TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**XỬ LÝ NGÔN NGỮ TỰ NHIÊN**

**LANGUAGE MODEL**

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

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

**Đào Việt Duy (MSSV: 51603070)**

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**

**LANGUAGE MODEL**

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

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

**Đào Việt Duy (MSSV: 51603070)**

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)

CHƯƠNG 1: MÔ HÌNH NGÔN NGỮ

1.1 Xác suất trong câu từ, cú pháp

Về mặt toán học xác suất của câu từ và cú pháp thể hiện mức độ hợp lý của các ký tự, từ vựng và trật tự của nó trong các câu văn bản hay văn nói. Giá trị xác suất này càng cao thì mức độ hợp lý của câu văn cũng như các thành phần của nó sẽ càng cao. Ví dụ xác suất của “**Trà** ngon đấy ông!” sẽ cao hơn “**Chà** ngon đấy ông!”, vì “**Trà** ngon đấy ông!” về mặt chính tả chính xác hơn nên giá trị xác suất của nó sẽ cao hơn. Ví dụ xác suất của “**Thôi**, tôi về nha” sẽ có giá trị xác suất cao hơn “**Hoy**, tôi về nha”, do “Hoy” là dạng tiếng lóng mô phỏng cho phát âm đặc trưng của vùng miền do đó “**Thôi**, tôi về nha” sẽ có giá trị xác suất cao hơn. Ví dụ “Mặt hồ **xanh** như mắt biếc” sẽ có giá trị xác suất cao hơn “Mặt hồ **trong** như mắt biếc” vì “mắt biết” là hình tượng con mắt xanh gắn liền hơn với “xanh” của “Mặt hồ”.

Xác suất của 1 câu hay của 1 chuỗi từ được tính bằng:

Trong đó:

* : là xác suất của câu hay chuỗi.
* : là các từ có trong câu hay chuỗi.

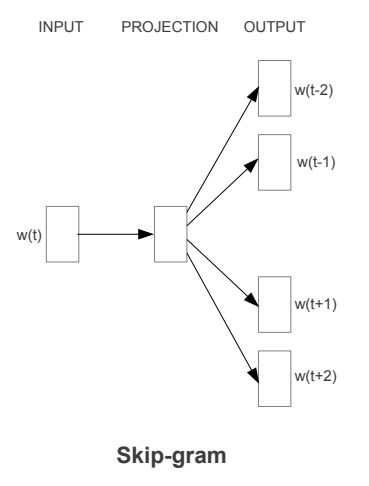
CHƯƠNG 2: MÔ HÌNH SKIP-GRAM

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:

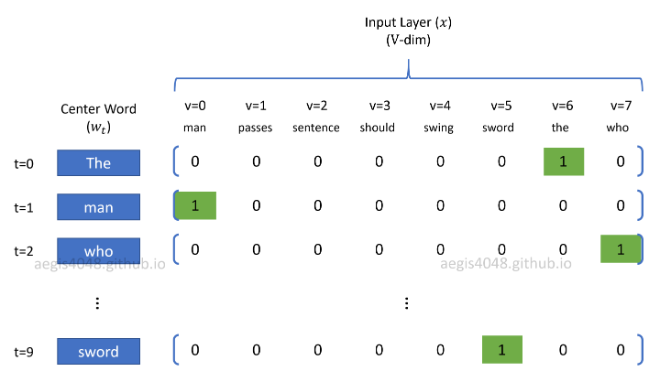
Dựa vào (6), ta có thể biểu diễn hàm mất thành dạng sau:

Tổng quát hóa cho T dữ liệu khi áp dụng vào toàn tập văn bản, 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”.

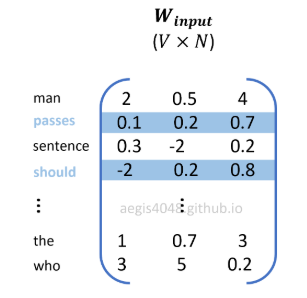
Sau đó ta sẽ vector hóa các từ này:



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

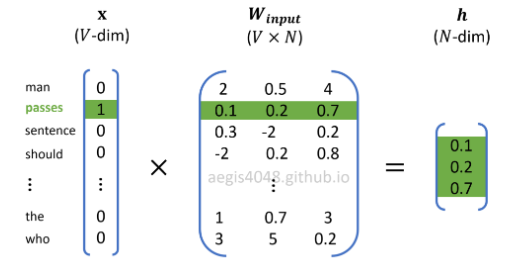
Ta có thể thấy trong hình 2.2, mỗi dòng là 1 vector đại diện biểu diễn cho từ tương ứng.

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.



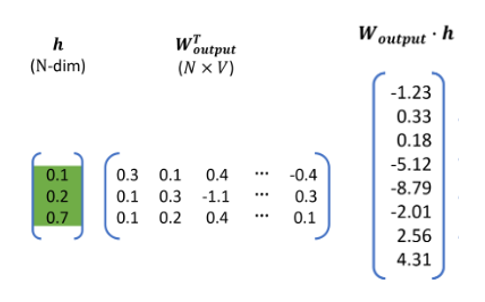
1. Chuyển vector từ thành word-embedding () [2]

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.



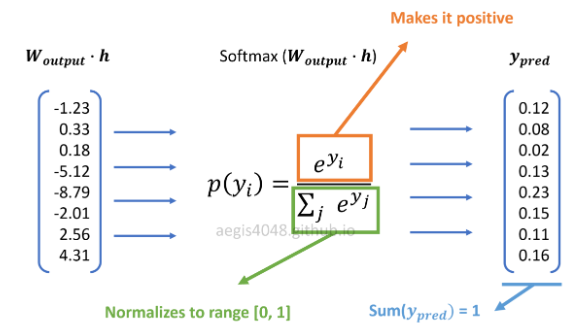
1. Biểu diễn cách tính [2]

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 .



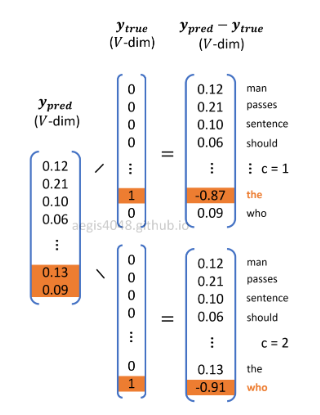
1. Giải mã kết quả về lại dạng word-vector [2]

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

Thay vì ta sẽ dùng word vector và đem vector này vào mô hình để dự đoán và sử dụng softmax để chuẩn hóa phân phối, thì ta sẽ dùng sigmoid function để học sự khác biệt giữa các cặp từ được tạo bởi từ noise distribution (phân phối nhiễu).

2.5.1 Phân phối nhiễu (noise distribution)

Phân phối của các từ được tính dựa trên số lần xuất hiện của từ đó trong tập văn bản (corpus). Với mỗi phân phối, để chuẩn hóa ta sẽ chia phân phối này cho giá trị chuẩn (normalization factor) Z. Cuối cùng ta sẽ nâng kết quả này lên bậc lũy thừa :

Giá trị giúp làm mượt các phân phối, giúp các phân phối trở nên cân bằng hơn, giúp phân phối của các từ hiếm gặp trở nên phổ biến hơn nhằm tăng khả năng được chọn cho các từ hiếm gặp.

2.5.2 Cách thức triển khai (Forward propagation)

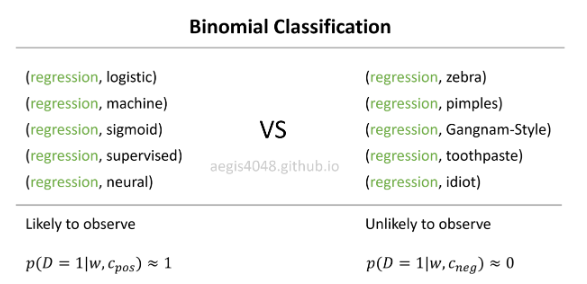
Ứng với mỗi từ đầu vào, ta sẽ tạo các cặp , với c là các từ xuất hiện trong tập văn bản. Xác suất để các cặp này xuất hiện sau khi mô hình tính được sẽ là . Ngược lại xác suất để các cặp này không xuất hiện sẽ là .

Ta sẽ phân các cặp này thành 2 loại:

* Một là với là các từ có khả năng đi liền hoặc bắt cặp được với w. Các có tồn tại trong tập văn bản, và có khả năng xuất hiện ở kết quả đầu ra là .
* Hai là với là các từ rất ít khả năng hoặc hầu như không có khả năng đi liền hoặc bắt cặp với w, tuy nhiên vẫn có khả năng các cặp này xuất hiện ở kết quả đầu ra . Ngược lại với , các không tồn tại trong tập văn bản, được ghép bằng cách lấy K phân phối từ ngẫu nhiên từ phân phối nhiễu.

Mục tiêu của mô hình hay của bài toán đó chính là làm tối đa hóa (maximize) các phân phối hay và tối thiểu hóa (minimize) các phân phối hay .

Hay nói các khác ta sẽ đưa bài toán dự đoán trở về bài toán phân loại nhị phân. Ta sẽ phân các cặp vào 2 nhóm duy nhất đó chính là: và .



1. Minh họa kết quả phân loại của bài toán [2]

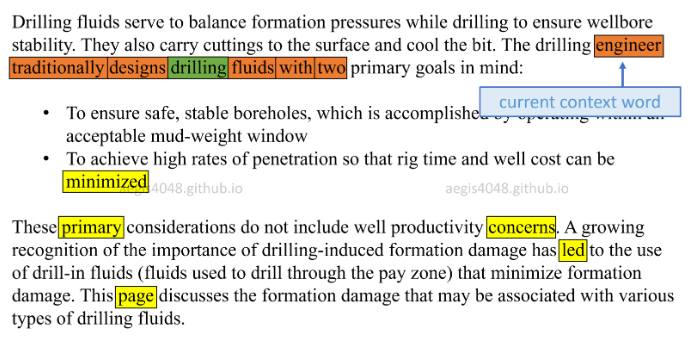
Xác suất để một từ c có thể đi liền hoặc bắt cặp với w ở kết quả đầu ra được tính bằng công thức:

Với là word vector nằm trong .

Ngoài ra ứng với mỗi từ đầu vào, thay vì ta đi xét trên toàn bộ tập từ vựng thì ta sẽ chỉ xét trên bộ K + 1 từ vựng mà thôi. Trong đó K từ ta sẽ lẫu nhiên từ tập phân phối nhiễu (ta gọi tập này là ) và 1 từ có liên quan với w xuất hiện trong tập văn bản (ta gọi từ này là ). Như vậy ta có:

Kết hợp (17) và (18) ta được:

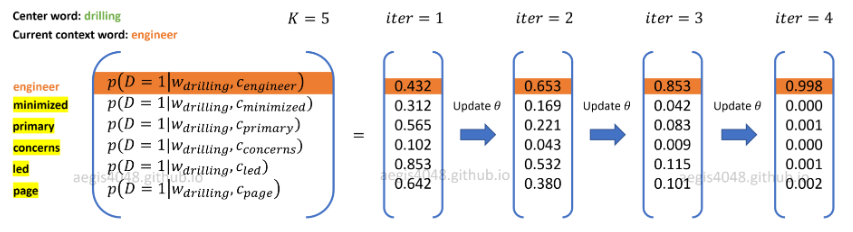
Để dễ hiểu ta sẽ xét ví dụ minh họa sau:



1. Minh họa về negative sampling [2]

Trong ví dụ hình 2.9, từ đầu vào (được tô màu xanh lá) là “drilling”, các từ đi liền với “drilling” là “engineer”, “traditionally”, “designs”, “fluids”, “with”, “two” do đó các từ này có . Và hiện đang xét là “engineer”. Và từ () được chọn ngẫu nhiên từ phân phối nhiễu là “minimized”, “primary”, “concerns”, “led”, “page” (các từ tô màu vàng).

Kết quả của ví dụ có thể được mô tả tuần tự như trong hình sau:



1. Từng vòng lặp của mô hình [2]

Ta có thể thấy trong hình 2.10, ở mỗi lần vòng lặp ta sẽ tiến hành dự đoán và hiệu chỉnh và các trong số, ngoài ra ta còn thấy được cứ sau mỗi vòng lặp thì lại càng tiến gần về 1 hơn, ngược lại các lại càng ngày tiến gần về 0 hơn.

2.5.3 Hàm chi phí mới

Bài toán của ta gồm 2 mục tiêu: thứ nhất là và thứ hai là . Hay nói cách khác đó chính là maximize và maximize . Vì vậy ta có hàm chi phí như sau:

Hay:

Để tránh tình trạng tích bị 0, ta sẽ thay thế bằng hàm log:

Hay:

Ta sẽ chuyển phép tích ở số hạng thứ 2 của (22) về dạng phép tổng, ta được:

Thế (17) vào (23) ta được:

Rút gọn lại ta được:

Ta đặt hàm sigmoid là:

Thế (26) vào (25), ta được:

Trong biểu thức (8) ta sẽ thế:

Thế (28) vào (9), ta được:

Do kích thước của toàn tập văn bản là rất lớn, nên thay vì đi huấn luyện cho toàn bộ tập văn bản thì ta sẽ thực hiện huấn luyện bằng Stochastic gradient descent. Do đó hàm chi phí cho mỗi sẽ là (với ):

2.5.4 Triển khai hiệu chỉnh (Back propagation)

Đầu tiên ta sẽ gradient descent cho hàm chi phí:

Tiếp theo ta sẽ gradient descent cho lớp output :

Hiệu chỉnh cho lớp output:

Tiếp tục ta sẽ gradient descend cho lớp ẩn h:

Ta gộp và lại với nhau:

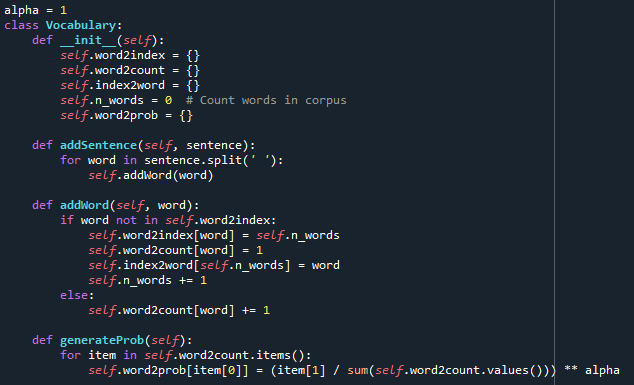
Với khi là và khi .

Hiệu chỉnh cho lớp ẩn :

2.6 Xây dựng mô hình từ lý thuyết

Đầu tiên ta sẽ tạo class Vocabulary nhằm lưu trữ một vài thông tin cơ bản:

* word2index: dùng để tham chiếu index (chỉ số thứ tự) của các từ.
* word2count: đếm số lần xuất hiện của các từ có trong corpus (tập văn bản).
* index2word: đánh chỉ số thứ tự (index) cho các từ.
* n\_words: đếm độc lập số lượng của các từ.
* word2prob: lưu phân phối của các từ .

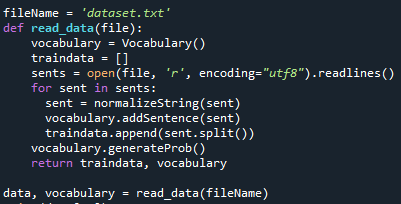


1. Tạo class lưu trữ các thông tin cơ bản của các từ

Trong class này có 3 hàm bổ trợ khác là:

* addSentence: tách câu thành các từ.
* addWord: lưu các thông tin cơ bản của các từ.
* generateProb: tính phân phối của các từ.

Viết hàm đọc dữ liệu, tiến hành đọc dữ liệu và lưu thông tin của các từ:



1. Tiến hành đọc dữ liệu vào chương trình

Ta định nghĩa một số biến quan trọng như: window\_size (kích thước cửa sổ, số lượng từ trước hoặc sau từ trung tâm mà ta sẽ xét), K – số lượng từ (phân phối ngẫu nhiên mà ta sẽ lấy ngoài, bên cạnh các từ mà ta xét ở window\_size).

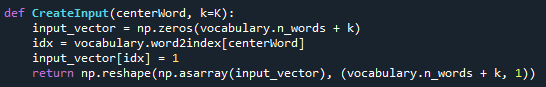


1. Định nghĩa window\_size và K

Theo hình 2.13 ta có window\_size = 5 tức là trước từ trung tâm ta sẽ xét 5 từ và sau từ trung tâm ta sẽ xét 5 từ. Tính thêm từ trung tâm thì khung ngữ cảnh của bài toán mà ta thực hiện sẽ có 11 từ.

Theo hình 2.13, ta sẽ lấy K=10 phân phối ngẫu nhiên của các từ nằm ngoài khung ngữ cảnh h. Theo như phần lý thuyết ở mục 2.5 thì ta sẽ gộp K=10 từ này với V từ của bộ từ điển để làm dữ liệu đầu vào cho mô hình.

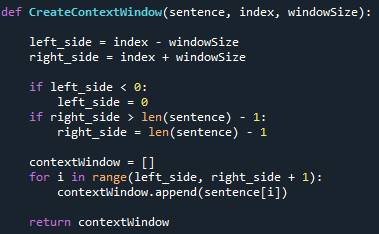
Tiếp theo ta sẽ viết hàm xây dựng tập dữ liệu đầu vào cho mô hình:



1. Hàm xây dựng tập dữ liệu đầu vào cho mô hình

Kết quả trả về của hàm được viết ở hình 2.14 là một tập vector one-hot coding. Trong đó từ trung tâm sẽ có giá trị 1, các từ V và K còn lại sẽ có giá trị là 0.

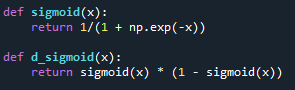
Tiếp theo ta sẽ viết hàm xây dựng khung ngữ cảnh h:



1. Hàm xây dựng khung ngữ cảnh h

Ứng với hình 2.9, ta có từ trung tâm là: “drilling”. Với window\_size là 5 ta sẽ xét phía trước từ trung tâm thêm 5 từ là: “The”, “drilling”, “engineer”, “traditionally”, “designs”. Và phía sau từ trung tâm thêm 5 từ là: “fluids”, “with”, “two”, “primary”, “goals”. Như vậy sau khi thực hiện hàm ở hình 2.15 ta sẽ có khung ngữ cảnh h gồm 11 từ với 5 từ trước từ trung tâm, từ trung tâm, và 5 từ sau từ trung tâm.

Cuối cùng ta sẽ viết thêm 2 hàm bổ trợ nữa là hàm sigmoid và đạo hàm của hàm sigmoid:



1. Hàm sigmoid và đạo hàm của hàm sigmoid

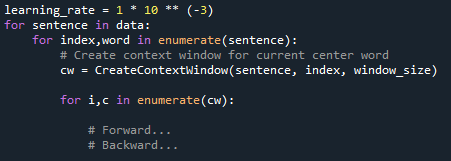
Đầu tiên ta sẽ khởi tạo 2 bộ trọng số: và .



1. Khởi tạo và

Cả 2 bộ trọng số đều có kích thước là V + K dòng và widow\_size x 2 + 1 cột. Các trọng số là các giá trị ngẫu nhiên trong phạm vi từ 0 đến 1.

Khung sườn của mô hình sẽ có dạng như sau:



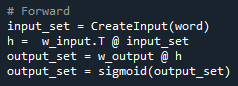
1. Khung sườn của mô hình

Ở đây ta sẽ chọn learning rate hay .

Theo hình 2.18, ta cho duyệt theo từng câu trong corpus.

Ứng với mỗi câu ta sẽ duyệt từng từ và xét từng từ làm từ trung tâm. Ứng với mỗi từ trung tâm ta sẽ tạo khung ngữ cảnh h. Ứng với mỗi từ trong khung ngữ cảnh, ta sẽ chạy giải thuật tiến (forward) trước và ta sẽ cho chạy giải thuật lùi (backward) để cập nhật các trọng số.

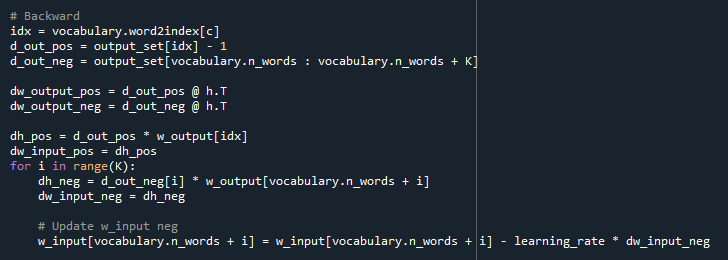
Giải thuật tiến của ta sẽ có dạng:



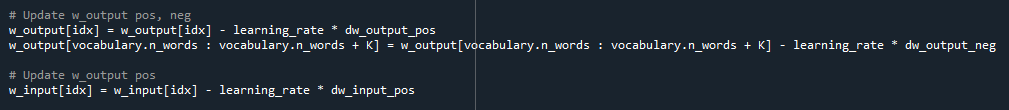
1. Giải thuật tiến

Trình tự của giải thuật tiến tuân theo lý thuyết đã đề cập ở mục 2.5.2.

Giải thuật lùi và tiến trình cập nhật trọng số của ta sẽ có dạng:



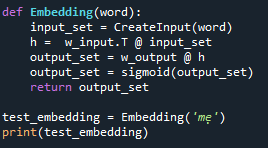
1. Giải thuật lùi



1. Quá trình cập nhật các trọng số

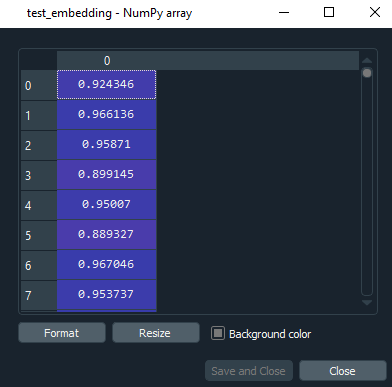
Giải thuật lùi và quá trình cập nhật các trọng số tuân theo lý thuyết ở mục 2.5.3 và 2.5.4.

Cuối cùng ta sẽ viết hàm embedding để sinh kết quả và cho chạy thử chương trình:



1. Chạy và kiểm thử mô hình

Kết quả sau khi embedding thử 1 từ:



1. Kết quả embedding của 1 từ

Như vậy là ta đã hoàn thiện xong quá trình xây dựng mô hình và thực hiện embedding cho 1 từ.

2.7 Ứng dụng

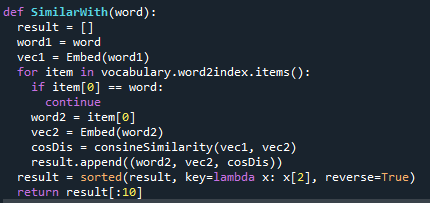
2.7.1 Tìm từ tương thích

Đầu tiên ta sẽ xây dựng hàm tính độ tương thích (tương tự). Ở đây ta sẽ sử dụng độ đo cosine:



1. Xây dựng hàm tính khoảng cách cosine

Cuối cùng ta sẽ đi xây dựng hàm tìm 10 từ có độ tương thích cao với từ muốn xét:

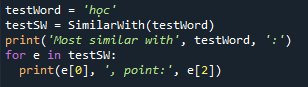


1. Hàm tìm các từ tương thích

Ta sẽ đi quét toàn bộ các từ có được trong bộ từ điền. Ta sẽ tính độ đo tương thích giữa từ đang xét (từ được truyền vào) với từng từ trong bộ từ điển.

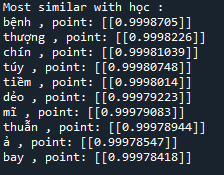
Cuối cùng ta sẽ sắp xếp độ tương thích theo chiều từ cao xuống thấp. Từ đó ta sẽ trả về 10 từ có độ tương thích cao nhất.

Ta có thể chạy thử chương trình bằng ví dụ sau:



1. Tìm thử từ tương tích bằng ví dụ

Kết quả của ví dụ sau khi chạy chương trình.



1. Kết quả tìm từ tương thích

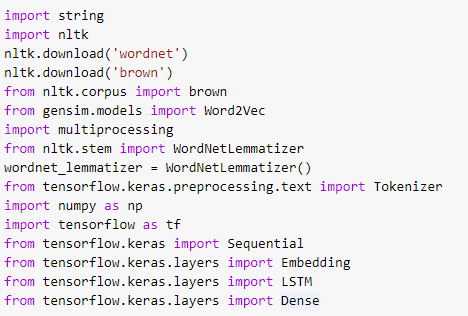
2.7.2 Language model ở mức từ vựng

Phần này ta sẽ mở rộng ra bằng bài toán xây dựng language model ở mức từ vựng dựa trên các word embedding.

Để đơn giản phần này ta sẽ sử dụng các hàm và các thư viện có sẵn của keras.

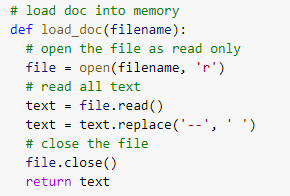
Ta sẽ xây dựng language model dựa trên cuốn sách “The republic” của Plato.

Đầu tiên ta cần chuẩn bị các thư viện:



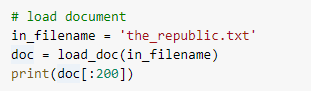
1. Các thư viện cần chuẩn bị

Tiếp theo ta viết hàm đọc dữ liệu:



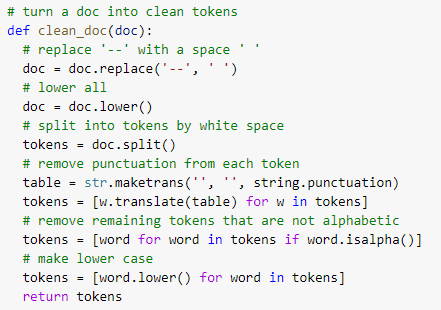
1. Hàm đọc dữ liệu

Tiếp đó là đọc dữ liệu từ sách “The republic”:



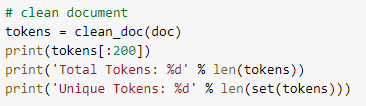
1. Đọc dữ liệu vào chương trình

Ta sẽ xây dựng hàm thực hiện lọc và tạo token cho dữ liệu.



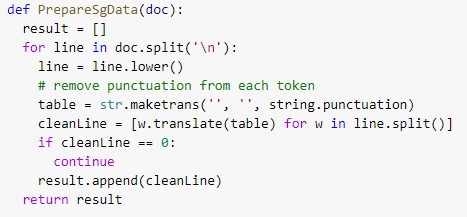
1. Hàm lọc và tạo token cho dữ liệu

Ta sẽ tiến hành lọc và tạo token cho dữ liệu.



1. Lọc và tạo token cho dữ liệu

Để sử dụng mô hình skip-gram của keras ta cần phải đưa dữ liệu về dạng chuẩn mà mô hình sẽ sử dụng.



1. Hàm chuẩn hóa dữ liệu trước khi train skip-gram của keras

Ta thực hiện chuyển dữ liệu.



1. Chuyển dữ liệu về dạng yêu cầu của keras

Ta tiến hành sử dụng hàm Word2Vec của thư viện keras để train mô hình skip-gram.



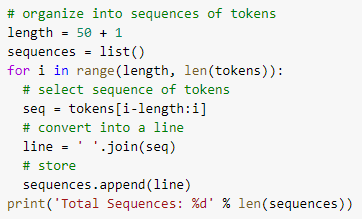
1. Tiến hành train mô hình skip-gram

Lưu lại mô hình về dạng file .bin và .txt để có sử dụng lại sau này.



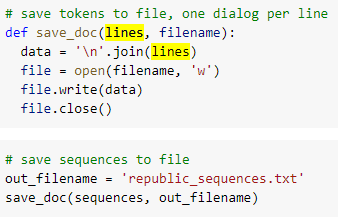
1. Lưu kết quả của mô hình skip-gram về file .txt và file .bin

Ta sẽ chuyển tập văn bản thành các chuỗi chứa 50 từ input và 1 từ output.



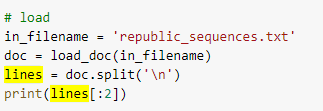
1. Chuyển văn bản thành tập dữ liệu để training

Lưu tập chuỗi này về file.



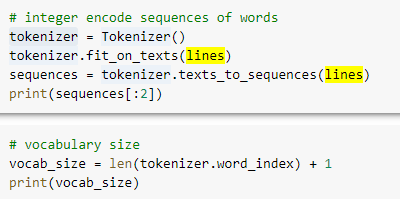
1. Lưu tập dữ liệu về file

Ta load lại tập chuỗi.



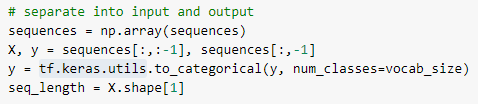
1. Load lại tập dữ liệu

Token tập chuỗi.



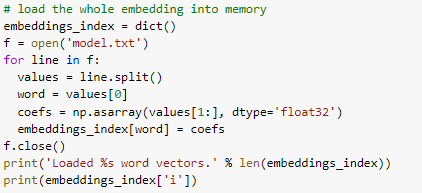
1. Token tập chuỗi

Phân tập dữ liệu thành tập huấn luyện và tập test.



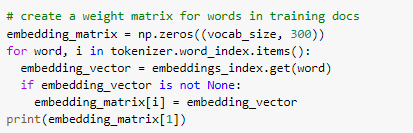
1. Phân tập chuỗi thành tập train và tập test

Ta tiến hành load lại kết quả của mô hình skip-gram.



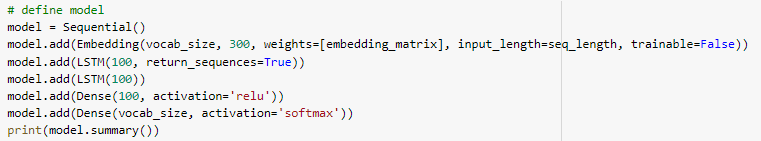
1. Load lại kết quả của mô hình skip-gram

Ta sẽ tiến hành embed cho từng từ trong bộ từ điển và lưu tạm vô 1 ma trận.



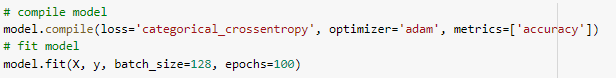
1. Embed cho từng từ và lưu vào ma trận

Ta sẽ khởi tạo mô hình dự đoán từ vựng. Ở đây ta sẽ dùng mô hình mạng LSTM để thực hiện dự đoán.



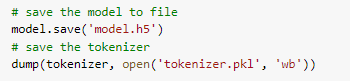
1. Khởi tạo mô hình dự đoán LSTM

Cuối cùng ta sẽ tiến hành huấn luyện mô hình.



1. Tiến hành huấn luyện mô hình

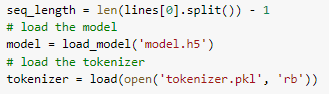
Và lưu lại kết quả của mô hình về file.



1. Lưu kết quả của mô hình về file

Ta sẽ tiến hành thử nghiệm mô hình ngôn ngữ mà ta vừa xây dựng được bằng sinh một đoạn text có n=50 kí tự.

Đầu tiên ta sẽ load lại kết quả xây dựng được của mô hình.



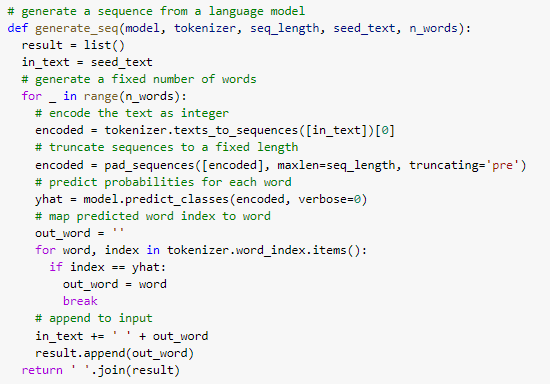
1. Load lại kết quả của mô hình

Chọn 1 câu ngẫu nhiên bất kỳ từ tập dữ liệu đầu vào.



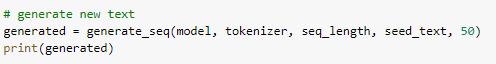
1. Chọn ngẫu nhiên 1 cầu input bất kỳ

Viết hàm sinh đoạn text.



1. Hàm sinh đoạn text

Cuối cùng ta sẽ thực hiện sinh đoạn text.



1. Thực hiện sinh đoạn text

Thông tin về mô hình sau khi ta tiến hành chạy chương trình.

1. Thông tin về mô hình ngôn ngữ LSTM

Kết quả sau khi kết thúc quá trình huấn luyện.

Kết quả sau khi kết thúc quá trình sinh text.

2.8 Tổng kết chương 2

Sau chương 2, ta đã biết cách thức hoạt động của mô hình skip-gram theo chiều tiến (feedforward) lẫn như chiều hiệu chỉnh (backpropogation).

Mô hình skip-gram bắt đầu bằng 1 từ ta gọi từ này là từ trung tâm, sau đó ta sẽ sinh khung ngữ cảnh cho từ này. Ta sẽ quét từng từ một trong khung ngữ cảnh. Ứng với mỗi từ quét ta sẽ tạo vector đầu vào là 1 one-hot vector với từ trung tâm có giá trị 1, ta sẽ nhân ma trận trọng số đầu vào với vector này như là một phép chiếu vector để tạo ra vector . Cuối cùng ta sẽ lấy ma trận đầu ra nhân với h như một phép chiếu vector khác để trả ra kết quả là 1 vector . Vector này chính là kết quả dự đoán cho các từ xung quanh từ trung tâm.

Dựa theo chain rule và hàm mất mát ta có thể tính và hiệu chỉnh sai số cho các trọng số của mô hình.

Ngoài ra trong chương 2, ta đã có thể tự xây dựng thủ công mô hình skip-gram và thực hiện bài toán tìm từ tương thức đơn giản. Không những thế trong chương 2 ta còn biết được cách sử dụng hàm word2vec của thư viện keras để train mô hình skip-gram cũng như sử dụng mô hình LSTM của keras để xây dựng mô hình ngôn ngữ.

CHƯƠNG 3: MÔ HÌNH GLOVE

* 1. Giới thiệu mô hình Glove:

GloVe thuộc mô hình frequency base embedding, dựa vào tần số xuất hiện của từ để tạo ra các vector từ

**GloVe (Global Vector)** là một trong những phương pháp mới để xây dựng vector từ (được giới thiệu vào năm 2014), nó thực chất được xây dựng dựa trên **Co-occurrence Matrix**. **GloVe** có bản chất là xác suất, ý tưởng xây dựng phương pháp này đến từ tỉ số sau:

Trong đó:

**P(k|i)** là xác suất xuất hiện của từ **k** trong ngữ cảnh của từ **i** , tương tự với **P(k|j)** .

Công thức của **P(k|i)** :

Trong đó:

**Xik** : số lần xuất hiện của từ **k** trong ngữ cảnh của từ **i** (hoặc ngược lại).

**Xi** : số lần xuất hiện của từ **i** trong ngữ cảnh của toàn bộ các từ còn lại ngoại trừ **i**.

Ý tưởng chính của **GloVe**: độ tương tự ngữ nghĩa giữa hai từ **i**, **j** có thể được xác định thông qua độ tương tự ngữ nghĩa giữa từ **k** với mỗi từ **i**, **j**, những từ **k** có tính xác định ngữ nghĩa tốt chính là những từ làm cho **Tỷ số** >>1 hoặc xấp chỉ bằng 0. Ví dụ, nếu **i** là “table”, **j** là “cat” và **k** là “chair” thì **Tỷ số**  sẽ khá lớn do “chair” có nghĩa gần hơn với “table” hơn là “cat”, ở trường hợp khác, nếu ta thay **k** là “ice cream” thì **Tỷ số** sẽ xấp xỉ bằng 1 do “ice cream” hầu như chẳng lên quan gì tới “table” và “cat”.

Dựa trên tầm quan trọng của Tỷ số, GloVe khởi đầu bằng việc là nó sẽ tìm một hàm F sao cho nó ánh xạ từ các vec-tơ từ trong vùng không gian V sang một giá trị tỉ lệ với **Tỷ số** . Việc tìm **F** không đơn giản, tuy nhiên, sau nhiều bước đơn giản hóa cũng như tối ưu, ta có thể đưa nó về bài toán hồi quy với việc minimum **cost function** sau:

Trong đó:

**wi**,**wj** là các vector từ.

**bi**, **bj** là các **bias** tương ứng (được thêm vào ở các bước đơn giản hóa và tối ưu).

**Xij**: mục nhập tương ứng với cặp từ **i**,**j** trong **Co-occurrence Matrix**.

Hàm **f** được gọi là **weighting function**, được thêm vào để giảm bớt sự ảnh hưởng của các cặp từ xuất hiện quá thường xuyên, hàm này thỏa 3 tính chất:

Có giới hạn tại 0.

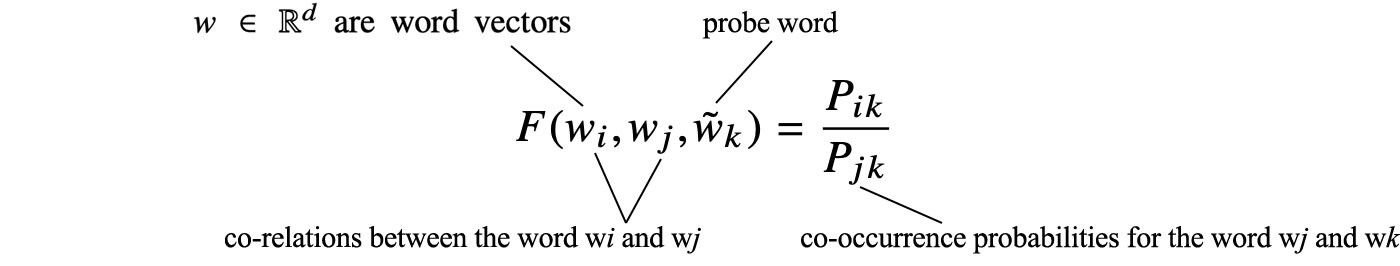
Là hàm không giảm.

Có giá trị nhỏ khi **x** rất lớn.

Thực tế, có nhiều hàm số thỏa các tính chất trên, nhưng ta sẽ lựa chọn hàm số sau:

Với α=3/4

Việc thực hiện minimum **cost function** **J** để tìm ra các vec-tơ từ **wi**, **wj** thể được thực hiện bằng nhiều cách, trong đó cách tiêu chuẩn nhất là sử dụng **Gradient Descent**.



CHƯƠNG 4: FASTTEXT

4.1 Giới thiệu về fastText

fastText là một thư viện mã nguồn mở được sử dụng để huấn luyện các bài toán liên quan đến các vấn đề về xử lý phân loại văn bản cũng như biểu diễn từ. fastText được xây dựng bởi nhóm nghiên cứu mã nguồn mở của Facebook.

fastText là một thư viện hiệu quả trong xử lý các bài toán về word representation hay các bài toán về phân loại câu. fastText được viết bằng C ++ và hỗ trợ đa xử lý trong quá trình huấn luyện mô hình. fastText cho phép ta huấn luyện theo kiểu giám sát (supervise) hoặc không giám sát (unsupervised) của các từ và câu. Các biểu diễn này có thể được sử dụng cho nhiều ứng dụng.

FastText hỗ trợ đào tạo mô hình CBOW và mô hình Skip-gram bằng phương thức negative sampling, softmax hoặc phân cấp softmax.

Mỗi từ được biểu diễn dưới dạng một túi các cụm n-gram ký tự, ví dụ, đối với từ “matter”, với n = 3, sẽ gồm các biểu diễn cho các cụm: <ma, mat, att, tte, ter, er>. Từ khóa <and> sẽ được thêm dưới dạng ký hiệu ranh giới để phân biệt giữa biểu diễn của các cụm n-gram với biểu diễn từ từ của chính nó. Vì vậy, ví dụ, nếu cụm mat nếu tồn tại trong bộ từ điển như là 1 từ thông thường, thì biểu diễn của “matter” sẽ chính là biểu diễn của từ <mat>. Điều này vừa giúp có thể duy trì nghĩa của các từ ngắn đồng thời có thể tận dụng để biểu diễn cho các từ dài hơn. Do đó, điều này cũng cho phép ta nắm bắt được ý nghĩa của các yếu tố tiền tố hoặc hậu tố của các từ.

Độ dài của n-gram ta sử dụng có thể được kiểm soát bằng các cờ -minn và -maxn cho số lượng ký tự tối thiểu và tối đa cho 1 cụm n-gram. Chúng kiểm soát phạm vi của các giá trị để có sinh các cụm n-gram phù hợp. Mô hình vẫn được coi là mô hình túi của các từ vì ngoài việc quét khung ngữ cảnh của các từ thì khung ngữ cảnh sẽ được áp dụng riêng cho từng cụm được phân ra từ đang xét, thứ tự của các cụm này không quan trọng. Ta có thể tắt hoàn toàn chế độ n-gram này bằng cách đặt cả hai biến -minn và -maxn thành 0. Điều này có thể hữu ích khi các từ trong mô hình của ta không phải là các từ dùng để biểu diễn cho các ngôn ngữ đặc thù và trong các trường hợp này việc n-gram các cụm ký tự sẽ không có nghĩa. Trường hợp sử dụng phổ biến nhất là khi bạn sử dụng các id của các từ để làm đại diện thay cho từ.

4.2 Yêu cầu về hệ thống để chạy fastText

fastText xây dựng trên các bản phân phối Mac OS và Linux hiện đại. Vì nó sử dụng các tính năng của C++ 11, nó yêu cầu một trình biên dịch có hỗ trợ C++ 11 tốt. Bao gồm:

* Gcc-4.6.3 hoặc các phiên bản mới hơn.
* Hoặc clang-3.3 hoặc các phiên bản mới hơn.

4.3 Cài đặt fastText để sử dụng trên python

Sau khi ta đã tải fastText về máy như hướng dẫn ở mục 4.2 thì ta tiếp tục sẽ phải chạy một số lệnh để có thể sử dụng fastText trên python.

Ta cài đặt fastText cho python bằng lệnh:



1. Cài đặt fastText cho python

Như vậy là python đã có thể sử dụng được thư viện fastText.

4.4 Xử lý biểu diễn từ bằng fastText

Đầu tiên ta chuẩn bị thư viện:



1. Chuẩn bị thư viện fasttext

Xây dựng và huấn luyện mô hình:



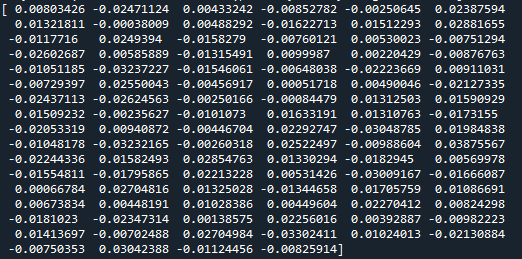
1. Xây dựng và huấn luyện mô hình biểu diễn từ

Ta thử biểu diễn 1 từ bằng fasttext:



1. Biểu diễn từ bằng fasttext

Kết quả biểu diễn của từ minh họa:



1. Kết quả của biểu diễn từ

4.5 Ứng dụng

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>