

Sau khi tính cổng update, ta sẽ đi thực hiện tính cổng reset r.

Trong đó:

* trọng số ứng với
* trọng số ứng với

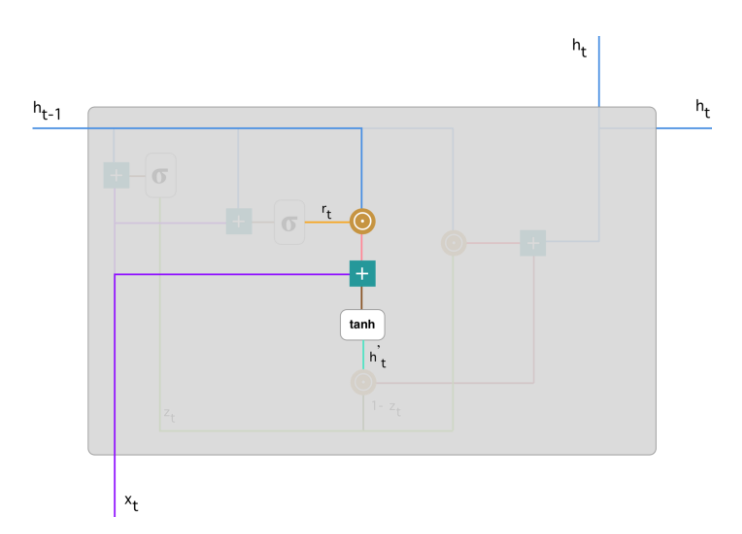


1. Mô tả cách tính cổng reset [2]

Cổng reset giúp neural quyết định xem nên bỏ đi bao nhiêu phần của input.

* + 1. Xuất kết quả

Đầu tiên ta sẽ tính giá trị :



1. Mô tả cách tính [2]

Cuối cùng ta sẽ tính giá trị đầu ra của neural:

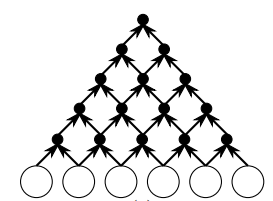
Ta sẽ áp dụng mạng neural RNN với mỗi neural là GRU cho cả 2 bộ là encoder và decoder.

Ở encoder đầu vào của mỗi neural sẽ là tập dữ liệu văn bản của ngôn ngữ gốc kèm với kết quả đầu ra của neural phía trước. Đầu ra của mỗi neural là kết quả tính được của mỗi neural.

Ở decoder đầu vào của mỗi neural gồm kết quả đầu ra của neural phía trước và kết quả ta dịch được tại bước phía trước (t – 1). Còn đầu ra sẽ gồm kết quả ta dịch được kèm với kết quả ta tính được ở neural hiện tại.

* 1. Convolutional Neural Network trong mô hình

Ngoài việc sử mô hình mạng neural RNN cho quá trình dịch máy, ta còn có thể sử dụng mô hình mạng neural Convolutional Neural Network (CNN) vào quá trình dịch. Mô hình dịch máy sử dụng CNN ta sẽ gọi là Gated Recursive CNN (grConv).



1. Cấu trúc của CNN trong encoder [3]

Ta có tập dữ liệu đầu vào:

Ở bước 0, hidden unit được tính như sau:

Trong đó U là trọng số tại bước 0.

Mạng CNN của ta sẽ có 4 trọng số chính là: , , và . Tại mỗi bước , đầu vào của neural j được tính dựa trên 3 giá trị đầu vào cơ bản: giá trị đầu vào bên trái (left - l), bên phải (right - r) và ở giữa (center - c).

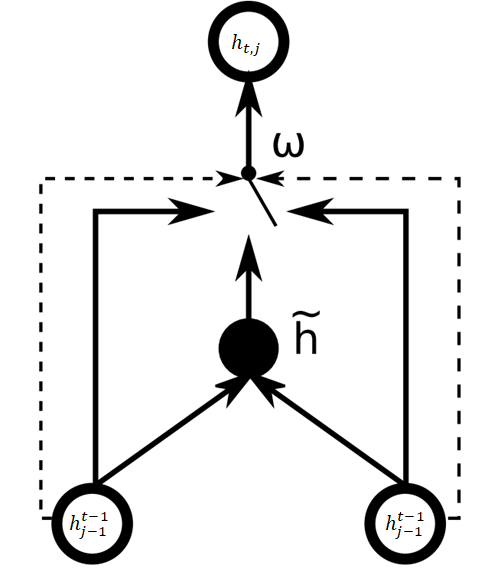
Giá trị ở giữa được tính như sau:

Giá trị của neural hiện tại được tính như sau:

Các giá trị , và được tính như sau:

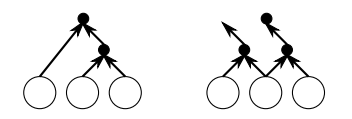
Trong đó và Z được tính như sau:

Theo kích hoạt này, ta có thể nghĩ đến việc kích hoạt (activate) một node duy nhất ở cấp độ đệ quy t chính là việc ta sẽ sự lựa chọn giữa kết quả của hàm activation mới tính toán được từ nhánh trái hay nhánh phải của bước t – 1 ở phía trước. Lựa chọn này cho phép cấu trúc tổng thể của tích chập đệ quy có thể thay đổi một cách linh hoạt đối với các mẫu đầu vào. Quá trình của một node có được biểu diễn đơn giản bằng hình sau:



1. Minh họa cách hoạt động của hàm activation ở mỗi node [3]

Về mặt này, chúng ta thậm chí có thể coi grConv là một giải thuật phân tích cú pháp không giám sát.

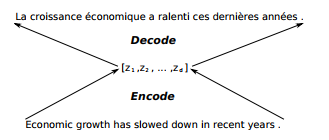


1. Minh họa cho một số cấu trúc của CNN [3]
   1. Mô hình Encoder-Decoder

Nhiệm vụ của dịch thuật có thể được hiểu theo quan điểm của học máy đó là việc học các phân phối có điều kiện của một câu đích (kết quả sau quá trình dịch) f dựa trên một câu nguồn e (câu muốn dịch). Bằng cách lấy mẫu (e,f) từ thực tế hoặc bằng cách sử dụng thuật toán tìm kiếm (gần đúng) mà ta có thể tìm được mức tối đa của phân phối.

Một số bài báo gần đây đã đề xuất sử dụng mạng lưới thần kinh để trực tiếp tìm hiểu phân phối có điều kiện từ một kho văn bản song ngữ (Kalchbrenner và Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). Ví dụ, các tác giả (Kalchbrenner và Blunom, 2013) đã đề xuất một cách tiếp cận liên quan đến mô hình n-gram tích chập để trích xuất một vectơ có độ dài cố định của một câu nguồn được giải mã bằng mô hình n-gram ngược được tăng cường bằng RNN. Trong (Sutskever et al., 2014), một RNN với các đơn vị LSTM đã được sử dụng để mã hóa một câu nguồn và bắt đầu từ trạng thái ẩn cuối cùng, để giải mã một câu đích. Tương tự, các tác giả của (Cho et al., 2014) đã đề xuất sử dụng RNN để mã hóa và giải mã một cặp cụm từ.

Tất cả các mô hình này đều được xây dựng dựa trên cấu trúc gồm 2 bộ phận chính là bộ encoder (bộ mã hóa) và bộ decoder (bộ giải mã) (xem hình 1.9). Bộ mã hóa xử lý một câu dữ liệu đầu vào có độ dài thay đổi (câu nguồn) từ đó xây dựng được một vectơ biểu diễn có độ dài cố định (ký hiệu là trong Hình 1.9). Dựa trên kết quả vector này, bộ giải mã sẽ tạo ra được một chuỗi có độ dài thay đổi tương ứng (câu đích).



1. Mô hình encoder-decoder cơ bản [3]

Trước đây trong tài liệu của (Sutskever et al., 2014) phương pháp encoder-decoder này đã được sử dụng chủ yếu như một phần của hệ thống dịch máy thống kê (Statistical Machine Translation – SMT) hiện có. Cách tiếp cận này được sử dụng để xếp hạng lại danh sách n đối tượng tốt nhất được tạo bởi hệ thống SMT trong (Kalchbrenner và Blunom, 2013) và các tác giả của (Cho et al., 2014) đã sử dụng phương pháp này để cung cấp điểm số bổ sung cho phần điểm hiện có của bảng cụm từ.

* + 1. Word embedding

Word embedding trong thực tế là một lớp các kỹ thuật trong đó các từ riêng lẻ được biểu diễn dưới dạng các vectơ có giá trị thực trong một không gian vectơ được xác định trước. Mỗi từ được ánh xạ tới một vectơ và các giá trị vectơ được học theo cách giống với mạng thần kinh, và do đó kỹ thuật này thường được đưa vào lĩnh vực học sâu.

Chìa khóa của cách tiếp cận là ý tưởng sử dụng một đại diện phân phối mật độ cho mỗi từ.

Mỗi từ được đại diện bởi một vectơ có giá trị thực, thường là hàng chục hoặc hàng trăm chiều, khác với one-hot encoding mỗi từ sẽ là một vector chứa các giá trị 0 và 1.

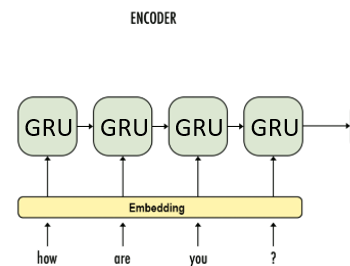
Các đại diện phân phối được học dựa trên việc xuất hiện của các từ. Các từ xuất hiện hay được sử dụng như thế nào thì sẽ có vector biểu diễn tương ứng, vô hình ta có thể tự nhiên nắm bắt ý nghĩa của chúng.

Có một lý thuyết ngôn ngữ sâu hơn đằng sau giúp một phần giải thích cách tiếp cận này. Lý thuyết này có tên là “giả thuyết phân phối” (“distributional hypothesis”) của Zellig S. Harris, có thể được tóm tắt như sau: những từ có ngữ cảnh tương tự sẽ có nghĩa tương tự. (Có thể tham khảo thêm trong tài liệu: Zellig S. Harris, 1954, Distributional Structure).

Trên thực tế, con người giao tiếp với nhau thông qua các kí tự, giọng nói, hình ảnh,… Tuy nhiên điều này vô cùng khó khăn cho máy tính khi xử lý với bất kỳ bài toán. Vì máy tính “quen” với việc xử lý trên các dữ liệu dạng số, vì vậy trước khi ta thực hiện quá trình dịch máy, ta sẽ phải word embedding dữ liệu để đưa dữ liệu về dạng các vector biểu diễn số.

* + 1. Encoder

Cấu trúc của một bộ GRU-RNN encoder có thể được minh họa như sau:



1. Cấu trúc của một bộ GRU-RNN encoder [4]

Đầu vào của neural đầu tiên chỉ có dữ liệu gốc, không có giá trị ẩn (hidden unit bằng 0).

Kết quả đầu ra của encoder sẽ là tập vector có độ dài cố định:

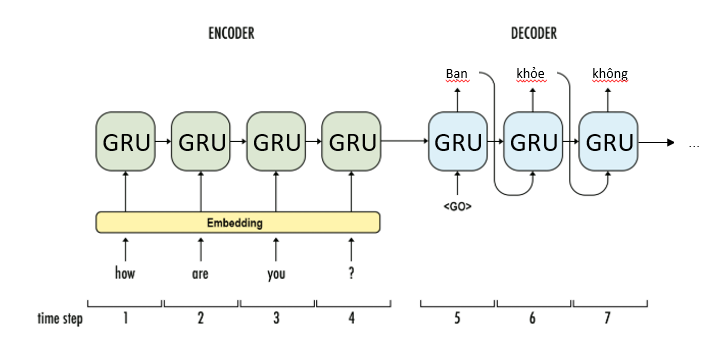
Cấu trúc của grConv encoder giống với phần mô tả ở mục 1.3.

* + 1. Decoder

*Decoder thuần:*

Một bộ decoder đơn thuần sẽ chỉ sử dụng tập kết quả đầu ra của encoder để làm đầu vào cho mình. Tập kết quả của encoder tạo được này đôi khi được gọi là vector bối cảnh vì nó mã hóa bối cảnh từ toàn bộ tập dữ liệu ban đầu. Vectơ bối cảnh này được sử dụng làm trạng thái ẩn (hidden unit) ban đầu cho decoder.

Ở mỗi bước giải mã (decode), input đầu vào, decoder sẽ được cung cấp bởi 2 giá trị: kết quả và trạng thái ẩn của neural phía trước (xem minh họa ở hình 1.11). Chuỗi decoder sẽ bắt đầu bằng chuỗi <GO> (hình 1.11) (do là neural đầu tiên trong decoder) và trạng thái ẩn đầu tiên chính là vectơ ngữ cảnh (kết quả cuối cùng của encoder).



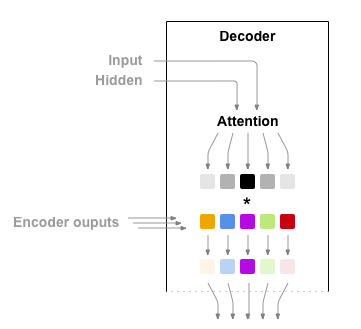
1. Minh họa cho decoder [4]

*Decoder sử dụng attention:*

Attention là một cơ chế kỹ thuật nhằm giúp giải quyết các giới hạn của kiến trúc encoder-decoder trên các chuỗi dài (chuỗi dữ liệu đầu vào). Attention sẽ giúp tăng tốc độ học tập và cải thiện khả năng dự đoán của mô hình.

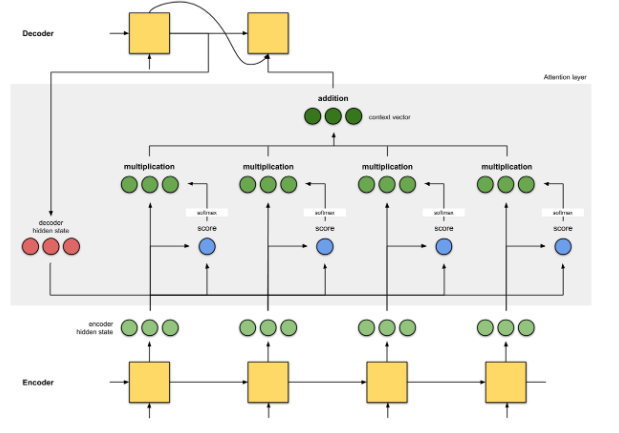
Do vector ngữ cảnh này có độ dài cố định và là kết quả sau quá trình mã hóa từ dữ liệu đầu vào có độ dài thay đổi, nên vector này sẽ chứa rất nhiều thông tin. Việc mã hóa (decode) vector này sẽ là một áp lực lớn đối với decoder.

Cơ chế attention cho phép tại mỗi bước input của decoder, decoder có thể tập trung vào từng phần khác nhau của encoder output. Đầu tiên, chúng ta sẽ tính tập trọng số attention (attention weight). Sau đó ta sẽ lấy tập trọng số này nhân với các vectơ đầu ra của encoder để tạo ra một tổ hợp có trọng số. Kết quả (được gọi là attn\_applied) này giúp nhận biết được từng phần của chuỗi đầu vào, và từ đó sẽ giúp decoder chọn được từ (word) đầu ra đúng.



1. Cơ chế attention [5]

Thay vì mã hóa chuỗi đầu vào thành một vectơ ngữ cảnh cố định duy nhất, mô hình chú ý phát triển một vectơ bối cảnh được lọc riêng cho từng bước thời gian t ở đầu ra.



1. Cấu trúc chi tiết của một attention [6]
   * 1. Linguistic Input Feature

Mỗi nước mỗi vùng sẽ có những ngôn ngữ khác nhau. Trong các ngôn ngữ đó, mỗi ngôn ngữ đều có nét đặc trưng riêng cho từng loại ngôn ngữ. Ta gọi chung các đặc trưng riêng này là linguistic feature (đặc trưng ngôn ngữ).

Chính vì tính đặc trưng của mỗi ngôn ngữ khác nhau như vậy sẽ dẫn đến tình trạng quá trình dịch sẽ không chính xác do phải phụ thuộc vào nhiều yếu tố như ngữ pháp, đồng nghĩa, đồng âm,…

Vì vậy việc xử lý các đặc trưng này trước khi đưa dữ liệu vào quá trình dịch sẽ giúp cho quá trình dịch trở nên một cách chính xác hơn. Điều này đã được thể hiện rõ trong tài liệu: Rico Sennrich, Barry Haddow, 2016, Linguistic Input Features Improve Neural Machine Translation.

Trong tài liệu Rico Sennrich, Barry Haddow, 2016, Linguistic Input Features Improve Neural Machine Translation giới thiệu cách xử lý 5 loại đặc thù của ngôn ngữ mà từ đó có thể giúp ta có thể cải thiện được độ chính xác của quá trình dịch. 5 loại đặc thù này gồm:

* Lemmas.
* Subword tag.
* Morphological feature.
* POS tag.
* Dependency label.

Tài liệu này xin tập trung chủ yếu vào cách xử lý đặc thù về lenmas (bổ đề).

Thông thường các trạng thái ẩn của encoder sẽ được tính như sau:

Trong đó:

* : vector trạng thái ẩn hiện tại của encoder.
* , : vector trọng số.
* : ma trận word embedding.
* : dữ liệu input.
* : vector trạng thái ẩn của encoder ở bước phía trước (t – 1).
* : kích thước của ma trận word embedding.
* : số lượng tế bào GRU (số lượng trạng thái ẩn).
* : kích thước của bộ từ điển.

Khi có thêm các yếu tố đặc thù ngôn ngữ , các trạng ẩn của encoder sẽ được tính như sau:

Trong đó:

* phép nối vector.
* : ma trận word embedding.
* : kích thước của ma trận word embedding.
* : kích thước của bộ từ điển có chứa yếu tố đặc thù ngôn ngữ.

Mô hình tổng quát này có thể sử dụng cho bất kỳ đặc thù ngôn ngữ tùy ý nào mà ta mong muốn.

Trong một ngôn ngữ, một từ có thể được viết lại thành nhiều dạng để phù hợp với ngữ cảnh, ngữ pháp,… Ví dụ trong tiếng anh, ta có từ: interest (thích thú, thú vị,…) đóng vai trò là động từ hoặc danh từ, tuy nhiên khi chuyển về tính từ, ta được: interesting hoặc interested. Ta có thể thấy rằng interesting và interested đều được tạo từ interest. Bổ đề (lemmas – từ gốc) của interesting, interested, interest sẽ là interest.

Dựa trên công thức (1), chính là quá trình ta sẽ thực hiện việc đặc thù hóa dữ liệu trước, sau đó ta sẽ đem đi word embedding.

Vì vậy việc lemmas hóa (Lemmatization) chính là quá trình ta sẽ tìm từ gốc cho toàn bộ các từ trong tập dữ liệu trước khi ta đưa dữ liệu đem đi huấn luyện.

* + 1. Beam search
  1. Tóm tắt chương 1

TÀI LIỆU THAM KHẢO

**Tiếng Anh:**

1. <https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346>
2. <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>
3. Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.
4. https://medium.com/@t.schnake/a-formalization-of-a-simple-sequential-encoder-decoder-b31be7e92988
5. <https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html>
6. <https://towardsdatascience.com/attn-illustrated-attention-5ec4ad276ee3>