TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ CƯƠNG CHI TIẾT LUẬN VĂN THẠC SĨ**

**CẢI THIỆN DỊCH MÁY NƠ-RON**

**VIỆT-ANH BẰNG GIÓNG HÀNG TỪ HƯỚNG DẪN**

*Người hướng dẫn*: **TS. Nguyễn Chí Thiện**

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

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ CƯƠNG CHI TIẾT LUẬN VĂN THẠC SĨ**

**CẢI THIỆN DỊCH MÁY NƠ-RON**

**VIỆT-ANH BẰNG GIÓNG HÀNG TỪ HƯỚNG DẪN**

*Người hướng dẫn*: **TS. Nguyễn Chí Thiện**

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

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021**

**ĐỀ CƯƠNG CHI TIẾT ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đề cương chi tiết của riêng tôi và được sự hướng dẫn của TS. 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 17 tháng 11 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](#_Toc66793280)

[MỤC LỤC 1](#_Toc66793281)

[TÓM TẮT 6](#_Toc66793282)

[Chương 1: TỔNG QUAN ĐỀ TÀI 7](#_Toc66793283)

[1.1 Tính cấp thiết của đề tài 7](#_Toc66793284)

[1.2 Mục tiêu nghiên cứu 8](#_Toc66793285)

[1.3 Tổng quan nghiên cứu 8](#_Toc66793286)

[1.4 Đối tượng nghiên cứu 9](#_Toc66793287)

[1.5 Phạm vi nghiên cứu 9](#_Toc66793288)

[1.6 Phương pháp nghiên cứu 10](#_Toc66793289)

[1.7 Ý nghĩa của đề tài 10](#_Toc66793290)

[1.8 Cấu trúc của đề cương 10](#_Toc66793291)

[Chương 2: MÔ HÌNH GIÓNG HÀNG TỪ 11](#_Toc66793292)

[2.1 Giới thiệu về dịch máy thống kê 11](#_Toc66793293)

[2.1.1 Mô hình dịch (translation model) – kĩ thuật gióng hàng từ 13](#_Toc66793294)

[2.1.2 Quá trình tìm kiếm 14](#_Toc66793295)

[2.2 Các mô hình dịch máy sử dụng kĩ thuật gióng hàng từ của IBM 14](#_Toc66793296)

[2.2.1 Mô hình 1 của IBM 16](#_Toc66793297)

[2.2.2 Mô hình 2 của IBM 19](#_Toc66793298)

[2.3 Kĩ thuật gióng hàng từ nhanh dựa trên mô hình 2 của IBM 20](#_Toc66793299)

[2.3.1 Các phép suy luận biên 22](#_Toc66793300)

[2.3.2 Các phép cải tiến 22](#_Toc66793301)

[2.4 Tổng kết chương 2 24](#_Toc66793302)

[Chương 3: MÔ HÌNH DỊCH MÁY NƠ-RON 25](#_Toc66793303)

[3.1 Cấu trúc của 1 mạng nơ-ron cơ bản 26](#_Toc66793304)

[3.1.1 Perceptron – mạng neural phi tuyến tính cơ bản 26](#_Toc66793305)

[3.1.2 Mạng nơ-ron phi tuyến tính nâng cao 27](#_Toc66793306)

[3.1.3 Giải thuật lan truyền ngược (Backpropagation algorithm) 30](#_Toc66793307)

[3.2 Mô hình dịch máy nơ-ron sequence-to-sequence (encoder-decoder) cơ bản 38](#_Toc66793308)

[3.2.1 Giới thiệu về dịch máy neural 39](#_Toc66793309)

[3.2.2 Giới thiệu về Recurrent Neural Network (RNN) 39](#_Toc66793310)

[3.2.3 Mô hình Encoder – Decoder 41](#_Toc66793311)

[3.2.4 Cơ chế Attention 43](#_Toc66793312)

[3.3 Mô hình dịch máy transformer 45](#_Toc66793313)

[3.3.1 Embedding và softmax 50](#_Toc66793314)

[3.3.2 Positional Encoding 50](#_Toc66793315)

[3.3.3 Cơ chế Multi-head Attention 51](#_Toc66793316)

[3.3.4 Mạng Position-wise Feed-Forward 53](#_Toc66793317)

[3.4 Kết hợp kĩ thuật gióng hàng từ vào mô hình transformer 54](#_Toc66793318)

[3.5 Tổng kết chương 3 55](#_Toc66793319)

[Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ 56](#_Toc66793320)

[4.1 Dữ liệu 56](#_Toc66793321)

[4.2 Xử lý tiền dữ liệu 56](#_Toc66793322)

[4.3 Thực hiện gióng hàng từ bằng công cụ hỗ trợ fast\_align 59](#_Toc66793323)

[4.4 Chạy chương trình bằng công cụ hỗ trợ fairseq 62](#_Toc66793324)

[4.4.1 Chuẩn bị 62](#_Toc66793325)

[4.4.2 Thực hiện huấn luyện và kiểm thử mô hình transformer thuần 63](#_Toc66793326)

[4.4.3 Thực hiện huấn luyện và kiểm thử mô hình transformer sử dụng gióng hàng từ 64](#_Toc66793327)

[4.5 Kết quả và đánh giá 66](#_Toc66793328)

[4.5.1 Kết quả của mô hình transformer thuần 66](#_Toc66793329)

[4.5.2 Kết quả của mô hình transformer có sử dụng gióng hàng từ 66](#_Toc66793330)

[4.5.3 So sánh và đánh giá 67](#_Toc66793331)

[4.6 Tổng kết 68](#_Toc66793332)

[TÀI LIỆU THAM KHẢO 69](#_Toc66793333)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1. Minh họa cho quá trình dịch máy thống kê [2] 12](#_Toc66793334)

[Hình 1.2. Minh họa cho quá trình decode [2] 13](#_Toc66793335)

[Hình 2.1. Minh họa cho phân phối xác suất gióng hàng từ 21](#_Toc66793336)

[Hình 3.1. Cấu trúc cơ bản của 1 neural ở người [5] 25](#_Toc66793337)

[Hình 3.2. Cấu trúc cơ bản của perceptron 26](#_Toc66793338)

[Hình 3.3. Minh họa về việc thay đổi 28](#_Toc66793339)

[Hình 3.4. Đồ thị của hàm sigmoid [6] 29](#_Toc66793340)

[Hình 3.5. Sơ lược cấu tạo hoạt động của RNN 40](#_Toc66793341)

[Hình 3.6. Cấu trúc tổng quát của mô hình encoder-decoder [7] 41](#_Toc66793342)

[Hình 3.7. Cấu trúc RNN encoder 42](#_Toc66793343)

[Hình 3.8. Minh họa cho decoder 43](#_Toc66793344)

[Hình 3.9. Tổng quan cấu trúc mô hình transformer [9] 46](#_Toc66793345)

[Hình 3.10. Đầu vào encoder và decoder của mô hình [9] 47](#_Toc66793346)

[Hình 3.11. Positional Encoding [9] 47](#_Toc66793347)

[Hình 3.12. Cấu trúc của 1 lớp trong encoder trong transformer [9] 48](#_Toc66793348)

[Hình 3.13. Cấu trúc 1 lớp của decoder trong transformer [9] 49](#_Toc66793349)

[Hình 3.14. Tạo giá trị dự đoán từ decoder [9] 49](#_Toc66793350)

[Hình 3.15. Cơ chế self-attention của transformer [9] 51](#_Toc66793351)

[Hình 3.16. Minh họa cho cơ chế mask 52](#_Toc66793352)

[Hình 3.17. Cơ chế multi-head attention [9] 53](#_Toc66793353)

[Hình 4.1. Chuẩn bị thư viện để xử lý tiền dữ liệu 56](#_Toc66793354)

[Hình 4.2. Hàm lọc bớt thực thể html 57](#_Toc66793355)

[Hình 4.3. Tiến hành đọc và làm sạch dữ liệu 57](#_Toc66793356)

[Hình 4.4. Tách từ và chuẩn bị dữ liệu cho mô hình gióng hàng từ 58](#_Toc66793357)

[Hình 4.5. Xuất kết quả quá trình tiền xử lý ra file 59](#_Toc66793358)

[Hình 4.6. Tải phần mềm fast\_align bằng github 59](#_Toc66793359)

[Hình 4.7. Biên dịch chương trình 59](#_Toc66793360)

[Hình 4.8. Chạy file biên dịch để cài đặt fast\_align 60](#_Toc66793361)

[Hình 4.9. Ví dụ minh họa kiểu dữ liệu để chạy fast\_align 60](#_Toc66793362)

[Hình 4.10. Chạy fast\_align 61](#_Toc66793363)

[Hình 4.11. Kết quả sau khi chạy fast\_align 61](#_Toc66793364)

[Hình 4.12. Minh họa cho kiểu dữ liệu trả về của fast\_align 62](#_Toc66793365)

[Hình 4.13. Tải và cài đặt fairseq 63](#_Toc66793366)

[Hình 4.14. Chuẩn bị các file cần thiết cho fairseq 63](#_Toc66793367)

[Hình 4.15. Thực hiện huấn luyện mô hình transformer thuần 63](#_Toc66793368)

[Hình 4.16. Thực hiện kiểm thử mô hình transformer thuần 64](#_Toc66793369)

[Hình 4.17. Thực hiện huấn luyện mô hình transformer sử dụng gióng hàng từ 65](#_Toc66793370)

[Hình 4.18. Thực hiện kiểm thử mô hình transformer sử dụng gióng hàng từ 65](#_Toc66793371)

[Hình 4.19. Kết quả huấn luyện của mô hình transformer thuần 66](#_Toc66793372)

[Hình 4.20. Kết quả kiểm thử của mô hình transformer thuần 66](#_Toc66793373)

[Hình 4.21. Kết quả huấn luyện mô hình transformer có sử dụng gióng hàng từ 67](#_Toc66793374)

[Hình 4.22. Kết quả kiểm thử mô hình transformer có sử dụng gióng hàng từ 67](#_Toc66793375)

**DANH MỤC BẢNG BIỂU**

[Bảng 1. Bảng so sánh kết quả giữa mô hình transform thuần và mô hình transformer có sử dụng gióng hàng từ 67](#_Toc66793376)

DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

**CÁC KÝ HIỆU**

*S Câu nguồn*

*Các từ trong câu nguồn*

*T Câu đích*

*Các từ trong câu đích*

*Xác suất*

*Hàm sigmoid*

**CÁC CHỮ VIẾT TẮT**

MT Machine Translation

RBMT Rule-based Machine Translation

SMT Statistical Machine Translation

NMT Neural Machine Translation

LSTM Long-short Term Memory

Đpcm Điều phải chứng minh

RNN Recurrent Neural Network

CNN Convolutional Neural Network

PE Positional Encoding

FFN Feed Forward Network

TÓM TẮT

Hiện nay việc áp dụng các giải thuật, mô hình vào quá trình dịch đang là một lĩnh vực nghiên cứu nhằm giúp cải thiện và tăng tính tự động của quá trình dịch. Quá trình áp dụng các giải thuật hay mô hình vào quá trình dịch được gọi tắt là dịch máy.

Tài liệu này xin trình bày mô hình dịch máy nơ-ron transformer làm chủ đạo trong quá trình thực hiện. Mô hình gióng hàng từ (word alignment) sẽ được dùng trong quá trình xây dựng mô hình transformer.

Bằng việc kết hợp này, ta hy vọng có thể cải thiện được chất lượng của quá trình dịch máy từ tiếng Việt sang tiếng Anh.

Quá trình thực nghiệm sẽ được tiến hành trên bộ dữ liệu Việt – Anh được đăng trên hội thảo quốc tế The International Workshop on Spoken Language Translation 2015 (IWSLT 2015).

Sau quá trình thực nghiệm, ta sẽ nghiệm thu, đánh giá mô hình dựa trên kết quả đạt được.

Chương 1: TỔNG QUAN ĐỀ TÀI

* 1. Tính cấp thiết của đề tài

Dịch thuật là sự truyền đạt ý nghĩa của văn bản ngôn ngữ nguồn sang văn bản ngôn ngữ mong muốn tương đương [1].

Dịch thuật cần người có am hiểu ngôn ngữ của cả ngôn ngữ gốc (ngôn ngữ nguồn) của văn bản lẫn ngôn ngữ mong muốn (ngôn ngữ đích) mà nhu cầu đời sống cần sử dụng. Vì thế dịch thuật là một quá trình tốn rất nhiều thời gian và công sức. Vì thế với sự phát triển của thời buổi công nghệ hiện nay, ta có thể thực hiện việc dịch văn bản một cách tự động bằng các chương trình trên máy tính, với hy vọng ta có thể làm giảm chi phí thời gian, công sức, lẫn con người của quá trình dịch thuật.

Tuy nhiên, dịch thuật bằng máy (dịch máy, Machine translation (MT)) vẫn còn nhiều hạn chế. Các yếu tố này thường là do: một từ có nhiều nghĩa giữa cả 2 ngôn ngữ, sự khác biệt trong cấu trúc ngữ pháp của cả 2 ngôn ngữ, cách vận dụng, tình huống xảy ra trong văn bản của ngôn ngữ nguồn,… Đặc biệt là việc dịch từ tiếng Việt sang tiếng Anh. Vì tiếng Việt được mọi người trên thế giới đánh giá là một ngôn ngữ có mức độ khó và phức tạp về từ vựng, ngữ pháp, ý nghĩa.

Dựa trên cách thực hiện, dịch máy thường có thể được phân thành các hướng sau: dịch máy dựa trên tri thức hay kinh nghiệm của các chuyên gia trong lĩnh vực dịch thuật (Rule-based machine translation (RBMT)), dịch máy dựa trên văn bản bằng các tính toán lẫn phân tích về mặt thống kê (Statistical machine translation (SMT)), dịch máy kết hợp giữa rule-based machine translation và statistical machine translation được gọi là Hybrid machine translation, cuối cùng một phương hướng mới đó là sử dụng các mô hình mạng thần kinh (nơ-ron – nerual) nhân tạo (Neural machine translation – NMT).

Với các khó khăn hiện nay đã nêu ở trên, việc tìm ra những phương pháp giúp cải thiện quá trình dịch máy là một điều cần thiết. Đề tài luận văn xin được tập trung giới hạn nghiên cứu cải thiện dịch máy từ văn bản ngôn ngữ tiếng Việt sang ngôn ngữ tiếng Anh bằng mô hình transformer (một dạng của mô hình mạng neural) kết hợp với kỹ thuật gióng háng từ hướng dẫn (alignment). Vì thế đề tài có tên là: “Cải thiện dịch máy nơ-ron Việt-Anh bằng gióng hàng từ hướng dẫn.

* 1. Mục tiêu nghiên cứu

Mục tiêu nghiên cứu của đề tài đó chính là sử dụng kỹ thuật gióng hàng từ vào mô hình mạng neural để cải thiện quá trình dịch máy từ văn bản tiếng Việt thành văn bản tiếng Anh. Mục tiêu cụ thể gồm:

* Tiền xử lý và tạo ra một tập dữ liệu phù hợp cho việc thực hiện kỹ thuật gióng hàng từ cũng như cho quá trình huấn luyện của mô hình transformer
* Chọn mô hình mạng neural và kỹ thuật gióng hàng từ cho quá trình dịch.
* Tiến hành thực nghiệm, đánh giá và so sánh giữa 2 phương pháp: phương pháp sử dụng mô hình mạng neural thuần với phương pháp sử dụng mô hình neural kết hợp với kỹ thuật alignment (cả 2 phương pháp sẽ đều dùng cùng một loại mô hình mạng neural).
  1. Tổng quan nghiên cứu

Dịch máy hiện nay thường được nghiên cứu dựa trên các mô hình mạng nơ-ron nhân tạo. Các mô hình nơ-ron này hoạt động dựa trên các cấu trúc hồi quy làm cốt lõi để trích xuất và nắm bắt cách phụ thuộc của các yếu tố trong cả 2 ngôn ngữ nguồn và đích được đem ra để dịch. Phương pháp chủ yếu của các giải thuật này đó chính là tối đa hóa các xác suất phụ thuộc trong quá trình huấn luyện, để từ đó mô hình có thể cho ra được kết quả dịch gần đúng và hợp lý.

Các mô hình mạng nơ-ron sơ khai ban đầu là các mạng nơ-ron với mỗi nhân trong mô hình là 1 hàm tuyến tính cơ bản, sau đó phát triển và được thay thế bằng các hàm phi tuyến tính. Theo thời gian, cấu trúc nhân của các mạng nơ-ron cũng được phát triển từ 1 hàm tính toán thành 1 tổ hợp các hàm tính toán nhằm làm giảm các hạn chế của việc chỉ sử dụng 1 hàm đơn thuần. Các cấu trúc tiêu biểu này thường là: Long-short Term Memory (LSTM) của Sepp Hochreiter; Jürgen Schmidhuber (1997), Gated Recurrent Unit (GRU) của Kyunghyun Cho và các cộng sự (2014),…

Bên cạnh sự phát triển nghiên cứu về nhân của nơ-ron, là sự phát triển trong nghiên cứu về cấu trúc của các mạng nơ-ron. Tiêu biểu là các mô hình Convolutional Neural Network (CNN) được Yann Lecun và cộng sự giới thiệu trong những thập niên 1980, mô hình Sequence-to-Sequence của Sutskever Ilya, Vinyals Oriol, Lê Quốc Việt (2014), Transformer của Ashish Vaswani và các cộng sự (2017),…

Thêm vào đó còn có các nghiên cứu về các kỹ thuật nhằm cải thiện chất lượng dịch. Tiêu biểu là các kỹ thuật alignment của Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio (2014), attention của Minh-Thang Luong, Hieu Pham, Christopher D. Manning (2015),…

Dựa trên các thành tựu nghiên cứu trên, ta sẽ tiến hành đi nghiên cứu và kết hợp các phương pháp nhằm cải thiện chất lượng dịch, nhất là quá trình dịch từ tiếng Việt sang tiếng Anh. Đó cũng chính là mục tiêu hướng tới của đề tài.

* 1. Đối tượng nghiên cứu

Mô hình dịch máy nơ-ron transformer.

Dữ liệu văn bản tiếng Việt, tiếng Anh: đăng trên hội thảo quốc tế The International Workshop on Spoken Language Translation 2015 (IWSLT 2015).

Kết quả giữa mô hình dịch máy transformer thuần với mô hình dịch máy transformer có sử dụng gióng hàng từ.

* 1. Phạm vi nghiên cứu

Nghiên cứu và ứng dụng kĩ thuật gióng hàng từ và mô hình transformer.

Sử dụng tập dữ liệu Việt-Anh phổ biến được đăng trên hội thảo quốc tế The International Workshop on Spoken Language Translation 2015 (IWSLT 2015) để thực nghiệm, đánh giá.

* 1. Phương pháp nghiên cứu

Tiến hành sử dụng dữ liệu để thực nghiệm và đánh giá kết quả của các mô hình dịch máy được nêu ở mục 1.4, sử dụng kết hợp với kỹ thuật gióng hàng từ (alignment). Từ đó rút ra kết luận cho đề tài.

* 1. Ý nghĩa của đề tài

Đóng góp vào việc cải thiện chất lượng của quá trình dịch máy nói chung cũng như quá trình dịch từ tiếng Việt sang tiếng Anh nói riêng nhằm góp phần đưa Việt Nam hội nhập hơn với thế giới.

* 1. Cấu trúc của đề cương

Đề tài được trình bày thành 4 chương:

Chương 1: Trình bày lý do nghiên cứu, mục tiêu nghiên cứu, tổng quan, phạm vi, đối tượng nghiên cứu cũng như ý nghĩa nghiên cứu của đề tài.

Chương 2: Trình bày cơ sở lý thuyết của kỹ thuật gióng hàng từ.

Chương 3: Trình bày cơ sở lý thuyết chung về mô hình no-ron cũng như mô hình nơ-ron cụ thể được thực nghiệm trong đề tài.

Chương 4: Trình bày quá trình thực nghiệm, kết quả sau khi thực nghiệm, tiến hành đánh giá, kết luận dựa trên kết quả thực nghiệm.

Chương 2: MÔ HÌNH GIÓNG HÀNG TỪ

2.1 Giới thiệu về dịch máy thống kê

Dịch máy thống kê (Statistical Machine Translation – SMT) là mô hình dịch tự động dựa trên các tham số thống kê được tính toán dựa trên các phân tích thống kê từ các tập dữ liệu văn bản song ngữ.

SMT lần đầu được giới thiệu bởi Warren Weaver vào năm 1947. Ông cho rằng các suy diễn logic của ngôn ngữ có thể được xem như các suy diễn logic của toán học. Kết quả dịch ở tập văn bản ngôn ngữ đích có thể được xác định dựa trên thống kê xuất hiện của các từ trên tập văn bản ngôn ngữ nguồn.

Quá trình dịch của SMT bắt đầu bằng các tập văn bản song ngữ đã được dịch một cách chính xác. Từ đây ta sẽ tiến hành đem các tập văn bản này đi thực hiện các phép thống kê để tạo ra 1 mô hình thống kê. Sau khi tạo được mô hình thống kê, ta sẽ tiến hành dịch các văn bản mong muốn dựa trên mô hình này. Kết quả dịch của văn chưa được dịch này sẽ tập các từ có xác suất phù hợp với tần số xuất hiện của các từ tương ứng trong văn bản nguồn.

Cho 1 cặp câu song ngữ với là câu của ngôn ngữ nguồn và là câu của ngôn ngữ đích được dịch từ tương ứng. Ta đi xây dựng mô hình dịch sao cho xác suất kết quả dịch của ngôn ngữ đích tương ứng có xác suất là cao nhất.

Ví dụ ta có câu trong văn bản tiếng Việt cần dịch là “Tôi ăn cơm” , ta xây dựng mô hình sao cho xác suất dịch xảy ra “I eat rice” lớn hơn xác suất dịch của “I eat dinner” , nói cách khác chính là .

Ta có kết quả dịch mong đợi:

Với là kết quả tính được của mô hình.

Điều ta mong muốn đó là có giá trị là cao nhất dẫn đến phải có giá trị cao nhất.

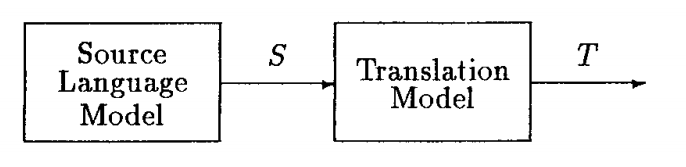
Ứng với mỗi của văn bản ngôn ngữ đích ta sẽ tiến hành đi tìm phù hợp ở văn bản ngôn ngữ nguồn có xác suất là cao nhất. Ta có xác suất Baye:

Với:

* là xác xuất của dựa vào .
* là xác suất số lần xuất hiện của trong tập dữ liệu.
* là xác suất số lần xuất hiện của khi biết .

Ta dễ dàng thấy không phụ thuộc vào , được tính dựa trên số lần xuất hiện của trong tập dữ liệu văn bản đích. Vì thế để đạt được cao nhất thì phải đạt được giá trị cao nhất.

Hệ thống dịch máy thống kê gồm 2 phần cơ bản: mô hình dịch (translation model), bộ giải mã (decoder).



1. Minh họa cho quá trình dịch máy thống kê [2]

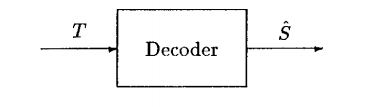
Trong hình 1.1 ta có thể thấy đầu vào của mô hình dịch (translation model) chính là các câu cần dịch và đầu ra là các kết quả dịch ứng với . Mô hình dịch sẽ tính toán các giá trị xác suất và từ đó tìm ra các giá trị khả thi sao cho giá trị đạt được là lớn nhất.

Ta đặt:

Nhiệm vụ của mô hình dịch chính là sử dụng giải thuật thống kê để tối ưu hóa .

Sau khi đã tính được từ mô hình dịch, ta sẽ đem đi thực hiện dịch bằng bộ giải mã (decoder).

Nhiệm vụ của decoder đó chính là ứng với mỗi câu T ở ngôn ngữ đích, decoder sẽ chọn các câu S sao cho giá trị là cực đại.



1. Minh họa cho quá trình decode [2]

Hình 1.2 ở trên là một minh họa về decoder. Đầu vào của decoder chính là các câu dịch ở ngôn ngữ đích, đầu ra là câu ngôn ngữ nguồn tương ứng, với:

Giá trị sẽ được tính dựa trên mô hình ngôn ngữ. Như ta đã biết là câu của ngôn ngữ nguồn, là một chuỗi gồm các từ .

2.1.1 Mô hình dịch (translation model) – kĩ thuật gióng hàng từ

Với cặp , (“Tôi uống trà”, “I drink tea”), ta có thể dễ dàng thấy rằng “Tôi” sẽ được dịch là “I”, “uống” sẽ được dịch là “drink”, “trà” sẽ được dịch là “tea”. Với cách dịch từ theo từ (word by word) như vậy, sẽ được gọi là gióng hàng từ, “Tôi” gióng thành “I”, “uống” gióng thành “drink”, “trà” gióng thành “tea”. Tuy nhiên, không phải lúc nào các cặp đều đơn giản như vậy. Ví dụ ta có cặp , (“Tôi đi ngủ”, “I go to sleep”), “Tôi” gióng thành “I”, “ngủ” gióng thành “sleep”, tuy nhiên “đi” gióng thành cho cả “go” và “to”. Đôi khi một từ trong tiếng Việt phải gióng với nhiều hoặc ít hơn một từ của tiếng Anh và ngược lại, đôi khi một từ của tiếng Anh phải gióng với nhiều hơn hoặc ít hơn một từ của tiếng Việt.

Ta đặt số lượng từ của được gióng với 1 từ của là fertility. Ta xét ví dụ: (“Tôi đi ngủ”, “I go to sleep”), , , . Xác suất của gióng hàng từ được tính bằng:

Với chính là xác suất gióng hàng từ của mỗi cặp gióng hàng từ.

Ví dụ:

Ta thấy rõ ràng trong cả 2 trường hợp (“Tôi uống trà”, “I drink tea”) và (“Tôi đi ngủ”, “I go to sleep”), vị trí của các từ Việt-Anh đều nằm ở gần nhau. Tuy nhiên, trong một vài trường hợp, một từ tiếng Việt khi dịch qua tiếng Anh, vị trí của 2 từ Việt-Anh này có thể nằm cách xa nhau. Những trường hợp này được gọi là “distortion” (dị biệt). Xác suất của distortion được tính bằng , với là vị trí của từ tiếng Anh tương ứng khi gióng, là vị trí của từ tiếng Việt, là số lượng từ gióng tương ứng.

2.1.2 Quá trình tìm kiếm

Quá trình tìm kiếm dựa trên kĩ thuật tìm kiếm ngăn xếp, ví dụ ta có câu: “Tôi uống trà”, quá trình tìm kiếm tuần tự sẽ là:

* .

Quá trình kết thúc khi kết quả dịch phù hợp với điều kiện yêu cầu.

2.2 Các mô hình dịch máy sử dụng kĩ thuật gióng hàng từ của IBM

Các mô hình được giới thiệu trong mục này được trích từ bài báo “The Mathematics of Statistical Machine Translation: Parameter Estimation” của nhóm tác giả ở trung tâm nghiên cứu T.J Watson của IBM (1993).

Các mô hình này chủ yếu giải quyết dựa trên xác suất đồng thời (joint distribution) , trong đó:

* là câu của ngôn ngữ nguồn.
* là câu của ngôn ngữ đích được dịch từ tương ứng.
* là biến ngẫu nhiên đại diện cho việc gióng hàng từ giữa và .

Ngoài ra còn có các biến và để biểu diễn cho độ dài của chuỗi và .

sẽ được viết lại thành:

là kết quả của phép tổng dựa trên tập các gióng hàng từ .

Việc gióng hàng từ của các mô hình cũng được giới hạn lại trong phạm vi 1 từ của sẽ chỉ gióng với 1 từ của hoặc không gióng với từ của . Nếu có từ và có từ, thì gióng hàng từ sẽ được biểu diễn thành cặp gióng hàng từ . Nếu 1 từ của ở vị trí gióng với 1 từ của ở vị trí , thì , còn nếu 1 từ của t ở vị trí j không gióng với bất kỳ từ nào của s thì .

Ta có:

Trong , ta có thể thấy rằng việc đầu tiên trước khi dịch đó là ta xem kết quả dịch sẽ gồm mấy từ. Ví dụ “Tôi đi ngủ” có thể dịch thành “I sleep” hoặc “I go to sleep” . Vì thế ta sẽ tính đầu tiên.

Tiếp theo ta sẽ tính tích xác suất của các cặp gióng hàng từ. Với một cặp gióng hàng từ ta sẽ tính tích của 2 giá trị xác suất. Giá trị thứ nhất chính là xác suất xảy ra từ thứ của sẽ được gióng với từ thứ của . Giá trị xác suất thứ hai chính là xác suất xảy ra sẽ được dùng để dịch.

2.2.1 Mô hình 1 của IBM

Trong mô hình 1, nhóm tác giả của IBM đã giả định rằng trong (Eq1), và là độc lập với nhau, vì thế sẽ chỉ còn phụ thuộc vào , chiều dài của chuỗi ngôn ngữ nguồn. Mỗi từ trong ngôn ngữ đích đều có khả năng không gióng với từ nào với ngôn ngữ nguồn hoặc có khả năng gióng từ 1 cho đến từ của ngôn ngữ nguồn, vì thế:

Còn xác suất xảy ra từ dịch sẽ chỉ còn phụ thuộc vào và .

Ta đặt:

Biểu thức sẽ trở thành:

Ứng với mỗi cũng đều có khả năng gióng từ 0 cho đến n từ của ngôn ngữ nguồn, vì thế:

Ta muốn kết quả dịch của ta là:

Vì thế ta sẽ vận dụng nhân tử Lagrange để đi tìm mục tiêu, sẽ trở thành:

Để tìm cực đại, ta sẽ tiến hành đạo hàm theo :

Rút gọn lại và giải phương trình, ta có điểm cực đại cần tìm:

Với và là hàm Kronecker :

Kết hợp , , , ta có thể rút gọn lại thành:

Ta có chính là tổng số lần mà gióng với .

Ta đặt số lần gióng mong đợi mà s gióng với t trong phép dịch chính là phép đếm (“”) của khi biết trong phép dịch, ta ký hiệu là .

Với:

Nếu ta chọn:

Thì biểu thức , trở thành:

Do phép dịch là tập hợp của phép dịch, nên:

Biểu thức trước khi áp dụng nhân tử Lagrange thì ta có thể rút gọn lại thành:

Ta thế biểu thức cho biểu thức và áp dụng Lagrange tương tự, ta sẽ được:

Ta sẽ kết hợp biểu thức và để tiến hành ước lượng cho các tham số theo trình tự sau:

1. Chọn các giá trị khởi tạo cho .
2. Ứng với mỗi cặp câu ta sẽ tính biểu thức . Các phép đếm trong khác không khi các và đều là các từ nằm trong . không phụ thuộc vào thứ tự sắp xếp của các từ, chỉ phụ thuộc vào số lần xuất hiện của các từ trong các cặp câu tương ứng.
3. Ứng với mỗi trong :

* Tính giá trị:
* Ứng với mỗi trong ta dùng để tính các giá trị .

1. Lặp lại bước 2 và 3 cho đến khi kết quả hội tụ đến giá trị mà ta mong muốn.

2.2.2 Mô hình 2 của IBM

Trong mô hình 1 của IBM, chúng ta không hề quan tâm đến vị trí của các từ trong các cặp câu dịch. Vì thế trong mô hình 2 chúng ta sẽ vừa kế thừa mô hình 1 vừa giả định rằng phụ thuộc vào và .

Ta đặt

Là xác suất gióng hàng từ, thỏa mãn:

Biểu thức được viết lại thành:

Áp dụng nhân tử Lagrange cho biểu thức , ta được:

Phép đếm mới sẽ là:

Áp dụng tương tự và , ta có:

Nếu cặp không có độ dài là hoặc không có độ dài là thì phép đếm tương ứng của sẽ bằng 0.

Ta có thể rút gọn lại thành:

Với biểu thức trên ta tạo được phép đếm:

Mô hình 1 là trường hợp đặc biệt của mô hình 2 vì trong mô hình 1 ta không quan tâm đến thứ tự gióng của phép dịch. Vì thế từ mô hình 1 mà ta có thể xây dựng nên mô hình tổng quát cho mô hình 2.

2.3 Kĩ thuật gióng hàng từ nhanh dựa trên mô hình 2 của IBM

Trong mục 2.2.1 và 2.2.2 ta đã tìm hiểu các giải thuật dịch máy của IBM dựa trên các phép tính thống kê. Trong mục này ta sẽ tìm hiểu kỹ thuật gióng hàng từ nhanh dựa trên mô hình 2 của IBM.

Kĩ thuật này được phát triển bởi nhóm nghiên cứu Chris Dyer, Victor Chahuneau, Noah A. Smith [4].

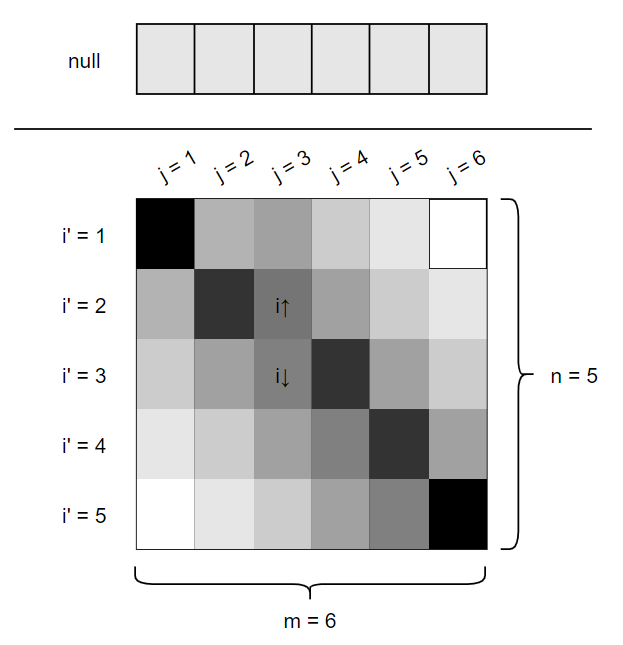
Các giá trị được sử dụng trong mục 2.2.1 và 2.2.2 là hàm Kronecker :

Giá trị trong mục này sẽ được tính bằng:

Với là xác suất dịch của từ vựng.

Trong hình 2.1, các giá trị xác suất gióng hàng từ tốt nhất chính là các giá trị nằm trên đường chéo chính.

Phân phối được tính dựa trên xác suất của gióng hàng từ khi không gióng với bất kỳ từ nào của ngôn ngữ nguồn và giá trị . Giá trị được dùng để kiểm soát mức độ gióng hàng từ giữa các từ. Trong trường hợp , phân phối sẽ càng tiến về dạng của mô hình 1, các giá trị xác suất gióng hàng từ càng sẽ tiến gần đến giá trị xác suất hoàn hoản (các giá trị nằm trên đường chéo chính được mô tả trong hình 2.1). Trong trường hợp càng lớn thì mức độ gióng hàng từ giữa các từ sẽ càng giảm, xác suất gióng hàng từ sẽ càng nằm xa so với giá trị nằm trên đường chéo chính.



1. Minh họa cho phân phối xác suất gióng hàng từ

Tiếp theo ta sẽ đi nghiên cứu về các phép suy luận (inference) biên (marginal) và các phép đánh giá cải tiến.

2.3.1 Các phép suy luận biên

Ta có thể tính xác suất xảy ra từ thứ trong câu đích:

Từ đó ta có thể tính hậu xác suất xảy ra của các gióng hàng từ bằng cách:

Cuối cùng ta tính được xác suất dịch bằng cách:

2.3.2 Các phép cải tiến

Phép tính ở biểu thức thời gian thực hiện sẽ là . Tuy nhiên nếu ta vận dụng phép chuỗi hình học (Courant, Robbins, 1996), sẽ giúp ta có thể giảm bớt thời gian thực thi.

Phép chuỗi hình học Courant, Robbins, 1996:

Ta sẽ lập ma trận xác suất gióng hàng từ như theo hình 2.1. Ta sẽ phân các thành 2 phần, các giá trị nằm trên đường chéo chính ta đặt là , và các giá trị nằm từ đường chéo trở về dưới ta đặt là . Các giá trị mới này được tính như sau:

Khi thế các giá trị mới này để tính, ta thấy các giá trị này sẽ giảm dần một tỉ lệ khi ta di chuyển các giá trị xác suất lên trên cũng như xuống từ đường chéo chính theo cột. Điều này giúp ta có thể áp dụng biểu thức cho biểu thức .

Biểu thức sẽ được tính lại theo giá trị mới như sau:

Ta có thể thấy rằng nếu ta tính biểu thức theo biểu thức và thì thời gian thực thi sẽ giảm xuống còn .

Để tối ưu hóa giá trị likelihood cho dữ liệu, kỹ thuật này sử dụng giải thuật tối đa hoá-kỳ vọng (Expectation–Maximization, gọi tắt là EM). Giải thuật này gồm 2 phần: giai đoạn tối đa hóa (Maximization step, gọi tắt là M-step (bước M)) và giai đoạn kỳ vọng (Expectation step, gọi tắt là E-step (bước E)). E-step sẽ tính kỳ vọng cho các giá trị log-likelihood dựa trên các dựa đoán và M-step sẽ tính toán các giá trị nhằm tối đa hóa các tham số của log-likelihood dựa trên các giá trị kỳ vọng tính được ở bước E. Các ước lượng tham số ở bước M này sau đó sẽ được cập nhật để tính ở các bước E tiếp theo.

Trong suốt quá trình ở bước M, các giá trị dựa trên sẽ được cập nhật theo một lượng:

Trong có chứa , vì thế ta cần đi tính đạo hàm cho . Nhờ nghiên cứu của Fernandez và cộng sự, 2006:

Mà ta vẫn có thể tính được đạo hàm cho trong thời gian thực thi :

Với .

2.4 Tổng kết chương 2

Trong chương 2 ta đã tìm hiểu dịch máy thống kê chính là quá trình dịch máy dựa trên các mô hình sử dụng các phép thống kê.

Ta đã tìm hiểu được ba mô hình thống kê được giới thiệu là mô hình IBM 1, mô hình IBM 2 và mô hình gióng hàng từ nhanh dựa trên mô hình 2 của IBM.

Cả 3 mô hình đều sử dụng kỹ thuật gióng hàng từ làm trung tâm. Kỹ thuật gióng hàng từ là kỹ thuật dịch từ theo từ, tức là ta sẽ dịch theo từng từ một. Mỗi một từ có thể được dịch thành 1 hoặc nhiều từ hoặc có thể là không thể dịch thành bất cừ từ nào ở ngôn ngữ còn lại.

Mô hình 1 IBM xây dựng các phép phân phối dựa trên số lượng gióng từ cũng như tần số xuất hiện (phép đếm) các từ của cặp câu (ngôn ngữ nguồn, ngôn ngữ đích).

Mô hình 2 IBM kế thừa mô hình 1 IBM, tuy nhiên khác với mô hình 1 IBM thay vì chỉ quan tâm đến số lượng gióng từ (không quan tâm đến thứ tự gióng) thì trong mô hình IBM ta sẽ tập trung chú ý đến xác suất gióng của các từ (thứ tự gióng của các từ).

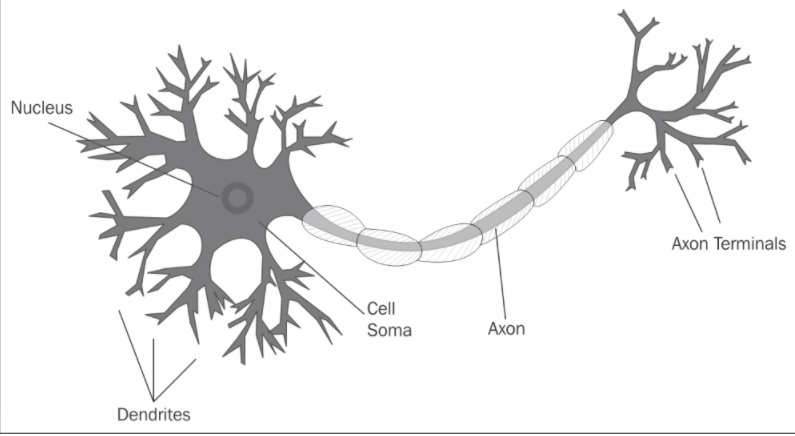
Mô hình gióng hàng từ nhanh dựa trên mô hình 2 IBM chủ yếu dựa trên các công thức cũng như các phép cải tiến nhằm tối ưu hóa thời gian chạy của các tham số.

Kết quả của mô hình gióng hàng từ nhanh này sẽ được sử dụng trong mô hình dịch máy neural sẽ được giới thiệu ở chương tiếp theo.

Chương 3: MÔ HÌNH DỊCH MÁY NƠ-RON

Bộ não của người là một mạng nơ-ron (neural) khép kín gồm nhiều neural liên kết với nhau chạy xuyên suốt nhằm thu nhận thông tin từ thế giới quan xung quanh, dần dần cứ qua từng neural một sẽ hình thành được nên tín hiệu điều khiển nhằm giúp cơ thể có thể biết nên phản ánh như thế nào với các tác động cũng như hoàn cảnh của môi trường.

Một nơ-ron (neural) thông thường gồm 3 phần chính thân tế bào neural (soma), các nhánh neural (dendrite) và đuôi neural (axon).



1. Cấu trúc cơ bản của 1 neural ở người [5]

Theo hình 3.1, ta thấy các dendrite sẽ có nhiệm vụ nhận tín hiệu (thông tin) được truyền tới từ các neural khác. Soma có nhiệm vụ xử lý tín hiệu nhận được từ các dendrite và trả kết quả xử lý cho axon. Cuối cùng axon có nhiệm vụ truyền kết quả đã xử lý xong tới dendrite của neural khác.

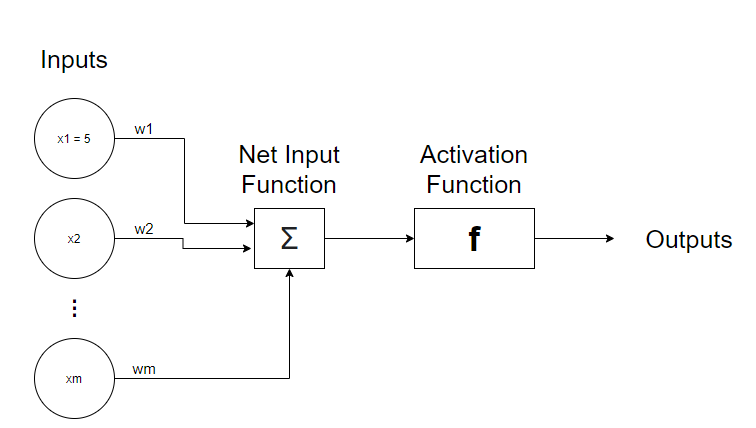
Dựa trên cấu trúc này, con người cũng mong muốn tạo ra một mạng neural nhân tạo có thể thu nhận thông tin, xử lý văn bản ngôn ngữ nguồn và từ đó có thể trả ra kết quả dịch một cách tự động.

3.1 Cấu trúc của 1 mạng nơ-ron cơ bản

3.1.1 Perceptron – mạng neural phi tuyến tính cơ bản

Perceptron là mạng neural nhân tạo đầu tiên được phát triển. Perceptron được công bố bởi Frank Rosenblatt vào năm 1957.

Perceptron là một giải thuật học có giám sát. Giải thuật cho phép học và xử lý đồng thời trên tập dữ liệu.



1. Cấu trúc cơ bản của perceptron

Theo hình 3.2, ta có thể thấy cấu trúc cơ bản của perceptron sẽ gồm 3 thành phần chính: đầu vào (inputs, weight, net input function), hàm xử lý (activation function) và đầu ra (output).

Trong phần đầu vào inputs là tập chứa dữ liệu sẽ được xử lý, weights là tập chứa các giá trị trọng số, net input function là hàm tính tổng các tích giữa dữ liệu và trọng số tương ứng. Net input function sẽ được tính như sau:

Trong đó:

* là số lượng dữ liệu đầu vào.
* là dữ liệu thứ trong tập dữ liệu.
* là trọng số thứ tương ứng với .

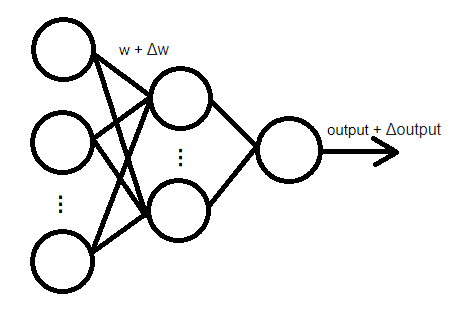
Phần trung tâm của perceptron là hàm xử lý (activation function). Hàm xử lý cơ bản nhất là hàm nhị phân. Hàm này sẽ trả về 2 kết quả duy nhất là 0 và 1. Kết quả là 0 khi kết quả của net input function đạt dưới ngưỡng (threshold) hoặc bằng ngưỡng được cho phép. Kết quả là 1 khi kết quả của net input function đạt trên ngưỡng cho phép.

Như vậy là ta đã tìm hiểu xong cấu trúc cũng như nguyên lý hoạt động cơ bản của perceptron, mạng neural cơ bản đầu tiên được phát triển.

3.1.2 Mạng nơ-ron phi tuyến tính nâng cao

Ở mục 3.1.1, ta đã tìm hiểu được cấu trúc của 1 mạng neural cơ bản hay còn gọi là perceptron. Ở mục 3.1.2, ta sẽ đi tìm hiểu một phiên bản cao cấp hơn của mạng perceptron, ta sẽ gọi là mạng neural phi tuyến tính nâng cao. Ta gọi nâng cao ở đây là bởi vì các mạng neural sử dụng các hàm phi tuyến tính biến thiên liên tục trong các ngưỡng nhất định chứ không giống perceptron chỉ biến thiên 0 và 1.

Ở mục 3.1.1, ta biết rằng output được tính dựa trên , và . Tuy nhiên ta không thể thay đổi vì đây là tập dữ liệu gốc, là yếu tố khách quan. Vì vậy ta chỉ có thể thay đổi và . Vì vậy để có thể đạt được kết quả output mong muốn, ta cần có 1 giải thuật giúp chúng ta có thể thay đổi và . Ta có thể hình dung của việc thay đổi qua minh họa sau:



1. Minh họa về việc thay đổi

Qua minh họa ở hình 3.3, ta thấy rằng khi trọng số thay đổi một lượng thì kết quả cũng sẽ thay đổi một lượng tương ứng.

Quá trình học bản chất chính là quá trình ta đi tìm lượng và sao cho ta thu được kết quả đầu ra là như mong muốn.

Ngoài ra còn 1 vấn đề khác nằm ở hàm activation của perceptron. Hàm activation của perceptron chỉ trả về kết về là 0 và 1. Kết quả này chuyển đổi quá nhanh khiến ta không thể điều chỉnh các trọng số kịp thời để kết quả đầu ra gần khớp với kết quả mà ta mong muốn. Vì vậy ta cần một hàm activation khác giúp ta có thể điều chỉnh kết quả đầu ra một cách từ từ nhằm giúp ta có thể điều chỉnh toàn bộ hệ thống sao cho sau quá trình huấn luyện đạt được kết quả mong muốn.

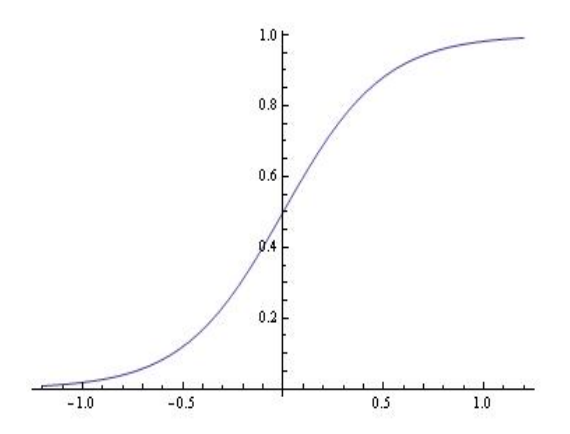
Một trong những hàm activation phổ biến dùng để thay thế perceptron là hàm sigmoid có dạng như sau:

Hay ta có thể viết lại thành:

Với:

Về mặt toán học, ta có:

Như vậy hàm sigmoid sẽ có đồ thị như sau:



1. Đồ thị của hàm sigmoid [6]

Như ta thấy ở đồ thị hình 3.4, giá trị của hàm sigmoid sẽ thay đổi một cách chậm rãi từ giá trị 0 cho đến giá trị 1 khi ta cho dữ liệu chạy từ ( cho đến ). Điều này đồng nghĩa với việc khi ta thay đổi một lượng nhỏ giá trị hay một lượng nhỏ giá trị hay thay đổi cả hai thì hàm số sẽ thay đổi một lượng nhỏ giá trị . Điều này sẽ giúp ta điều chỉnh sao cho chương trình ra được kết quả dễ dàng hơn, cũng như gần chính xác hơn với giá trị mà ta mong muốn.

Ta đặt kết quả đầu ra của hàm activation là trong đó .

Ta đặt lượng giá trị đầu ra bị thay đổi là

Lượng được tính như sau:

Áp dụng chuỗi Taylor, ta có thể tính được như sau:

Như vậy là ta đã tìm hiểu xong cách thức hoạt động của mạng sigmoid. Đồng thời ta cũng biết được ý tưởng ban đầu của việc học của 1 mạng neural.

3.1.3 Giải thuật lan truyền ngược (Backpropagation algorithm)

Backpropagation đã được giới thiệu vào thập niên 1970, tuy nhiên mãi tới năm 1986 mới được ra ấn phẩm giấy chính thức bởi David Rumelhart, Geoffrey Hinton và Ronald Williams.

Ta sẽ đặt các trọng số w là:

Với:

* là layer (lớp) hiện tại.
* là node thứ của layer hiện tại .
* là node thứ của layer phía trước .

Tương tự ta sẽ đặt với phần bù đại số b là:

Và kết quả sau khi tính xong của hàm activation là:

Áp dụng biểu thức , ta có:

Từ đây ta tính được giá trị như sau:

Với K tổng số node mà lớp hiện đang có.

Như vậy ở mỗi lớp, ta hình thành được vector chứa các giá trị kết quả mà hàm activation tính được ở các node của layer .

Như vậy ta có thể rút gọn biểu thức về thành:

Với:

* là vector trọng số chứa tất cả các trọng số của layer hiện tại.
* là vector kết quả chứa tất cả các kết quả tính được của hàm activation của layer phía trước .
* là vector phần bù đại số chứa tất cả các giá trị phần bù đại số của layer hiện tại.

Ta có thể rút gọn biểu thức bằng cách đặt:

Biểu thức được rút thành:

Ta gọi là giá trị đầu vào có trọng số (weighted input) do nó là giá trị đầu vào và nó có chứa trọng số và phần bù đại số kèm theo. Ta sẽ sử dụng giá trị này ở các phần sau của chương.

Ta gọi là sai số được sinh ra từ tính toán của neural (node) thứ j của lớp thứ .

Backpropagation sẽ cung cấp cho chúng ta một quy trình (mỗi chuỗi các thủ tục) giúp chúng ta có thể tính được và từ đó giúp chúng ta tạo được mối liên hệ giữa với .

Tưởng tượng rằng, trước khi hàm activation của 1 neural hoạt động, đầu vào của neural này vô tình không phải là mà ta sẽ thay bằng . Ta thêm nhằm giúp làm giảm sai số do tính toán của layer phía trước. Nếu là một giá trị lớn (một sai số lớn) thì lượng sẽ giúp ta làm giảm bớt lượng sai số đi. Tuy nhiên, nếu là một giá trị nhỏ (sai số rất ít) thì ta không cần lượng này nữa, ta có thể phán định rằng phép tính của neural là hiệu quả (tối ưu). Vì vậy sẽ giúp chúng ta biết được rằng phép tính của neural có tối ưu hay không.

Do đó ta có thể đặt:

Là sai số xuất phát từ tính toán của neural thứ j trong layer thứ .

Biểu thức tính sai số ở lớp cuối cùng (lớp output) được tính như sau:

Trong đó:

* cho biết tốc độ thay đổi của sai số của kết quả mà hàm activation thứ j tính được.
* cho biết tốc độ thay đổi của kết quả của hàm activation với đầu vào là .

Ta viết biểu thức về thành:

Trong đó:

* là vector chứa các giá trị đạo hàm của hàm loss theo .

Khai triển đạo hàm của hàm loss, ta được:

Với biểu thức sẽ giúp ta dễ dàng tính toán hơn.

Ta có thể tính sai số ở một lớp ẩn bất kỳ bằng biểu thức:

Trong đó:

* ma trận nghịch đảo chứa các trọng số của lớp thứ .
* là kết quả mức độ sai số mà ta đã tính được ở lớp trước.
* là phép đạo hàm hàm activation có biến thế bằng giá trị .

Do đây là lớp ẩn, là lớp nằm ở giữa không phải là lớp cuối nên kết quả của các activation tính được không phải là kết quả cuối cùng vì vậy ta sẽ không sử dụng hàm loss, thay vào đó giá trị loss này sẽ kế thừa từ lớp phía trước . Ở các lớp ẩn , chỉ có 1 yếu tố ảnh hưởng đến sự sai lệch của chương trình đó chính là kết quả tính toán của các hàm activation. Chính vì thế ở các lớp ẩn , ta chỉ đi tính đạo hàm của activation và dùng giá trị để thế vào và tính ra kết quả.

Để tính xem mức độ ảnh hưởng các giá trị bias (phần bù đại số) ảnh hưởng như thế nào trong sự sai lệch kết quả, ta sẽ sử dụng biểu thức:

Ta tính mức độ sai số của các trọng số bằng biểu thức:

Trong đó:

* là kết quả của hàm activation của neural thứ k ở lớp phía trước .
* là kết quả mức độ sai số ở lớp hiện tại.

Vị trí của các trọng số đó chính là nằm giữa hàm activation của lớp hiện tại với kết quả đầu ra của activation ở lớp phía trước. Vì vậy mức độ sai số của trọng số sẽ được tính dựa trên 2 giá trị này.

Trong mục trước ta đã tìm hiểu xong các biểu thức chính của backpropagation theo khuynh hướng tích hadamard. Tuy nhiên ta có thể biểu diễn các biểu thức này theo khuynh hướng dễ hiểu hơn.

Ta có thể biểu diễn biểu thức thành:

Trong đó:

* là ma trận vuông có đường chéo chính là các giá trị . Các phần tử còn lại sẽ là giá trị 0.

Ta có thể biểu diễn biểu thức thành:

Ta gộp 2 biểu thức và , ta sẽ được:

Với cách làm này sẽ giúp ta dễ hiểu hơn.

Ta sẽ chứng minh các biểu thức , , và .

Đầu tiên, ta bắt đầu xem xét mức độ sai số giữa kết quả đầu ra của chương trình với kết quả mà ta mong muốn thông qua hàm loss, bằng biểu thức:

Áp dụng chain rule, ta tiếp tục xem mức độ ảnh hưởng của hàm activation dẫn tới sự sai khác.

Ta rút gọn biểu thức thành:

Vì và là 2 cách biểu diễn tương nhau, nên ta viết gọn thành:

Do backpropagation là giải thuật truyền ngược, nên ta sẽ bắt đầu với việc tiếp nhận kết quả tính toán mức độ sai số mà ta đã tính được ở lớp :

Sử dụng chain rule ta được:

Mà , suy ra:

Mà:

Do đó:

Thế vào :

Rút gọn ta sẽ được biểu thức .

Tiếp theo ta sẽ chứng minh .

Để chứng minh , đầu tiên ta sẽ chứng minh ở lớp cuối trước. Ta bắt đầu chứng minh lại bằng biểu thức:

Ta sử dụng chain rule:

Mà , ta được:

Áp dụng tương tự với biểu thức , , ta được:

Tiếp tục ta chứng minh với lớp ẩn thứ . Ta bắt đầu bằng biểu thức:

Áp dụng chain rule:

Mà , ta được:

Áp dụng tương tự với , , , , ta được:

Từ và , ta suy ra được: (đpcm)

Đầu tiên ta sẽ chứng minh ở lớp cuối trước. Ta bắt đầu chứng minh lại bằng biểu thức:

Ta sử dụng chain rule:

Mà , ta được:

Áp dụng tương tự với biểu thức , , ta được:

Tiếp tục ta chứng minh với lớp ẩn thứ . Ta bắt đầu bằng biểu thức:

Áp dụng chain rule:

Mà , ta được:

Áp dụng tương tự với , , , , ta được:

Từ và , ta suy ra được: (đpcm)

Cuối cùng ta đúc kết nên giải thuật lan truyền ngược theo trình tự sau:

Bước 1: Chuẩn bị dữ liệu input cho lớp neural đầu tiên. Và chạy hàm activation ở lớp đầu tiên

Bước 2: Với mỗi lớp , từ lớp thứ 2 cho đến lớp output (lớp L). Ta thực hiện tính:

Bước 3: Tính mức độ sai số của lớp output:

Bước 4: Thực hiện backpropagation cho từng lớp ẩn (với mỗi mỗi lớp từ trở về lớp ), ta tính:

Bước 5: Tính gradient sai số của trọng số và bias:

Bước 6: Cập nhật giá trị mới cho trọng số và bias (nếu sử dụng Stochastic gradient descent):

Nếu không sử dụng Stochastic gradient descent, ta thực hiện việc cập nhật như sau:

3.2 Mô hình dịch máy nơ-ron sequence-to-sequence (encoder-decoder) cơ bản

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.

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 Việt sang tiếng Anh, encoder sẽ chuyển tập dữ liệu tiếng Việt 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 Anh tương ứng ở đầu ra.

Trong mục 3.1 ta đã tìm hiểu cấu tạo, nguyên lý hoạt động cũng như cách mà 1 mạng neural có thể học và chỉnh sửa.

Trong mục này ta sẽ tập trung vào việc phân tích mô hình dịch máy neural bằng cách sử dụng RNN cho encoder và decoder.

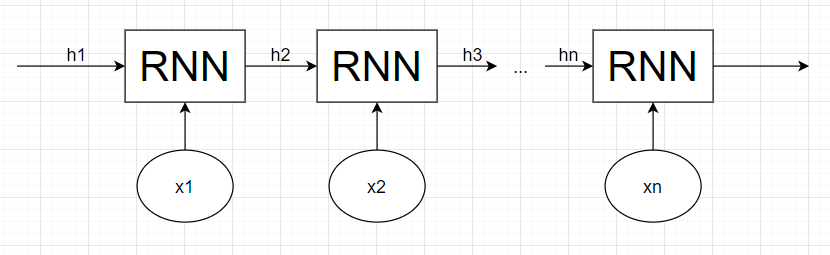
3.2.1 Giới thiệu về dịch máy neural

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.

3.2.2 Giới thiệu về Recurrent Neural Network (RNN)

Ta có tập dữ liệu đầu vào:



1. Sơ lược cấu tạo hoạt động của RNN

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.
* còn có tên gọi khác là

Kết quả của đầu ra được chọn theo giá trị lớn nhất của softmax được gọi là Greedy search.

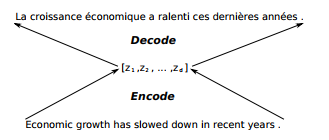
Từ đó ta tính được phân bố xác suất cho toàn tập:

Trên hình 3.5 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).

3.2.3 Mô hình Encoder – Decoder

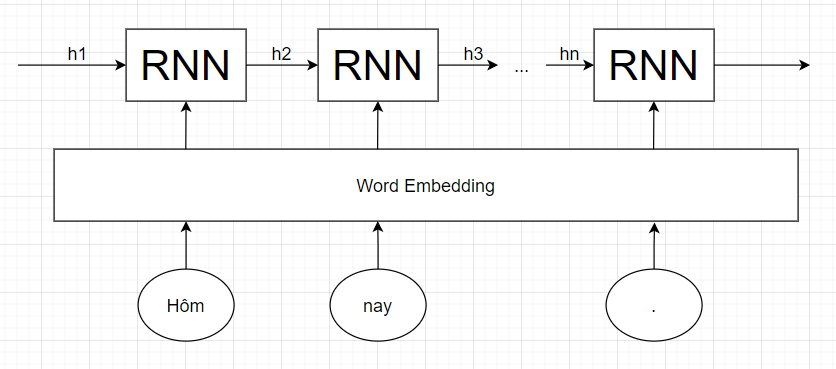
Nhiệm vụ của dịch thuật có thể được hiểu theo quan điểm của học máy đó là việc học các phân phối có điều kiện của một câu đích (kết quả sau quá trình dịch) dựa trên một câu nguồn (câu muốn dịch).

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 3.6). 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 3.6). 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. Cấu trúc tổng quát của mô hình encoder-decoder [7]

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



1. Cấu trúc RNN encoder

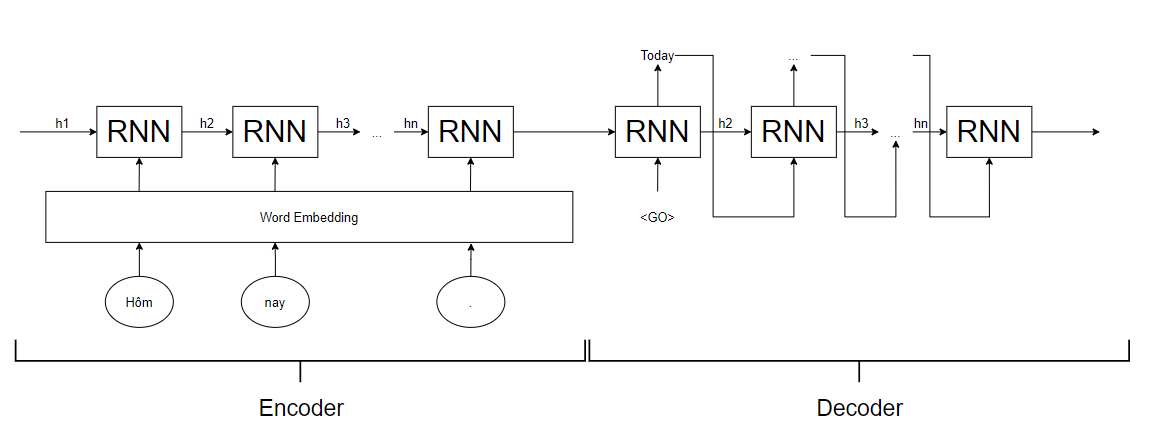
Đầu vào của neural đầu tiên chỉ có dữ liệu gốc, không có giá trị ẩn (hidden unit bằng 0).

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

Tập vector có độ dài cố định này được gọi là vector ngữ cảnh (context vector).

Một bộ decoder đơn thuần sẽ chỉ sử dụng tập kết quả đầu ra của encoder (context vector) để làm đầu vào cho mình. Context vector được gọi như vậy bởi vì nó mã hóa ngữ cảnh từ toàn bộ tập dữ liệu ban đầu. Vectơ ngữ cảnh này được sử dụng làm trạng thái ẩn (hidden unit) đầu tiên 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 3.8). Chuỗi decoder sẽ bắt đầu bằng chuỗi <GO> (hình 3.8) (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

3.2.4 Cơ chế Attention

Như ta đã biết, chức năng của encoder đó chính là nén toàn bộ dữ liệu đầu vào (đã được embedding) thành 1 tập vector có độ dài cố định. Với các câu ngắn (hay dữ liệu đầu vào ít) thì chuyện này rất dễ. Tuy nhiên với các câu đầu vào dài (hay dữ liệu đầu vào nhiều) thì rất khó để có thể nén hết được thông tin, có thể dẫn đến mất mát, gây hiệu ứng cổ chai.

Tương tự với decoder, việc tập vector này có độ dài cố định cũng gây khó khăn cho decoder. Decoder phải tìm kiếm các kết quả cho đầu ra dựa trên một tập dữ liệu đầu vào hữu hạn.

Vì thế attention được sinh ra nhằm đánh giá việc cặp từ này được bao nhiêu “điểm”, “điểm” của cặp càng cao thì việc gióng hàng từ của cặp này càng tốt. Với cơ chế tập trung vào quan hệ của các cặp này sẽ giúp giảm bớt khó khăn cho encoder và decoder trong việc tìm kiếm thông tin.

Cơ chế attention lần đầu được giới thiệu bởi Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, 2015 trong “NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE”.

Các context vector thay vì dùng kết quả hidden unit của encoder thì nay được tính bằng:

Trong đó:

* là bước thứ thực hiện.
* là vector ngữ cảnh tại bước , context vector dùng để sinh từ thứ trong câu đích.
* là tập chứa các hidden state của encoder.
* là gióng hàng từ giữa .

được tính bởi:

Trong đó:

Với:

* được gọi là mô hình gióng hàng từ (alignment model)
* là hidden unit của decoder ở bước trước.
* là hidden unit của encoder.
* là hàm tính điểm. Hàm tính điểm có thể là các hàm activation phi tuyến tính như: sigmoid, tanh (Bahdanau, 2015),… hoặc chỉ là phép tích giữa tham số (Luong, 2015) hoặc là hàm (Luong, 2015),…

Từ đây hidden unit cho bước của decoder được tính bởi:

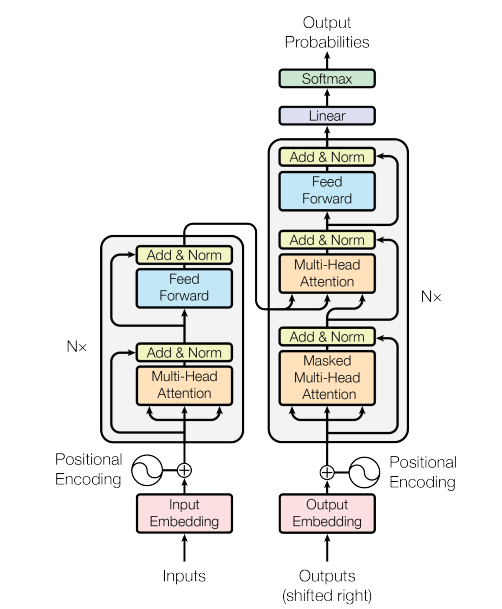
Với cách làm này sẽ tạo được cho decoder cơ chế tập trung quan sát giữa lớp phía trước của nó với từng phần hidden unit của encoder. Điều này giúp decoder có thể quan sát không những trong phạm vi của nó mà còn giúp decoder có thể quyết định dựa trên các quan sát từng bộ phận của câu ở ngôn ngữ nguồn dựa trên các điểm đánh giá. Decoder đã có thể dễ dàng chọn lọc thông tin, điều này giúp giảm tải gánh nặng cho encoder khi phải nén, mã hóa tất cả thông tin vào một vector có độ dài cố định.

3.3 Mô hình dịch máy transformer

Transformer là một mô hình dựa trên cấu trúc mô hình sequence-to-sequence (encoder-decoder) và cơ chế attention (self-attention).

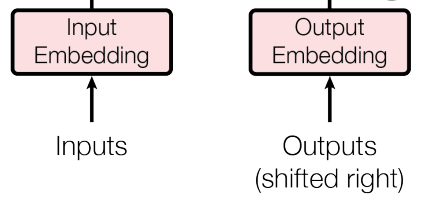
Encoder là một chồng mã hóa gồm có lớp đồng nhất. Mỗi lớp gồm có 2 lớp con. Lớp con thứ nhất chứa cơ chế multi-head self-attention và lớp con thứ hai chứa mạng position-wise feed-forward. Giữa các lớp con là các chuẩn hóa lớp (layer normalization, viết tắt là LayerNorm). Đầu ra của các lớp con sẽ là , với là kết quả của lớp con hiện tại. Các vector dữ liệu, embedding, kết quả của các lớp con,… trong encoder đều có số chiều .

Decoder cũng là một chồng giải mã gồm có lớp đồng nhất. Mỗi lớp cũng có 2 lớp con giống với encoder, tuy nhiên decoder có thêm một lớp multi-head self-attention được dùng để quan sát thêm các kết quả của encoder. Giống với encoder giữa các lớp con đều có các LayerNorm. Để đảm bảo rằng dự đoán ở bước i chỉ phụ thuộc vào các bước trước đó, ta sẽ thêm “mask” nhằm đánh dấu các vị trí.



1. Tổng quan cấu trúc mô hình transformer [9]

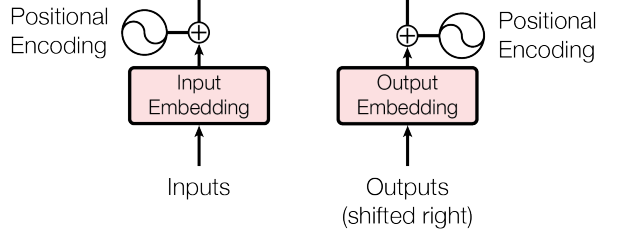
Ở phần đầu của mô hình:



1. Đầu vào encoder và decoder của mô hình [9]

Đầu vào của encoder là dữ liệu của câu ở ngôn ngữ nguồn, còn đầu vào của decoder cũng là dữ liệu tuy nhiên là dữ liệu của câu ở ngôn ngữ đích và được duyệt chạy lần lượt theo chiều từ trái qua phải.

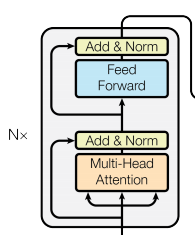
Sau khi nạp, các dữ liệu này sẽ được embedding để tạo thành các vector nhằm đưa vào mô hình để thực hiện (hình 3.10). Một số tả về embedding và softmax được chú thích trong mục 3.3.1.



1. Positional Encoding [9]

Vì mô hình transformer không sử dụng các mạng RNN hay CNN nên đầu vào của encoder sẽ không chạy tuần tự vì thế không thể biết vị trí của các từ. Chính vì thế phải có cơ chế để chèn các vị trí tương đối vào dữ liệu. Cơ chế này được gọi là Positional Encoding (gọi tắt là PE) (hình 3.11). PE sẽ được nói rõ hơn ở mục 3.3.2 của tài liệu.

Cấu trúc của 1 lớp encoder có dạng như sau:



1. Cấu trúc của 1 lớp trong encoder trong transformer [9]

Nhiệm vụ của encoder là ánh xạ tất cả chuỗi dữ liệu đầu vào thành một biểu diễn liên tục có thể chứa được toàn bộ thông tin học được từ chuỗi dữ liệu đầu vào này.

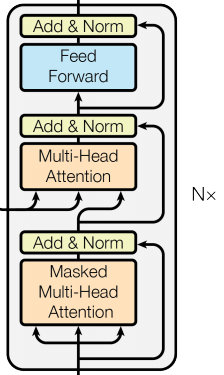
Một lớp của encoder gồm 2 lớp con là Multi-head attention (multi-head self-attention) và một mạng Feed-forward (position-wise feed-forward). Multi-head attention sẽ được nói rõ hơn ở mục 3.3.3 và mạng feed-forward sẽ được nói rõ hơn ở mục 3.3.4.

Giữa các lớp con này là các chuẩn hóa lớp. Đầu ra của LayerNorm ứng với Multi-head attention sẽ là . Còn đầu ra của LayerNorm ứng với mạng feed-forward (Feed Forward Network, gọi tắt là FFN) sẽ là .

Kết quả sẽ được dùng làm đầu vào cho FFN. FFN sẽ được nói rõ hơn trong mục 3.3.4 của tài liệu.

Như vậy, encoder sẽ là tập hợp gồm có 6 lớp và mỗi lớp sẽ cấu trúc giống như hình 3.12.

Cấu trúc 1 lớp của decoder trong transformer như sau:



1. Cấu trúc 1 lớp của decoder trong transformer [9]

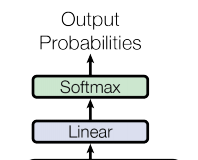
Lớp con đầu tiên của decoder chính là Masked Multi-head attention. Masked multi-head attention gần giống với Multi-head attention chỉ khác là có thêm cơ chế mask. Cơ chế mask giúp decoder không tập trung vào các từ chưa được đọc, chỉ tập trung vào các từ đã được thực thi và dự đoán ở các bước phía trước. Cơ chế mask sẽ được nói rõ hơn ở mục 3.3.3.

Lớp con ở giữa của decoder là sự kết hợp giữa kết quả của Masked multi-head attention của decoder và kết quả trả ra của encoder. Attention của lớp này hoạt động bình thường theo đúng cơ chế multi-head attention bình thường.

Giống với encoder, cuối lớp decoder sẽ là lớp FFN.

Với 3 lớp con này sẽ tạo nên 1 lớp cho decoder. Ta xếp chồng 6 lớp như vậy sẽ tạo thành bộ decoder.

Sau khi kết thúc 6 lớp của decoder sẽ là phép chiếu tuyến tính và softmax để tạo ra kết quả dự đoán (hình 3.14).



1. Tạo giá trị dự đoán từ decoder [9]

3.3.1 Embedding và softmax

Giống với các mô hình sequence-to-sequence khác, ta có thể sử dụng các giải embedding phổ biến (count-based, one-hot vector, word2vec,…) để chuyển dữ liệu thành các vector dạng số có số chiều .

Trước khi xuất kết quả, giống với các mô hình sequence-to-sequence khác, transformer decoder cũng dùng các hàm biến đổi tuyến tính cũng như softmax để chuyển đổi kết quả đầu ra của mình thành các giá trị dự đoán.

Embedding và hàm biến đổi tuyến tính của decoder đều có cùng ma trận trọng số.

Các ma trận trọng số trong các lớp embedding được nhân thêm một lượng .

3.3.2 Positional Encoding

Vì mô hình transformer không sử dụng các mạng RNN hay CNN nên đầu vào của encoder sẽ không chạy tuần tự vì thế không thể biết vị trí của các từ. Chính vì thế phải có cơ chế để chèn các vị trí tương đối vào dữ liệu. Cơ chế này được gọi là Positional Encoding (gọi tắt là PE).

PE có cùng số chiều với embedding vì thế mà ta có thể cộng 2 giá trị này lại với nhau (hình 3.11).

Có nhiều cách thực hiện cho PE. Mô hình transformer của nhóm tác giả đã lựa chọn, có PE được tính như sau:

Trong đó:

* là vị trí của từ.
* là số chiều.

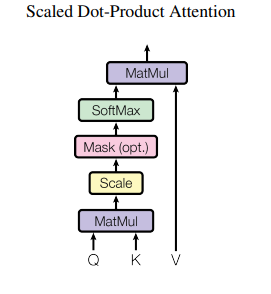
Mỗi chiều của PE ứng với 1 đường hình sin. Tạo thành một dãy cấp số nhân từ . Nhóm tác giả chọn cách tính PE này vì nhóm tác giả cho rằng mô hình sẽ dễ dàng học các giá trị tương đối này bởi với phần bù (offset) nào, thì cũng đều có thể biểu diễn dưới dạn hàm tuyến tính . Các vị trí lẻ sẽ dùng , còn các vị trí chẵn sẽ dùng .

3.3.3 Cơ chế Multi-head Attention

Multi-head attention dựa theo cơ chế self-attention.

Cơ chế self-attention cho phép một từ có thể tập trung quan sát với các từ còn lại trong câu. Một từ trong câu sẽ được biểu diễn thành tập của 3 giá trị (key (K), query (Q), value (V)). Trong đó V đại diện cho các giá trị đầu vào, K đại diện cho các khóa tìm kiếm của V, còn Q đại diện cho cách truy vấn V theo K.

Key (K) và Query (Q), vector sẽ có số chiều là . Còn Value (V) sẽ có số chiều là .



1. Cơ chế self-attention của transformer [9]

Giá trị đầu ra của self-attention sẽ được tính bằng:

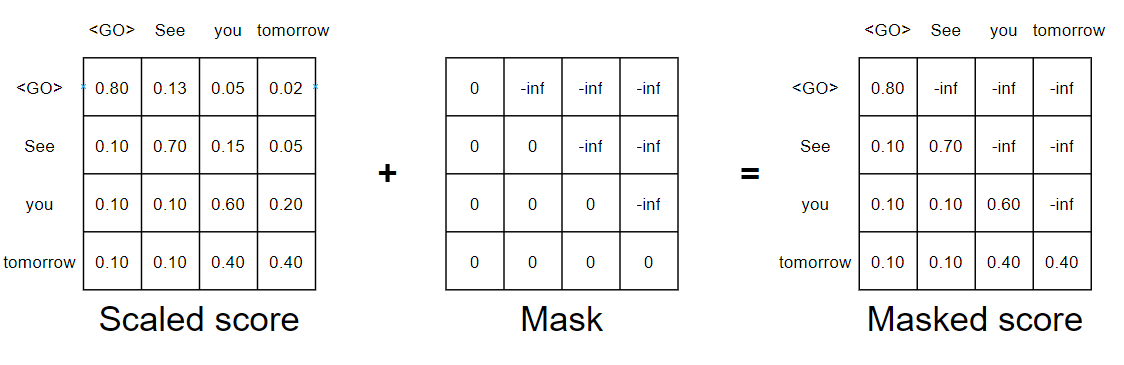
Hình 3.15 mô tả trình tự thực hiện của biểu thức .

chính là bảng điểm đánh giá mức độ cần quan sát của các từ trong câu.

Q và K càng có số chiều càng lớn sẽ dẫn tới kết quả của hàm softmax càng nhỏ dẫn tới tình trạng gradient vanishing, vì thế ta sẽ chia ma trận tích cho để tránh tình trạng này.

Tuy nhiên self-attention của decoder có chút khác biệt. Nó khác so với encoder ở chỗ self-attention của decoder có thêm mask.

Ví dụ, ta có cặp câu (“Hẹn gặp bạn ngày mai”, “See you tomorrow”), đầu vào của decoder đang là “you” và ta mong muốn ở bước dịch này sẽ cho ra kết quả “tomorrow” cho phiên dịch “ngày mai”. Vì thế từ “tomorrow” chưa xuất hiện trong ngữ cảnh này, do đó từ “you” chỉ có thể quan sát chính nó hoặc quan sát với từ “See”. Mask sẽ có chức năng giúp điểm đánh giá giữa “you” và “tomorrow” trở nên rất thấp để mô hình không tập trung vào cặp này nữa. Ta có thể xem ví dụ minh họa qua hình 3.16 sau:



1. Minh họa cho cơ chế mask

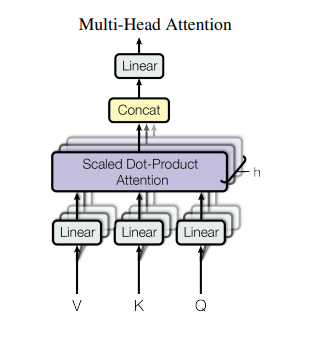
Cơ chế mask sẽ thực hiện sau khi tính . Ta sẽ lấy ma trận kết quả cộng với ma trận mask rồi từ đó mới đem đi softmax. Các giá trị điểm rất nhỏ này (-inf) sau khi softmax sẽ có giá trị gần bằng 0. Với các giá trị điểm gần bằng 0 này mô hình sẽ hầu như không tập trung quan sát vào nó nữa, thỏa mãn được mục tiêu mà ta yêu cầu cho decoder.

Multi-head attention là self-attention, tuy nhiên khác ở chỗ thay vì thực hiện tuần tự cho từng dữ liệu đầu vào thì ta có thể thực hiện cho dữ liệu đầu vào. Bằng cách sử dụng các phép chiếu tuyến tính và các ma trận trọng số ta có thể đưa về dạng self-attention (hình 3.15):

Trong đó:

Với:

* là phép nối vector.
* là ma trận tham số dùng để chiếu cho Q.
* là ma trận tham số dùng để chiếu cho K.
* là ma trận tham số dùng để chiếu cho V.
* là ma trận tham số dùng để chiếu cho .



1. Cơ chế multi-head attention [9]

3.3.4 Mạng Position-wise Feed-Forward

Mạng position-wise feed-forward (FFN) là một tập hợp gồm 2 phép tuyến tính nằm giữa là 1 lớp ReLU. Mạng FFN được tính như sau:

Vector đầu vào và đầu ra của FFN đều đảm bảo có số chiều , tuy nhiên số chiều của FFN .

3.4 Kết hợp kĩ thuật gióng hàng từ vào mô hình transformer

Theo như bài báo Jointly Learning to Align and Translate with Transformer Models của nhóm tác giả Apple [10], khi ta lập thành ma trận thì ta có thể xem kết quả của transformer là các gióng hàng từ (alignment).

Ta có thể huấn luyện các trọng số dựa trên sai khác (phân kỳ Kullback-Leibler) giữa alignment sinh được từ của transformer và kết quả alignment sinh được từ các mô hình mà ta đã trình bày ở chương 2. Để tối thiểu hóa (minimized) phân kỳ Kullback-Leibler, ta sẽ tiến hành tối thiểu hóa hàm mất mát cross-entropy sau:

Với:

* là hàm mất mát cross-entropy, biểu thị mất mát trong gióng hàng từ.
* là ma trận gióng hàng từ sinh được từ mô hình ở chương 2. khi từ thứ ở ngôn ngữ nguồn gióng với từ thứ ở ngôn ngữ đích. Chuẩn hóa ma trận ta thu được ma trận .
* là ma trận thu được từ bất kỳ kết hợp giữa encoder (V,K) và decoder (Q).

Ngoài sự sai khác gióng hàng từ, còn có sự mất mát trong quá trình dịch:

Kết hợp và ta ra được hàm mất mát cần dùng:

3.5 Tổng kết chương 3

Sau chương 3, ta đã tìm hiểu xong cấu tạo, nguyên lý hoạt động của transformer. Cũng như cách dùng alignment của chương 2 để đào tạo và huấn luyện cho attention của transformer.

Giống với các mô hình sequence-to-sequence, transformer cũng có 2 phần là encoder và decoder.

Encoder là một chồng gồm 6 lớp, mỗi lớp gồm có 2 lớp con là Multi-head attention và lớp position-wise feed-forward.

Decoder là một chồng gồm 6 lớp, mỗi lớp gồm có 3 lớp con là lớp Masked multi-head attention, lớp multi-head attention dùng kể kết hợp giữa encoder-decoder và lớp position-wise feed-forward (FFN).

Lớp multi-head attention dựa trên cơ chế self-attention đánh giá điểm giúp mô hình có thể tập trung quan sát thông tin trong cùng một câu một cách dễ dàng. Thêm vào đó multi-head attention là phiên bản mở rộng của self-attention vì nó có thể tập trung và đánh giá nhiều câu đồng thời cùng một lúc nhờ các phép chiếu tuyến tính.

Cơ chế mask trong multi-head attention giúp decoder không “ăn gian” trong việc tập trung vào các từ phía trước chưa được xét.

Lớp FFN có thể được xem như một lớp neural cơ bản vì có nhân là hàm ReLU activation.

Cuối cùng là ta biết được cách để huấn luyện cho attention dựa trên kỹ thuật áp dụng alignment của chương 2 với attention của transformer. Ta tối ưu hóa attention của mô hình bằng cách làm tối thiểu hóa sự mất mát kết hợp từ sự mất mát chất lượng dịch với sự sai khác gióng hàng từ của mô hình ở chương 2 với attention của transformer (Kullback- Leibler, ).

Chương 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ

Trong chương này ta sẽ áp dụng các cơ sở lý thuyết ở chương 2 và chương 3 để tiến hành thực nghiệm cho quá trình từ tiếng Việt sang tiếng Anh. Sau khi thực nghiệm ta sẽ tiến hành lấy kết quả và thực hiện đánh giá.

* 1. Dữ liệu

Dữ liệu được thực hiện trong đề tài được lấy từ nguồn hội thảo quốc tế The International Workshop on Spoken Language Translation 2015 (IWSLT 2015).

Bộ dữ liệu gồm có 2 loại dữ liệu: loại thứ nhất dùng để huấn luyện (train), loại thứ hai dùng để kiểm thử (test).

Dữ liệu dùng để huấn luyện gồm có 2 file: “train.en” chứa các câu văn viết bằng tiếng Anh và “train.vi” chứa các câu văn viết bằng tiếng Việt. Mỗi file đều chứa hơn 133 ngàn câu văn bản. Mỗi dòng tiếng Việt của “train.vi” ứng với câu dịch tiếng Anh tương ứng với mỗi dòng của “train.en”.

Dữ liệu để test gồm 2 bộ: bộ thứ nhất tạm gọi là bộ năm 2012 và bộ thứ hai tạm gọi là bộ năm 2013. Bộ năm 2012 gồm 2 file “tst2012.en” và “tst2013.vi” đều chứa các cặp câu Anh-Việt (hơn 1500 cặp câu Anh-Việt). Tương tự bộ năm 2013 cũng có 2 file “tst2013.en” và “tst2013.vn” cũng đều chứa các cặp câu Anh-Việt (hơn 1200 cặp câu Anh-Việt).

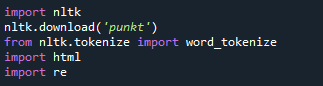
Tất cả đều là dữ liệu văn bản dạng thô, có lẫn cả các thực thể html, cũng như các ký tự văn bản bình thường.

* 1. Xử lý tiền dữ liệu

Vì dữ liệu của ta vẫn còn các thực thể html vì thế ta sẽ tiến hành đi lọc bớt các thực thể này.

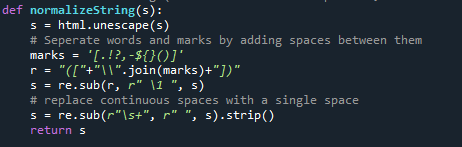
Ngoài ra ta sẽ tiến tách từ (word tokenize) cho dữ liệu.

Đầu tiên ta tiến hành chuẩn bị thư viện cho chương trình:



1. Chuẩn bị thư viện để xử lý tiền dữ liệu

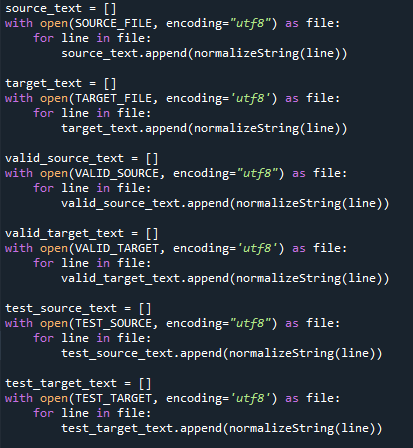
Tiếp theo ta sẽ viết hàm để xử lý lọc bớt các thực thể html:



1. Hàm lọc bớt thực thể html

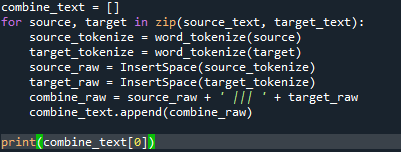
Trong hình 4.2, hàm html.unescape được dùng để lọc bớt các thực thể html. Các đoạn mã còn lại của hàm được dùng để tạo khoảng trắng giữa các dấu câu với các từ.

Sau khi xây dựng hàm làm sạch dữ liệu ta sẽ tiến hành đọc dữ liệu, ta sẽ vừa tiến hành đọc dữ liệu vừa làm sạch dữ liệu nhờ vào hàm ở hình 4.2.



1. Tiến hành đọc và làm sạch dữ liệu

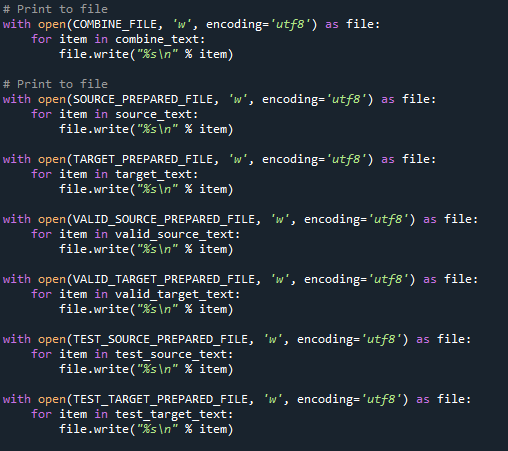
Sau khi làm sạch dữ liệu ta sẽ tiến hành tách từ (tokenize) bằng word\_tokenize của thư viện nltk.tokenize:



1. Tách từ và chuẩn bị dữ liệu cho mô hình gióng hàng từ

Trong hình 4.4, ta còn tiến hành gộp câu nguồn và câu đích về dạng “<Câu nguồn> ||| <Câu đích>” để tiến hành đem đi thực hiện mô hình gióng hàng từ nhanh ở mục 4.3 (lý thuyết được trình bày ở mục 2.3).

Sau khi hoàn tất ta tiến hành in kết quả ra file để thực hiện cho các mục sau này.



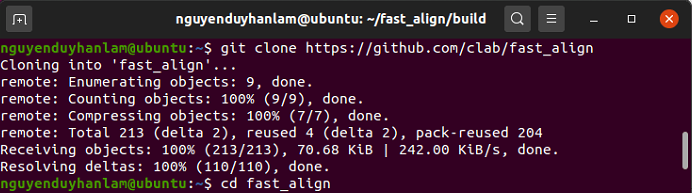
1. Xuất kết quả quá trình tiền xử lý ra file
   1. Thực hiện gióng hàng từ bằng công cụ hỗ trợ fast\_align

Fast\_align là công cụ nhằm thực hiện gióng hàng từ dựa trên mô hình gióng hàng từ nhanh ở mục 2.3.

Công cụ fast\_align được chính nhóm tác giả phát triển, là phần mềm mã nguồn mở và được đính trong bài báo A Simple, Fast, and Effective Reparameterization of IBM Model 2.

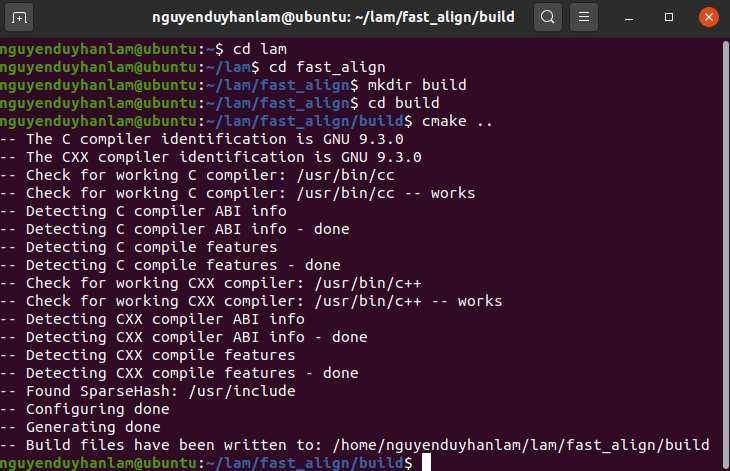
Phần này tài liệu sẽ trình bày cách thực hiện chạy công cụ fast\_align.

Đầu tiên ta tải công cụ từ github bằng lệnh:

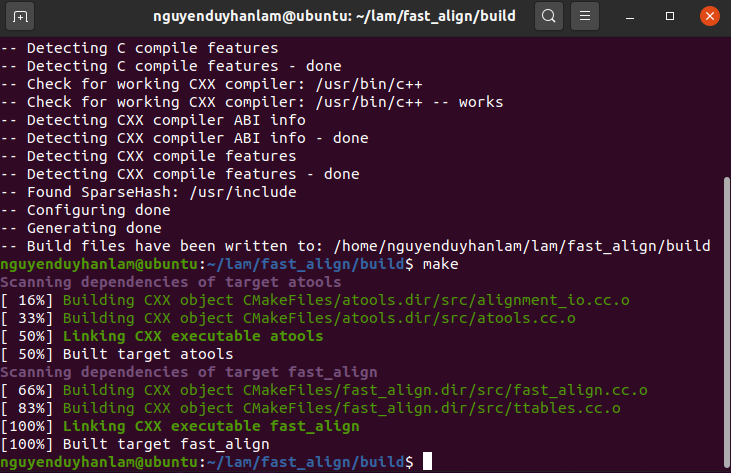


1. Tải phần mềm fast\_align bằng github

Tiến hành cài đặt phần mềm:

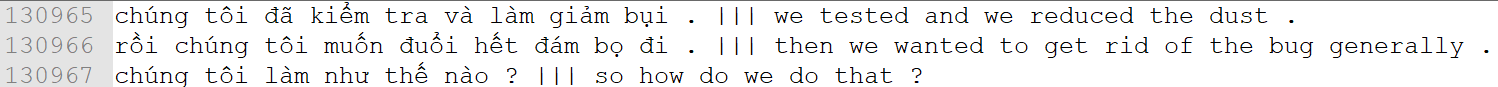


1. Biên dịch chương trình



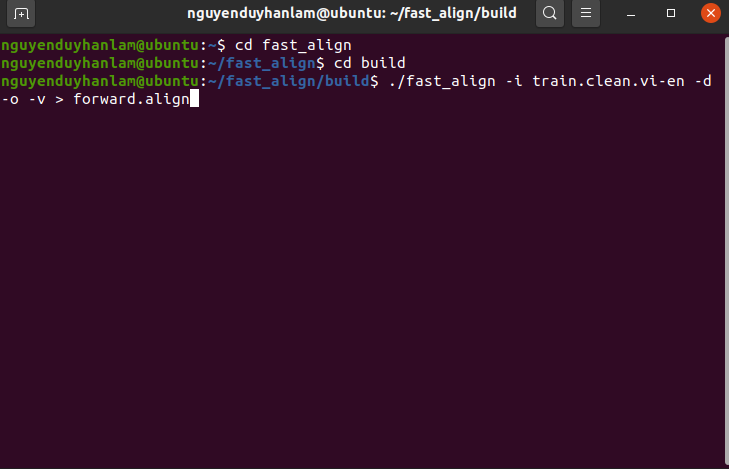
1. Chạy file biên dịch để cài đặt fast\_align

Sau đó ta cần chuẩn bị file text, từng dòng có dạng: “<Câu nguồn> ||| <Câu đích>”. Để dễ hình dung ta có thể xem minh họa sau:



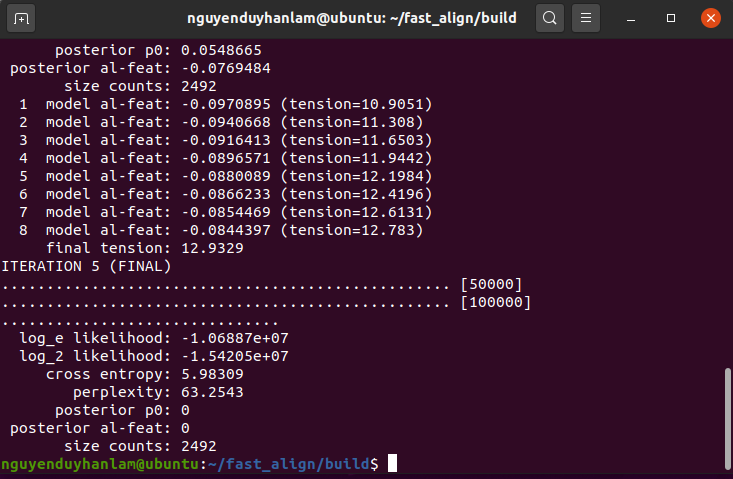
1. Ví dụ minh họa kiểu dữ liệu để chạy fast\_align

Vì ta đã chuẩn bị file này ở mục 4.3 vì thế ta sẽ tiến hành chạy fast\_align, để chạy fast\_align ta thực hiện lệnh:



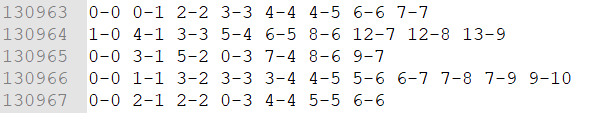
1. Chạy fast\_align

Sau khi chạy hoàn tất chương trình thông báo kết quả:



1. Kết quả sau khi chạy fast\_align

Ta thu được là file gióng hàng từ, mỗi dòng trong file chứa các cặp gióng hàng từ của các từ ứng với mỗi cặp .



1. Minh họa cho kiểu dữ liệu trả về của fast\_align
   1. Chạy chương trình bằng công cụ hỗ trợ fairseq

Fairseq là một bộ công cụ được phát triển bởi nhóm nghiên cứu Facebook AI Research.

Fairseq cung cấp nhiều mô hình dựng sẵn mô phỏng theo các mô hình được đăng trên các bài báo khoa học. Trong đó fairseq có hỗ trợ cả mô hình transformer thuần và mô hình transformer có sử dụng gióng hàng từ Jointly Learning to Align and Translate with Transformer Models.

* + 1. Chuẩn bị

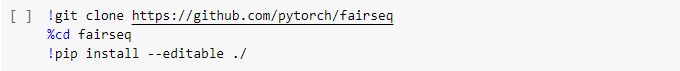
Đầu tiên ta cần tạo 2 thư mục trống sau:

* Bin dùng để lưu các file như từ điển, file log,… mà fairseq cần để chạy, huấn luyện, kiểm thử chương trình.
* Checkpoint dùng để lưu các mô hình sau từng đợt huấn luyện.

Về dữ liệu, dữ liệu:

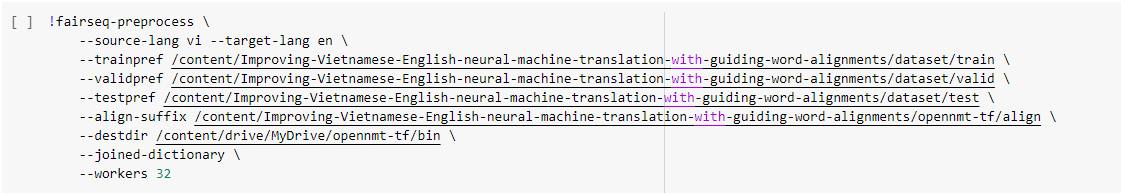
* Huấn luyện: là bộ 2 file train.en, train.vi sau khi được làm sạch dữ liệu và tokenize từ mục 4.2.
* Validation: là bộ 2 file tst2012.en, tst2012.vi sau khi được làm sạch dữ liệu từ mục 4.2.
* Kiểm thử: là bộ 2 file tst2013.en, tst2013.vi sau khi được làm sạch dữ liệu từ mục 4.2.

Ta sẽ tiến hành tải và cài đặt công cụ fairseq bằng lệnh sau:



1. Tải và cài đặt fairseq

Tiến hành xây dựng bộ từ điển và tạo các file cần thiết để fairseq chạy mô hình:

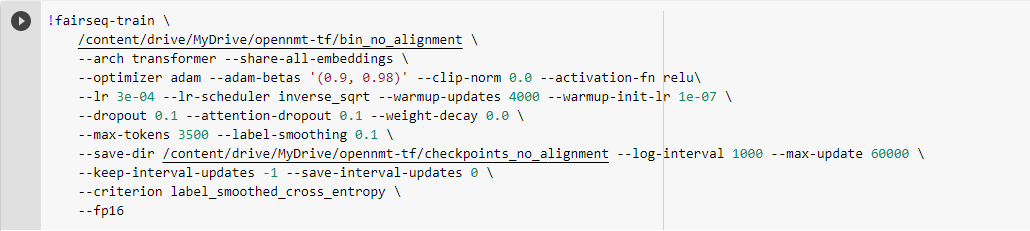


1. Chuẩn bị các file cần thiết cho fairseq

Trong đó:

* Source-lang: ngôn ngữ nguồn (ở đây là “vi” – tiếng Việt).
* Target-lang: ngôn ngữ đích (ở đây là “en” – tiếng Anh).
* Trainpref: tên tiền tố của file huấn luyện.
* Validpref: tên tiền tố của file validation.
* Align-suffix: tên hậu tố của file gióng hàng từ.
* Destdir: nơi lưu trữ các file cần thiết cho fairseq (ta dẫn đến thư mục Bin ta vừa mới tạo ở trên).
* Joined-dictionary: xây dựng bộ từ điển song ngữ.
* Workers: số luồng chạy song song.
  + 1. Thực hiện huấn luyện và kiểm thử mô hình transformer thuần

Ta tiến hành huấn luyện mô hình transformer thuần bằng lệnh sau:

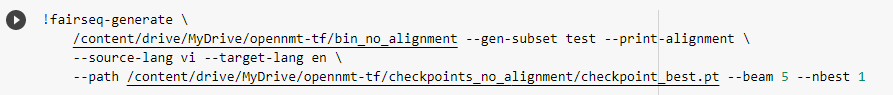


1. Thực hiện huấn luyện mô hình transformer thuần

Trong đó:

* Arch: kiến trúc mô hình sử dụng (ta chọn mô hình “transformer” thuần).
* Hàm activation: ReLU (giống với mục 3.3)
* Optimizer: kĩ thuật tối ưu hóa (ta chọn Adam, ngưỡng tối ưu ).
* Lr: Learning rate (tỉ lệ học): 3e-04.
* Tỉ lệ dropout là 0.1.
* Save-dir: nơi lưu mô hình huấn luyện (ta dẫn đến thư mục Checkpoint mà ta vừa mới tạo ở trên).
* Criterion: hàm mất mát ta chọn là hàm cross entropy dùng cho label smoothing.

Sau khi huấn luyện, ta thực hiện kiểm thử bằng lệnh sau:



1. Thực hiện kiểm thử mô hình transformer thuần

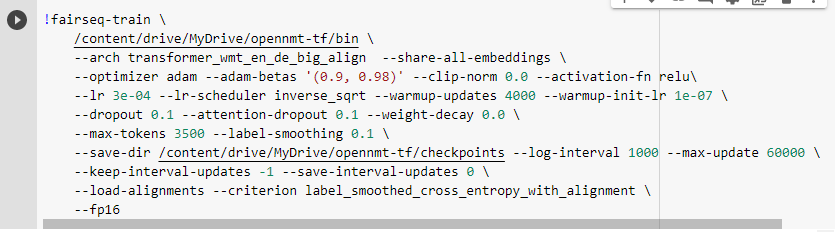
Trong đó:

* Beam: số lượng beam được dùng để tìm kiếm.
* Path: nơi lưu trữ file mô hình sau khi huấn luyện.

Cuối cùng ta bấm chạy để thực thi chương trình.

* + 1. Thực hiện huấn luyện và kiểm thử mô hình transformer sử dụng gióng hàng từ

Ta tiến hành huấn luyện mô hình transformer có sử dụng gióng hàng từ nhanh bằng lệnh sau:

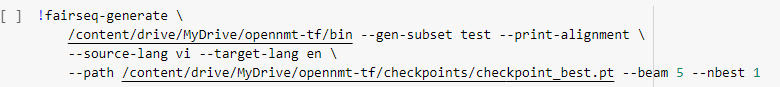


1. Thực hiện huấn luyện mô hình transformer sử dụng gióng hàng từ

Trong đó:

* Arch: kiến trúc mô hình sử dụng (ta chọn mô hình “transformer\_wmt\_en\_de\_big\_align”, mô hình transformer có sử dụng gióng hàng từ).
* Hàm activation: ReLU (giống với mục 3.3)
* Optimizer: kĩ thuật tối ưu hóa (ta chọn Adam, ngưỡng tối ưu ).
* Lr: Learning rate (tỉ lệ học): 3e-04.
* Tỉ lệ dropout là 0.1.
* Save-dir: nơi lưu mô hình huấn luyện (ta dẫn đến thư mục Checkpoint mà ta vừa mới tạo ở trên).
* Criterion: hàm mất mát ta chọn là hàm cross entropy dùng cho label smoothing.

Sau khi huấn luyện, ta thực hiện kiểm thử bằng lệnh sau:



1. Thực hiện kiểm thử mô hình transformer sử dụng gióng hàng từ

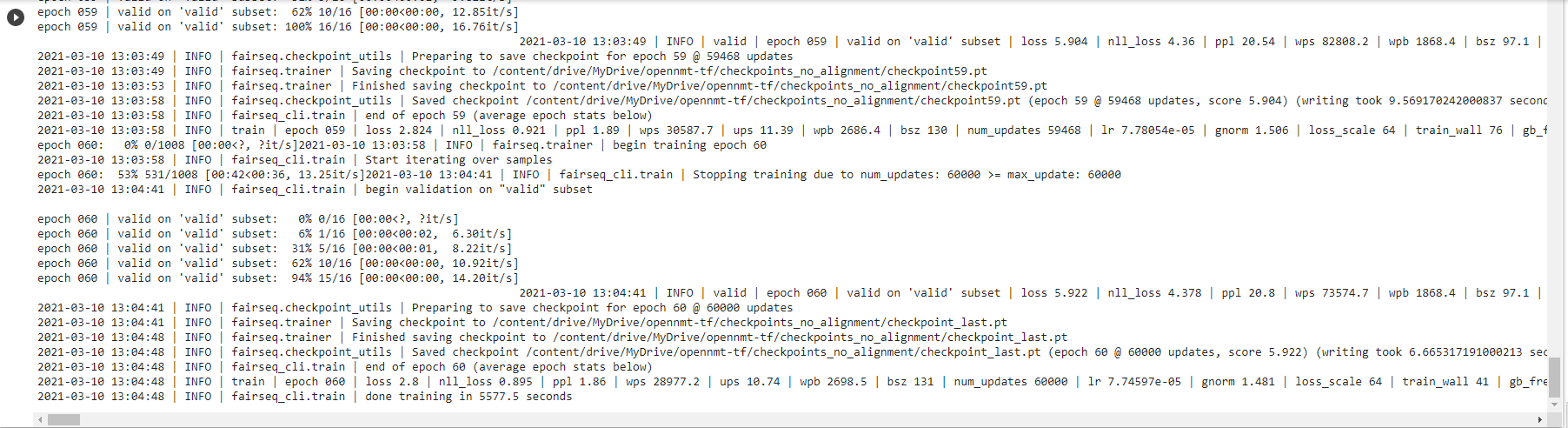
Trong đó:

* Beam: số lượng beam được dùng để tìm kiếm.
* Path: nơi lưu trữ file mô hình sau khi huấn luyện.

Cuối cùng ta bấm chạy để thực thi chương trình.

* 1. Kết quả và đánh giá
     1. Kết quả của mô hình transformer thuần

Sau khi huấn luyện, mô hình transformer thuần cho ra kết quả sau:

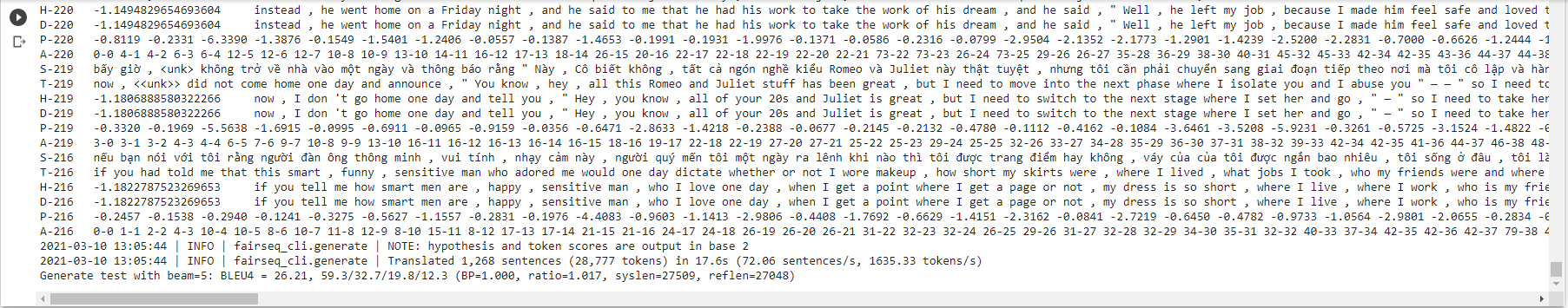


1. Kết quả huấn luyện của mô hình transformer thuần

Giá trị mất mát của tập:

* Huấn luyện mà fairseq tính được: 2.8, Negative Log-Likelihood (NLL): 0.895.
* Validation mà fairseq tính được: 5.922, Negative Log-Likelihood (NLL): 4.378.

Sau khi kiểm thử, mô hình transformer thuần cho ra kết quả sau:



1. Kết quả kiểm thử của mô hình transformer thuần

BLEU score của quá trình kiểm thử là: 26.21.

* + 1. Kết quả của mô hình transformer có sử dụng gióng hàng từ

Sau khi huấn luyện, mô hình transformer sử dụng gióng hàng từ cho ra kết quả sau:

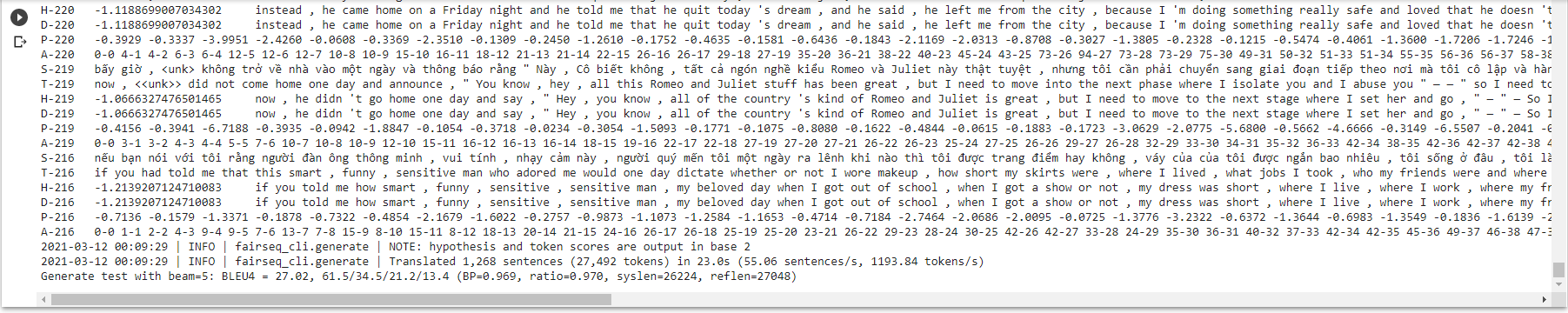


1. Kết quả huấn luyện mô hình transformer có sử dụng gióng hàng từ

Giá trị mất mát của tập:

* Huấn luyện mà fairseq tính được: 2.446, Negative Log-Likelihood (NLL): 0.523.
* Validation mà fairseq tính được: 5.993, Negative Log-Likelihood (NLL): 4.532.

Sau khi kiểm thử, mô hình transformer sử dụng gióng hàng từ cho ra kết quả sau:



1. Kết quả kiểm thử mô hình transformer có sử dụng gióng hàng từ

BLEU score của quá trình kiểm thử là 27.02

* + 1. So sánh và đánh giá

Từ kết quả ở mục 4.5.1 và 4.5.2 ta thu được bảng sau:

1. Bảng so sánh kết quả giữa mô hình transform thuần và mô hình transformer có sử dụng gióng hàng từ

|  |  |  |  |
| --- | --- | --- | --- |
|  | Train (Loss) | Validation (Loss) | Test (BLEU) |
| Transformer thuần | 2.8 | **5.922** | 26.21 |
| Transformer có sử dụng gióng hàng từ | **2.446** | 5.933 | **27.02** |

Ta thấy rằng transformer có sử dụng gióng hàng từ cho ra kết quả đánh giá BLEU score cao hơn mô hình transformer thuần.

Nhờ có cơ chế huấn luyện Attention dựa trên gióng hàng từ giúp cải thiện được cơ chế mà vì thế mô hình transformer có sử dụng gióng hàng từ cho ra kết quả tốt hơn.

* 1. Tổng kết

Với việc sử dụng gióng hàng từ mà ta đã có thể cải thiện chất lượng dịch của mô hình transformer. Nhờ có cơ chế huấn luyện Attention dựa trên gióng hàng từ giúp cải thiện được cơ chế mà vì thế mô hình transformer có sử dụng gióng hàng từ cho ra kết quả tốt hơn, thông qua quá trình thực nghiệm ta thấy rõ được điều này.

TÀI LIỆU THAM KHẢO

1. Namit Bhatia, 1992, The Oxford Companion to the English Language, trang (pp.) 1051 – 1054.
2. Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, Paul S. Roossin, 1990, A STATISTICAL APPROACH TO MACHINE TRANSLATION.
3. Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, Robert L. Mercer, 1993, The Mathematics of Statistical Machine Translation: Parameter Estimation.
4. Chris Dyer, Victor Chahuneau, Noah A. Smith, 2013, A Simple, Fast, and Effective Reparameterization of IBM Model 2.
5. James D. Miller , Rui Miguel Forte, 2017, Mastering Predictive Analytics with R - Second Edition.
6. Nikolay Kyurkchiev, Svetoslav Markov, 2015, SIGMOID FUNCTIONS: SOME APPROXIMATION, AND MODELLING ASPECTS.
7. 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.
8. Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, 2015, NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE.
9. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, 2017, Attention Is All You Need.
10. Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, Matthias Paulik, 2019, Jointly Learning to Align and Translate with Transformer Models.