BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh Viên: Liêu Vương Phúc Minh

Giảng Viên Hướng Dẫn: Ths.Nguyễn Thị Xuân Hương

Hải Phòng - 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

TÌM HIỂU VỀ MÔ HÌNH WORD2VEC VÀ ỨNG DỤNG XỬ LÝ CHO DỮ LIỆU TIẾNG VIỆT

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY
NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh Viên: Liêu Vương Phúc Minh
Giảng Viên Hướng Dẫn: Ths.Nguyễn Thị Xuân Hương

Hải Phòng - 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Liêu Vương Phúc Minh **Mã SV**: 1712111002

Lớp : CT2101M

Ngành : Công nghệ thông tin

Tên đề tài: Tìm hiểu về mô hình Word2Vec và ứng dụng xử lý cho dữ liệu tiếng Việt

LỜI CẢM ƠN

Lời đầu tiên cho em gửi lời cảm ơn sâu sắc đến gia đình, người thân của em. Đã động viên, giúp đỡ, cổ vũ, tạo cho em thêm động lực để em có thể hoàn thành đồ án trong thời gian được giao.

Em xin gửi lời cảm ơn đến Ban Giám Hiệu Trường Đại học Quản lý và Công nghệ Hải Phòng, các Ban, Ngành đã hỗ trợ hết mức tạo điều kiện tốt nhất để em có thể đăng kí đồ án tốt nghiệp.

Em xin cảm ơn đến các thầy, các cô Khoa Công nghệ thông tin, Trường Đại học Quản lý và Công nghệ Hải Phòng, đã giúp em có những kiến thức cực kì bổ ích trong vòng 4 năm vừa qua, giúp em có được nền tảng kiến thức vững chắc để em có thẻ thực hiện được đồ án.

Em xin gửi lời cảm ơn chân thành đến cô Ths. Nguyễn Thị Xuân Hương, đã dành rất nhiều thời gian công sức, cả về vật chất và tinh thần giúp em có thể thể hoàn thành được đồ án một cách trơn tru nhất

Em xin chân thành cảm ơn!

Hải Phòng, ngày.....tháng.....năm 2021

Sinh viên

Liêu Vương Phúc Minh

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ VÀ CÁC BẢNG

BẢNG CÁC TỪ VIẾT TẮT

Viết tắt	Đầy đủ	Ý nghĩa
RNN	Recurrent Neural Network	Mạng neural hồi quy
ANN	Artificial Neural Network	Mạng neural nhân tạo
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
LSTM	Long short-term memory	Mạng neural cải tiến giải quyết vấn đề phụ thuộc từ quá dài
CNN	Convolutional Neural network	Mạng neural tích chập
SVM	Support Véc-tơ Machine	Máy véc-tơ hỗ trợ
NCE	Noise Contrastive Estimation	Ước lượng tương phản nhiễu

MỞ ĐẦU

Từ xa xưa, con người ta đã sử dụng ngôn ngữ là hình thức giao tiếp trong cuộc sống hằng ngày. Ngôn ngữ của con người là một hệ thống được xây dựng một cách đặc trưng để chuyển tải ý nghĩa, và không được tạo ra bằng bất cứ loại biểu hiện hình thể nào. Để thể hiện nội dung của mình muốn đề cập đến, chúng ta có thể sử dụng từ ngữ hoặc dấu hiệu để diễn tả, điều đó được thể hiện qua lời nói, chữ viết hoặc các hình ảnh.

Với sự phát minh ra máy tính để trợ giúp con người trong rất nhiều các hoạt động trong đời sống, kinh tế, chính trị, xã hội, v.v. con người mong muốn bằng cách nào đó có thể dạy cho máy tính hiểu được thứ ngôn ngữ của mình để trong các bài toán để thực hiện hiệu quả những nhiệm vụ liên quan đến ngôn ngữ của như: tương tác giữa người và máy, cải thiện hiệu quả giao tiếp giữa con người với con người, hoặc đơn giản là nâng cao hiệu quả xử lý văn bản và lời nói. Với những yêu cầu đó, Xử lý ngôn ngữ Tự nhiên ra đời tập trung vào các nghiên cứu trên ngôn ngữ của con người.

Đã có nhiều bài toán được nghiên cứu trong lĩnh vực xử lý ngôn ngữ tự nhiên như: kiểm tra lỗi chính tả, tìm kiếm từ khóa, tìm từ đồng nghĩa, phân tích thông tin từ các trang web, tài liệu, phân tích ngữ nghĩa, khuyến nghị, hệ thống hỏi đáp, phân tích quan điểm người dùng, v.v.

Thách thức lớn nhất trong Xử lý ngôn ngữ tự nhiên là đó là làm thế nào máy tính có thể hiểu được ý nghĩa của một từ. Không giống như con người, máy tính chỉ có thể đọc được và mã hóa dữ liệu dưới dạng các dãy bit. Vì thế người ta sẽ tìm cách ánh xạ từ một dãy các từ thành các dãy số mà máy tính có thể "hiểu" đươc. Do đó việc nghiên cứu mã hóa từ thành các véc-tơ là cần thiết. Một trong những phương pháp đó là đưa từ vào một không gian mới người ta thường gọi là nhúng từ (embedding). Các mã hóa đơn giản là one-hot [1] tức là tạo một bộ từ vựng (Vocabulary) cho dữ liệu và mã hóa các từ trong tài liệu (document) thành những véc-tơ, nếu từ đó có trong văn bản thì mã hóa là 1 còn không có sẽ là 0. Kết quả tạo ra một ma trận thưa (sparse matrix), tức là ma trận mà hầu hết là 0. Mã hóa này có nhiều nhược điểm đó là thứ nhất là số chiều của nó rất lớn (NxM, N là số tài liệu còn M là số từ vựng, thứ hai là các từ không có quan hệ với nhau. Điều đó dẫn đến người ta tạo ra một mô hình mới có tên là nhúng từ (embedding), trong đó các từ sẽ có quan hệ với nhau về ngữ nghĩa tức là ví dụ như Paris-Tokyo, nam-nữ, con trai – con gái, v.v. những cặp từ này sẽ có khoảng cách gần nhau hơn trong không gian nhúng từ [2][3].

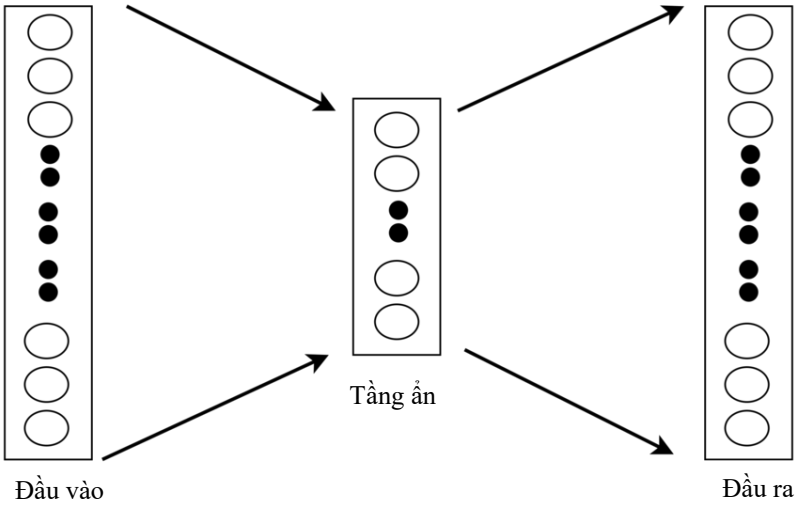
Nhiều mô hình nhúng từ đã được phát triển, nổi bật trong đó là Phân tích ngữ nghĩa ẩn (Latent Semantic Analysis -LSA**)** [4] và Phân bố Dirichlet ẩn (Latent Dirichlet Allocation-LDA) [5]. Về cơ bản, các mô hình này sử dụng từ trong từ điển như là đầu vào và biến chúng thành những véc-tơ trong không gian với chiều thấp hơn, sau đó thay đổi trọng số, các tham số thông qua học lan truyền ngược (back-propagation) để tạo thành lớp nhúng.

Word2Vec hay chuyển đổi từ thành Vec mô hình chuyển từ thành véc-tơ được phát triển và ứng dụng rộng rãi trong thời gian gần đây. Hầu hết trong các mô hình học sâu hiện nay như: CNN, RNN, LSTM, v.v.sử dụng đầu vào là các véc-tơ từ và đã được chứng minh cho kết quả tốt nhất trong nhiều bài toán xử lý ngôn ngữ tự nhiên. Do đó, em đã chọn đề tài "Mô hình Word2Vec và ứng dụng xử lý cho dữ liệu tiếng Việt" làm đồ án tốt nghiệp của mình trong đó em tìm hiểu về phương pháp véc- tơ hóa từ, một số phương pháp học sâu và ứng dụng cho bài toán phân loại bình luận tiếng Việt.

CHƯƠNG 1 WORD2VEC

1.1. Giới thiệu về Word2vec

Word2vec là phương pháp biểu diễn một từ dưới dạng một phân bố quan hệ với các từ còn lại. Mỗi từ được biểu diễn bằng một véc-tơ có các phần tử mang giá trị là phân bố quan hệ của từ này đối với các từ khác trong từ điển. Năm 2013, Google đã bắt đầu với dự án word2vec với dữ liệu được sử dụng từ Google News. Bộ dữ liệu được coi là đồ sộ nhất cho tới bây giờ với 100 tỷ từ [6].



Hình 1.1. Mô hình Word2vec

Word2vec là một mạng neural 2 lớp với duy nhất 1 tầng ẩn, lấy đầu vào là một ngữ liệu lớn và sinh ra không gian véc-tơ (với số chiều khoảng vài trăm), với mỗi từ duy nhất trong ngữ liệu được gắn với một véc-tơ tương ứng trong không gian. Các véc-tơ từ được xác định trong không gian véc-tơ sao cho những từ có chung ngữ cảnh trong ngữ liệu được đặt gần nhau trong không gian

Giả sử bộ từ vựng của ta chỉ có 5 từ: Vua, Hoàng hậu, Đàn ông, Phụ nữ và Trẻ con. Ta sẽ mã hóa cho từ Hoàng hậu như sau:

0	**1**	**0**	**0**	**0**
Vua	Hoàng hậu	Đàn ông	Phụ nữ	Trẻ con

Hình 1.1.1. Mã hóa 1-of-N

Trong Word2Vec, một biểu diễn phân tán của một từ được sử dụng. Tạo ra một véc-tơ với kích thước vài trăm chiều. Mỗi từ được biểu diễn bởi tập các trọng số của từng phần tử trong nó. Vì vậy, thay vì sự kết nối 1-1 giữa một phần tử trong véc-tơ với một từ, biểu diễn từ sẽ được dàn trải trên tất cả các thành

phần trong véc-tơ, và mỗi phần tử trong véc-tơ góp phần định nghĩa cho nhiều từ khác nhau.

Nếu ta gán nhãn các kích thước cho một véc-tơ từ giả thuyết, nó trông giống như hình sau:

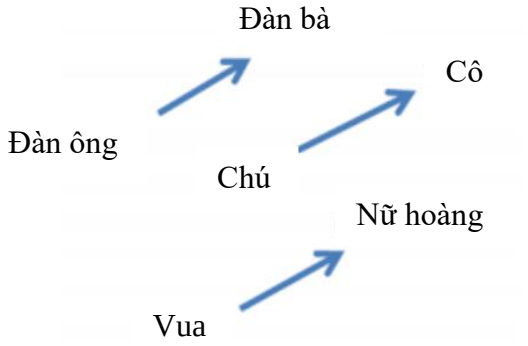
		Vua	Hoàng hậu	Phụ nữ	Công chúa
Hoàng gia		0.99	0.99	0.02	0.98
Nam tính		0.99	0.05	0.01	0.02
Nữ tính		0.05	0.93	0.999	0.94
Tuổi		0.7	0.6	0.5	0.1

Hình 1.1.2. Giả thuyết về biểu diễn 1 véc-tơ có gán nhãn các kích thước

Như vậy một véc-tơ sẽ đại diện cho ý nghĩa được tóm tắt của một từ. Và ta sẽ thấy tiếp theo, đơn giản bằng việc kiểm tra một tập văn bản lớn, nó có thể học các véc-tơ từ, ta có thể nắm bắt mối quan hệ giữa các từ theo một cách đáng ngạc nhiên. Ta cũng có thể sử dụng các véc-tơ các đầu vào cho một mạng Nơ-ron.

Mô hình Skip-Gram là mô hình dự đoán từ xung quanh dựa vào một từ cho trước. Ví dụ cho một câu: “I love you so much”. Khi dùng một cửa sôt tìm kiếm có cỡ bằng 3 ta thu được: {(I, you), love}, {(love, so), you}, {(you, much), so}. Nhiệm vụ của nó là khi cho một từ trung tâm ví dụ là love thì phải dự đoán các từ xung quang là i, you.

Mô hình CBOW (Continuous Bag of Word) khác với mô hình Skip-Gram là dựa vào ngữ cảnh các từ xung quanh để đoán từ ở giữa. Trên thực tế hai mô hình này thường được lựa chọn để véc-tơ hóa từ trong văn bản. Mô hình CBOW huấn luyện nhanh hơn so với Skip-Gram nhưng độ chính xác kém hơn Skip-Gram và ngược lại.



Hình 1.1.3. Phân bố quan hệ giữa từ trong word2vec

Ví dụ bài toán kinh điển Vua + Đàn ông – Đàn bà =? Việc nhúng các từ trong không gian véc-tơ cho thấy sự tương tự giữa các từ. Giả sử như tại hình 3.1 là một sự khác biệt về mặt giới tính giữa các cặp từ (“Đàn ông”, “đàn bà”), (“Chú”, “Cô”), (“Vua”, “Nữ hoàng”)

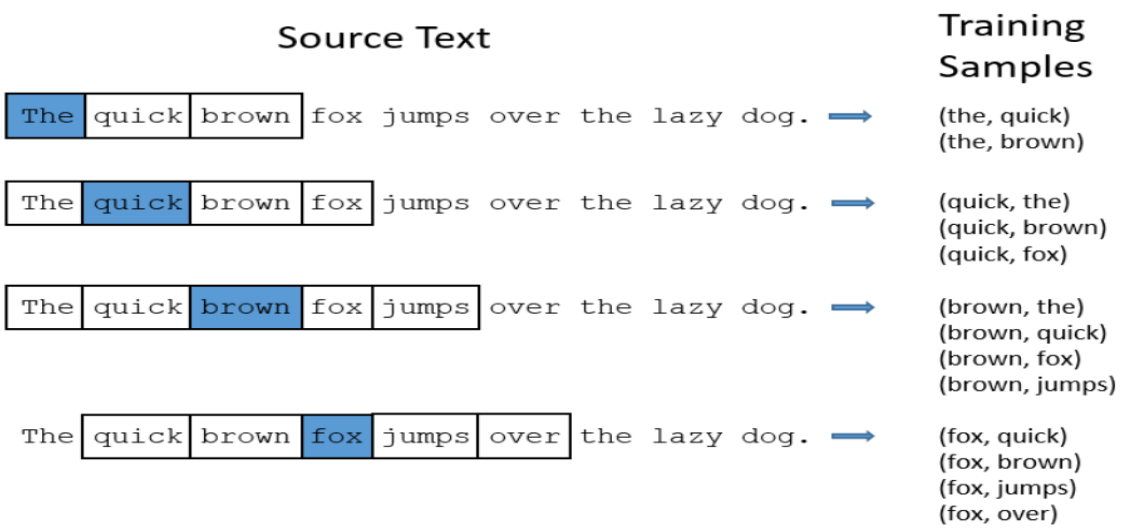
W(“đàn bà”) – W(“Đàn ông”) ≈ W(“Cô”) – W(“Chú”)

W(“đàn bà”) – W(“Đàn ông”) ≈ W(“Nữ hoàng”) – W(“Vua”)

Từ đó, kết quả của Vua + Đàn ông – Đàn bà = Nữ hoàng. Để xây dựng được véc-tơ mô tả phân bố quan hệ với tập từ điển, bản chất mô hình Word2Vec sử dụng một mạng nơ-ron đơn giản với một lớp ẩn. Sau khi được huấn luyện trên toàn bộ tập văn bản, toàn bộ lớp ẩn sẽ có giá trị mô hình hóa quan hệ của từ trong tập văn bản được huấn luyện ở mức trừu tượng. Trong ngữ cảnh, từ sẽ được huấn luyện việc sử dụng thuật toán CBOW và Skip-Gram, mô hình Skip-Gram cho kết quả tốt hơn với tập dữ liệu lớn.

Khi sử dụng mô hình Skip-Gram thì đầu vào là một từ trong câu, thuật toán sẽ nhìn vào những từ xung quanh nó. Giá trị số từ xung quanh nó được xét gọi là “window size”. Một window size bằng 5 có nghĩa sẽ xét 5 từ trước nó và 5 từ sau nó. Xác suất đầu ra sẽ liên quan tới khả năng tìm thấy các từ xung quanh từ hiện tại đang xét. Ví dụ nếu đã huấn luyện với từ đầu vào là “bóng đá”, xác suất đầu ra sẽ cao hơn đối với những từ “quả bóng” hay “cầu thủ” so với các từ không liên quan như “dưa hấu” hay “Nam Phi”. Thực hiện huấn luyện mạng nơ-ron này bằng cách cho xét từng cặp từ gồm từ được xét và từ xung quanh nó.

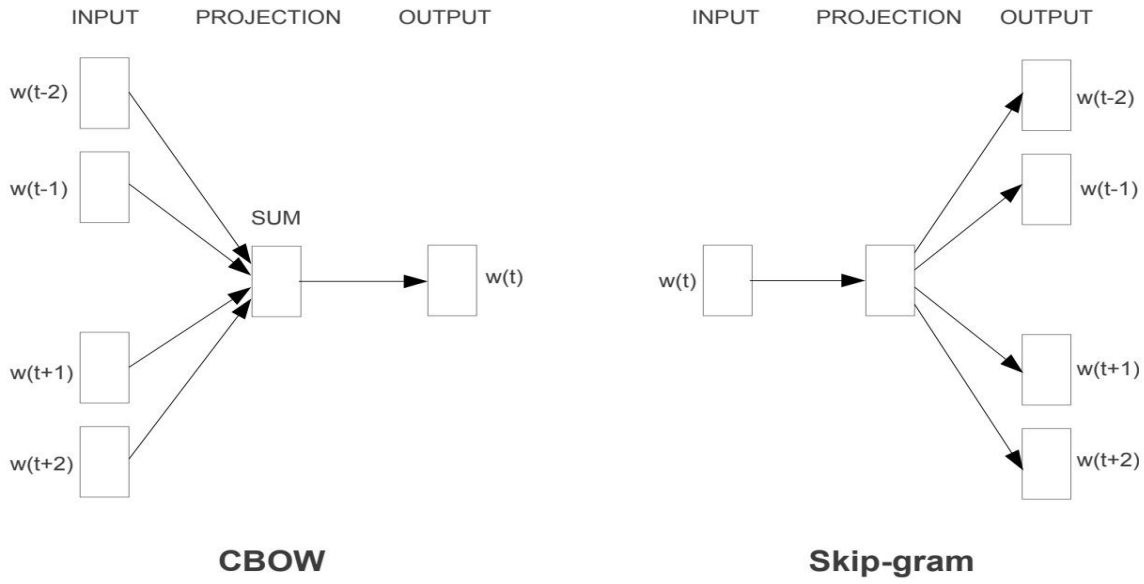
Xét câu “The quick brown fox jumps over the lazy dog” với window size bằng 2. Từ được bôi đậm là từ đầu vào.



Hình 1.1.4. Mô hình Skip-Gram trong Word2vec

Mạng nơ-ron sẽ được huấn luyện từ số liệu thống kê số lần cặp ghép xuất hiện. Do vậy, mạng sẽ nhận được nhiều cặp mẫu huấn luyện (“bóng đá”,” cầu thủ”) hơn là (“bóng đá”,” dưa hấu”). Khi quá trình huấn luyện kết thúc, nếu đưa ra từ “bóng đá” như đầu vào thì sẽ có một giá trị xác suất cao hơn cho “cầu thủ” và “quả bóng” so với “dưa hấu” và “Nam Phi”.

1.2. Chi tiết mô hình



Hình 1.2.1. Mô hình CBOW và SKIP-GRAM

1.2.1. Mô hình CBOW và Skip-Grams

Mô hình CBOW: ý tưởng chính là dựa vào các từ xung quanh để dự đoán từ ở giữa. CBOW có điểm thuận lợi là huấn luyện mô hình nhanh hơn so với mô

hình Skip-Gram, thường cho kết quả tốt hơn với các từ thường xuất hiện trong văn bản [7][8].

Skip-Gram dùng từ đích để dự đoán các từ xung quanh. Skip-Gram huấn luyện chậm hơn. Do đặc trưng của mô hình nên khả năng véc-tơ hóa cho các từ ít xuất hiện tốt hơn CBOW.

1.2.2. Lớp làm giả

Nhiệm vụ lớp làm giả sẽ huấn luyện mạng nơ-ron để làm công việc sau. Cho một từ đầu vào (ở giữa một câu), nhìn vào các từ gần đó và chọn một từ ngẫu nhiên. Mạng sẽ cho biết xác suất cho mỗi từ trong từ vựng là "từ gần" đã chọn. "Gần" ở đây phụ thuộc vào kích thước của cửa sổ (window size) cho thuật toán. Kích thước cửa sổ điển hình có thể là 5, có nghĩa là 5 từ phía sau và 5 từ phía trước (tổng cộng 10 từ).

Các xác suất đầu ra sẽ liên quan đến khả năng nó tìm thấy mỗi từ vựng gần từ đầu vào của câu. Ví dụ, nếu bạn cho mạng được đào tạo từ đầu vào "SoViet", xác suất đầu ra sẽ cao hơn nhiều với các từ như "Union" và "Russia" so với các từ không liên quan như "watermelon" và "kangaroo".

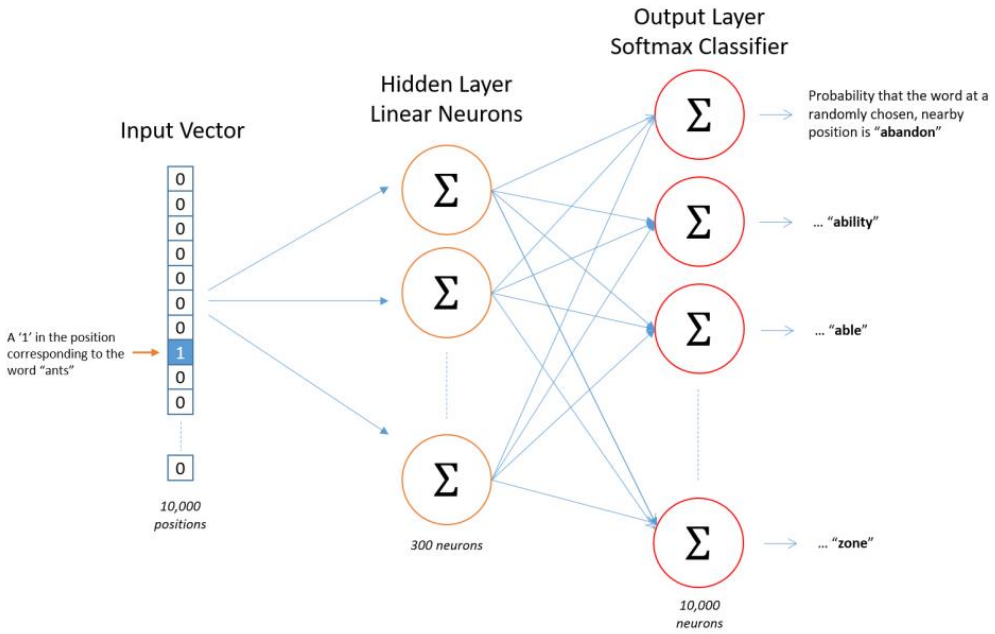
Trước tiên, xây dựng từ điển của các từ trong dữ liệu huấn luyện, giả sử từ vựng gồm 10.000 từ không trùng lặp. Cho "ants" là một từ đầu vào được coi như một véc-tơ one-hot. Véc-tơ này sẽ có 10.000 thành phần (một cho mỗi từ trong từ điển) và đặt "1" ở vị trí tương ứng với từ "ants" và 0 ở tất cả các vị trí khác.

1.2.3. Kiến trúc mạng nơ-ron:

Trước hết, đưa từ vào mạng nơ-ron một lớp ẩn kể trên. Để có thể huấn luyện được, từ được véc-tơ hóa để cho vào mạng. xây dựng kho từ điển từ tập dữ liệu văn bản sau đó sử dụng one-hot-véc-tơ để diễn tả từng từ trong kho từ điển.

Giả sử, từ điển gồm 10.000 từ, véc-tơ one-hot sẽ gồm 10.000 thành phần đại diện cho mỗi từ trong từ điển trong đó toàn bộ giá trị bằng 0, chỉ có chỉ số tương ứng với vị trí của từ trong từ điển có giá trị bằng 1.

Ví dụ từ "ants" sẽ biểu diễn bằng véc-tơ 10.000 phần tử. gồm toàn số 0, duy nhất số 1 tại vị trí tương ứng với từ "ants" trong từ điển.



Hình 1.2.3. Mô hình mạng nơ-ron 1 lớp ẩn của Word2vec

Đầu vào là one-hot-véc-tơ mỗi word sẽ có dạng x_1, x_2...x_v trong đó V là số từ vựng trong từ điển, là một véc-tơ trong đó mỗi từ sẽ có giá trị 1 tương đương với chỉ số trong từ điển và còn lại sẽ là 0.

Ma trận trọng số (Weight matrix) giữa đầu vào và lớp ẩn (hidden layer) là ma trận W(có số chiều là VxN) có hàm kích hoạt (active function) là tuyến tính (linear), trọng số giữa lớp ẩn và đầu ra là W, ′W′ (có số chiều là NxV) hàm kích hoạt của đầu ra là soft max.

Mỗi hàng của W là véc-tơ N chiều đại diện cho υ_w là mỗi từ trong lớp đầu vào (input layer). Mỗi hàng của W là $\upsilon\frac{t}{w}$. Với đầu vào là 1 one-hot véc-tơ (sẽ có dạng 000100) chỉ có 1 phần tử bằng 1 nên.

$$h = W^T x = v_w^T$$

Từ lớp ẩn đến đầu ra là ma trận $W'=w'_{i,j}$. Ta tính điểm (score) u_i cho mỗi từ trong từ điển.

$$u_j = v'_{w_j} h$$

Trong đó υ_{wj} là véc-tơ cột j trong W′. Tiếp đó ta sử dụng hàm Softmax

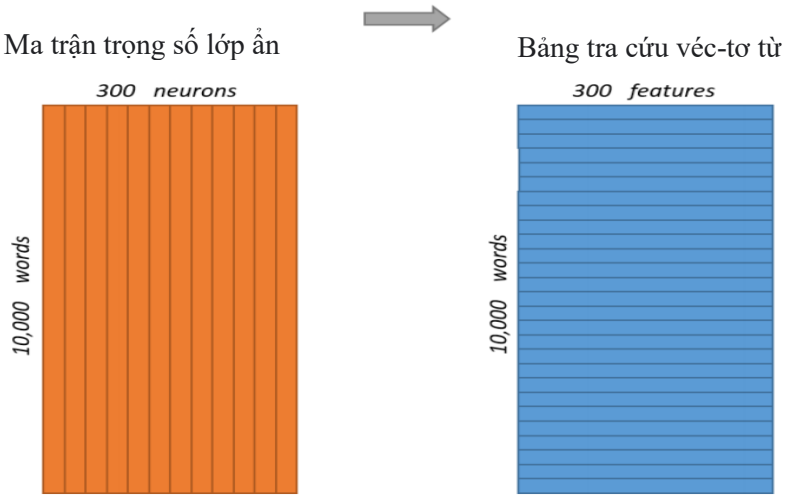
$$P(w_j|w_I) = y_i = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j'})} == \frac{exp(v'^T_{w_j} v_{w_I})}{\sum_{j'=1}^{V} exp(v'^T_{w_{j'}} v_{w_I})}$$

Trong đó υ_w và $\upsilon_{w'}$là 2 véc-tơ đại diện cho từ w đến từ ma trận W và W′

Người ta dùng hợp lý cực đại (maximum like hood) với kỹ thuật giảm chiều gradient (gradient descent) để giải quyết bài toán này nhưng vì số lượng từ vựng lớn nên tính toán mẫu số nó tính trên toàn bộ từ điển là rất lớn nên người ta dùng 2 phương pháp giải quyết là SoftMax phân cấp (Hierarchical SoftMax) hoặc lấy mẫu phủ đinh (Negative Sampling).

1.2.4. Lớp ẩn

Lớp ẩn giả sử gồm 300 nơ-ron, thường không sử dụng hàm kích hoạt, nhưng đầu ra thì sử dụng hàm SoftMax để lấy giá trị xấp xỉ. Đầu ra sẽ là véc-tơ cũng là một véc-tơ có độ lớn 10.000 và giá trị tương ứng với mỗi vị trí là xác suất xuất hiện gần từ đã chọn của từ gần vị trí đó. Kích thước 300 nơ-ron ở lớp ẩn là một siêu tham số (hyperparameter) của mô hình, nó được gọi là số chiều hay số đặc trưng của word2vec. Con số 300 được Google sử dụng trong mô hình huấn luyện từ tập ngữ liệu Google News. Giá trị siêu tham số có thể được thay đổi sao cho phù hợp với mô hình, dữ liệu của người nghiên cứu.



Hình 1.2.4. Ma trận trọng số của lớp ẩn của mô hình word2vec

Kết quả của việc huấn luyện trên toàn tập ngữ liệu là tìm ra ma trận trọng số tại lớp ẩn. Với đầu vào của mô hình là 1 từ được biểu diễn dưới dạng one-hot véc-tơ tức là một véc-tơ có các giá trị toàn bằng 0, chỉ có một vị trí bằng 1 tương

ứng với vị trí của từ đầu vào theo thứ tự từ điển. Việc nhân véc-tơ one-hot đầu vào với ma trận trọng số bản chất là việc tìm kiếm trên ma trận trọng số một véc-tơ đặc trưng có chiều dài bằng số chiều bằng số chiều của ma trận trọng số.

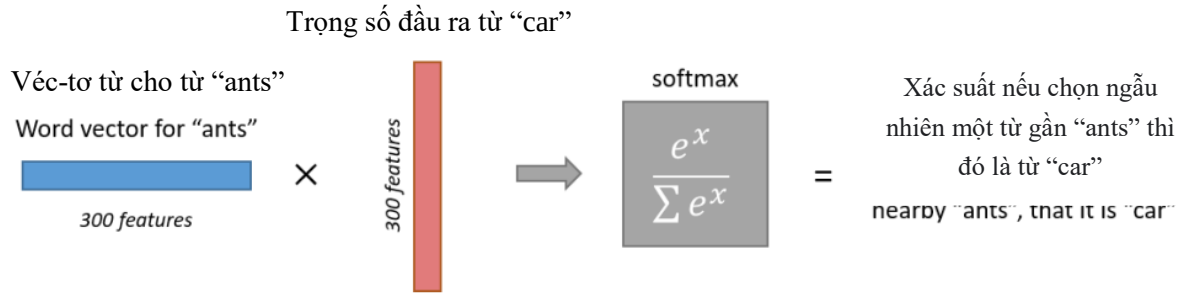
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Hình 1.2.4.1. Lớp ẩn của mô hình hoạt động như một bảng tra cứu

1.2.5. Lớp đầu ra

Đầu ra của mô hình Word2vec là một bộ phân loại sử dụng hàm SoftMax để tính xác suất. Ưu điểm của hàm SoftMax là luôn tạo giá trị xác suất dương và tổng tất cả các xác suất thành phần là bằng 1.

Giả sử tính mối tương quan giữa từ "ants" và từ "car", hai từ này sẽ được véc-tơ hóa dựa vào ma trận trọng số của lớp ẩn đã huấn luyện. Đầu ra qua hàm SoftMax sẽ có ý nghĩa là xác suất từ "car" xuất hiện gần từ được chọn "ants"



Hình 1.2.5. Mối qua hệ giữ từ "ants" và từ "car"

1.3. Nhúng từ (Word Embedding)

Nhúng từ là một biểu diễn véc-tơ số của văn bản trong kho ngữ liệu ánh xạ đến từng từ trong kho từ vựng với một tập hợp các vectơ có giá trị thực trong không gian N chiều được xác định trước.

Các biểu diễn véc-tơ có giá trị thực này cho mỗi từ trong kho từ vựng được học thông qua các kỹ thuật có giám sát như mô hình mạng nơron được đào tạo về các nhiệm vụ như phân tích cảm tính và phân loại tài liệu hoặc thông qua các kỹ thuật không được giám sát như phân tích thống kê tài liệu [9][10].

Word Embeddings cố gắng nắm bắt ý nghĩa ngữ nghĩa, ngữ cảnh và cú pháp của từng từ trong kho từ vựng dựa trên cách sử dụng những từ này trong câu. Các từ có ý nghĩa ngữ nghĩa và ngữ cảnh tương tự cũng có các biểu diễn véc-tơ tương tự trong khi đồng thời mỗi từ trong từ vựng sẽ có một tập biểu diễn véc-tơ duy nhất.

1.4. Tính hiệu quả

Các bản nhúng từ được tạo bởi Word2Vec không chỉ được nén nhiều hơn so với các bản mã hóa một lần, mà còn chứa thông tin theo ngữ cảnh về từ được nhúng! Thông tin này có thể được hiển thị bằng khả năng thao tác nhúng. Ví dụ: lấy từ nhúng cho 'lớn nhất', trừ từ nhúng cho 'lớn' và thêm từ nhúng cho 'nhỏ' sẽ trả về từ nhúng cho 'nhỏ nhất'!

Cả hai mô hình Word2Vec cũng có thể tìm hiểu mối quan hệ giữa các khu vực địa lý. Bảng mã Skip-Gram cho từ “Athens” rất gần với từ nhúng cho “Hy Lạp”. Tương tự, cách nhúng của Skip-Gram cho “Chicago” rất giống với “Illinois”. Điều đáng chú ý là các nhúng từ có thể nắm bắt thông tin tốt như thế nào!

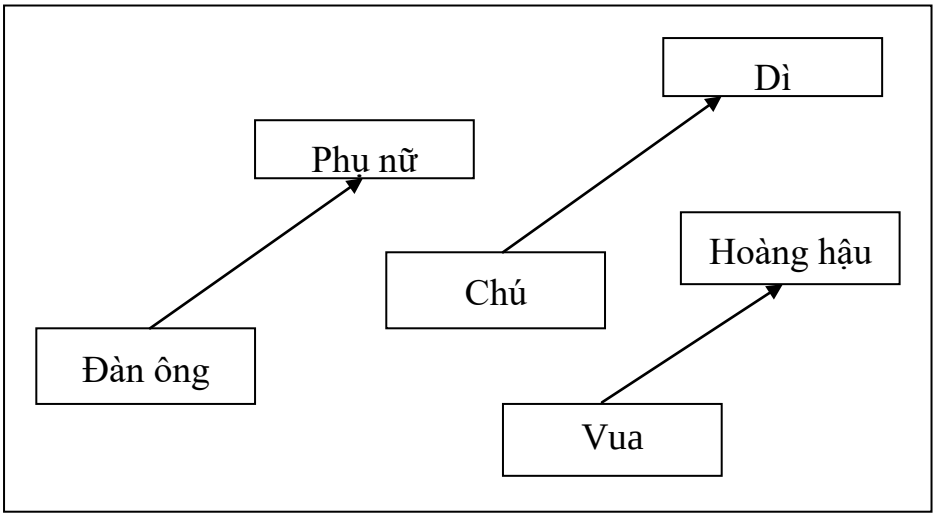
Như vậy, các bản nhúng của Word2Vec cực kỳ tốt trong việc mã hóa các nghĩa sâu hơn của từ. Họ có thể nắm bắt các kết nối phức tạp, đan xen giữa các từ khác nhau, một kỳ công là một tiến bộ to lớn trong việc tạo ra máy tính có thể xử lý ngôn ngữ tự nhiên một cách hoàn hảo. Máy tính đang phát triển nhanh chóng và Word2Vec là bằng chứng về điều đó.

1.5. Lập luận với Véc-tơ từ

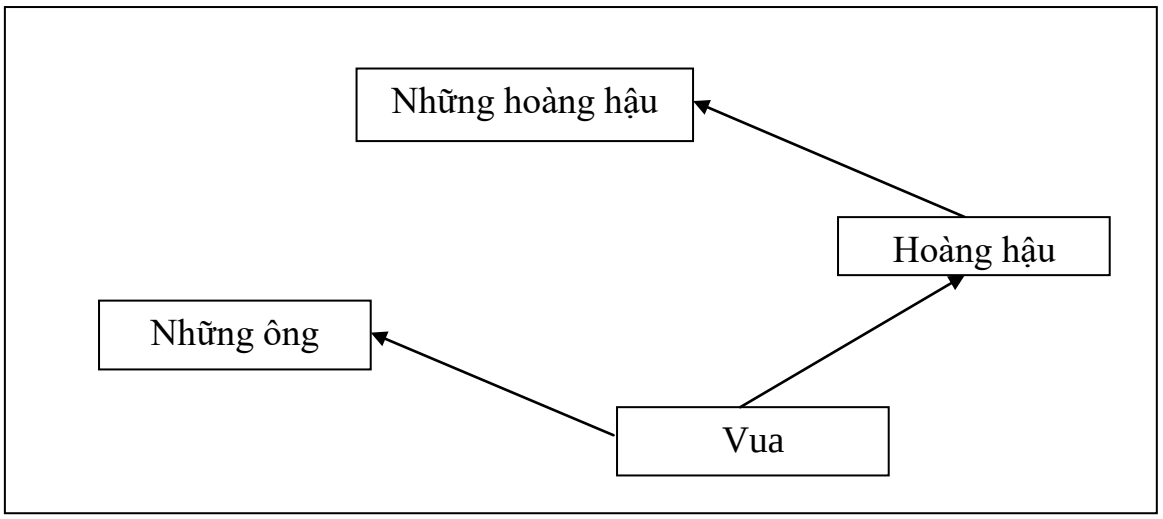
Ta thấy rằng các đại diện từ được nghiên cứu trong thực tế nắm bắt quy tắc cú pháp và ngữ nghĩa có ý nghĩa theo một cách rất đơn giản. Cụ thể, các quy tắc được quan sát các giá trị bù véc-tơ không đổi giữa các cặp từ chia sẻ một mối quan hệ đặc biệt. Ví dụ, nếu ta ký hiệu véc-tơ cho chữ i là X_i, và tập trung vào mối quan hệ số ít/số nhiều, ta sẽ quan sát thấy rằng X_{apple} - X_{apples} ≈ X_{car} X_{cars}, X_{family} - $X_{families}$ ≈ X_{car} - X_{cars}, v.v. Ta thấy rằng đây cũng là trường hợp cho một loạt các quan hệ ngữ nghĩa được đo bởi mối quan hệ tương đồng.

Các véc-tơ rất tốt khi trả lời câu hỏi tương tự dạng a là dành cho b như c là dành cho? Ví dụ, Đàn ông (đàn ông) là dành cho Đàn bà (phụ nữ) như uncle

(chú) là dành cho? Aunt (thím, dì) sử dụng một phương pháp các giá trị bù véc-tơ đơn giản dựa vào khoảng cách cousin.



Hình 1.5.1. Giá trị bù véc-tơ cho 3 cặp từ mô phỏng mối quan hệ về giới

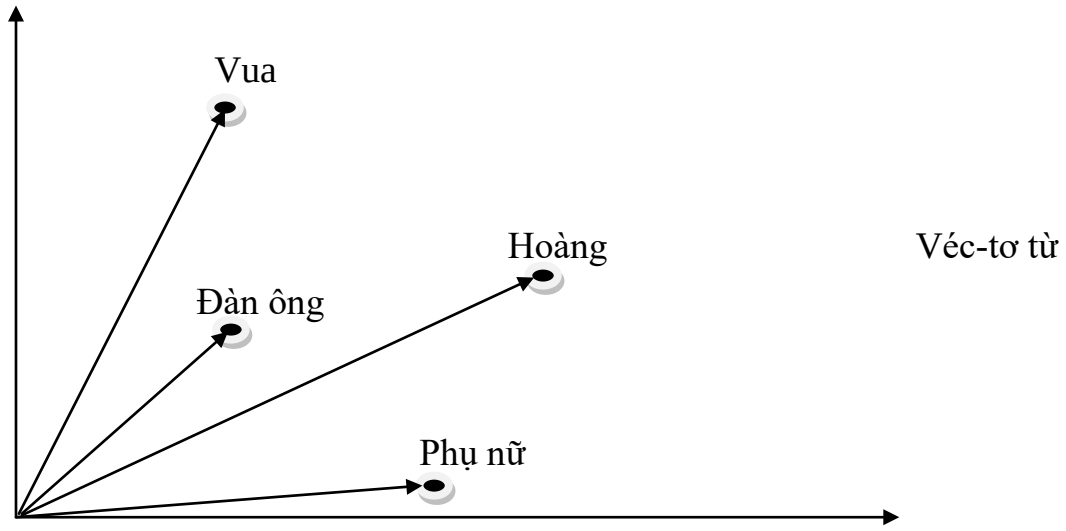


Hình 1.5.2. Mối quan hệ giữa số nhiều và số ít

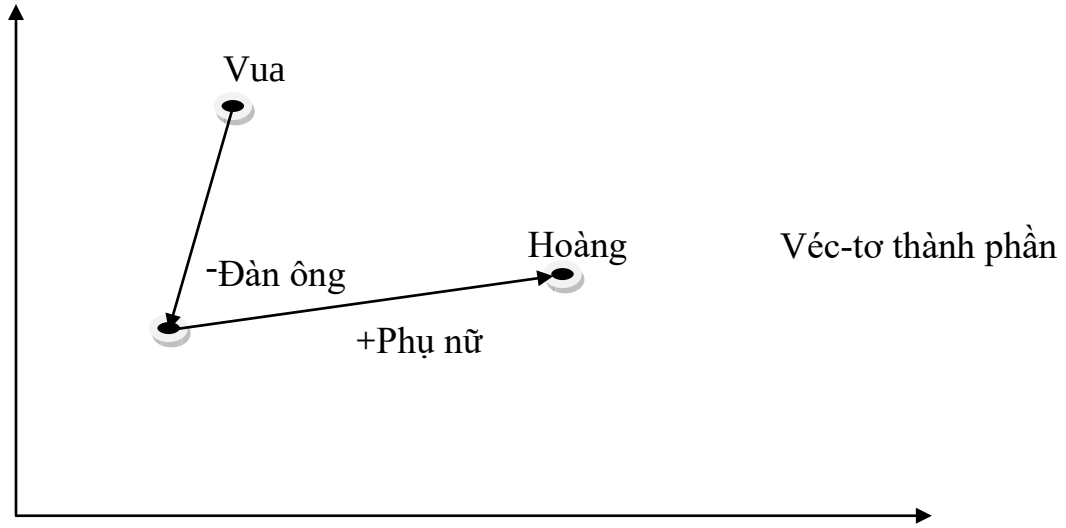
Đây là sự hợp thành véc-tơ cũng cho phép ta trả lời câu hỏi "Vua – Đàn ông + Phụ nữ =?" và đi đến kết quả "Hoàng hậu"! Tất cả đều thực sự đáng chú ý khi bạn nghĩ rằng các kiến thức này chỉ đơn giản là xuất phát từ việc nhìn vào rất nhiều từ trong ngữ cảnh (ta sẽ thấy ngay) mà không có thông tin khác được cung cấp về ngữ nghĩa của nó.

Khá là ngạc nhiên để nhận thấy rằng sự giống nhau của các đại diện từ nằm ngoài các quy luật ngữ nghĩa đơn giản. Sử dụng kỹ thuật về giá trị bù từ nơi các phép toán đại số đơn giản được thực hiện trên các véc-tơ từ, điều đó đã được chỉ

ra, ví dụ véc-tơ (“Vua”) - véc-tơ (“Đàn ông”) + véc-tơ (“Phụ nữ”) cho kết quả trong một véc-tơ gần nhất với đại diện véc-tơ của từ “Hoàng hậu”.



Hình 1.5.3. Véc-tơ từ cho Vua, Đàn ông, Hoàng hậu và Phụ nữ

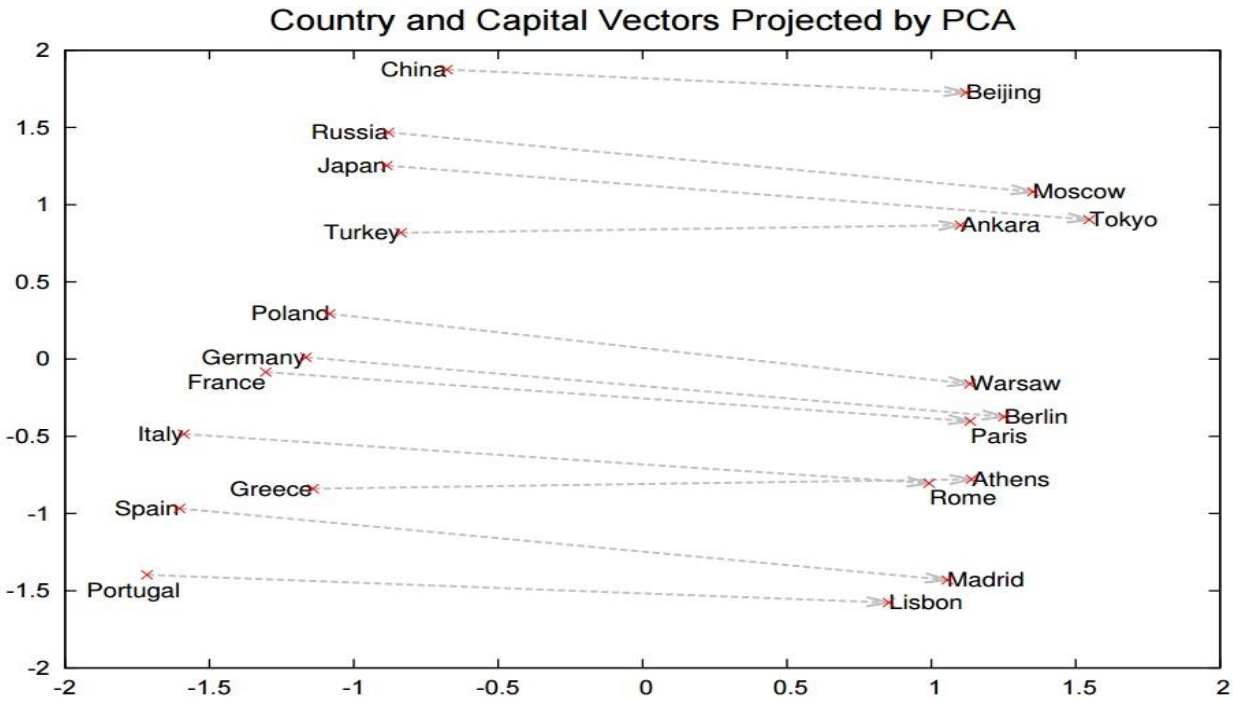


Hình 1.5.4. Kết quả sự cấu thành Véc-tơ Vua – Đàn ông + Phụ nữ =?

Quan hệ	Ví dụ 1	Ví dụ 2	Ví dụ 3

France – Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
Big – bigger	Small: larger	Cold: colder	Quick: quicker
Miami – Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein – scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy – France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
Copper – Cu	Zinc: Zn	Gold: Au	Uranium: plutonium
Berlusconi – Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft – Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft – Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Bảng 1.5.5. Ví dụ về các mối quan hệ giữ các cặp từ



Hình 1.5.6. Mối quan hệ thủ đô – quốc gia

Dưới đây là mối quan hệ thủ đô-quốc gia (country-capital city) trông giống như 2 phép chiếu nhận diện hình ảnh 2 chiều:

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airline
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogel’s	Amazon

Bảng 1.5.7. Ví dụ của các dạng câu hỏi “a là dành cho b như c là dành cho?”

Ta cũng có thể sử dụng thêm thành phần tương ứng của các thành phần véc-tơ để đặt câu hỏi chẳng hạn như 'Đức + các hãng hàng không” và bằng cách nhìn vào các dấu hiệu gần nhất với véc-tơ phức hợp đưa ra được câu trả lời ấn tượng:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress

Koruna	Hanoi	Airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	Carrier Lufthansa	Volga River	Vanessa Paradis
Polish zloty	Viet Nam	Flag Carrier Lufthansa	Upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Bảng 1.5.8. Trả lời cho câu hỏi dạng "a là dành cho b như c là dành cho?"

Véc-tơ từ với các mối quan hệ ngữ nghĩa như vậy có thể được sử dụng để cải thiện nhiều ứng dụng NLP hiện có, chẳng hạn như biên dịch bằng máy, hệ thống tìm kiếm thông tin và hệ thống câu hỏi/trả lời, và còn có thể cho phép các ứng dụng khác trong tương lai được phát minh.

Việc thử nghiệm mối quan hệ từ về ngữ nghĩa-cú pháp để hiểu về hàng loạt mối quan hệ như được thể hiện phía dưới. Sử dụng các Véc-tơ từ 640 chiều, mô hình Skip-Gram đạt được độ chính xác 55% về mặt ngữ nghĩa và 59% về mặt cú pháp.

1.6. Ngữ cảnh

1.6.1. Ngữ cảnh của một từ

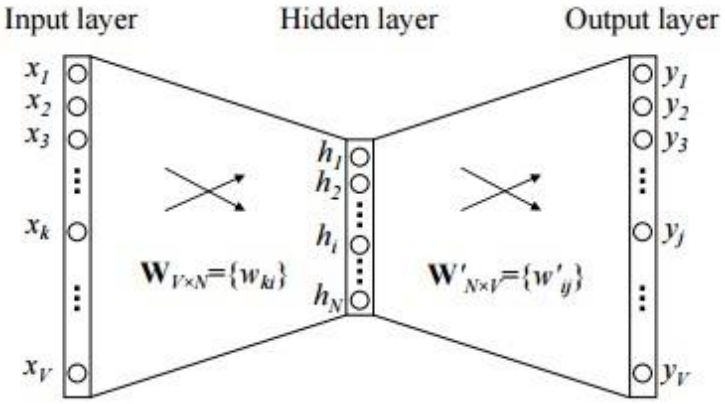
Trong mô minh CBOW được giới thiệu bởi Mikolov và cộng sự. Ta giả định rằng chỉ có một từ được xem xét trong ngữ cảnh, có nghĩa là mô hình sẽ dự đoán một từ mục tiêu để xác định ngữ cảnh của từ, cái đó giống như mô hình Bigram [9].

Hình 1.6.1 sau đây biểu diễn mô hình mạng, sự định nghĩa ngữ cảnh đã được đơn giản hóa. Trong thiết lập của ta, quy mô từ vựng là V, và quy mô lớp ẩn là N. Các đơn vị trên lớp liền kề được kết nối đầy đủ, đầu vào là một véc-tơ được mã hóa one – hot, có nghĩa là cho một từ trong ngữ cảnh đầu vào được nhắc đến, chỉ có một trong số các đơn vị V, $\{x_1 \ldots, x_V\}$, sẽ là 1, và tất cả các đơn vị khác là 0.

Input layer: Lớp đầu vào

Hidden layer: Lớp ẩn

Output layer: Lớp đầu ra



Hình 1.6.1. Mô hình CBOW

Các trọng số giữa lớp đầu vào và lớp đầu ra có thể được biểu diễn lại bằng một ma trận W kích thước V x N. Mỗi hàng của W là đại diện véc tơ N-chiều Vω của từ liên kết của lớp đầu vào. Để xác định một ngữ cảnh (một từ), giả sử $x_k = 1$ và $x_{k'} = 0$ cho $k' \neq k$, theo đó:

$$h = \mathrm{W}^T x = \mathrm{W}^T_{(k,.)} := v^T_{\omega I'} \qquad \text{CT 1.6.1.}$$

trong đó chủ yếu là sao chép dòng thứ k của W tới h. $v\omega I$ là đại diện véc-tơ của từ vựng đầu vào ω_I. Điều này ngụ ý rằng hàm liên kết (kích hoạt) của các đơn vị lớp ẩn là tuyến tính đơn giản (tức là, trực tiếp đi qua tổng trọng của đầu vào tới lớp tiếp theo).

Từ lớp ẩn tới lớp đầu ra, đó là một ma trận trọng số khác W '={vij}, mà là một ma trận N x V. Sử dụng những trọng số này ta có thể tính toán một điểm uj cho mỗi từ trong bộ từ vựng,

$$u_j = {v'_{\omega_j}}^T h \qquad \text{CT 1.6.1.1.}$$

với $v'_{\omega j}$ là cột thứ j của ma trận W'. Sau đó, ta có thể sử dụng SoftMax, một mô hình phân lớp log-tuyến tính, để đạt được sự phân bố sau của các từ vựng, đây là sự phân phối đa thức.

$$p(\omega_j|\omega_I)=y_j=\frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_{j'})}$$ CT 1.6.1.2.

trong đó y_j là đầu ra của đơn vị thứ j trong lớp đầu ra. Thay vào sẽ được:

$$p(\omega_j|\omega_I)=\frac{\exp(v'_{\omega_j}{}^{T} v_{\omega_I})}{\sum_{j'=1}^{V}\exp(v'_{\omega_{j'}}{}^{T} v_{\omega_I})}$$ CT 1.6.1.3.

Trong đó $v\omega$ và $v'\omega$ là hai đại diện của từ ω. $v\omega$ của dòng W, là đầu vào ma trận trọng số ẩn, và $v'\omega$ đến từ các cột của W‟ là ẩn ma trận đầu ra. Trong phân tích tiếp theo, ta gọi $v\omega$ là "véc-tơ đầu vào", và $v'\omega$ như "véc-tơ đầu ra" của từ ω.

* Cập nhật phương trình cho ẩn trọng số

Bây giờ suy ra phương trình cập nhật trọng số đối với mô hình này. Mặc dù việc tính toán hiện tại không thực tế (được giải thích phía dưới), ta đang suy luận để đạt được những hiểu biết về mô hình ban đầu này mà không có thủ thuật nào được áp dụng. Mục tiêu huấn luyện (đối với một mẫu huấn luyện) là tối đa hóa, xác suất có điều kiện của việc quan sát từ đầu ra thực tế ω0 (biểu thị chỉ số của nó trong lớp đầu ra như j*) được xác định nhóm các từ cùng ngữ cảnh đầu vào w_I chỉ quan tâm đến các trọng số. Để đưa ra thuật toán tính xác suất có điều kiện và sử dụng nó để xác định hàm tổn thất.

$$\begin{aligned}\log p(\omega_O|\omega_I) &= \log y_{j^*} \\ &= u_{j^*} - \log \sum_{j'=1}^{V}\exp(u_{j'}) := -E\end{aligned}$$ CT 1.6.1.4.

$E= -\log p(\omega_O|\omega_I)$ là hàm tổn thất, và j* là chỉ số của từ đầu ra thực tế. Lưu ý rằng hàm tổn thất có thể được hiểu như là một trường hợp đặc biệt của phép đo cross-entropy giữa hai phân phối xác suất.

Bây giờ lấy được các phương trình cập nhật của các trọng số giữa lớp ẩn và lớp đầu ra. Lấy đạo hàm của E đối với đầu vào u_j của đơn vị thứ j sẽ được:

$$\frac{\partial E}{\partial u_j}=y_j-t_j:=e_j$$ CT 1.6.1.5.

Trong công thức (1.6.1.5) t_j=1(j=j*), tức là t_j sẽ là 1 trong khi các đơn vị thứ j là từ vựng đầu ra thực tế, nếu không t_j = 0. Lưu ý rằng đạo hàm này là lỗi dự đoán e_j của lớp đầu ra.

Tiếp theo lấy đạo hàm trên ω'_{ij} để có được độ chênh lệch trên các trọng số ẩn các trọng số đầu ra:

$$\frac{\partial E}{\partial \omega'_{ij}}=\frac{\partial E}{\partial u_j}.\frac{\partial u_j}{\partial \omega'_{ij}}=e_j.h_i$$ CT 1.6.1.6.

Vì vậy, sử dụng sự giảm độ chênh lệch ngẫu nhiên, ta được phương trình cập nhật trọng số cho ẩn trọng số đầu ra:

$$\omega'^{(new)}_{ij}=\omega'^{(old)}_{ij}-\eta.e_j.h_i$$ CT 1.6.1.7.

Hoặc:

$$v'^{(new)}_{\omega_j}=v'^{(old)}_{\omega_j}-\eta.e_j.h \qquad for\ j=1,2,...,V$$ CT 1.6.1.8.

Trong công thức trên $\eta > 0$ là tỷ lệ huấn luyện $e_j = y_j - t_j$ và h_i là đơn vị thứ i trong lớp ẩn; $v'_{\omega j}$ là véc-tơ đầu ra của ω_j. Lưu ý phương trình cập nhật này ngụ ý rằng ta phải đi qua tất cả các từ có thể trong lớp từ vựng, kiểm tra xác suất đầu ra y_j của nó, và so sánh y_j với xác suất đánh giá t_j (hoặc là 0 hoặc là 1). Nếu $y_j > t_j$ ("đánh giá quá cao"), sau đó ta trừ một tỷ lệ h của véc-tơ ẩn (tức là: $v_{\omega j}$) từ $v'_{\omega j}$, rồi làm cho $v'_{\omega j}$ xa $v_{\omega I}$; nếu $y_j < t_j$ ("đánh giá thấp"), ta thêm một số h cho $v'_{\omega o}$, sau đó làm cho $v'_{\omega o}$ gần $v_{\omega I}$ hơn. Nếu y_j là rất gần với t_j thì căn cứ theo các phương trình cập nhật, rất ít thay đổi sẽ được thực hiện đối với các trọng số. Lưu ý một lần nữa rằng v_ω (véc-tơ đầu vào) và v'_ω (véc-tơ đầu ra) là hai đại diện véc-tơ khác nhau của từ ω.

* Cập nhật phương trình cho các trọng số đầu vào trọng số ẩn

Sau khi thu được các phương trình cập nhật cho W", bây giờ ta có thể chuyển sang W. Ta lấy đạo hàm của E ở đầu ra của các lớp ẩn, ta được:

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^{V} \frac{\partial E}{\partial u_j} \cdot \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^{V} e_j \cdot \omega'_{ij} = EH_i$$

CT 1.6.1.9.

Trong công thức hi là đầu ra của đơn vị thứ i của lớp ẩn; u_j được định nghĩa trong (CT 1.6.1.1), đầu vào thực của đơn vị thứ j trong lớp đầu ra; và $e_j = y_j - t_j$ là lỗi dự đoán của từ thứ j trong lớp đầu ra. EH, một véc-tơ N-chiều, là tổng của các véc-tơ đầu ra của tất cả các từ trong bộ từ vựng, được đánh trọng số bởi lỗi dự đoán của chúng.

Tiếp theo lấy đạo hàm của E trên W. Đầu tiên, nhớ lại rằng các lớp ẩn thực hiện một tính toán tuyến tính trên các giá trị từ lớp đầu vào. Mở rộng các ký hiệu véc-tơ, có được:

$$h_i = \sum_{k=1}^{V} x_k \cdot \omega_{ki}$$

CT 1.6.1.10.

Lấy đạo hàm của E đối với mỗi phần tử của W, thì nhận được:

$$\frac{\partial E}{\partial \omega_{ki}} = \frac{\partial E}{\partial h_i} \cdot \frac{\partial h_i}{\partial \omega_{ki}} = EH_i \cdot x_k$$

CT 1.6.1.11.

điều này tương đương với tích ten xơ (tensor) của x và EH, tức là:

$$\frac{\partial E}{\partial \mathrm{W}} = x \otimes EH = xEH^T$$

CT 1.6.1.12.

từ đó có một ma trận kích thước V x N. Vì chỉ có một thành phần của x là khác 0, chỉ là một dòng của $\frac{\partial \mathrm{E}}{\partial \mathrm{W}}$ khác 0, và giá trị của hàng đó là EHT, và một véc-tơ N-chiều. Ta được phương trình cập nhật của W như sau:

$$v_{\omega_I}^{(new)} = v_{\omega_I}^{(old)} - \eta . EH^T$$

CT 1.6.1.13.

Trong công thức (CT 1.6.1.13) $v_{\omega I}$ là một hàng của W, "véc-tơ đầu vào" của nhóm từ cùng ngữ cảnh duy nhất, và là hàng duy nhất của W mà đạo hàm của nó

khác 0. Tất cả các hàng khác của W sẽ vẫn không thay đổi sau sự lặp đi lặp lại này, bởi vì đạo hàm của chúng bằng 0.

Bằng trực giác, vì véc-tơ EH là tổng các véc-tơ đầu ra của tất cả các từ trong bộ từ vựng được đánh trọng số bởi lỗi dự đoán của chúng $e_j = y_j - t_j$, nên ta có thể hiểu (CT 1.6.1.13) thêm một phần của tất cả các véc-tơ đầu ra trong bộ từ vựng vào véc-tơ đầu vào của nhóm từ cùng ngữ cảnh. Nếu trong lớp đầu ra, xác suất của một từ j ω là từ đầu ra được đánh giá quá cao ($y_j > t_j$), sau đó các véc-tơ đầu vào của nhóm từ cùng ngữ cảnh ω_I sẽ có xu hướng di chuyển ra xa véc-tơ đầu ra của ω_j; trái lại, nếu xác suất ω_j là từ đầu ra được đánh giá thấp ($y_j < t_j$), thì các véc-tơ đầu vào ω_I sẽ có xu hướng di chuyển gần hơn tới véc-tơ đầu ra của ω_j; nếu xác suất ω_j là dự đoán tương đối chính xác, thì nó sẽ có chút ảnh hưởng đến sự di chuyển của các véc-tơ đầu vào của ω_I. Sự di chuyển của véc-tơ đầu vào của ω_I được xác định bởi lỗi dự đoán của tất cả các véc-tơ trong vốn từ vựng; lỗi dự đoán càng lớn thì tác động càng lớn, một từ sẽ di chuyển trên véc-tơ đầu vào của nhóm từ cùng ngữ cảnh.

Cập nhật các thông số mô hình lặp đi lặp lại bằng việc bỏ qua cặp từ trong ngữ cảnh mục tiêu được tạo ra từ một tập huấn luyện, các kết quả trên các véc-tơ sẽ tích lũy. Có thể tưởng tượng rằng các véc-tơ đầu ra của một từ w bị "kéo" đi tới đi lui bởi các véc-tơ đầu vào của các từ đứng gần w cùng xảy ra, như thể có sợi dây vật lý giữa các véc-tơ của w và véc-tơ của các từ xung quanh nó. Tương tự như vậy, một véc-tơ đầu vào cũng có thể bị kéo bởi nhiều véc-tơ đầu ra. Việc giải thích này có thể nhắc nhở về lực hấp dẫn, hoặc sơ đồ, đồ thị lực có hướng. Sau nhiều lần lặp lại, các vị trí tương đối của các véc-tơ đầu vào và đầu ra cuối cùng sẽ ổn định.

1.6.2. Ngữ cảnh của cụm từ

Hình sau đây cho thấy mô hình CBOW với thiết lập ngữ cảnh của cụm từ. Khi tính toán đầu ra của lớp ẩn, thay vì trực tiếp sao chép véc-tơ đầu vào của nhóm từ cùng ngữ cảnh đầu vào, thì mô hình CBOW lấy trung bình các véc-tơ của các nhóm từ cùng ngữ cảnh đầu vào, và sử dụng các kết quả của ma trận trọng số đầu vào ma trận trọng số ẩn và véc-tơ trung bình như đầu ra:

$$h = \frac{1}{C} W^T (x_1 + x_2 + ... + x_C) \quad \text{CT 1.6.2.}$$

$$= \frac{1}{C} (v_{\omega 1} + v_{\omega 2} + ... + v_{\omega C})^T \quad \text{CT 1.6.2.1.}$$

trong đó C là số các từ trong ngữ cảnh, ω_1; …; ω_C là các từ trong ngữ cảnh, và v_ω là véc-tơ đầu vào của một từ ω. Hàm tổn thất là:

$$E = -\log p(\omega_O | \omega_{I,1}, ..., \omega_{I,C} \quad \text{CT 1.6.2.2.}$$

$$= -u_{j*} + \log \sum_{j'=1}^{V} \exp(u_{j'}) \quad \text{CT 1.6.2.3.}$$

$$= -v'_{\omega o}{}^T . h + \log \sum_{j'=1}^{V} \exp(v'_{\omega_j}{}^T . h) \quad \text{CT 1.6.2.4.}$$

giống như công thức (1.6.1.4), mục tiêu của mô hình one-word-context (từ một ngữ cảnh), ngoại trừ h là khác biệt, giống như định nghĩa trong công thức (1.6.2.1) thay vì công thức (1.6.1).

1.7. SoftMax phân cấp (Hierarchical SoftMax)

Một phép tính xấp xỉ hiệu quả của toàn bộ SoftMax là SoftMax phân cấp. Trong ngữ cảnh của các mô hình ngôn ngữ mạng nơ-ron, được giới thiệu lần đầu tiên bởi Morin và Bagnio. Đối với mỗi nút, SoftMax phân cấp sử dụng một cây đại diện nhị phân của lớp đầu ra với các từ W như lá của nó, đối với mỗi nút, rõ ràng các đại diện xác suất tương đối của các nút con của nó. Những điều này định nghĩa một bước đi ngẫu nhiên được cho là xác suất đối với các từ.

Chính xác hơn, mỗi từ ω có thể đạt được bằng một đường từ gốc của cây. Gọi n (ω, j) là nút thứ j trên con đường từ gốc đến ω và gọi L (ω) là độ dài của đường đi đó, thì n (ω,1) = gốc và n(ω,L(ω)) = ω. Hơn nữa, đối với bất kỳ nút bên trong n nào, gọi ch(n) là tập con tùy ý và gọi $_x$ là 1 nếu x là đúng và sai là -1.

Vậy SoftMax phân cấp xác định p(ωo|ωI) như sau:

$$p(\omega|\omega_I) = \prod_{j=1}^{L(\omega)-1} \sigma(\ n(\omega,j+1)=ch(n(\omega,j))\ .v'_{n(\omega,j)}{}^T v_{\omega_I}) \quad \text{CT 1.7.}$$

Trong công thức $\sigma(x)=1/(1+\exp(-x))$ có thể được xác định rằng $\sum^{W}_{\omega}=1^{p}(\omega|\omega_I)=1$. Điều này ngầm chỉ ra rằng trị giá của phép tính log p $(\omega_O|\omega_I)$ và ∇ log $p(\omega_O|\omega_I)$ là tỷ lệ thuận với L(ωo), trị giá trung bình không lớn hơn logW.

Cũng không giống như công thức SoftMax chuẩn của Skip-Gram mà gán hai đại diện v_ω và v'_ω đối với mỗi từ ω, công thức SoftMax phân cấp có một đại diện v_ω đối với mỗi từ ω và một đại diện v'_n đối với mỗi nút trong n của cây nhị phân.

Cấu trúc của cây được sử dụng bởi SoftMax phân cấp có tác dụng đáng kể về hiệu suất. Mnih và Hinton đã khám phá một số phương pháp để xây dựng các cấu trúc cây và các hiệu ứng trên cả thời gian huấn luyện và tính chính xác của mô hình kết quả. Trong công trình của họ sử dụng một cây Huffman nhị phân, như nó gán mã ngắn đối với các từ thường gặp mà tạo kết quả nhanh. Nó đã được quan sát trước khi nhóm các từ với nhau bằng tần suất của chúng hoạt động tốt như một kỹ thuật tăng tốc đơn giản cho mạng nơ-ron dựa trên các mô hình ngôn ngữ.

1.7.1. Lấy Mẫu phủ định (Negative Sampling)

Một thay thế cho SoftMax phân cấp là ước lượng tương phản nhiễu (Noise Contrastive Estimation – NCE) được Gutman và Hyvarinen giới thiệu và Mnih và Teh đã áp dụng cho mô hình ngôn ngữ. NCE thừa nhận rằng một mô hình tốt nên có khả năng phân biệt dữ liệu nhiễu bằng các phương tiện hồi quy logistic. Điều này cũng tương tự như việc mất đi điểm mấu chốt mà Collobert và Weston đã sử dụng họ là những người huấn luyện các mô hình bằng cách xếp hạng các dữ liệu nhiễu.

Trong khi NCE có thể được hiển thị để tối đa hóa xác suất log của SoftMax, thì mô hình Skip-Gram lại chỉ quan tâm đến việc nghiên cứu đại diện véc-tơ chất lượng cao, vì vậy ta được tự do để đơn giản hóa NCE miễn là các đại diện véc-tơ giữ được chất lượng của chúng. Ta xác định lấy mẫu phủ định (NEG) là mục tiêu:

$$\log\sigma({v'\omega_O}^T v_{\omega_I})+\sum_{i=1}^{k} \mathrm{E}_{\omega_i\sim P_n(\omega)}\left[\log\sigma(-{v'\omega_i}^T v_{\omega_I})\right] \qquad \text{CT 1.7.1.}$$

1.7.2. Lựa chọn mẫu phụ của các từ thường gặp (Subsampling of Frequent Words)

Trong một tập văn lớn, các từ thường thấy nhất có thể dễ gặp hàng trăm triệu lần (ví dụ, "in", "the", và "a"). Những từ như vậy thường cung cấp giá trị thông tin ít hơn những từ hiếm gặp. Ví dụ, trong khi những lợi ích mô hình Skip-Gram từ việc quan sát các sự xuất hiện đồng thời của "France" và "Paris", nó giúp ích ít nhiều từ việc quan sát sự đồng xuất hiện thường xuyên của "France" và "the", như hầu hết các từ cùng xuất hiện thường xuyên trong một câu với "the". Ý tưởng này cũng có thể được áp dụng theo hướng ngược lại; các đại diện véc-tơ của các từ thường gặp không làm thay đổi đáng kể sau khi thực hiện trên vài triệu ví dụ.

Để tránh sự mất cân bằng giữa các từ hiếm và thường gặp, ta đã sử dụng một phương pháp tiếp cận mẫu phụ đơn giản: mỗi từ w_i trong tập huấn luyện được loại bỏ với xác suất tính theo công thức:

$$P(\omega_i)=1-\sqrt{\frac{t}{f(\omega_i)}} \qquad \text{CT 1.7.2.}$$

Phương pháp	Thời gian (phút)	Cú pháp (%)	Ngữ nghĩa (%)	Tổng độ chính xác (%)
NEG-5	38	63	54	59
NEG-15	97	63	58	61
HSHuffman	41	53	40	47
NCE-5	38	60	45	53
Nhưng kết quả sau sử dụng 10^{-5} mẫu phụ				
NEG-5 NEG-15	14 36	61 61	58 61	60 61
HS-Huffman	21	52	59	55

Bảng 1.7.2. Độ chính xác của nhiều mô hình Skip-Gram 300-chiều

Trong đó f (ω_i) là tần số của từ ω_i và t là một ngưỡng được chọn, thường khoảng 10^{-5}. Ta đã lựa chọn công thức mẫu phụ này vì nó cho thấy các từ mẫu phụ có tần số lớn hơn t trong khi vẫn giữ thứ hạng của các tần số. Mặc dù công thức mẫu phụ này đã được lựa chọn một cách kín đáo nhưng ta đã ứng dụng rất ổn trong thực tế. Nó làm tăng tốc việc nghiên cứu và thậm chí cải thiện đáng kể độ chính xác của các véc-tơ đã được nghiên cứu của những từ hiếm gặp.

CHƯƠNG 2 MỘT SỐ MÔ HÌNH HỌC SÂU

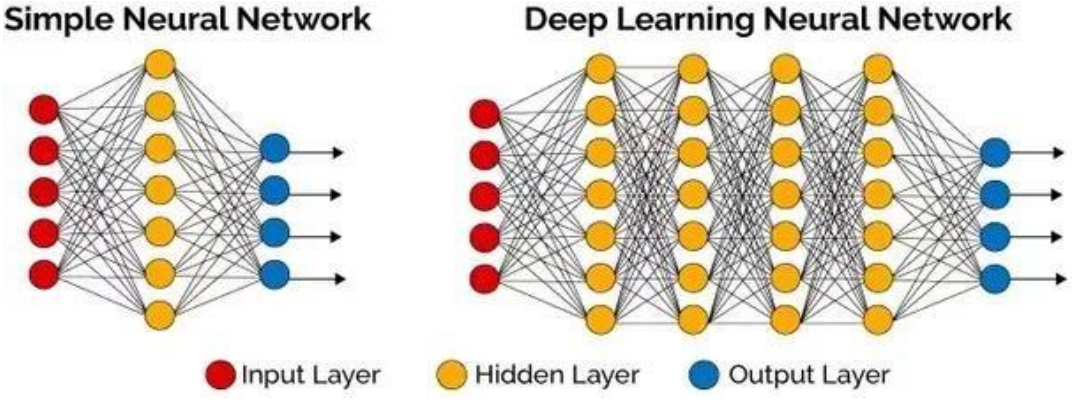
2.1. Học sâu - Deep Learning

Học máy (Machine Learning-ML) là một lĩnh vực của trí tuệ nhân tạo (Artificial Intelligence - AI). Các thuật toán học máy cho phép máy tính đào tạo đầu vào dữ liệu và sử dụng phân tích thống kê để đưa ra các giá trị nằm trong một phạm vi cụ thể [11].

Ngày nay, những người sử dụng công nghệ đều được hưởng lợi từ việc học máy. Công nghệ nhận diện khuôn mặt giúp người dùng gắn thẻ và chia sẻ ảnh của bạn bè. Công nghệ nhận dạng ký tự quang học (OCR) chuyển đổi hình ảnh văn bản sang dạng di chuyển.

Khi mà khả năng tính toán của máy tính được nâng lên một tầm cao mới cùng với lượng dữ liệu khổng lồ được thu thập, ML đã tiến thêm một bước dài và Deep Learning (DL) một lĩnh vực mới được ra đời.

Deep Learning được xây dựng từ mạng nơ-ron sinh học và bao gồm nhiều lớp trong mạng nơ-ron nhân tạo được tạo thành từ phần cứng và GPU. Deep Learning sử dụng một tầng các lớp đơn vị xử lý phi tuyến để trích xuất hoặc chuyển đổi các tính năng (hoặc biểu diễn) của dữ liệu. Đầu ra của một lớp phục vụ như là đầu vào của lớp kế tiếp. Deep learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, dịch tự động (machine translation), xử lý ngôn ngữ tự nhiên…



Hình 2.1. Mô hình Deep Learning

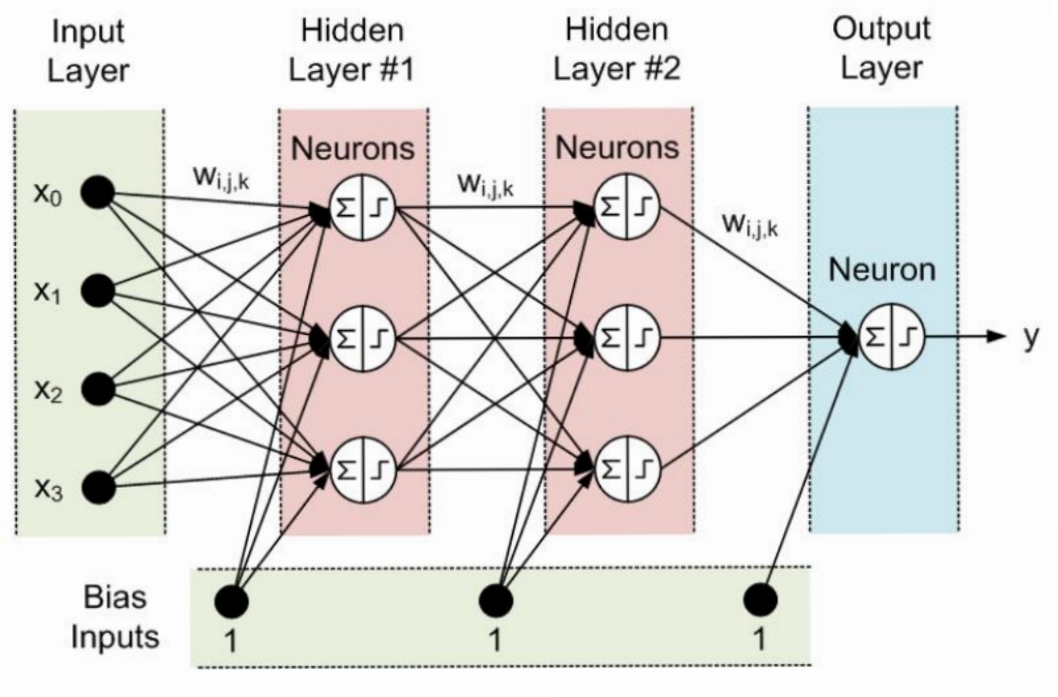
Trong số các thuật toán học máy hiện đang được sử dụng và phát triển, học sâu thu hút được nhiều nghiên cứu nhất và có thể đánh bại con người trong một số nhiệm vụ nhận thức. Do những đặc tính nổi bật và kết quả tối ưu, học tập sâu

đã trở thành phương pháp tiếp cận được nghiên cứu và ứng dụng trong giải quyết nhiều bài toán thuộc lĩnh vực trí tuệ nhân tạo.

2.2. Mạng nơ-ron hồi quy RNN (Recurrent Neural Network)

2.2.1. Giới thiệu mạng nơ-ron hồi quy (RNN)

Mạng nơ-ron hồi quy RNN (Recurrent Neural Network) được giới thiệu bởi John Hopfield năm 1982, là một trong những mô hình học sâu - Deep learning**.** Recurrent có nghĩa là thực hiện lặp lại cùng một tác vụ cho mỗi thành phần trong chuỗi. Trong đó, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước đó. Mạng nơ-ron thông thường, mỗi sự kiện đầu vào x được xử lý một cách độc lập và đưa ra đầu ra y tương ứng mà không có sự trao đổi thông tin thu thập được tại mỗi đầu vào x trong mạng [11][12].



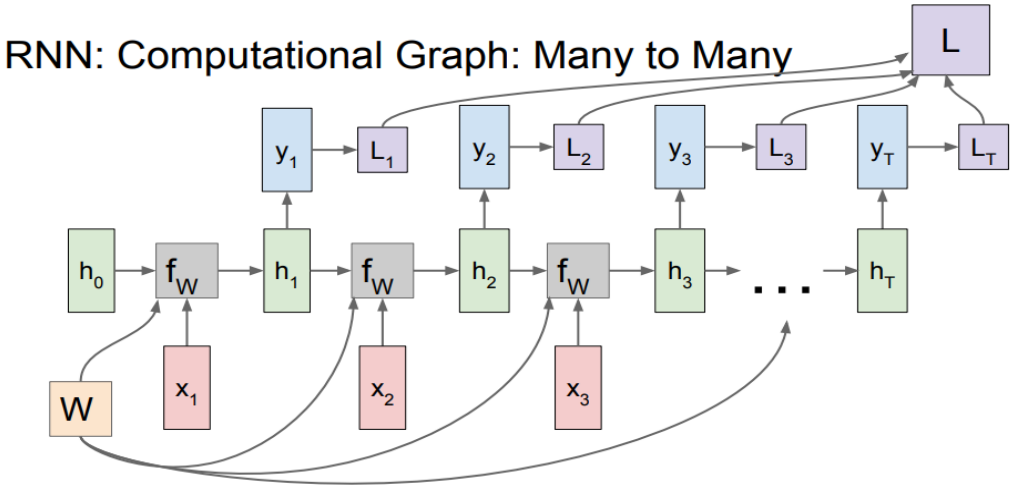
Hình 2.2.1. Mô hình mạng nơ-ron

RNN là một mô hình có trí nhớ **(**memory**)**, có khả năng nhớ được thông tin đã tính toán trước đó. Không như các mô hình mạng nơ-ron truyền thống trước đó là thông tin đầu vào hoàn toàn độc lập với thông tin đầu ra**.** Hầu hết RNN được thiết kế như là một chuỗi các mô-đun được lặp đi lặp lại, các mô-đun này thường có cấu trúc đơn giản chỉ có một lớp mạng tank. Huấn luyện RNN tương tự như huấn luyện nơ-ron truyền thống. Giá trị tại mỗi đầu ra không chỉ phụ

thuộc vào kết quả tính toán của bước hiện tại mà còn phụ thuộc vào kết quả tính toán của các bước trước đó.

2.2.2. Cấu trúc của RNN

Các đầu vào x_t sẽ được kết hợp với tầng ẩn h_{t-1} bằng hàm f_w để tính toán ra các tầng ẩn h_t hiện tại và đầu ra y_t sẽ được tính từ h_t. W là tập các trọng số được cập nhật ở tất cả các cụm và L_1,L_2,… L_t là các hàm mất mát. Khi đó kết quả được tính từ các quá trình trước được nhớ bằng cách kết hợp thêm h_{t-1} để tính ra h_t nhằm tăng độ chính xác cho những dự đoán ở hiện tại.



Hình 2.2.2 Các dạng RNN

- Quá trình tính toán được xác định như sau:

$h_t = f_w(h_{t-1},x_t)$

Trường hợp hàm f_w sử dụng hàm tank thì công thức trên được viết như sau:

H_t= tank($W_{hh}h_{t-1}$+ $W_{xh}x_t$)

$Y_t = W_{hy}h_t$

Trong đó, ma trận trọng số W_{hh}, W_{xh}, W_{hy} được cập nhật cho 2 quá trình tính toán: (Mạng RNN chỉ sử dụng một ma trận trọng số duy nhất)

W_{hh} kết ợp với "bộ nhớ trước" h_{t-1} và W_{xh} kết hợp với x_t để tính ra "bộ nhớ bước hiện tại"

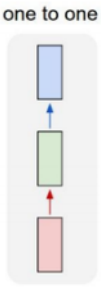
W_{hy} để tính ra y_t

2.2.3. Các dạng của RNN

One to one:

Mẫu bài toán cho Neural Network (NN) và Convolutional Neural Network (CNN), 1 đầu vào và 1 đầu ra

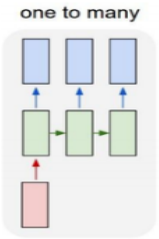
Ví dụ với CNN đầu vào là ảnh và đầu ra là ảnh được segment.



Hình 2.2.3. RNN dạng One to One

One to Many: bài toán có 1 đầu vào nhưng nhiều đầu ra

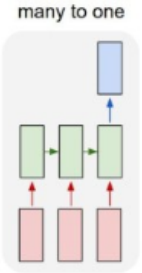
Ví dụ: bài toán gán nhãn cho ảnh, đầu vào là 1 ảnh nhưng đầu ra là nhiều chữ mô tả cho ảnh đấy, dưới dạng một câu.



Hình 2.2.3.1. RNN dạng One to Many

Many to one: bài toán có nhiều đầu vào nhưng chỉ có 1 đầu ra

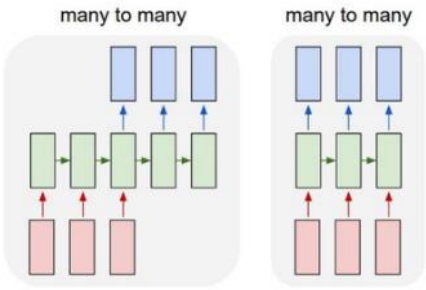
Ví dụ bài toán phân loại hành động trong video, đầu vào là nhiều ảnh (frame) tách ra từ video, đầu ra là hành động trong video.



Hình 2.2.3.2. RNN dạng Many to One

Many to Many: bài toán có nhiều đầu vào và nhiều đầu ra

Ví dụ bài toán dịch từ tiếng anh sang tiếng việt, đầu vào là 1 câu gồm nhiều chữ: “I love Vietnam” và đầu ra cũng là 1 câu gồm nhiều chữ “Tôi yêu Việt Nam”.



Hình 2.2.3.3. RNN dạng Many to Many

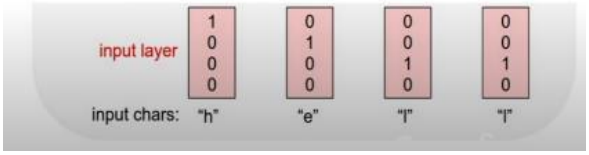
2.2.4. Ví dụ ứng dụng

Cho tập đầu vào

x = [h,e,l,o]

Sử dụng mô hình RNN để tạo ra một từ có nghĩa.

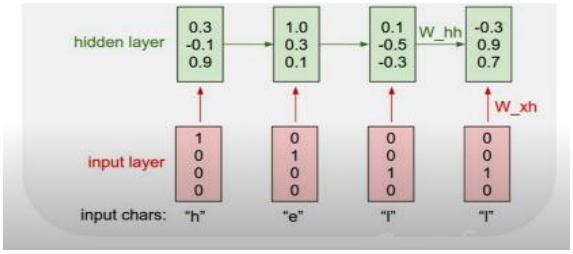
Trong trường hợp này, các chữ cái được mã hóa dạng one hot véc-tơ.

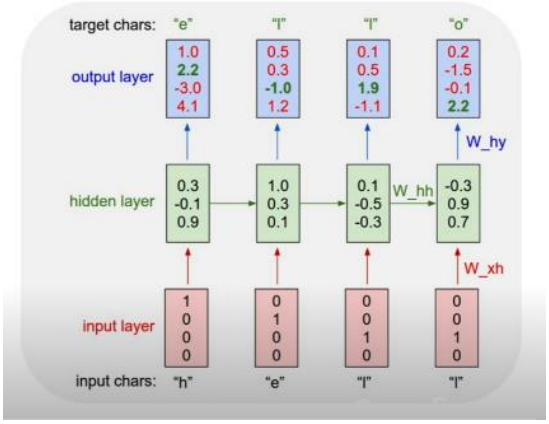


Hình 2.2.4. Các chữ cái mã hóa dạng one hot véc-tơ

Sử dụng hàm tank để kết hợp tính toán đầu ra cho các tầng ẩn ht cho kết quả như sau:

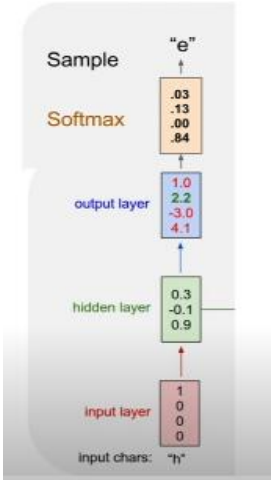
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$





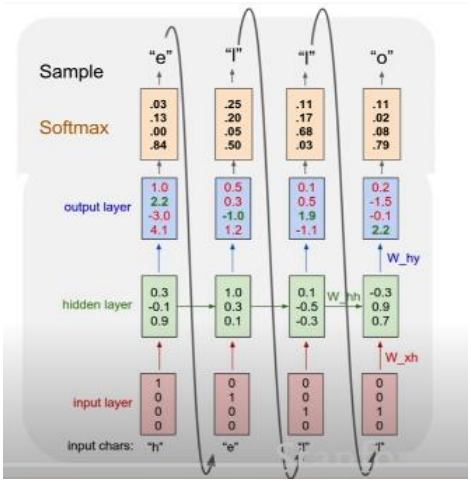
Hình 2.2.4.1. Sử dụng hàm tank để kết hợp tính toán đầu ra

Tại mỗi bước, kết quả dự đoán như sau: (sử dụng hàm SoftMax để tổng hợp kết quả đầu ra).



Hình 2.2.4.2. sử dụng hàm SoftMax để tổng hợp kết quả đầu ra

Mỗi đầu ra lại được sử dụng là đầu vào cho dự đoán ở bước sau:



Hình 2.2.4.3. Kết quả

2.2.5. Một số ứng dụng của RNN

Speech to text: Chuyển giọng nói sang text.

Sentiment classification: phân loại quan điểm (Phân loại đánh giá 1*,2*,3*,4*,5* cho một bình luận)

Ví dụ: đầu vào là bình luận: "ứng dụng tốt", đầu ra: 4 sao.

Machine translation: Bài toán dịch tự động giữa các ngôn ngữ.

Video recognition: Nhận diện hành động trong video.

2.2.6. Nhận xét

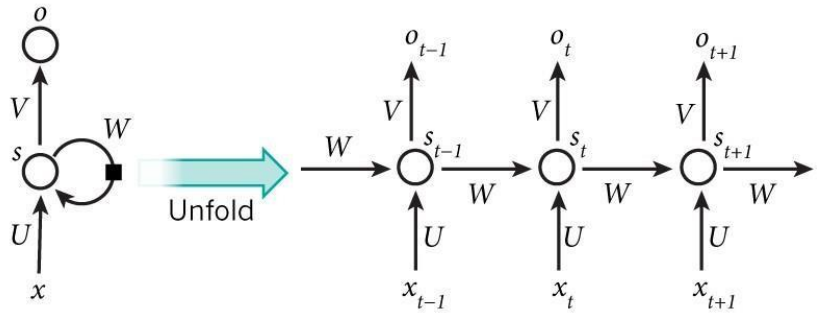
RNN có thể xử lý thông tin dạng chuỗi (sequence/ time-series).

RNN có thể mang thông tin từ các tầng (layer) trước đến các tầng sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng trạng thái nhất định, mô hình chỉ học được từ các trạng thái gần nó được gọi là bộ nhớ ngắn hạn (Short term memory). Trong một số trường hợp, ví dụ bài toán là dự đoán từ tiếp theo trong đoạn văn.

Đoạn đầu tiên "Mặt trời mọc ở hướng …", ta có thể chỉ sử dụng các từ trước trong câu để đoán là đông. Tuy nhiên, với đoạn, "Tôi là người Việt Nam. Tôi đang sống ở nước ngoài. Tôi có thể nói trôi chảy tiếng …" thì rõ ràng là chỉ sử dụng từ trong câu đấy hoặc câu trước là không thể dự đoán được từ cần điền là Việt. Ta cần các thông tin từ trạng thái ở trước đó rất xa do đó cần bộ nhớ dài (long term memory), điều này RNN không làm được

Để giải quyết vấn đề này, ta cần một mô hình mới được gọi là Mạng bộ nhớ dài-ngắn (Long short-term memory - LSTM)

2.2.7. Quá trình xử lý thông tin trong mạng RNN



Hình 2.2.7. Quá trình xử lý thông tin trong mạng RNN

RNN có khả năng biểu diễn mối quan hệ phụ thuộc giữa các thành phần trong chuỗi (nếu chuỗi đầu vào có 6 từ thì RNN sẽ dàn ra thành 6 lớp, mỗi lớp ứng với mỗi từ, chỉ số mỗi từ được đánh từ 0 đến 5. Trong Hình 2.8 ở trên, x_t là đầu vào tại thời điểm thứ t, s_t là trạng thái ẩn (hidden state) tại thời điểm thứ t, được tính dựa trên các trạng thái ẩn trước đó kết hợp với đầu vào của thời điểm hiện tại với công thức:

$$S_t = \text{tank}\ (U_{xt} + W_{st-1})$$

S_{t-1} là trạng thái ẩn được khởi tạo là 1 véc-tơ O. O_t là đầu ra tại thời điểm thứ t, là một véc-tơ chứa xác xuất của toàn bộ các từ trong từ điển.

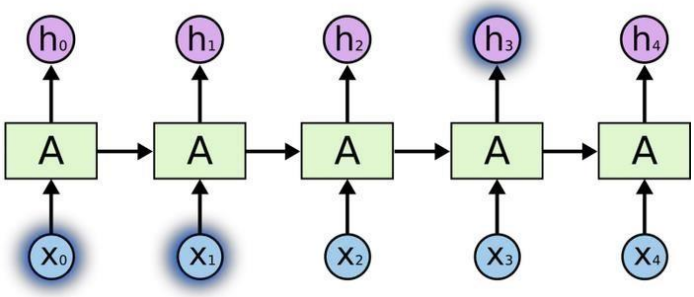
$$O_t = \text{SoftMax}(V_{st})$$

Không như mạng nơ-ron truyền thống, tại mỗi lớp cần phải sử dụng một tham số khác, RNNs chỉ sử dụng một bộ các tham số (U, V, W) cho toàn bộ các bước.

Ý tưởng ban đầu của RNN là kết nối những thông tin trước đó nhằm hỗ trợ cho các xử lý hiện tại. Nhưng đôi khi, chỉ cần dựa vào một số thông tin gần nhất để thực hiện tác vụ hiện tại.

Ví dụ, dự đoán từ cuối cùng trong câu "chuồn_chuồn bay thấp thì mưa", thì sẽ không cần truy tìm quá nhiều từ trước đó, ta có thể đoán ngay từ tiếp theo sẽ là "mưa". Trong trường hợp này, khoảng cách tới thông tin liên quan được rút ngắn lại, mạng RNN có thể học và sử dụng các thông tin quá khứ.

Quá trình này những mạng nơ-ron thông thường không thể mô phỏng được.

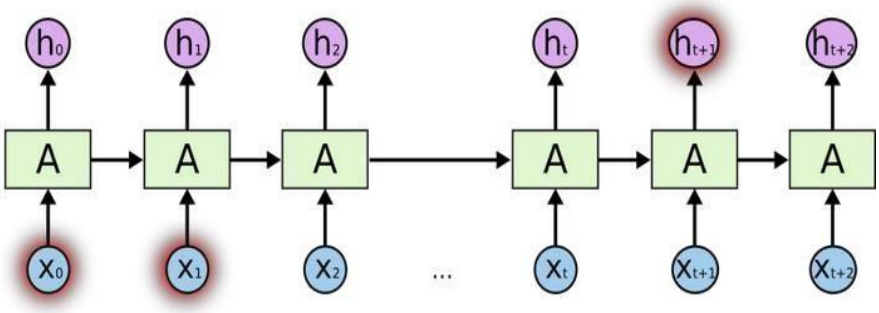


Hình 2.2.7.1. RNN phụ thuộc short-term

Trường hợp có nhiều thông tin hơn trong một câu, nghĩa là phụ thuộc vào ngữ cảnh. Ví dụ nhưng khi dự đoán từ cuối cùng trong đoạn văn bản **"Tôi sinh ra và lớn lên ở Việt_Nam … Tôi có_thể nói thuần_thục Tiếng_Việt."** Từ thông tin gần nhất cho thấy rằng từ tiếp theo là tên một ngôn ngữ, nhưng khi

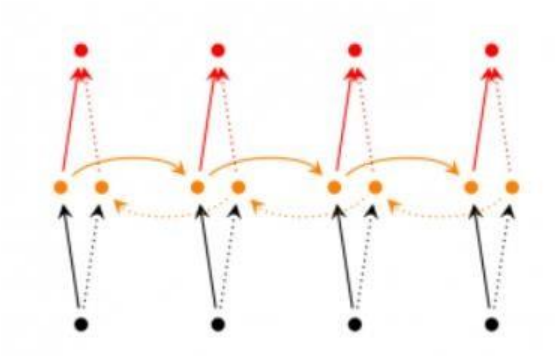
muốn biết cụ thể ngôn ngữ nào, thì cần quay về quá khứ xa hơn, để tìm được ngữ cảnh **Việt_Nam**. Và như vậy, RNN có thể phải tìm những thông tin có liên quan và số lượng các điểm đó trở nên rất lớn.

Không được như mong đợi, RNN không thể học để kết nối các thông tin lại với nhau.

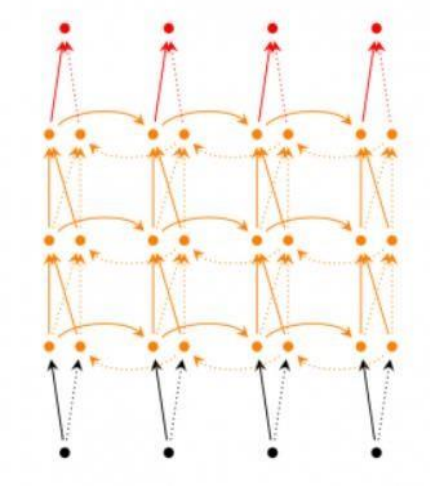


Hình 2.2.7.2. RNN phụ thuộc long-term

Về lý thuyết, RNN có thể nhớ được thông tin của chuỗi có chiều dài bất kì, nhưng trong thực tế mô hình này chỉ nhớ được thông tin ở vài bước trước đó. RNN có các phiên bản mở rộng như: Bidirectional RNN **(**RNN hai chiều**),** Deep (Bidirectional) RNN**,** Long short-term memory networks (LSTM)**.**



Hình 2.2.7.3. Bidirectional RNN



Hình 2.2.7.4. Deep (Bidirectional) RNN

2.3. Mạng Bộ nhớ dài ngắn (Long-short term memory-LSTM)

2.3.1. Giới thiệu

Là một dạng đặc biệt của mạng Nơ-ron hồi quy (RNN) dựa trên Gradient. Trong quá trình hoạt động nó cho phép cắt bỏ những Gradient dư thừa. Trong quá trình học LSTM có thể thu hẹp thời gian trễ dư thừa của các bước thực hiện thông qua tập hằng số lỗi (theo Hoch Reiter & Schmid Huber- 1997) [13].

LSTM được giới thiệu bởi Hoch Reiter & Schmid Huber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều nhà nghiên cứu trong học máy. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau.

LSTM được thiết kế để giải quyết vấn đề phụ thuộc dài (long-term dependency). Đặc tính của mạng là có thể nhớ thông tin trong suốt thời gian dài chứ không cần phải huấn luyện mạng để nhớ được (nội tại của mạng đã có thể ghi nhớ được mà không cần bất kì can thiệp nào) [14].

LSTM có các thành phần cơ bản sau:

+ Trạng thái tế bào (cell state)

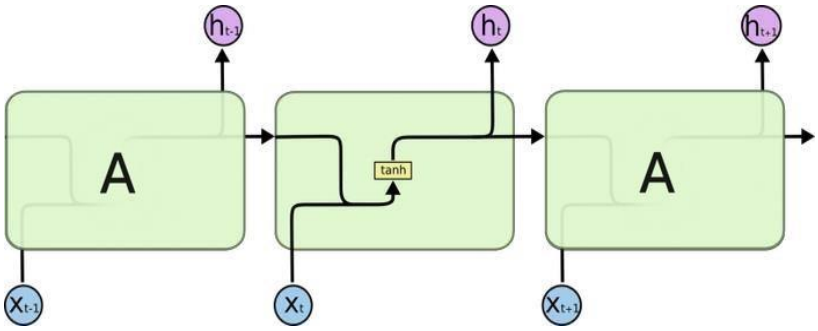
+ Cổng (gates)

+ Sigmoid

+ Tank

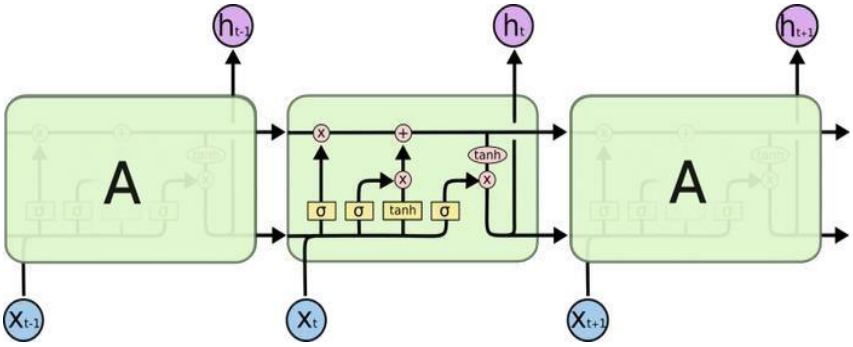
LSTM được thiết kế nhằm loại bỏ vấn đề phụ thuộc quá dài. Ta quan sát lại mô hình RNN bên dưới, các layer đều mắc nối với nhau thành các mô-đun neural

network. Trong RNN chuẩn, mô-đun repeating này có cấu trúc rất đơn giản chỉ gồm một lớp đơn giản là tầng tank (**tank layer)**.

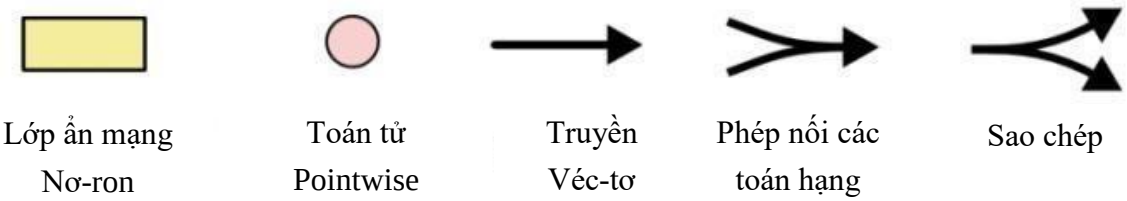


Hình 2.3.1. Các mô-đun lặp của mạng RNN chứa một layer

Về kiến trúc mạng LSTM**:** giống như RNN, nó là một chuỗi các mô-đun được lặp đi lặp lại. Tuy nhiên, xét về cấu trúc thì LSTM có 4 tầng mạng nơ-ron tương tác với nhau một cách rất đặc biệt, gọi đó là các tầng ẩn (hidden layer). Một số biến thể của LSTM được thực hiện dựa trên việc thay đổi vị trí kết nối giữa các tầng và cổng.



Hình 2.3.1.1. Các mô-đun lặp của mạng LSTM chứa bốn layer



Hình 2.3.1.2. Các kí hiệu sử dụng trong mạng LSTM

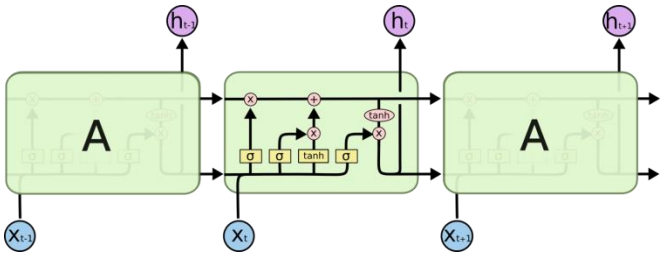
Các ký hiệu sử dụng trong mạng LSTM được như hình 1.15.3 sau đây:

- Hình chữ nhật là các lớp ẩn của mạng nơ-ron
- Hình tròn biểu diễn toán tử Pointwise
- Đường kẻ gộp lại với nhau biểu thị phép nối các toán hạng

- Và đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác

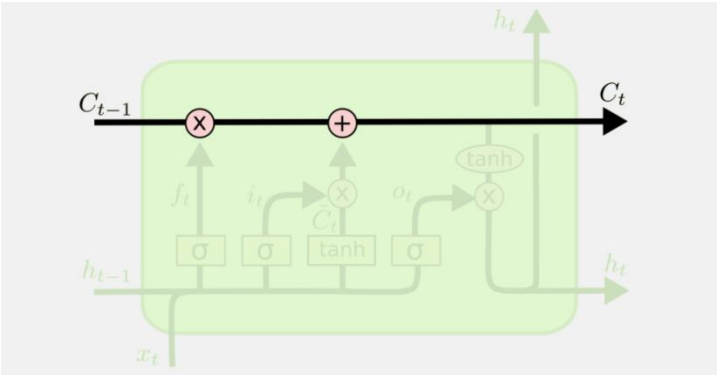
2.3.2. Phân tích mô hình LSTM

LSTM là một mạng nơ ron hồi quy có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình 2.3.2.1. Tế bào trạng thái LSTM giống như một băng truyền

Chìa khóa của LSTM là trạng thái tế bào (cell state). Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Nhờ đó các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.



LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

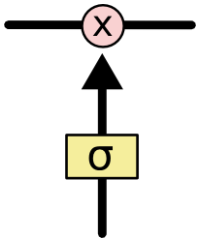
Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.

Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1][0,1], mô tả có bao nhiêu thông tin có thể được thông qua.

Khi đầu ra là 00 thì có nghĩa là không cho thông tin nào qua cả

Khi là 11 thì có nghĩa là cho tất cả các thông tin đi qua nó.

Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

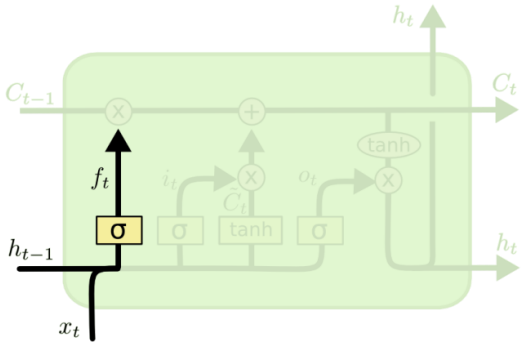


Hình 2.3.2.2. Cổng trạng thái LSTM

Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là "tầng cổng quên" (forget gate layer).

Nó sẽ lấy đầu vào là h_{t-1} và x_t rồi đưa ra kết quả là một số trong khoảng [0, 1][0,1] cho mỗi số trong trạng thái tế bào C_{t-1}

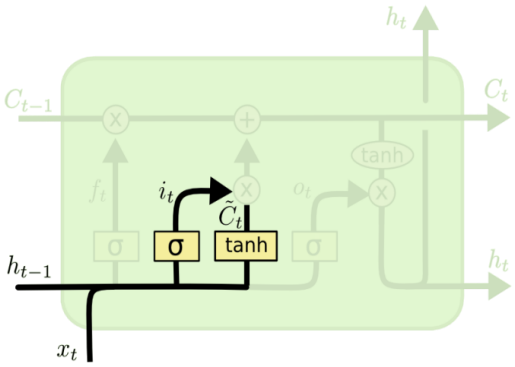
Đầu ra là 11 thể hiện rằng nó giữ toàn bộ thông tin lại, còn 00 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Hình 2.3.2.3. LSTM focus f

Đầu tiên là sử dụng một tầng sigmoid được gọi là "tầng cổng vào" (input gate layer) để quyết định giá trị nào sẽ cập nhật.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

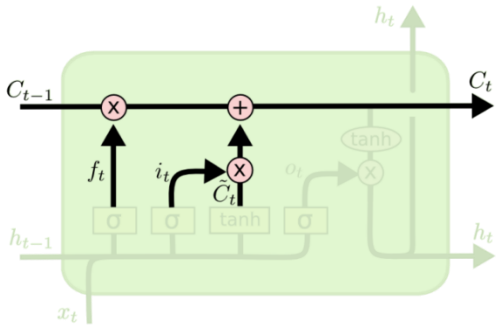
Hình 2.3.2.4. LSTM focus i

Tiếp theo là một tầng tanh tạo ra một véc-tơ cho giá trị mới nhằm thêm vào cho trạng thái. Sau đó kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái.

Cập nhập trạng thái tế bào cũ C_{t-1} thành trạng thái mới C_t.

Thực hiện nhân trạng thái cũ với f_t để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm i_t C_t.

Trạng thái mới thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

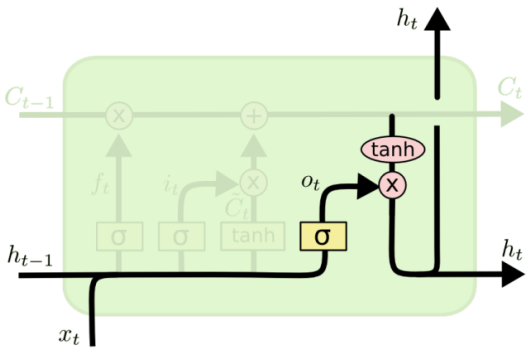


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hình 2.3.2.5. LSTM focus c

Cuối cùng, ta cần quyết định xem đầu ra là gì.

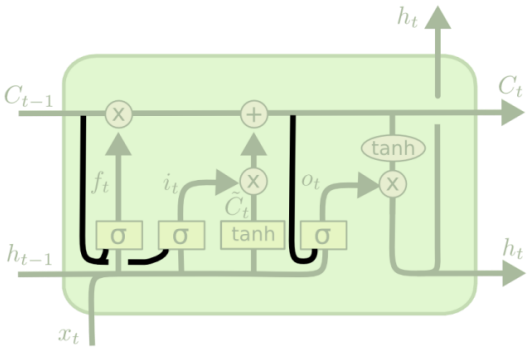
Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bảo qua một hàm *tank* để có giá trị nó về khoảng [-1, 1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

2.3.3. Một số biến thể của LSTM

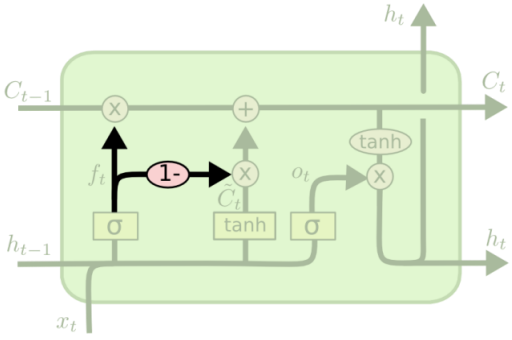
Một dạng LTSM phổ biến được giới thiệu bởi Gers & Schmidhuber (2000) được thêm các đường kết nối "peephole connections", làm cho các tầng cổng nhận được giá trị đầu vào là trạng thái tế bào.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t]\ +\ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t]\ +\ b_o\right)$$

Một biến thể khác là nối 2 cổng loại trừ và đầu vào với nhau.

Thay vì phân tách các quyết định thông tin loại trừ và thông tin mới thêm vào, ta sẽ quyết định chúng cùng với nhau luôn. Bỏ đi thông tin khi mà ta thay thế nó bằng thông tin mới đưa vào. Đưa thông tin mới vào khi ta bỏ thông tin cũ nào đó đi.

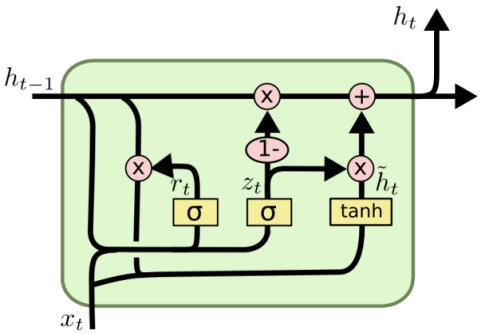


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

Hình 2.3.3. Biến thể nối 2 cổng loại trừ và đầu vào với nhau.

Một biến thể khác của LSTM là Gated Recurrent Unit, hay GRU được giới thiệu bởi Cho, et al. (2014).

Nó kết hợp các cổng loại trừ và đầu vào thành một cổng "cổng cập nhập" (update gate). Nó cũng hợp trạng thái tế bào và trạng thái ẩn với nhau tạo ra một thay đổi khác. Kết quả là mô hình của ta sẽ đơn giản hơn mô hình LSTM chuẩn và ngày càng trở nên phổ biến.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Hình 2.3.3.1. Biến thể Gated Recurrent Unit

2.3.4 Kết luận

LSTM là một cải tiến cho RNN với mục đích giúp cho tất cả các bước của RNN có thể truy vấn được thông tin từ một tập thông tin lớn hơn.

Ví dụ, nếu bạn sử dụng RNN để tạo mô tả cho một bức ảnh, nó có thể lấy một phần ảnh để dự đoán mô tả từ tất cả các từ đầu vào.

LSTM hoạt động thực sự tốt hơn nhiều so với RNN cho nhiều bài toán

CHƯƠNG 3 ỨNG DỤNG WORD2VEC CHO XỬ LÝ CHO DỮ LIỆU TIẾNG VIỆT

3.1. Bài toán phân loại quan điểm bình luận

Phân tích tình cảm (Sentiment Analysis) hay khai phá quan điểm (Opinion Mining) người dùng là lĩnh vực đã và đang thu hút được sự quan tâm của cộng đồng các nhà nghiên cứu cũng như các nhà phát triển ứng dụng. Cùng với sự phát triển của mạng máy tính toàn cầu và các thiết bị di động, người dùng đã tạo ra một lượng dữ liệu đánh giá khổng lồ trong quá trình họ tương tác trên các trang mạng xã hội, các trang diễn đàn, các trang đánh giá sản phẩm, v.v. Người dùng hay nhà sản xuất thường muốn biết sản phẩm hay dịch vụ mà họ quan tâm được đánh giá là tích cực hay tiêu cự để quyết định lựa chọn hay sản xuất nó. Do đó, việc khai thác các thông tin hữu ích từ dữ liệu đã được bình luận trên mạng sẽ giúp họ nắm được xu thế đang được đánh giá, bình luận hay thể hiện tình cảm về các sản phẩm, dịch vụ, sự kiện, v.v. là khen hay chê và được thể hiện như thế nào.

Một trong những bài toán đã và đang được nghiên cứu và triển khai cho các ứng dụng thực tế là bài toán phân loại quan điểm người dùng (sentiment classification) là tích cực hay tiêu cực được phát biểu như sau:

Input: Cho một tài liệu .csv mang dữ liệu gồm 3 cột, chứa điểm đánh giá (rate), chứa các bình luận mang quan điểm (review), gán nhãn quan điểm cho bình luận ‘-1’ hoặc ‘-1’ (label).

Output: Kết quả hiển thị ra sau khi chạy chương trình gồm điểm đánh giá và độ chính xác của bình luận sau khi xử lí, phân loại xem quan điểm của bình luận đó là tích cực hay tiêu cực.

Cấu trúc file CSV bao gồm: 3 cột

Cột 1(Rate) chứa điểm đánh giá quan điểm của bình luận

Cột 2 (Review) chứa các bình luận mang quan điểm

Cột 3 (Label) chứa các nhãn 1 hoặc -1 đã được gán tương ứng với từng bình luận.

Ví dụ 3.1. về bình luận chứa thông tin chủ quan: “Ngoài việc ít rom cho em này ra, ko có gì để chê bai em A650 vào thời điểm này vì nhìn vào giá hàng xách

tay Hàn Quốc thì em này cấu hình cao mà giá lại rất tốt, nhưng ngược lại máy lại có sự ổn định cao."

Ví dụ 3.2. Về bình luận chứa thông tin khách quan: "Chào các bác. Em dùng bé treo 650, sang nay thức đậy tự nhiên em nó liệt một lúc 3 phím backspace, space và cái phím đen đen nằm dưới chữ A nữa. Thế là ko nhắn tin được nữa. Mấy bro có thể chỉ cho em biết nó bị sao ko ạ? với lại ở sài gòn thì nên mang em nó đến đâu để sửa?"

Thực hiện phân loại tình cảm cho bình luận chứa quan điểm trong ví dụ 3.1: "Ngoài việc ít rom cho em này ra, ko có gì để chê bai em A650 vào thời điểm này vì nhìn vào giá hàng xách tay Hàn Quốc thì em này cấu hình cao mà giá lại rất tốt, nhưng ngược lại máy lại có sự ổn định cao".

Kết quả phân loại phân cực cho tài liệu này sẽ là: "tích cực"

Trong ứng dụng phân loại bình luận tiếng Việt. Đồ án sử dụng tập dữ liệu bình luận về dịch vụ ăn uống đã được gán nhãn. Thông tin mỗi bình luận gồm: Rate: điểm đánh giá tính chủ quan của bình luận, Review: Nội dung bình luận, Label: nhận một trong hai giá trị: 1: đánh giá tích cực, -1: đánh giá tiêu cực. Tập dữ liệu gồm có 4086 bình luận được lưu trữ trong tệp CSV gồm:

1765 bình luận tiêu cực được gán nhãn quan điểm = '-1'.

	A	B	C
1	Rate	Review	Label
2	4.0	Mình thể là mình ko thể cảm nổi đồ ăn ở aeon mall Mình đã thử ăn từ nhà hàng đến f	-1
3	3.8	Đôi khi thèm lên là bất chấp nắng nóng phi Và bao giờ sang cũng chỉ đóng đô ở quầy	-1
4	3.8	Ngõ treo biển cafe trứng đúng kiểu phố cổ hà nội rất sâu và nhỏ tẹo vừa đủ người đi	-1
5	3.8	Mình thấy địa chỉ cafe Giảng ở Nguyễn Hữu Huân chỉ ngon hồi ngày xưa còn bây giờ	-1
6	2.2	Mình là người Hà Nội và cũng cực kỳ khó tính trong chuyện ăn Nhà mình ở phố Hàng	-1
7	3.4	Mình là một người khá khó tính trong chuyện ăn theo mọi người nhận xét là sành Nhà	-1
8	3.8	Giá phù hợp sinh viên học cơ mà lần đầu tiên đến cơ sở Tạ Quang Bửu thì thái độ phụ	-1
9	1.0	Mình đặt hàng cốc từ quán xác nhận đơn hàng mà đến giờ đã hẹn thấy mình gọi điện	-1
10	1.0	Mếu tìm tkấy	-1
11	4.0	Hôm nay đi dạo qua nổi hứng muốn thử kem ốc quế vì trước nay chỉ mới thử kem que	-1
12	1.4	Đang định có thiện cảm vì kem ốc quế lại ngon và giòn như trước thì lúc mua kem que	-1
13	3.6	Kem ở đây không có j đặc biệt so với ở SGon giá ko kem ko khác mấy chỗ khác nếu kh	-1
14	3.8	Kem ở đây không còn như ngày xưa nữa ngay từ hương vị cho đến chất Không biếvs c	-1

2321 bình luận tích cực được gán nhãn quan điểm = '1'

	A	B	C
1	Rate	Review	Label
2	9.0	Khu ẩm thực với đa dạng đồ lại còn bày trí đẹp nên là rất	1
3	9.0	Lúc nào đến aeon là lúc đấy phải tống một đống thứ vào mồ	1
4	10.0	Bánh ngon lại rẻ chê đâu được gần hết các loại bánh luôn r	1
5	9.0	Ngon rẻ	1
6	9.6	Tôi sắp chết vì ngập trong sushi mấtttt Lên hà nội lần nào	1
7	9.0	Trong này rộng nên cứ đi lòng và lòng vòng mãi cũng thấy	1
8	10.0	Đến đây rất tiện vì cod xe bus rộng ăn tầng siêu lần đầu mì	1
9	9.0	Đến Aeon mall lần mặc dù mình đã có hẹn với bạn sẽ ăn tại	1
10	9.8	Nhắc đến Aeon Mình chỉ có thể nói đây là nơi thật sự RẤT T	1
11	9.0	Đồ đồ uống đều phong đa dạng và Mình đi buổi sáng nên r	1
12	10.0	Mình cũng đã từng qua Nói chung là rộng rãi nhưng hàng	1
13	9.8	Ghé lần thích ơi là thích	1
14	9.2	nghe danh cafe trứng của Giảng đã lâu mà nay mới có dịp (	1

Đồ án sử dụng mô hình LSTM trong thư viện Keras của Python để phân loại các bình luận là tích cực hay tiêu cực, các nhúng véc-tơ từ bằng công cụ Embedding trong Keras.

3.2 Ứng dụng thực nghiệm

3.2.1. Thư viện sử dụng

Sklearn	Scikit-learn là một thư viện máy học phần mềm miễn phí cho ngôn ngữ lập trình Python
Numpy	Numpy là một thư viện lõi phục vụ cho khoa học máy tính của Python, hỗ trợ cho việc tính toán các mảng nhiều chiều, có kích thước lớn với các hàm đã được tối ưu áp dụng lên các mảng nhiều chiều đó
Pandas	Pandas là một thư viện mã nguồn mở, hỗ trợ đắc lực trong thao tác dữ liệu. Đây cũng là bộ công cụ phân tích và xử lý dữ liệu mạnh mẽ của ngôn ngữ lập trình python. Thư viện này được sử dụng rộng rãi trong cả nghiên cứu lẫn phát triển các ứng dụng về khoa học dữ liệu
Keras	Keras là một thư viện phần mềm mã nguồn mở cung cấp giao diện Python cho các mạng nơ-ron nhân tạo. Keras hoạt động như một giao diện cho thư viện TensorFlow
Tokenizer	Tokenizer là một thuật toán có nhiệm vụ tách từ, cụm từ trong văn bản.
LSTM	là một mạng thần kinh hồi quy nhân tạo được sử dụng trong lĩnh vực học sâu. Không giống như các mạng thần kinh truyền thẳng tiêu chuẩn, LSTM có chứa các kết nối phản hồi. Mạng không chỉ xử lý các điểm dữ liệu đơn

	lẻ, mà còn xử lý toàn bộ chuỗi dữ liệu

Bảng 3.2.1. Các thư viện sử dụng

3.2.2. Python

Python được Guido van Rossum phát triển vào cuối những năm tám mươi và đầu những năm chín mươi tại Viện nghiên cứu quốc gia về toán học và khoa học máy tính ở Hà Lan.

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum. Nó dễ dàng để tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình. Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Python có cấu trúc dữ liệu cấp cao mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng. Cú pháp lệnh của Python là điểm cộng vô cùng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết script và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.

3.2.3. TensorFlow

Tensorflow là một thư viện mã nguồn mở cung cấp khả năng xử lí tính toán số học dựa trên biểu đồ mô tả sự thay đổi của dữ liệu, trong đó các node là các phép tính toán học còn các cạnh biểu thị luồng dữ liệu. Trong tesorflow có một vài khái niệm cơ bản sau.

Tensor là cấu trúc dữ liệu trong tensorflow đại diện cho tất cả các loại dữ liệu. Nói cách khác, tất cả các kiểu dữ liệu khi đưa vào trong tensorflow thì đều được gọi là Tensor. Vậy nên có thể hiểu được Tensorflow là một thư viện mô tả, điều chỉnh dòng chảy của các Tensor. Tensor có 3 thuộc tính cơ bản là rank, shape và type:

Rank là số bậc của tensor. Ví dụ Tensor = [1] thì có rank = 1, Tensor = [[3,4],[5,6]] thì sẽ có rank = 2. Việc phân rank này khá quan trọng vì nó đồng thời cũng giúp phân loại dữ liệu của Tensor. Khi các rank đặc biệt cụ thể, Tensor có những tên gọi riêng như sau:

Scalar: Khi Tensor có rank bằng 0.

Vector: Vector là một Tensor rank 1.

Matrix: Đây là một Tensor rank 2 hay mảng hai chiều theo khái niệm của Python.

N-Tensor: Khi rank của Tensor tăng lên lớn hơn 2, chúng được gọi chung là N-Tensor.

Shape là chiều của tensor. Vi dụ Tensor = [[[1,1,1], [178,62,74]]] sẽ có Shape = (1,2,3), Tensor = [[1,1,1], [178,62,74]] sẽ có Shape = (2,3).

Type kiểu dữ liệu của các elements trong Tensor. Vì một Tensor chỉ có duy nhất một thuộc tính Type nên từ đó cũng suy ra là chỉ có duy nhất một kiểu Type duy nhất cho toàn bộ các elements có trong Tensor hiện tại.

3.2.3.1. Cách cài đặt TensoFlow

Yêu cầu hệ thống

Tensorflow hỗ trợ Python 2.7 và Python 3.3+, do dùng Ubuntu lên Python được cài đặt sẵn.

Bản Tensorflow GPU cần cài đặt Cuda Toolkit 7.0 và cuDNN v2.

Cài pip:

```
# Ubuntu/Linux 64-bit
$ sudo apt-get install python-pip python-dev

# Mac OS X
$ sudo easy_install pip
```

Sau đó cài Tensorflow:

```
# Ubuntu/Linux 64-bit, CPU only:
$ sudo pip install --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-0.6.0-cp27-none-linux_x86_64.whl
```

Cài bản có hỗ trợ GPU:

```
# Ubuntu/Linux 64-bit, GPU enabled:
$ sudo pip install --upgrade https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow-0.6.0-cp27-none-linux_x86_64.whl
```

3. 3. Các bước thực hiện

3.3.1. Import các thư viện cần thiết

```
import keras.models
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras_preprocessing.text import Tokenizer
```

```
from sklearn.feature_extraction.text import CountVectorizer
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re
```

3.3.2. Lọc các cột cần thiết bằng cách sử dụng DataFrame của pandas

```
path = './data/SentNew.csv'
data = pd.read_csv(path)
data = data[['Review','Label']]
print(data[['Review','Label']])
```

Trong tập dữ liệu bình luận cho bài toán này chỉ chứa các bình luận được gán nhãn là tích cực tương ứng với nhãn = ‘1’ và tiêu cực tương ứng với nhãn = ‘-1’. Xác định số đặc trưng tối đa là 2000, và sử dụng Tokenizer để véc-tơ hóa và chuyển đổi văn bản thành chuỗi. Từ đó có thể xử lý dữ liệu đầu vào cho mạng.

```
data['Review'] = data['Review'].apply(lambda x: x.lower())
data['Review'] = data['Review'].apply((lambda x: re.sub('[^a-zA-z0-9\s]','',x)))

print(data[ data['Label'] == '1'].size)
print(data[ data['Label'] == '-1'].size)

for idx,row in data.iterrows():
    row[0] = row[0].replace('rt',' ')

max_fatures =2000
X = data['Review'].values //posts

tokenizer: Tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(data['Review'].values)
X = tokenizer.texts_to_sequences(data['Review'].values)
X = pad_sequences(X)
```

```
2021-07-14 07:51:17.251862: W tensorflow/stream_executor/platform/default/dso_
2021-07-14 07:51:17.251883: I tensorflow/stream_executor/cuda/cudart_stub.cc:2
                                                  Review  Label
0       Mình thể là mình ko thể cảm nổi đồ ăn ở aeon ...     -1
1       Đôi khi thèm lên là bất chấp nắng nóng phi Và...     -1
2       Ngõ treo biển cafe trứng đúng kiểu phố cổ hà ...     -1
3       Mình thấy địa chỉ cafe Giảng ở Nguyễn Hữu Huâ...     -1
4       Mình là người Hà Nội và cũng cực kỳ khó tính ...     -1
...                                                   ...    ...
4402                                    Phục vụ tốt ah       1
4403                                  Đồ uống siêu ngon      1
4404                      Nhân viên siêu siêu nhiệt tình       1
4405                                       thật là toẹt      1
4406   Trà sữa ngon nhân viên bán hàng nhiệt Likeeeeeee      1

[4407 rows x 2 columns]
```

Thiết lập mạng LSTM. Các biến embed_dim, lstm_out, batch_size, droupout_x là các giá trị siêu tham số, các giá trị này phải trực quan hóa để chương trình chạy có kết quả xử lý tốt nhất. Thực hiện nhúng từ, bằng công cụ của Keras.

```
embed_dim = 128
lstm_out = 196

model = Sequential()
model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))
#model.add(SpatialDropout1D(0.4))
model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(2,activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 930, 128)          256000
_________________________________________________________________
lstm (LSTM)                  (None, 196)               254800
_________________________________________________________________
dense (Dense)                (None, 2)                 394
=================================================================
Total params: 511,194
Trainable params: 511,194
Non-trainable params: 0
_________________________________________________________________
```

3.3.3. Thiết lập tập dữ liệu cho việc huấn luyện và thử nghiệm

```
Y = pd.get_dummies(data['Label']).values
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.33, random_state = 42)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

Kết quả:

```
(2952, 930) (2952, 2)
(1455, 930) (1455, 2)
```

3.3.4. Huấn luyện mô hình

```
batch_size = 32
model.fit(X_train,Y_train,batch_size=batch_size,epochs=7,verbose =1 )
```

```
2021-07-14 07:51:20.120093: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2499950000 Hz
Epoch 1/7
93/93 [==============================] - 251s 2s/step - loss: 0.6247 - accuracy: 0.6646
Epoch 2/7
93/93 [==============================] - 210s 2s/step - loss: 0.3717 - accuracy: 0.8328
Epoch 3/7
93/93 [==============================] - 205s 2s/step - loss: 0.2793 - accuracy: 0.8860
Epoch 4/7
93/93 [==============================] - 207s 2s/step - loss: 0.2146 - accuracy: 0.9072
Epoch 5/7
93/93 [==============================] - 212s 2s/step - loss: 0.1875 - accuracy: 0.9272
Epoch 6/7
93/93 [==============================] - 210s 2s/step - loss: 0.2192 - accuracy: 0.9169
Epoch 7/7
93/93 [==============================] - 203s 2s/step - loss: 0.2608 - accuracy: 0.8978
18/18 [==============================] - 6s 293ms/step - loss: 0.3586 - accuracy: 0.8487
```

3.3.5. Đánh giá độ chính xác của mô hình:

```
validation_size = 880
X_validate = X_test[-validation_size:]
Y_validate = Y_test[-validation_size:]
X_test = X_test[:-validation_size]
Y_test = Y_test[:-validation_size]
score,acc = model.evaluate(X_test, Y_test, verbose =1 , batch_size = batch_size)
print("Điểm đánh giá: %.2f" % (score))
print("Độ chính xác: %.2f" % (acc))
```

```
Điểm đánh giá: 0.36
Độ chính xác: 0.83
```

Đánh giá tập dữ liệu test: Trong dữ liệu này các bình luận tích cực nhiều hơn là các bình luận tiêu cực. Do đó việc đánh giá các bình luận tích cực là tốt hơn so với bình luận tiêu cực.

Phần lớn các mô hình học sâu cho kết quả tốt hơn khi có tập dữ liệu lớn.

```
pos_cnt, neg_cnt, pos_correct, neg_correct = 0, 0, 0, 0
for x in range(len(X_validate)):
    result = model.predict(X_validate[x].reshape(1,X_test.shape[1]),batch_size=1,verbose =
2)[0]
    if np.argmax(result) == np.argmax(Y_validate[x]):
        if np.argmax(Y_validate[x]) == 0:
            neg_correct += 1
        else:
            pos_correct += 1
    if np.argmax(Y_validate[x]) == 0:
        neg_cnt += 1
    else:
        pos_cnt += 1

print("Độ chính xác với bình luận tích cực", pos_correct/pos_cnt*100, "%")
print("Độ chính xác với bình luận tiêu cực", neg_correct/neg_cnt*100, "%")
```

3.3.6. Kết quả đánh giá dữ liệu test:

```
Độ chính xác với bình luận tích cực 90.07633587786259 %
Độ chính xác với bình luận tiêu cực 79.7752808988764 %
```

Sử dụng mô hình để phân loại một bình luận là tích cực hay tiêu cực:

```
#Test bình luận là positive
twt = ['Lần đầu tiên vào về ngay lập tức bị nghiện trà đào và bông lan trứng muối chắc
ngày nào cũng phải ra đây thôi']
```

```
twt = tokenizer.texts_to_sequences(twt)
twt = pad_sequences(twt, maxlen=28, dtype='int32', value=0)
print(twt)
sentiment = model.predict(twt,batch_size=1,verbose = 2)[0]
if(np.argmax(sentiment) == 0):
   print("Tiêu cực")
elif (np.argmax(sentiment) == 1):
   print("Tích cực")
```

Kết quả thực hiện

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0 144 501  44]]
WARNING:tensorflow:Model was constructed with shape (None, 930) for input KerasTensor(type_spec=
1/1 - 1s
Tích cực
```

Đồ án đã xây dựng chương trình thực nghiệm để phân loại dữ liệu bình luận tiếng Việt là tích cực hay tiêu cực. Chương trình mới chỉ mang tính thử nghiệm phương pháp và kết quả thực nghiệm cho bài toán chưa cao với độ chính xác của mô hình là 0.83% do bộ dữ liệu huấn luyện nhỏ và chưa thực hiện được các tối ưu về tham số. Trong thời gian tiếp theo, em sẽ tiếp tục nghiên cứu và thử nghiệm với các bộ dữ liệu lớn hơn và thực hiện các bước tiền xử lý dữ liệu nhằm nâng cao hiệu năng của mô hình.

KẾT LUẬN

Trong đồ án này em đã tìm hiểu về một số phương pháp biểu diễn từ bằng véc-tơ, một số phương pháp học sâu sử dụng đặc trưng véc-tơ từ làm đầu vào xử lý và úng dụng phương pháp LSTM sử dụng véc-tơ từ là đặc trưng cho bài toán phân loại quan điểm bình luận tiếng Việt.

Đồ án sử dụng tập dữ liệu bình luận quan điểm người dùng về các dịch vụ ăn uống đã được gán nhãn quan điểm được lưu trữ trong tệp dữ liệu gồm 4086 bình luận trong đó 1765 bình luận tiêu cực và 2321 bình luận tích cực. Các bình luận tích cực được gán nhãn giá trị là '1' bình luận tiêu cực là '-1'.

Đồ án đã thực hiện cài đặt TensorFlow, chương trình thực nghiệm được viết trên ngôn ngữ Python. Sử dụng phương pháp nhúng từ và phân loại LSTM trong thư viện Keras để phân loại quan điểm với độ chính xác mô hình đạt 83%.

Độ chính xác của mô hình chưa thật sự cao vì dữ liệu còn một số hạn chế do chưa tiền xử lý văn bản do các lỗi viết tắt trong các bình luận và dữ liệu huấn luyện còn nhỏ. Trong thời gian tới em sẽ tiếp tục nghiên cứu để cải thiện các hạn chế nhằm nâng cao độ chính xác cho bài toán.

TÀI LIỆU THAM KHẢO

[1] Python Machine Learning By Example by Yuxi Liu, 2017.

[2] Neural Network Embeddings Explained by Will Koehrsen, 2018

[3] Using latent semantic analysis to improve access to textual information, ST Dumais-GW Furnas-TK Landauer-S. Deerwester-R. Harshman, Released 1922.

[4] Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013.

[5] A Beginner's Guide to Latent Dirichlet Allocation(LDA)

[6] XinRong, Word2vec Parameter Learning Explained, 2014.

[7] Chris McCormick - Word2Vec Tutorial - The Skip-Gram Model, 2016.

[8] Word2Vec Tutorial: The Continuous Bag-of-Words Model bt Alex Minnaar, 2015.

[9] Introduction to N-grams by Tobias Sterbak, 2019

[10] Introduction to word embeddings and its applications, 2020

[11] Deep Learning: Recurrent Neural Networks in Python: LSTM, GRU, and more RNN machine learning architectures in Python and Theano , 2016

[12] Andrew Ng, Machine Learning course ,2020

[13] Christopher Olah (2015), Understanding LSTM networks in Colah's blog

[14] Improved Semantic Representations FromTree-Structured Long Short-Term Memory Networks ,2015.