TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**WORD REPRESENTATION**

*Người hướng dẫn*: **Thầy Lê Anh Cường**

*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**



**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**WORD REPRESENTATION**

*Người hướng dẫn*: **Thầy Lê Anh Cường**

*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 Lê Anh Cường. 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 30 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](#_Toc38785913)

[MỤC LỤC 1](#_Toc38785914)

[CHƯƠNG 1: DỊCH MÁY NEURAL 4](#_Toc38785915)

[1.1 Giới thiệu về dịch máy neural 4](#_Toc38785916)

[1.2 Recurrent Neural Network trong mô hình 5](#_Toc38785917)

[1.2.1 Cổng update 7](#_Toc38785918)

[1.2.2 Cổng reset 7](#_Toc38785919)

[1.2.3 Xuất kết quả 8](#_Toc38785920)

[1.3 Convolutional Neural Network trong mô hình 9](#_Toc38785921)

[1.4 Mô hình Encoder-Decoder 11](#_Toc38785922)

[1.4.1 Word embedding 12](#_Toc38785923)

[1.4.2 Encoder 13](#_Toc38785924)

[1.4.3 Decoder 13](#_Toc38785925)

[1.4.4 Linguistic Input Feature 16](#_Toc38785926)

[1.5 Tóm tắt chương 1 18](#_Toc38785927)

[CHƯƠNG 2: THỰC HIỆN MÔ PHỎNG ENCODER-DECODER 19](#_Toc38785928)

[2.1 Chuẩn bị nguồn dữ liệu và thư viện 19](#_Toc38785929)

[2.2 Xử lý dữ liệu và tạo các đối tượng quản lý 19](#_Toc38785930)

[2.3 Xây dựng encoder-decoder 23](#_Toc38785931)

[2.4 Huấn luyện 27](#_Toc38785932)

[2.5 Đánh giá 33](#_Toc38785933)

[2.6 Kiểm thử 35](#_Toc38785934)

[2.7 Chạy chương trình và kết quả 36](#_Toc38785935)

[2.8 Tóm tắt 37](#_Toc38785936)

[TÀI LIỆU THAM KHẢO 39](#_Toc38785937)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1. Sơ lược cấu tạo hoạt động của RNN [1] 5](#_Toc38785871)

[Hình 1.2. Cấu trúc neural theo GRU [2] 6](#_Toc38785872)

[Hình 1.3. Mô tả cách tính của cổng update [2] 7](#_Toc38785873)

[Hình 1.4. Mô tả cách tính cổng reset [2] 8](#_Toc38785874)

[Hình 1.5. Mô tả cách tính [2] 8](#_Toc38785875)

[Hình 1.6. Cấu trúc của CNN trong encoder [3] 9](#_Toc38785876)

[Hình 1.7. Minh họa cách hoạt động của hàm activation ở mỗi node [3] 10](#_Toc38785877)

[Hình 1.8. Minh họa cho một số cấu trúc của CNN [3] 11](#_Toc38785878)

[Hình 1.9. Mô hình encoder-decoder cơ bản [3] 12](#_Toc38785879)

[Hình 1.10. Cấu trúc của một bộ GRU-RNN encoder [4] 13](#_Toc38785880)

[Hình 1.11. Minh họa cho decoder [4] 14](#_Toc38785881)

[Hình 1.12. Cơ chế attention [5] 15](#_Toc38785882)

[Hình 1.13. Cấu trúc chi tiết của một attention [6] 15](#_Toc38785883)

[Hình 2.1. Các thư viện cần chuẩn bị 19](#_Toc38785884)

[Hình 2.2. Các thư viện phục vụ linguistic input feature 19](#_Toc38785885)

[Hình 2.3. Class quản lý dữ liệu 20](#_Toc38785886)

[Hình 2.4. Xử lý và chuẩn hóa dữ liệu 21](#_Toc38785887)

[Hình 2.5. Hàm đọc dữ liệu 22](#_Toc38785888)

[Hình 2.6. Chuẩn hóa số lượng từ của câu và lemmas hóa dữ liệu 22](#_Toc38785889)

[Hình 2.7. Hàm đọc dữ liệu và thực hiện nạp dữ liệu vào chương trình 23](#_Toc38785890)

[Hình 2.8. Xây dựng bộ encoder 23](#_Toc38785891)

[Hình 2.9. Xây dựng bộ decoder cơ bản 24](#_Toc38785892)

[Hình 2.10. Cấu trúc hoạt động của attention 25](#_Toc38785893)

[Hình 2.11. Class decoder sử dụng attention (hàm khởi tạo của class) 26](#_Toc38785894)

[Hình 2.12. Hàm khởi tạo tiến trình và hàm feedforward 27](#_Toc38785895)

[Hình 2.13. Các hàm tạo vector input và output cho quá trình huấn luyện 28](#_Toc38785896)

[Hình 2.14. Hàm thực hiện huấn luyện 29](#_Toc38785897)

[Hình 2.15. Giai đoạn tiền encoder 29](#_Toc38785898)

[Hình 2.16. Giai đoạn chạy encoder 30](#_Toc38785899)

[Hình 2.17. Giai đoạn hậu encoder và tiền decoder 30](#_Toc38785900)

[Hình 2.18. Giai đoạn chạy decoder 30](#_Toc38785901)

[Hình 2.19. Giai đoạn hậu decoder 31](#_Toc38785902)

[Hình 2.20. Hàm thông báo lượng thời gian chương trình đã và đang chạy 31](#_Toc38785903)

[Hình 2.21. Hàm vẽ biểu đồ thay đổi của giá trị loss 32](#_Toc38785904)

[Hình 2.22. Hàm huấn luyện tổng quát 33](#_Toc38785905)

[Hình 2.23. Hàm thực hiện đánh giá 34](#_Toc38785906)

[Hình 2.24. Hàm thực hiện chọn đánh giá ngẫu nhiên 35](#_Toc38785907)

[Hình 2.25. Các hàm phục vụ trình bày kết quả 35](#_Toc38785908)

[Hình 2.26. Tiến hành huấn luyện mô hình 36](#_Toc38785909)

[Hình 2.27. Kết quả của từng đợt huấn luyện 36](#_Toc38785910)

[Hình 2.28. Chạy thử 1 câu ví dụ 37](#_Toc38785911)

[Hình 2.29. Kết quả chạy thử 1 câu ví dụ 37](#_Toc38785912)

TÓM TẮT NỘI DUNG

Học máy là một lĩnh vực quan trọng trong thời đại công nghệ hiện nay. Trong đó dữ liệu là 1 phần cốt lõi của lĩnh vực vì chương trình sẽ phải dựa trên nguồn tập dữ liệu này mới có thể đào tạo, học và từ đó mới sinh ra được mô hình để có thể từ đó giải quyết bài toán.

Dữ liệu càng lớn, các đối tượng trong dữ liệu càng chứa nhiều thuộc tính (hay còn gọi là chiều dữ liệu) sẽ càng khiến chương trình trở nên nặng nề, khó đào tạo và từ đó sẽ khó có thể đào tạo được ra một mô hình chuẩn xác để giải quyết bài toán. Bằng các kĩ thuật tính toán ta có thể hồi quy đưa số chiều của dữ liệu lớn trở nên nhỏ hơn nhằm giảm bớt gánh nặng trong việc đào tạo ra mô hình. Kĩ thuật tính toán này được gọi là shrinkage method (các phương thức co rút chiều dữ liệu).

Tài liệu này bao gồm các phần chính:

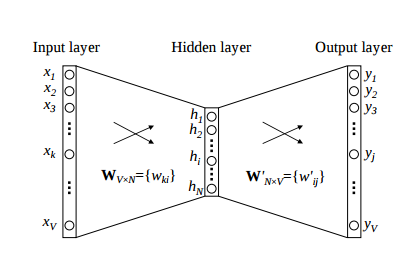
* Giới thiệu lại về mô hình tuyến tính.
* Các phương pháp hồi quy thô.
* Các phương pháp hồi quy chính (shrinkage method).
* Áp dụng vào chương trình mẫu.

Chương 1: MÔ HÌNH CBOW

1.1 Giói thiệu mô hình

Mô hình CBOW thuộc prediction-based embedding, Mục tiêu của mô hình là xây dựng cố gắng dự đoán từ một từ được chọn dựa theo các từ xung quanh, ngữ cảnh của chúng.

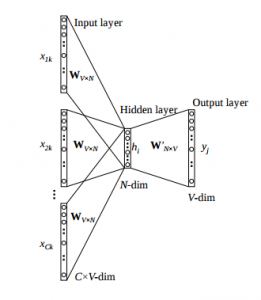
CBOW hoạt động dựa trên cách thức là nó sẽ dự đoán xác suất của một từ được đưa ra theo ngữ cảnh, ngữ cảnh đó có thể gồm một hay nhiều từ quanh từ được chọn, với input là một hoặc nhiều One-hot vector của các từ ngữ cảnh có chiều dài V (với V là độ lớn của từ điển), output sẽ là một vector xác suất cũng với chiều dài V của từ liên quan hoặc còn thiếu, Hidden Layer có chiều dài N, N cũng chính là độ lớn của vector từ biểu thị. Dưới đây là mô hình CBOW với ngữ cảnh là 1 từ đơn:



Ma trận số hóa các từ sẽ được đưa vào mô hình mạng neural 3 lớp với lớp output đầu ra là lớp softmax được để tổng các xác suất đầu ra tầng ẩn thành 1.

Mô hình thuật toán có thể được miêu tả như sau:

* Input được vector hóa one-hot encoded
* Giữa các lớp chỉ sử dụng hàm kích hoạt tuyến tính.
* Input được nhân trọng số tới hàm ẩn điều chỉnh trong số tính toán và từ tầng ẩn tiếp tục nhân trọng số rồi đưa vào hàm softmax đưa ra output là 1 vector xác suất, output này sẽ được so sánh với output mong muốn và tính toán độ lỗi
* Dựa vào độ lỗi này mà mạng neuron sẽ lan truyền ngược trở lại để cập nhật các giá trị của các ma trận trọng số.
* Trọng số giữa lớp ẩn và lớp đầu ra được lấy làm đại diện vectơ từ của từ.



1.2 Ưu điểm của CBOW

So với các phương pháp xác định xác suất ngữ cảnh của từ, CBOW cho ra kết quả tính toán rất tốt.

Không tốn nhiều bộ nhớ so với phương pháp ma trận ngữ cảnh.

1.3 Nhược điểm của CBOW

Xác suất ngữ cảnh một từ CBOW cho ra có thể là trung bình của 2 kết quả. Ví dụ Apple có thể là trái táo vừa là tên công ty, CBOW sẽ đưa ra xác suất ngữ cảnh nằm giữa táo và công ty. Dẫn tới việc các từ giống nhau nhưng nghĩa khác nhau vẫn chỉ được biểu diễn bằng 1 vec-tơ từ duy nhất.

Train mô hình CBOW từ đầu rất khó cho ra kết quả tốt nếu không được tối ưu hóa đúng cách.

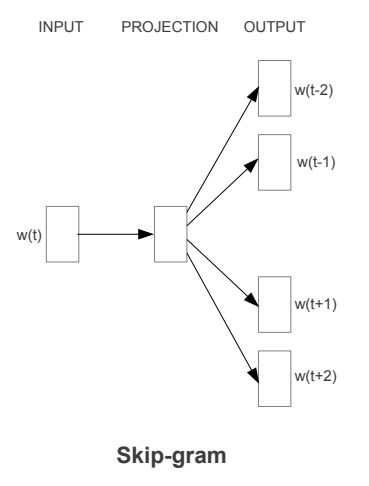
CHƯƠNG 2: MÔ HÌNH SKIP-GRAM

2.1 Mô hình Skip-gram

Một mô hình khác thường được sử dụng để huấn luyện các dữ liệu phân phối của các vector biểu diễn từ mà ta sẽ tìm hiểu ở chương này chính là mô hình skip-gram.

Mô hình skip-gram bắt đầu với vector giá trị của từ được xét ở bước hiện tại. Sau đó vector này sẽ được vào mô hình để dự đoán các giá trị ở đầu ra.

Mô hình skip-gram có thể được minh họa như sau:



1. Mô hình skip-gram [1]

Theo như hình 2.1 ta có thể thấy ở bước hiện tại t mô hình sử dụng vector w hiện tại để đem vào mô hình (projection) kết quả đầu ra của mô hình là các kết quả w khác.

Mô hình Skip-Gram tìm cách tối ưu hóa ma trận trọng số từ (embedding word) bằng cách dự đoán chính xác các ngữ cảnh, dựa trên một vector biểu diễn từ ở đầu vào (). Nói cách khác, mô hình muốn tối đa hóa độ chính xác của các xác suất dự đoán của các kết quả đầu ra cùng một lúc. Tối ưu hóa các xác suất dự đoán của các từ chính là việc ta sẽ phải đi tối ưu hóa ma trận trọng số từ (). Ma trận là sự kết hợp của 2 ma trận gồm ma trận trọng số đầu vào và ma trận trọng số đầu ra, , như được mô tả dưới đây. Kết thúc mô hình ta sẽ đi tính chi phí bằng hàm chi phí J và từ đó ta sẽ đi tối ưu hóa mô hình. Về mặt toán học, mô hình có thể được biểu thị như sau:

Trong đó:

* C là số lượng kết quả ở đầu ra.
* : là các giá trị kết quả ở đầu ra.
* : dữ liệu đầu vào.

Ta có thể thế (1) bằng hàm log:

Trong mô hình skip-gram, đại lượng p sẽ được tính bằng hàm softmax như sau:

Thế (3) vào (2) ta được:

Để tối đa hóa độ chính xác của các xác suất ta cần phải tối thiểu hóa giá trị lỗi của mô hình, hàm chi phí của mô hình được tính sau:

Ta có thể viết (5) lại thành:

Ta có thể tách (6) thành phép tổng sau:

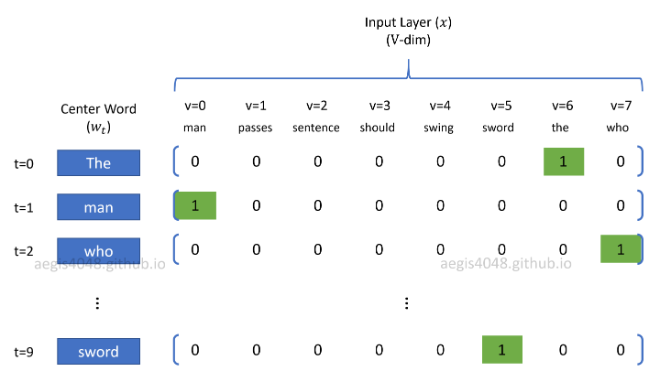
Dựa vào (6), ta có thể biểu diễn hàm mất thành dạng sau:

Tổng quát hóa cho T dữ liệu khi áp dụng Stochastic gradient descent, ta được:

2.2 Giải thuật tiến (Forward Propagation)

Đầu tiên ta sẽ trích xuất V các từ riêng biệt được sử dụng trong câu. Ví dụ ta có câu: “The man who passes the sentence should swing the sword.”. Ta sẽ trích xuất từ câu ra có 8 (V = 8) từ riêng biệt: “the”, “man”, “who”, “passes”, “sentence”, “should”, “swing”, “sword”.

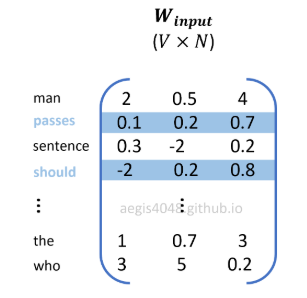
Sau đó ta sẽ vector hóa các từ này:



1. Vector hóa các từ trong câu dưới dạng nhị phân [2]

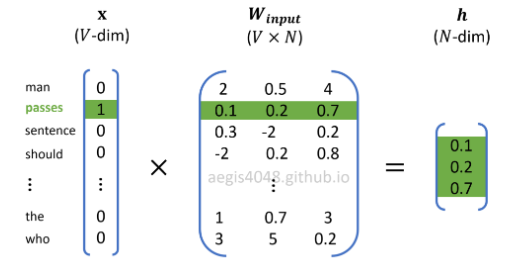
Ta có thể thấy trong hình 2.2, mỗi dòng là 1 vector đại diện biểu diễn cho từ tương ứng.

Tiếp theo ta sẽ chuyển vector này thành ma trận word-embedding, mỗi phần tử (các từ trong câu – các dòng trong ma trận) sẽ có N chiều (N cột). Nói gọn lại là mỗi dòng trong ma trận chính là 1 vector biểu diễn cho từ đó trong không gian N chiều.



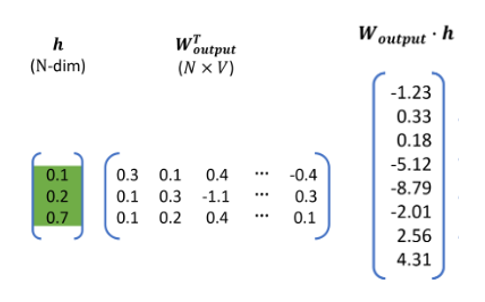
1. Chuyển vector từ thành word-embedding () [2]

Lớp ẩn của mô hình () sẽ tiến hành đem nhân 2 ma trận: vector từ (mỗi dòng ở hình 2.2, ) và (ở hình 2.3) lại với nhau.



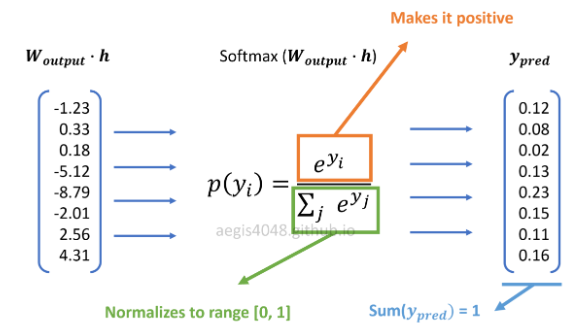
1. Biểu diễn cách tính [2]

Ta phải giải mã lại kết quả của h về lại vector biểu diễn từ bằng cách nhân h với một ma trận word-embedding khác. Ma trận word-embedding này ta sẽ gọi là . Vì muốn đổi về lại thành vector từ nên số chiều (số cột) của sẽ là V, và số dòng của nó sẽ là N ứng với N giá trị trong ma trận .



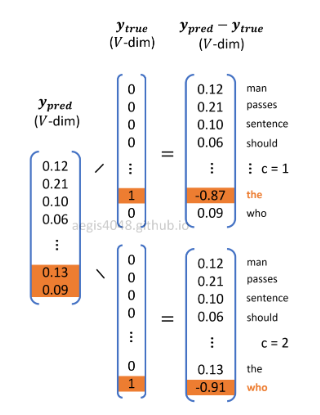
1. Giải mã kết quả về lại dạng word-vector [2]

Cuối cùng ta sẽ đi chuẩn hóa kết quả bằng hàm softmax:



1. Chuẩn hóa kết quả dự đoán bằng softmax [2]

Ta sẽ so kết quả dự đoán từ mô hình với kết quả thực. Ta sẽ lấy 2 kết quả này trừ với nhau để ra độ lỗi của mô hình.



1. Tính lỗi ở các kết quả đầu ra [2]

Sau đó ta gộp , tạo thành ma trận trọng số :

Cuối cùng ta đi tính chi phí của lượt chạy của mô hình ở thời điểm hiện tại:

2.3 Giải thuật lùi (Backward Propagation)

Ta sẽ bắt đầu đi gradient descent cho trọng số lớp output trước:

Ta tiến hành hiệu chỉnh :

Ta sẽ tiến hành gradient descent cho lớp ẩn:

Tiếp theo ta sẽ hiệu chỉnh cho :

2.4 Nhược điểm của softmax

Ở cuối mỗi kết quả của mô hình ta sẽ chuẩn hóa kết quả bằng hàm softmax:

Ta có thể thấy rằng cứ mỗi lần tính softmax ta đều phải tính tổng:

Nếu bộ từ điển của ta lớn (V lớn), ta không những phải tính tổng rất nhiều mà mỗi lần quét các từ ta đều phải thực hiện lại tính tổng này. Dẫn đến việc phép tính mà ta thực hiện sẽ rất nhiều làm giảm hiệu suất của chương trình.

Chưa kể việc V càng lớn thì sẽ khiến xác suất p ngày càng nhỏ dễ dẫn đến tình trạng vanishing gradient, lượng trọng số của mỗi lần hiệu chỉnh sẽ rất nhỏ, góp phần làm giảm hiệu suất của chương trình.

2.5 Phương hướng cải tiến

TÀI LIỆU THAM KHẢO

**Tiếng Anh:**

1. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, 2013, Efficient Estimation of Word Representations in Vector Space.
2. <https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling#Derivation-of-Cost-Function>