

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH  
KHOA CÔNG NGHỆ THÔNG TIN



## TIỂU LUẬN CHUYÊN NGÀNH

---

# Tìm hiểu Word Embedding cho biểu diễn văn bản

---

GVHD: ThS. Quách Đình Hoàng

SVTH1: Trần Trọng Nghĩa (17133040)

SVTH2: Vũ Anh Thái Dương (17133010)

THÀNH PHỐ HỒ CHÍ MINH, 1/2021

# Lời cam đoan

Luận văn này là công trình nghiên cứu của chúng tôi, được thực hiện dưới sự hướng dẫn khoa học của Ths. Quách Đình Hoàng. Các số liệu, những kết luận nghiên cứu được trình bày trong luận văn này hoàn toàn trung thực.

Chúng tôi xin hoàn toàn chịu trách nhiệm về lời cam đoan này.

**Sinh viên**

**Trần Trọng Nghĩa**

**Vũ Anh Thái Dương**

# Lời cảm ơn

Với tình cảm sâu sắc và chân thành nhất, chúng tôi xin được bày tỏ lòng biết ơn đến với Ths. Quách Đình Hoàng, giảng viên khoa công nghệ thông tin trường Đại học Sư phạm Kỹ thuật TP. Hồ Chí Minh đã tận tình chỉ bảo cho chúng tôi các kiến thức cần thiết trong thời gian vừa qua để chúng tôi có thể hoàn thành đề tài này.

Cuối cùng, chúng tôi xin cảm ơn đến các tác giả của những bài báo khoa học mà chúng tôi đã tham khảo. Các bài báo này giúp cho chúng tôi tiếp thu thêm được nhiều kiến thức mới và quan trọng là hiểu rõ hơn về đề tài đang nghiên cứu.

**Sinh viên**

**Trần Trọng Nghĩa**

**Vũ Anh Thái Dương**

# Mục lục

<b>1</b>	<b>TỔNG QUAN VỀ ĐỀ TÀI</b>	<b>1</b>
1.1	Giới thiệu đề tài . . . . .	1
1.2	Mục tiêu . . . . .	2
1.3	Bố cục của báo cáo . . . . .	2
<b>2</b>	<b>CÁC KHÁI NIỆM CƠ BẢN</b>	<b>4</b>
2.1	Tổng quan về word embedding . . . . .	4
2.1.1	Khái niệm về word embedding . . . . .	4
2.1.2	Tại sao không sử dụng one-hot vector? . . . . .	6
2.2	Các dạng nhúng từ phổ biến . . . . .	8
2.2.1	Nhúng từ dựa trên tần suất . . . . .	8
2.2.1.1	Count vector . . . . .	8
2.2.1.2	TF-IDF vector . . . . .	9
2.2.1.3	Ma trận đồng xuất hiện . . . . .	11
2.2.2	Nhúng từ dựa trên dự đoán . . . . .	13
<b>3</b>	<b>MÔ HÌNH WORD2VEC</b>	<b>14</b>
3.1	Tổng quan về word2vec . . . . .	14
3.1.1	Giới thiệu . . . . .	14
3.1.2	Kiến trúc tổng quát . . . . .	15

3.1.3	Dữ liệu huấn luyện . . . . .	16
3.1.4	Hidden Layer (Lớp nhúng) . . . . .	17
3.1.5	Output layer . . . . .	18
3.2	Mô hình skip-gram . . . . .	19
3.2.1	Khái niệm . . . . .	19
3.2.2	Huấn luyện mô hình . . . . .	21
3.2.3	Cách hoạt động của mô hình . . . . .	22
3.2.3.1	Forward propagation . . . . .	23
3.2.3.2	Backward propagation . . . . .	25
3.3	Mô hình continuous bag of words (CBOW) . . . . .	32
3.3.1	Khái niệm . . . . .	32
3.3.2	Huấn luyện mô hình . . . . .	33
3.4	Đánh giá mô hình . . . . .	34
<b>4</b>	<b>MÔ HÌNH GLOBAL VECTOR</b>	<b>36</b>
4.1	Khái niệm Global Vector . . . . .	36
4.2	Ý tưởng triển khai mô hình . . . . .	37
4.3	Huấn luyện mô hình . . . . .	43
4.4	Đánh giá mô hình . . . . .	47
<b>5</b>	<b>XÂY DỰNG HỆ THỐNG PRODUCT RECOMMENDATION SỬ DỤNG WORD2VEC</b>	<b>49</b>
5.1	Mục tiêu . . . . .	49
5.2	Mô tả tập dữ liệu . . . . .	50
5.3	Giải pháp . . . . .	50
5.4	Thực hiện . . . . .	50
<b>6</b>	<b>KẾT LUẬN</b>	<b>58</b>

6.1	Đóng góp của tiểu luận . . . . .	58
6.2	Hạn chế của tiểu luận . . . . .	58
6.3	Phương hướng phát triển tiểu luận trong tương lai . . . . .	59
<b>Tài liệu tham khảo</b>		<b>60</b>

# Danh sách hình vẽ

2.1	Biểu diễn dưới dạng phân tán . . . . .	5
2.2	Biểu diễn phân tán của các vector từ trên không gian 3 chiều . .	6
2.3	Biểu diễn các vector từ ở dạng one-hot và tích hai vector one-hot	7
2.4	Ma trận count vector . . . . .	9
2.5	Bảng TF-IDF . . . . .	10
2.6	Ma trận đồng xuất hiện . . . . .	12
3.1	Kiến trúc neural network của mô hình word2vec . . . . .	16
3.2	Xử lý dữ liệu thô sang dữ liệu huấn luyện cho mô hình . . . . .	16
3.3	Ma trận nhúng từ . . . . .	18
3.4	Tính toán đầu ra của tế bào thần kinh đầu ra cho từ “car” . . .	19
3.5	Kiến trúc của mô hình skip-gram . . . . .	20
3.6	Cấu trúc neural network của mô hình skip-gram . . . . .	23
3.7	Sử dụng hàm softmax để tính các giá trị xác suất . . . . .	25
3.8	Tính toán sai số dự đoán trên từng từ ngữ cảnh . . . . .	26
3.9	Tính toán tổng sai số dự đoán trên tất cả các từ ngữ cảnh . . .	27
3.10	Tổng sai số dự đoán qua các lần cập nhật để hội tụ về giá trị 0 .	27
3.11	Tính toán gradient của ma trận trọng số đầu vào $\nabla W_{input}$ . . . .	29
3.12	Tính toán gradient của ma trận trọng số đầu ra $\nabla W_{output}$ . . . .	30
3.13	Cập nhật ma trận $W_{input}$ . . . . .	31

3.14	Cập nhật ma trận $W_{output}$ . . . . .	31
4.1	Mô hình kiến trúc của glove . . . . .	37
4.2	Xác suất đồng xuất hiện . . . . .	38
4.3	Mô phỏng khoảng cách vector ứng với $w_i - w_j$ . . . . .	40
4.4	Hàm trọng số $f$ với $\alpha = 3/4$ . . . . .	43
4.5	Minh họa cho ma trận đồng xuất hiện . . . . .	44
4.6	Ma trận xác suất đồng xuất hiện P . . . . .	45
4.7	Phân rã ma trận đồng xuất hiện [VxV] ban đầu thành hai ma trận con [VxV] và [VxV] . . . . .	46
4.8	Cách thức hoạt động để thu được ma trận nhúng từ [VxN] . . .	47
5.1	Kiểm tra dữ liệu bán hàng từ file excel . . . . .	51
5.2	Tiền xử lý dữ liệu . . . . .	51
5.3	Chia tập dữ liệu thành train data and validation data . . . . .	52
5.4	Tạo một chuỗi các sản phẩm đã mua của một khách hàng trên tập train data . . . . .	52
5.5	Tạo một chuỗi các sản phẩm đã mua của một khách hàng trên tập validation data . . . . .	52
5.6	Xây dựng word2vec embeddings cho các sản phẩm . . . . .	53
5.7	Các thông số của mô hình sau khi đã train . . . . .	53
5.8	Các sản phẩm mà khách hàng đã mua . . . . .	54
5.9	Tạo danh sách gồm mã sản phẩm-tên sản phẩm . . . . .	55
5.10	Function tìm sự tương đồng giữa các vector sản phẩm . . . . .	55
5.11	Đề xuất các sản phẩm tương tự cho sản phẩm có tên là (" <i>YEL- LOW COAT RACK PARIS FASHION</i> ") . . . . .	55
5.12	Function trả về giá trị trung bình của các vector sản phẩm của một khách hàng đã mua . . . . .	56



5.13 Đề xuất các sản phẩm trong tương lai mà khách hàng có thể mua	57
5.14 Đề xuất các sản phẩm dựa trên 10 sản phẩm mà khách hàng đã mua gần đây nhất . . . . .	57

# Chương 1

## TỔNG QUAN VỀ ĐỀ TÀI

### 1.1 Giới thiệu đề tài

Ngày nay, con người luôn có nhu cầu được giao tiếp. Chính từ nhu cầu cơ bản này mà một lượng lớn dữ liệu dạng văn bản được tạo ra mỗi ngày. Dữ liệu dạng văn bản rất đa dạng từ mạng xã hội, ứng dụng trò chuyện, email, đánh giá sản phẩm, tài liệu nghiên cứu và sách báo. Việc giúp máy tính nói chung và các mô hình machine learning và deep learning nói riêng có thể hiểu và xây dựng các ứng dụng từ dữ liệu này. Nhằm mục đích là đưa ra cách thức hỗ trợ hoặc giúp đưa ra quyết định dựa trên ngôn ngữ của con người là một việc trở nên quan trọng. Các ứng dụng này có thể là phân tích đánh giá sản phẩm, đề xuất các bài hát tương tự theo ngữ cảnh, dịch thuật. Điểm chung các ứng dụng này là đều xử lý với một lượng lớn dữ liệu dạng văn bản. Nhưng các mô hình này không thể hiểu được trực tiếp dữ liệu dạng văn bản. Vì vậy, biểu diễn vector từ dưới dạng số (nhúng từ) là điều kiện tiên quyết cho hầu hết các mô hình machine learning và deep learning hiện nay.

Các kỹ thuật biểu diễn vector từ dưới dạng số (nhúng từ) truyền thống như one-hot rất dễ để triển khai. Tuy nhiên, các kỹ thuật này lại làm mất đi thông tin về mặt ý nghĩa, mối quan hệ ngữ nghĩa giữa các từ, vốn là thành phần quan trọng để xây dựng nên các ứng dụng xử lý ngôn ngữ tự nhiên.

Để khắc phục những hạn chế từ các kỹ thuật truyền thống này thì một kỹ thuật mới được ra đời. Đó là kỹ thuật word embedding cho biểu diễn văn bản. Đây là một kỹ thuật biểu diễn phân tán (distributed representations) vector từ mới, nơi các từ được ánh xạ tới các vector số mà vẫn giữ lại được thông tin về mặt ý nghĩa, mối quan hệ ngữ nghĩa giữa các từ.

## 1.2 Mục tiêu

Trong tiểu luận này chúng tôi sẽ tìm hiểu về những khái niệm cơ bản của kỹ thuật word embedding. Cụ thể về lý thuyết, chúng tôi sẽ tìm hiểu mô hình nổi tiếng của word embedding đó là mô hình word2vec và mô hình glove. Về ứng dụng, chúng tôi tiến hành xây dựng hệ thống product recommendation (đề xuất sản phẩm) đơn giản cho trang bán hàng trực tuyến sử dụng word2vec.

## 1.3 Bố cục của báo cáo

Cấu trúc của bài báo cáo dự định sẽ được chia thành các chương sau đây:

- Chương 2: Những khái niệm cơ bản
- Chương 3: Mô hình Word2vec
- Chương 4: Mô hình Global Vector
- Chương 5: Xây dựng hệ thống product recommendation sử dụng Word2vec

- Chương 6: Kết luận

## Chương 2

# CÁC KHÁI NIỆM CƠ BẢN

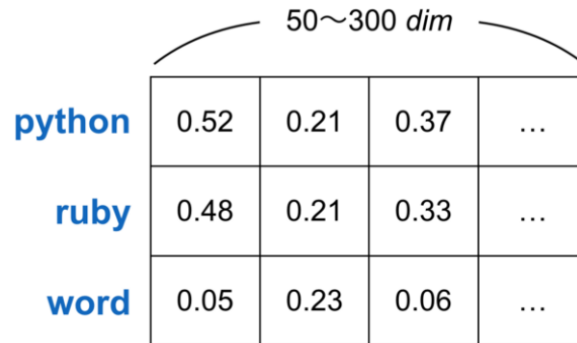
### 2.1 Tổng quan về word embedding

#### 2.1.1 Khái niệm về word embedding

Kỹ thuật ánh xạ từ ngữ trong kho ngữ liệu đến một tập hợp các vector số trong một không gian chiều  $N$  được xác định trước được gọi là kỹ thuật word embedding. Trong vài năm gần đây, word embedding dần trở thành kiến thức cơ bản trong xử lý ngôn ngữ tự nhiên.

Word embeddings cố gắng nắm bắt ý nghĩa, mối quan hệ ngữ nghĩa của mỗi từ trong kho ngữ liệu dựa trên việc sử dụng các từ này trong câu. Các từ có ý nghĩa và mối quan hệ ngữ nghĩa tương tự nhau sẽ có các biểu diễn vector tương tự nhau. Đồng thời mỗi từ trong kho ngữ liệu sẽ có một tập hợp đại diện vector duy nhất được xem là vector đặc trưng của một từ.

Biểu diễn phân tán (nhúng từ) là biểu diễn của một từ dưới dạng vector có các giá trị số thực chiều thấp [1]. Nó thường được thể hiện từ khoảng 50 đến 300 chiều. Ví dụ, hình 2.1 biểu diễn các từ dưới dạng biểu diễn phân tán trong không gian  $N$  chiều [2] như sau:



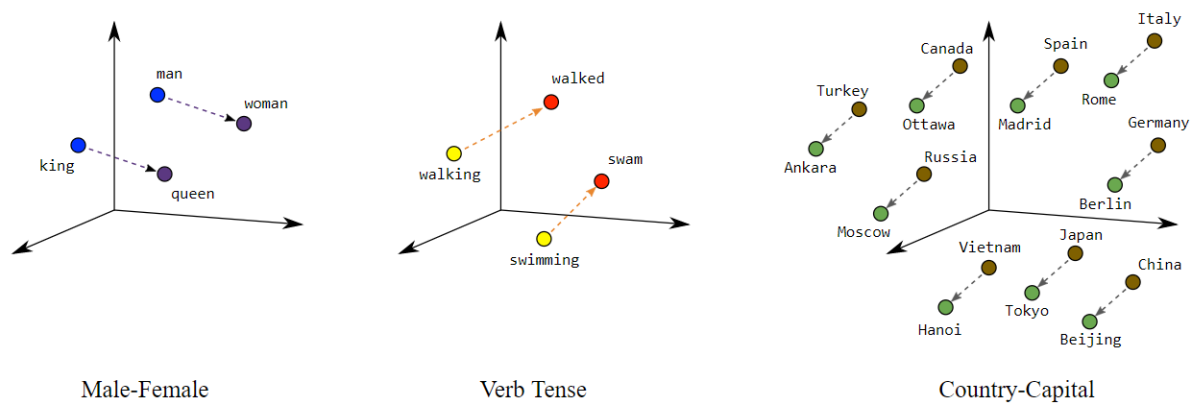
The diagram shows a table representing word embeddings. Above the table, a curved line spans the width of the data columns, labeled "50~300 dim". The table has three rows, each labeled on the left with a word in blue: "python", "ruby", and "word". Each row contains four cells: the first three contain numerical values, and the fourth contains an ellipsis "...".

<b>python</b>	0.52	0.21	0.37	...
<b>ruby</b>	0.48	0.21	0.33	...
<b>word</b>	0.05	0.23	0.06	...

Hình 2.1: Biểu diễn dưới dạng phân tán

Bằng cách sử dụng biểu diễn phân tán, chúng ta có thể tính toán độ giống nhau giữa các từ bằng phép toán vector mà các phương pháp truyền thống khó có thể làm được. Nhìn vào hình 2.1 ở trên, các giá trị được biểu diễn trong mỗi vector từ cho chúng ta thấy được rằng sự tương đồng giữa hai vector từ “*python*” và “*ruby*” là cao hơn so sự tương đồng giữa hai vector từ “*python*” và “*word*”. Ngoài ra, ngay cả khi số lượng từ vệt tăng lên, chúng ta cũng không cần tăng số chiều của mỗi từ mà vẫn giữ được các mối quan hệ ý nghĩa.

Hình 2.2, mô tả việc biểu diễn phân tán của các vector từ trên không gian 3 chiều. Chúng ta có thể thấy rằng khoảng cách giữa cặp từ “*Vietnam*” , “*Hanoi*” có khoảng cách bằng với cặp từ “*Janpan*” , “*Tokyo*”, điều này cho chúng ta thấy hai cặp từ này có mối quan hệ gần gũi với nhau.

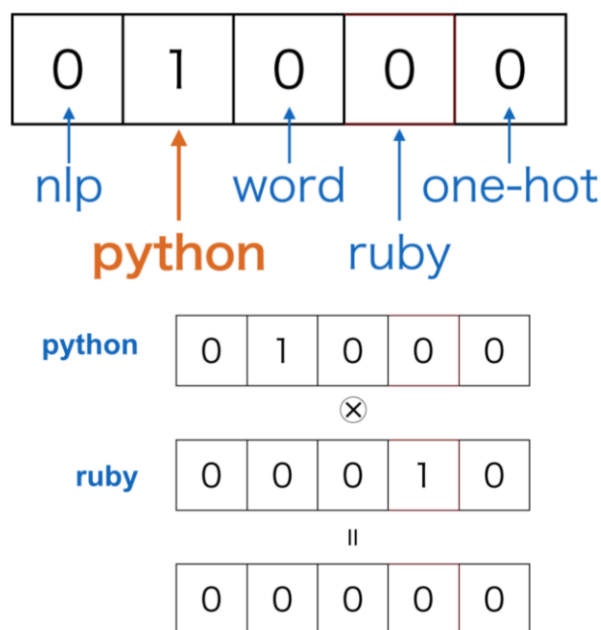


Hình 2.2: Biểu diễn phân tán của các vector từ trên không gian 3 chiều

### 2.1.2 Tại sao không sử dụng one-hot vector?

Một trong những cách đơn giản nhất để biểu diễn vector từ là one-hot vector. Mỗi từ tương ứng một-một với các số nguyên liên tiếp từ 0 đến  $V - 1$  ( $V$  là kích thước tập từ vựng), được gọi là chỉ số của từ. Giả sử chỉ số của một từ là  $i$ . Để thu được biểu diễn one-hot vector của từ đó, chúng ta tạo một vector có  $V$  phần tử có giá trị là 0 và đặt phần tử thứ  $i$  có giá trị bằng 1 [3].

Hình 2.3, mô tả việc biểu diễn các vector từ ở dạng one-hot và tích của hai vector one-hot [1].



Hình 2.3: Biểu diễn các vector từ ở dạng one-hot và tích hai vector one-hot

Tuy phương pháp one-hot dễ thực hiện nhưng nó sẽ gặp phải một số vấn đề sau đây:

- Chi phí tính toán lớn: Trong thực tế, chúng ta sẽ có hàng triệu câu và hàng triệu từ trong kho ngữ liệu. Ta có thể tưởng tượng kích thước của một one-hot vector lớn đến cỡ nào khi kho ngữ liệu của chúng ta có kích thước hàng triệu. Một số "1" tại vị trí từ đó và hàng loạt số "0" không mang lại nhiều ý nghĩa. Điều này sẽ dẫn đến sự kém hiệu quả về thời gian thực hiện và tốn kém về mặt tài nguyên lưu trữ.
- Độ tương tự giữa các từ: Khi một vector từ được biểu diễn bằng one-hot vector thì nó sẽ làm mất đi ý nghĩa và mối quan hệ ngữ nghĩa của từ đó. Vì vậy, các vector one-hot không thể biểu diễn một cách chính xác độ tương tự giữa các từ khác nhau. Trong ví dụ trên, thì chúng ta thấy rằng tích giữa hai vector "*python*" và "*ruby*" là 0, giá trị này thì không mang



lại nhiều ý nghĩa. Đó là vì độ tương tự cô-sin giữa hai vector one-hot bất kỳ đều bằng 0, nên rất khó sử dụng vector one-hot để biểu diễn độ tương tự giữa các từ khác nhau [3].

## 2.2 Các dạng nhúng từ phổ biến

### 2.2.1 Nhúng từ dựa trên tần suất

Nhúng từ dựa trên tần suất là phương pháp rất cơ bản, dễ dàng và nhanh chóng để biểu diễn vector từ. Chúng hoạt động dựa trên số lượng từ trong mỗi tài liệu. Nó có thể kể đến như là count vector, TF-IDF vector, ma trận đồng xuất hiện.

#### 2.2.1.1 Count vector

Count vector là xây dựng mô hình cho từng tài liệu bằng cách đếm số lần mỗi từ xuất hiện trong tài liệu [2].

Ví dụ sau đây sẽ giúp chúng ta dễ hình dung hơn, ở đây chúng ta có  $D$  tài liệu và  $T$  là số lượng các từ vựng. Sau đó, kích thước của ma trận count vector sẽ là  $D \times T$ . Chúng ta xem hai câu sau:

$D1$ : “*The cat sat on the hat*”.

$D2$ : “*The dog ate the cat and the hat*”.

Từ hai tài liệu này, vốn từ vựng của chúng ta như sau: “*the*”, “*cat*”, “*sat*”, “*on*”, “*hat*”, “*dog*”, “*ate*”, “*and*”.

Bây giờ, chúng ta đếm số lần xuất hiện của mỗi từ trong mỗi tài liệu. Trong *D1*, từ “*the*” xuất hiện hai lần, và “*cat*”, “*sat*”, “*on*” và “*hat*” mỗi từ xuất hiện một lần, thực hiện tương tự cho *D2* chúng tôi được ma trận count vector được thể hiện qua hình 2.4 như sau:

	the	cat	sat	on	hat	dog	ate	and
Document 1	2	1	1	1	1	0	0	0
Document 2	3	1	0	0	1	1	1	1

Hình 2.4: Ma trận count vector

Bây giờ, các giá trị trong một cột được hiểu là vector từ tương ứng trong ma trận *M*. Ví dụ, vector từ “*cat*” trong ma trận *M* là [1,1]. Ở đây, các hàng tương ứng với các tài liệu trong kho ngữ liệu và các cột tương ứng với các từ trong từ điển.

Tuy nhiên, phương pháp này gặp một nhược điểm đó là khi dữ liệu của chúng ta lớn, một số từ có tần suất xuất hiện lớn nhưng lại mang theo rất ít thông tin có ý nghĩa về nội dung thực tế của tài liệu (cụ thể là các mạo từ trong tiếng Anh như: *a*, *an*, *the*, ...). Và nếu chúng ta thống kê cả số lượng data này thì tần suất của các từ này sẽ làm mờ đi giá trị của những từ mang nhiều thông tin nhưng ít gặp hơn.

#### 2.2.1.2 TF-IDF vector

Đây là một phương pháp khác cũng dựa trên tần suất xuất hiện của các từ trong tài liệu. Nhưng khắc phục được một số nhược điểm của phương pháp count vector. Nó không chỉ tính đến tần suất xuất hiện của các từ trong mỗi tài liệu mà còn tính nó ở toàn bộ kho ngữ liệu [2].

Một số từ như “*a*”, “*an*”, “*the*”, xuất hiện thường xuyên hơn các từ khác trong mọi tài liệu. Những từ này dường như không làm thay đổi về mặt ý nghĩa, mối quan hệ ngữ nghĩa, ngữ cảnh của câu. Vì vậy, chúng tôi sẽ giảm trọng lượng các từ thường xuyên xuất hiện này trong hầu hết các tài liệu bằng công thức như sau:

$$TF - IDF(d, t) = TF(d, t) * IDF(t) \quad (2.1)$$

Trong đó:

- $TF(d, t) = (\text{Số lần từ } t \text{ xuất hiện trong một tài liệu } d) / (\text{Tổng số từ trong một tài liệu } d)$ .
- $IDF(d, t) = \log(N/n)$ , trong đó,  $N$  là tổng số tài liệu và  $n$  là số lượng tài liệu mà từ  $t$  này xuất hiện trong đó

Để dễ hình dung chúng ta sẽ xem tiếp bảng TF-IDF vector được thể hiện qua hình 2.5 như sau [4].

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	$\log(2/2) = 0$	0	0
Car	1/7	0	$\log(2/1) = 0.3$	0.043	0
Truck	0	1/7	$\log(2/1) = 0.3$	0	0.043
Is	1/7	1/7	$\log(2/2) = 0$	0	0
Driven	1/7	1/7	$\log(2/2) = 0$	0	0
On	1/7	1/7	$\log(2/2) = 0$	0	0
The	1/7	1/7	$\log(2/2) = 0$	0	0
Road	1/7	0	$\log(2/1) = 0.3$	0.043	0
Highway	0	1/7	$\log(2/1) = 0.3$	0	0.043

Hình 2.5: Bảng TF-IDF

Ở đây chúng ta dễ dàng nhận thấy rằng TF-IDF của các từ thường xuyên xuất hiện trong tài liệu mà không mang quá nhiều ý nghĩa “*the*”, “*is*”, “*on*” thì TF-IDF của chúng sẽ mang giá trị là 0. Còn TF-IDF của “*car*”, “*truck*”, “*road*”, and “*highway*” thì mang giá trị khác 0. Điều này cho thấy rằng các từ này mang nhiều ý nghĩa hơn.

Tuy nhiên, nếu lượng từ vựng quá nhiều thì ma trận TF-IDF được tạo ra sẽ rất lớn và sẽ tốn rất nhiều bộ nhớ và việc xử lý ma trận sẽ rất tốn kém.

### 2.2.1.3 Ma trận đồng xuất hiện

Ma trận đồng xuất hiện mô tả việc xuất hiện của các từ cùng với nhau, từ đó lần lượt nắm bắt được mối quan hệ ý nghĩa giữa các từ. Ma trận đồng xuất hiện được tính đơn giản bằng cách đếm hai hoặc nhiều từ xuất hiện cùng với nhau trong một kho ngữ liệu nhất định [5].

Giả sử chúng tôi có kho ngữ liệu như sau:

*“tôi yêu công\_việc”.*

*”tôi thích NLP”.*

*“tôi ghét ở một\_mình”.*

Từ đó chúng tôi xây dựng được ma trận đồng xuất hiện được thể hiện như hình 2.6 như sau:

	tôi	yêu	công_việc	thích	NLP	ghét	ở	một_mình	.
tôi	0	1	0	1	0	1	0	0	0
yêu	1	0	1	0	0	0	0	0	0
công_việc	0	1	0	0	0	0	0	0	1
thích	1	0	0	0	1	0	0	0	0
NLP	0	0	0	1	0	0	0	0	1
ghét	1	0	0	0	0	0	1	0	0
ở	0	0	0	0	0	1	0	1	0
một_mình	0	0	0	0	0	0	1	0	0
.	0	0	1	0	1	0	0	0	0

Hình 2.6: Ma trận đồng xuất hiện

Các giá trị hiển thị ở trên được gọi là tần số bigram [6]. Với một kho ngữ liệu có  $V$  từ vựng, chúng ta cần một ma trận có kích thước  $V \times V$  để đại diện cho tần số bigram của tất cả các cặp từ có thể. Một ma trận như vậy rất thưa thớt vì hầu hết các giá trị bằng 0.

Tuy nhiên, ma trận đồng xuất hiện này không phải là phương pháp biểu diễn vector từ thường dùng. Thay vào đó, ma trận đồng xuất hiện này bị phân rã bằng cách sử dụng các kỹ thuật thu giảm số chiều như PCA, SVD. Thành các thành phần khác và sự kết hợp của các thành phần này tạo thành biểu diễn phân tán vector từ (nhúng từ). Mà điều này, chúng tôi sẽ đề cập chi tiết hơn ở chương mô hình glove ở phần sau.

Ưu điểm:

- Ma trận đồng xuất hiện bảo tồn mối quan hệ ý nghĩa và ngữ nghĩa giữa các từ.

Nhược điểm:

- Ma trận đồng xuất hiện đòi hỏi bộ nhớ rất lớn để lưu trữ.

### 2.2.2 Nhúng từ dựa trên dự đoán

Nhúng từ dựa trên dự đoán là phương pháp xây dựng các biểu diễn vector từ dựa vào các mô hình dự đoán (xác suất). Tiêu biểu nhất chính là mô hình word2vec, nó gồm hai kiến trúc: continuous bag of words (CBOW) và skip-gram [6]. Cả hai kiến trúc này đều được xây dựng dựa trên một neural network gồm 3 lớp:

- Một input layer.
- Một hidden layer.
- Một output layer.

Mục đích chính của neural network [7] này là học các trọng số biểu diễn vector từ. Điều này, chúng tôi sẽ đề cập chi tiết hơn ở chương mô hình word2vec.

## Chương 3

# MÔ HÌNH WORD2VEC

### 3.1 Tổng quan về word2vec

#### 3.1.1 Giới thiệu

Mô hình word2vec được giới thiệu lần đầu tại ra Google vào năm 2013 bởi Mikolov et al [8]. Đây là một mô hình dự đoán dựa trên deep learning để tính toán và tạo ra các biểu diễn vector từ có chất lượng cao, nắm bắt được thông tin về mặt ý nghĩa, mối quan hệ ngữ nghĩa của các từ [9].

Mô hình word2vec cho phép chúng ta cộng và trừ các vector từ như thể là chúng đang nắm bắt ý nghĩa của một từ. Ví dụ nổi tiếng vector từ “*Queen*” được tính bằng cách trừ vector từ “*Man*” từ vector từ “*King*” và cộng vector từ “*Woman*” ( $\text{King} - \text{Man} + \text{Woman} = \text{Queen}$ ).

Có hai mô hình kiến trúc khác nhau của mô hình word2vec để tạo ra các đại diện nhúng từ. Đó là:

- Mô hình skip-gram.

- Mô hình continuous bag of words (CBOW).

### 3.1.2 Kiến trúc tổng quát

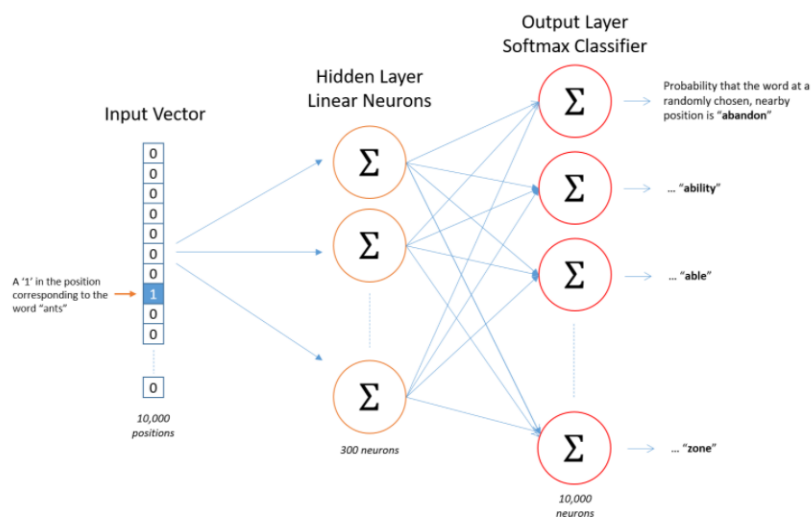
Mô hình word2vec sử dụng một kiến trúc neural network đơn giản với một hidden layer duy nhất để thực hiện một nhiệm vụ nhất định, nhưng sau đó chúng tôi sẽ không thực sự sử dụng neural network này cho mục tiêu mà chúng tôi đã đào tạo nó. Thay vào đó, mục tiêu thực sự của chúng tôi chỉ là để tìm hiểu ma trận trọng số của hidden layer. Các trọng số này chính là các vector từ mà chúng tôi đang cố gắng tìm hiểu.

Mô hình này gồm 3 thành phần chính [10]:

- Input layer: Vector được biểu diễn one-hot.
- Hidden layer: Lớp nhúng.
- Output layer: Vector được biểu diễn one-hot (ở giai đoạn đào tạo). Trong quá trình đánh giá thì output layer là một phân bố xác suất.

Sau đây là kiến trúc neural network của mô hình word2vec [11] được thể hiện qua hình 3.1.

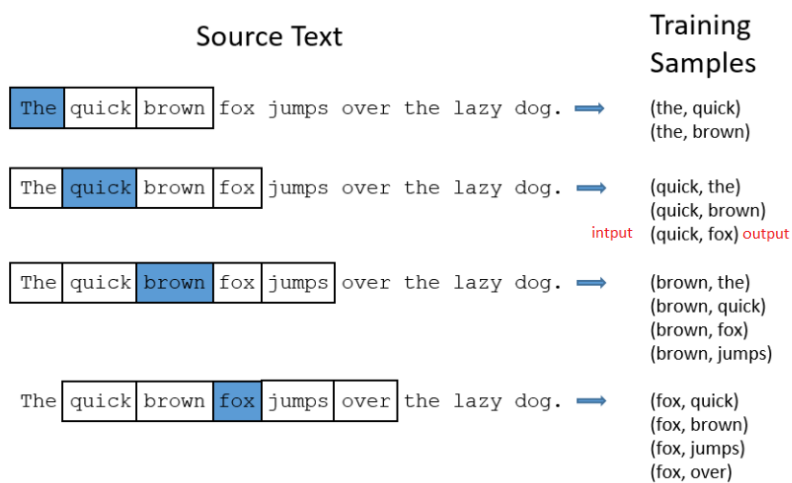




Hình 3.1: Kiến trúc neural network của mô hình word2vec

### 3.1.3 Dữ liệu huấn luyện

Chúng tôi sẽ thao tác đơn giản với kho ngữ liệu thô để đưa nó vào cấu trúc nhất định. Hình 3.2, mô tả cách thức mà chúng tôi xử lý dữ liệu thô sang dữ liệu huấn luyện cho mô hình.



Hình 3.2: Xử lý dữ liệu thô sang dữ liệu huấn luyện cho mô hình

Chúng tôi sử dụng ô màu xanh đại diện cho từ trung tâm (input) và từ này sẽ được biểu diễn one-hot. Tiếp theo, các ô màu trắng đại diện cho từ ngữ cảnh (output) và cũng được biểu diễn one-hot. Bất kỳ từ nào trong cửa sổ ngữ cảnh ngoại trừ từ trung tâm thì được gọi là từ ngữ cảnh. Hai điểm dữ liệu được tạo từ một cửa sổ ngữ cảnh duy nhất. Kích thước của cửa sổ ngữ cảnh là một siêu tham số do người dùng xác định.

Trong ví dụ trên [11] chúng tôi sử dụng kích thước cửa sổ ngữ cảnh là 2. Kích thước cửa sổ ngữ cảnh càng cao, hiệu suất của mô hình càng tốt. Nhưng khi có kích thước cửa sổ ngữ cảnh lớn, chúng ta sẽ mất thêm nhiều thời gian tính toán, khi lượng dữ liệu tăng lên đáng kể.

### 3.1.4 Hidden Layer (Lớp nhúng)

Lớp nhúng lưu trữ các vector từ của tất cả các từ có trong kho từ vựng. Đây là một ma trận có kích thước ([kích thước kho từ vựng x kích thước nhúng]). Kích thước nhúng (số chiều) này là một siêu tham số có thể điều chỉnh theo mô hình. Nó càng cao, mô hình sẽ hoạt động tốt hơn. Nhưng để mô hình được tối ưu nhất thì kích thước không nên vượt quá 500.

Ma trận này [10] được khởi tạo ngẫu nhiên và được tinh chỉnh từng chút một, trong quá trình tối ưu hóa để các vector của mỗi từ có thể dự đoán những từ xung quanh.

Mục đích cuối cùng của chúng tôi là tìm ra ma trận nhúng từ tối ưu nhất. Ma trận nhúng từ được thể hiện qua hình 3.3 như sau:

cat	0.01	0.2	0.9	....	0.64
dog	0.0	0.2	0.89	....	0.71
walk	0.3	0.0	0.6	....	0.09
⋮					
pencil	0.6	0.76	0.1	....	0.29

←----- Embedding size -----→

↑  
Vocabulary size  
↓

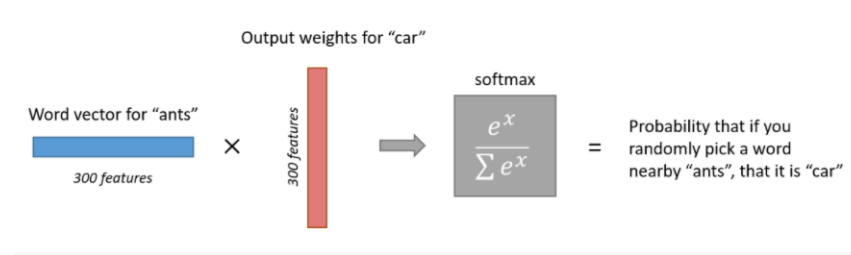
Hình 3.3: Ma trận nhúng từ

### 3.1.5 Output layer

Output layer được tính bởi một softmax regression [12]. Mục đích chính của nó là mỗi tế bào thần kinh đầu ra (các từ trong kho từ vựng) là tạo ra các giá trị xác suất đầu ra có trị trong khoảng từ 0 đến 1, và tổng của tất cả các giá trị đầu ra này sẽ là 1.

Cụ thể, mỗi tế bào thần kinh đầu ra có một vector trọng số, mà khi nó nhân với vector trọng số của hidden layer, sau đó kết quả này sẽ được đưa vào hàm softmax ( $\exp(x)$ ) để thu được các giá trị xác suất.

Hình 3.4, mô tả ví dụ về tính toán đầu ra của tế bào thần kinh đầu ra cho từ “car”.



Hình 3.4: Tính toán đầu ra của tế bào thần kinh đầu ra cho từ “car”

Tuy nhiên, neural network không biết bất cứ điều gì về sự liên quan về mặt ý nghĩa giữa từ đầu vào và từ đầu ra. Để hiểu được ý nghĩa, nó phụ thuộc vào dữ liệu đào tạo của chúng tôi. Ví dụ, mỗi lần xuất hiện duy nhất của từ “*york*” được đi trước bởi từ “*new*”. Đó là, ít nhất là theo dữ liệu đào tạo, có một xác suất 100% rằng từ “*new*” sẽ ở xung quanh từ “*york*”. Tuy nhiên đối với dữ liệu đào tạo khác, nếu chúng tôi lấy 10 từ trong xung quanh từ “*york*” và chọn ngẫu nhiên một từ trong số đó, thì xác suất xuất hiện của từ “*new*” với từ “*york*” không còn là 100%.

## 3.2 Mô hình skip-gram

### 3.2.1 Khái niệm

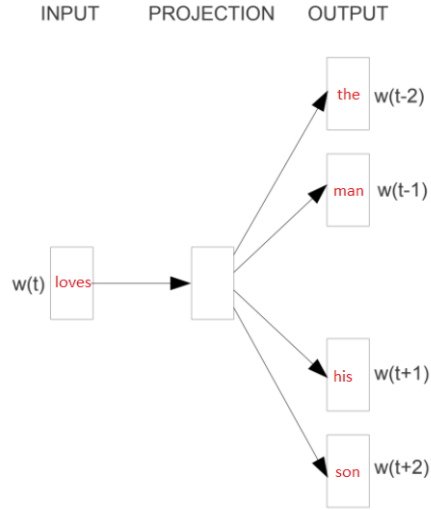
Mô hình skip-gram giả định rằng một từ có thể được sử dụng để sinh ra các từ xung quanh nó trong một chuỗi văn bản. Mô hình này quan tâm đến xác suất có điều kiện sinh ra các từ ngữ cảnh với một từ đích trung tâm cho trước [3].

Ví dụ, giả sử chúng tôi có chuỗi văn bản gồm các từ là “*the*”, “*man*”, “*loves*”, “*his*” và “*son*”. Chúng tôi sử dụng “*loves*” làm từ đích trung tâm và đặt kích thước của sổ ngữ cảnh (window size) là 2. Thì xác suất có điều kiện sinh ra các

từ ngữ cảnh (“*the*”, “*man*”, “*his*” và “*son*”) dựa trên từ đích trung tâm “*loves*” được mô tả như sau:

$$P(\text{“the”}|\text{“loves”}).P(\text{“man”}|\text{“loves”}).P(\text{“his”}|\text{“loves”}).P(\text{“son”}|\text{“loves”}).$$

Hình 3.5 dưới đây, mô tả kiến trúc của mô hình skip-gram.



Hình 3.5: Kiến trúc của mô hình skip-gram

Trong mô hình skip-gram, mỗi từ được biểu diễn bằng hai vector  $d$ -chiều để tính xác suất có điều kiện. Giả sử chúng ta có chỉ số của một từ trong từ điển là  $i$ , vector của từ được biểu diễn là  $\mathbf{v}_i \in \mathbb{R}^d$  khi từ này là từ đích trung tâm và là  $\mathbf{u}_i \in \mathbb{R}^d$  khi từ này là một từ ngữ cảnh. Gọi  $c$  và  $o$  lần lượt là chỉ số của từ đích trung tâm  $w_c$  và từ ngữ cảnh  $w_o$  trong từ điển. Chúng tôi thu được xác suất có điều kiện sinh ra từ ngữ cảnh cho một từ đích trung tâm cho trước bằng hàm softmax trên tích vô hướng của vector được định nghĩa như sau [3]:

$$P(w_o | w_c) = \frac{\exp(\mathbf{u}_o^\top \mathbf{v}_c)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c)} \quad (3.1)$$

Trong đó, tập chỉ số trong bộ từ vựng là  $\mathcal{V} = \{0, 1, \dots, |\mathcal{V}| - 1\}$ . Giả sử trong một chuỗi văn bản có độ dài  $T$ , từ tại bước thời gian  $t$  được ký hiệu là  $w^{(t)}$ . Giả sử rằng các từ ngữ cảnh được sinh độc lập với từ trung tâm cho trước. Khi kích thước của sổ ngữ cảnh là  $m$ , hàm likelihood của mô hình skip-gram là xác suất kết hợp sinh ra tất cả các từ ngữ cảnh với bất kỳ từ trung tâm cho trước nào. Hàm likelihood này được định nghĩa như sau:

$$\prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w^{(t+j)} | w^{(t)}) \quad (3.2)$$

### 3.2.2 Huấn luyện mô hình

Các tham số trong mô hình skip-gram là vector từ đích trung tâm và vector từ ngữ cảnh cho từng từ riêng lẻ. Trong quá trình huấn luyện, chúng tôi sẽ học các tham số mô hình bằng cách cực đại hóa hàm likelihood, còn gọi là ước lượng hợp lý cực đại. Việc này sẽ tương tự với việc giảm thiểu hàm mất mát (loss function) sau đây:

$$-\sum_{t=1}^T \sum_{-m \leq j \leq m, j \neq 0} \log P(w^{(t+j)} | w^{(t)}) \quad (3.3)$$

Chúng tôi dùng SGD (stochastic gradient descent) [13], thì trong mỗi vòng lặp, chúng tôi chọn ra một chuỗi con nhỏ hơn bằng việc lấy mẫu ngẫu nhiên để tính toán hàm mất mát (loss function) cho chuỗi con đó, rồi sau đó tính gradient để cập nhật các tham số mô hình. Điểm then chốt của việc tính toán gradient là tính gradient của logarit xác suất có điều kiện cho vector từ trung tâm và vector từ ngữ cảnh. Đầu tiên, theo định nghĩa chúng tôi có:

$$\log P(w_o | w_c) = \mathbf{u}_o^\top \mathbf{v}_c - \log \left( \sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \mathbf{v}_c) \right) \quad (3.4)$$

Thông qua phép tính đạo hàm, chúng tôi nhận được giá trị gradient  $v_c$  như sau:

$$\frac{\partial \log P(w_o | w_c)}{\partial v_c} = u_o - \sum_{j \in \mathcal{V}} P(w_j | w_c) u_j \quad (3.5)$$

Phép tính cho ra xác suất có điều kiện cho tất cả các từ có trong kho từ vựng với từ đích trung tâm  $w_c$  cho trước. Sau đó, chúng tôi lại sử dụng phương pháp này để tìm gradient cho các vector từ khác.

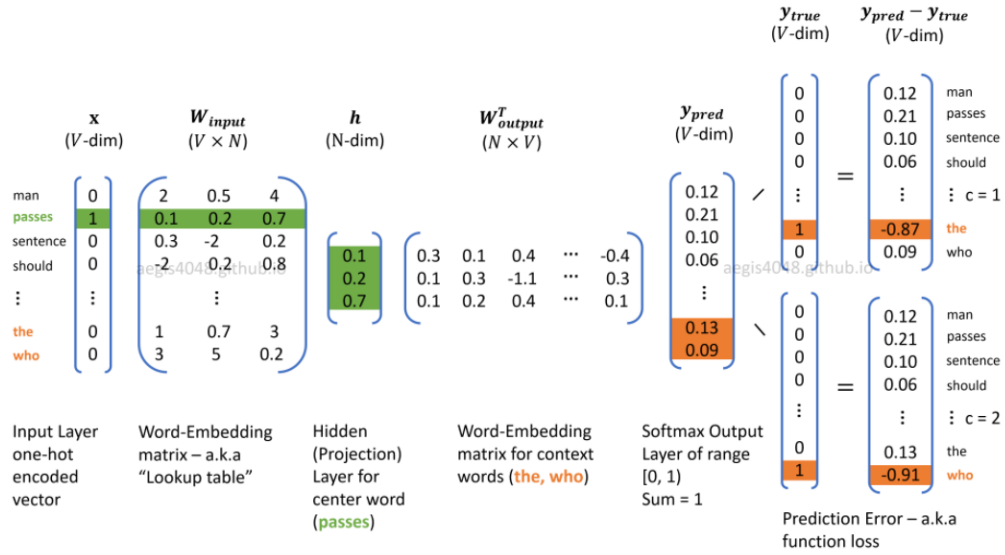
Sau khi huấn luyện xong, với từ bất kỳ có chỉ số là  $i$  trong từ kho từ vựng, chúng tôi sẽ nhận được tập hai vector từ  $\mathbf{v}_i$  và  $\mathbf{u}_i$ . Trong các ứng dụng xử lý ngôn ngữ tự nhiên, vector từ đích trung tâm  $\mathbf{v}_i$  trong mô hình skip-gram sẽ được sử dụng để làm vector biểu diễn một từ.

### 3.2.3 Cách hoạt động của mô hình

Để dễ hình dung, chúng ta hãy xem ví dụ cụ thể về cấu trúc neural network của mô hình skip-gram [14]. Do hoạt động dựa trên cấu trúc neural network nên mô hình chia làm hai giai đoạn như sau:

- Forward propagation: Truyền xuôi.
- Backward propagation: Lan truyền ngược lại.

Hình 3.6, mô tả cấu trúc neural network của mô hình skip-gram.



Hình 3.6: Cấu trúc neural network của mô hình skip-gram

### 3.2.3.1 Forward propagation

#### Input layer

Input layer của chúng tôi là một từ đích trung tâm được biểu diễn dưới dạng one-hot. Chúng tôi sẽ sử dụng kích thước của sổ ngữ cảnh (window size) là 1 và giả sử từ “*pass*” là từ đích trung tâm tạo nên các từ ngữ cảnh “*who*” và “*the*”.

#### Hidden layer

Ở hidden layer, vector từ trung tâm ( $x$ ) sẽ được nhân với ma trận  $W_{input}$  ( $V \times N$ ) khởi tạo ngẫu nhiên với  $V$  là kích thước tập từ vựng và  $N$  là số chiều mà chúng tôi cần nhúng. Giá trị hidden layer sẽ được tính như sau:

$$h = W_{input}^T \cdot x \in R^N \quad (3.6)$$



## Output layer

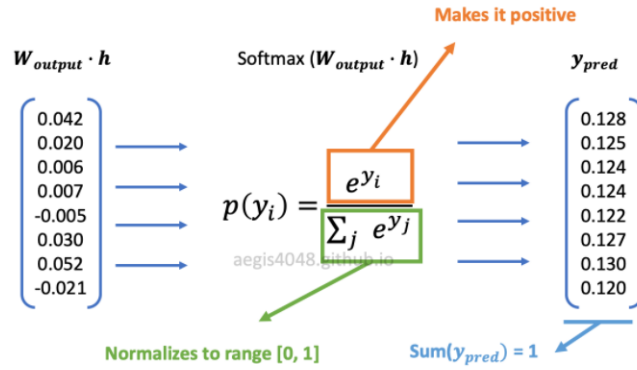
Output layer là một phân phối xác suất của tất cả các từ trong kho từ vựng với một từ đích trung tâm cho trước. Nó được tính bằng cách sử dụng hàm softmax như đã đề cập ở trên. Sau đây là công thức tổng quát trên toàn bộ tập từ vựng  $V$ :

$$\begin{bmatrix} p(w_1 | w_{\text{center}}) \\ p(w_2 | w_{\text{center}}) \\ p(w_3 | w_{\text{center}}) \\ \vdots \\ p(w_V | w_{\text{center}}) \end{bmatrix} = \frac{\exp(W_{\text{output}} \cdot h)}{\sum_{i=1}^V \exp(W_{\text{output}_{(i)}} \cdot h)} \in \mathbb{R}^V \quad (3.7)$$

Trong đó:

- $\mathcal{V} = \{w_1, \dots, w_V\}$  là tất cả các từ trong kho từ vựng.
- $w_{\text{center}}$  là đích trung tâm.
- $W_{\text{output}_{(i)}}$  là vector hàng thứ  $i$  có kích thước  $1 \times N$  từ ma trận  $W_{\text{output}}(N \times V)$ .
- $W_{\text{output}_{\text{context}}}$  là một vector từ ngữ cảnh cũng có kích thước  $1 \times N$  từ ma trận  $W_{\text{output}}(N \times V)$ .
- $h$  là hidden layer có kích thước  $N \times 1$ .

Output layer ở đây sẽ mang giá trị của một phân phối xác suất có điều kiện trong khoảng  $[0, 1]$ . Hình 3.7 sau đây, mô tả việc sử dụng hàm softmax để tính các giá trị xác suất.



Hình 3.7: Sử dụng hàm softmax để tính các giá trị xác suất

Tổng giá trị các xác suất có điều kiện trên toàn bộ tập từ vựng  $V$  sẽ là 1.

### 3.2.3.2 Backward propagation

#### Sum prediction error

Mô hình skip-gram tối ưu hóa ma trận trọng số để giảm sai số dự đoán (prediction error). Sai số dự đoán là sự chênh lệch giữa các giá trị phân phối xác suất của các từ được tính từ hàm softmax ở output layer ( $y_{pred}$ ) và các giá trị phân phối xác suất thật ( $y_{true}$ ) của từ ngữ cảnh thứ  $c$ . Giống với input layer,  $y_{true}$  là vector được biểu diễn one-hot.

$$PredictionError = y_{pred} - y_{true} \quad (3.8)$$

Hình 3.8, ví dụ về tính toán sai số dự đoán trên từng từ ngữ cảnh.

[illegible]

Hình 3.8: Tính toán sai số dự đoán trên từng từ ngữ cảnh

Sau khi tính được sai số dự đoán (prediction error) trên từng từ ngữ cảnh, chúng tôi tiếp tục tính tổng sai số dự đoán (sum prediction error) trên tất cả các từ ngữ cảnh. Công thức này được định nghĩa như sau:

$$\sum_{c=1}^C e_c \quad (3.9)$$

Trong đó:  $e_c$  là sai số dự đoán (prediction error) trên một từ ngữ cảnh.

- C: kích thước của số ngữ cảnh.
- $e_c$  là sai số dự đoán (prediction error) trên một từ ngữ cảnh.

Hình 3.9, ví dụ về tính toán tổng sai số dự đoán trên tất cả các từ ngữ cảnh.

$$\sum_{c=1}^C e_c = \begin{matrix} c=1 \\ \begin{bmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ -0.87 \\ 0.09 \end{bmatrix} \end{matrix} + \begin{matrix} c=2 \\ \begin{bmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ -0.91 \end{bmatrix} \end{matrix} = \begin{matrix} \begin{bmatrix} 0.24 \\ 0.42 \\ 0.20 \\ 0.12 \\ \vdots \\ -0.74 \\ -0.82 \end{bmatrix} \end{matrix} \begin{matrix} \text{man} \\ \text{passes} \\ \text{sentence} \\ \text{should} \\ \vdots \\ \text{the} \\ \text{who} \end{matrix}$$

Hình 3.9: Tính toán tổng sai số dự đoán trên tất cả các từ ngữ cảnh

Quá trình huấn luyện mô hình neural network của mô hình skip-gram sau mỗi lần huấn luyện thì nó sẽ cập nhật lại giá trị tổng sai số dự đoán (sum prediction error). Các giá trị này sẽ dần dần hội tụ 0 (giá trị tối ưu) thì khi đó quá trình huấn luyện mô hình kết thúc.

Hình 3.10, cho thấy các giá trị trong tổng sai số dự đoán (sum prediction error) dần hội tụ về 0.

$$\sum_{c=1}^C e_c = \begin{matrix} \text{iter} = 1 \\ \begin{bmatrix} 0.24 \\ 0.42 \\ 0.20 \\ 0.12 \\ \vdots \\ -0.74 \\ -0.82 \end{bmatrix} \end{matrix} \xrightarrow{\text{Update } \theta} \begin{matrix} \text{iter} = 2 \\ \begin{bmatrix} 0.18 \\ 0.30 \\ 0.12 \\ 0.09 \\ \vdots \\ -0.32 \\ -0.46 \end{bmatrix} \end{matrix} \xrightarrow{\text{Update } \theta} \begin{matrix} \text{iter} = 3 \\ \begin{bmatrix} 0.08 \\ 0.12 \\ 0.04 \\ 0.02 \\ \vdots \\ -0.12 \\ -0.13 \end{bmatrix} \end{matrix} \xrightarrow{\text{Update } \theta} \begin{matrix} \text{iter} = 4 \\ \begin{bmatrix} 0.01 \\ 0.02 \\ 0.01 \\ 0.00 \\ \vdots \\ -0.01 \\ -0.02 \end{bmatrix} \end{matrix} \begin{matrix} \text{man} \\ \text{passes} \\ \text{sentence} \\ \text{should} \\ \vdots \\ \text{the} \\ \text{who} \end{matrix}$$

Hình 3.10: Tổng sai số dự đoán qua các lần cập nhật để hội tụ về giá trị 0

### Tính toán $\nabla W_{input}$

Bằng cách sử dụng SGD (stochastic gradient descent), chúng tôi tính được gradients của ma trận trọng số đầu vào ( $\frac{\partial J}{\partial W_{input}}$ ) bằng công thức được định

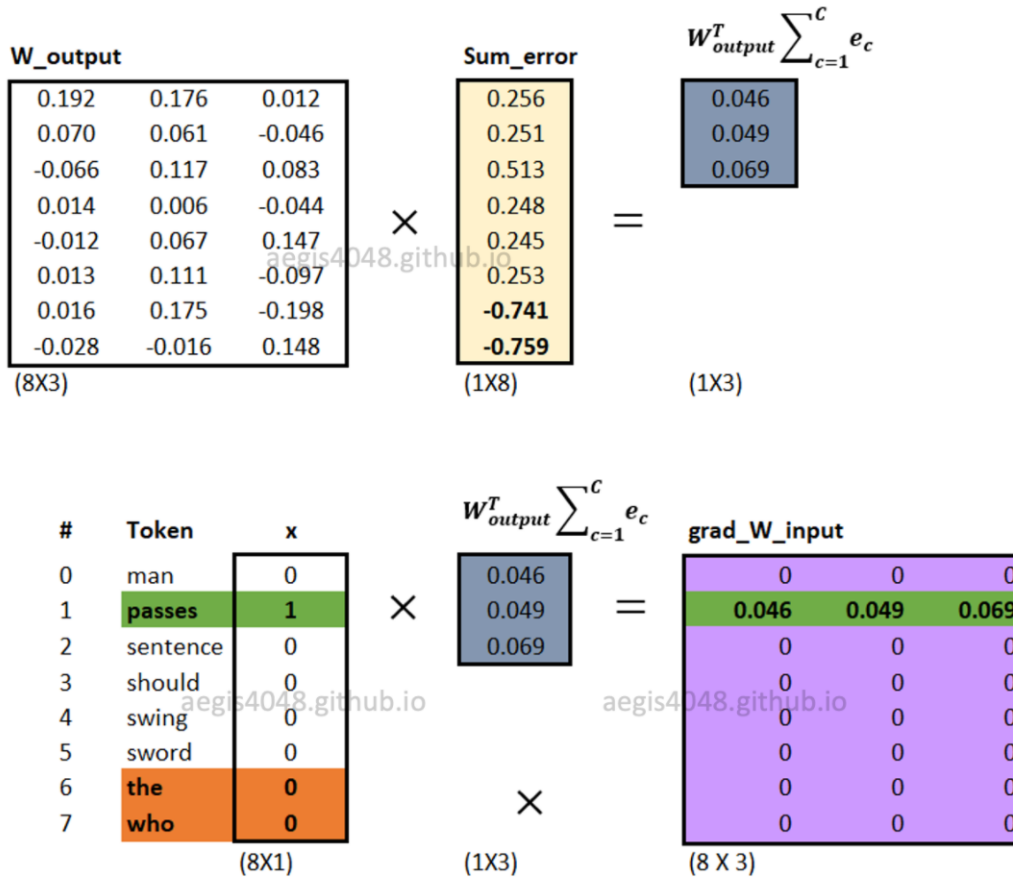
nghĩa như sau:

$$\frac{\partial J}{\partial W_{input}} = x \cdot \left( W_{output}^T \sum_{c=1}^C e_c \right) \quad (3.10)$$

Trong đó:

- $x$ : Vector đầu vào được biểu diễn one-hot.
- $W_{output}^T$ : Ma trận trọng số đầu ra.
- $\sum_{c=1}^C e_c$ : Tổng sai số dự đoán (sum prediction error).

Hình 3.11 là một ví dụ về tính toán gradients của ma trận trọng số đầu vào  $\nabla W_{input}$ .



Hình 3.11: Tính toán gradient của ma trận trọng số đầu vào  $\nabla W_{input}$

### Tính toán $\nabla W_{output}$

Tiếp tục, bằng cách sử dụng SGD (stochastic gradient descent), chúng tôi tính được gradients của ma trận trọng số đầu ra ( $\frac{\partial J}{\partial W_{output}}$ ) bằng công thức được định nghĩa như sau:

$$\frac{\partial J}{\partial W_{output}} = h \cdot \sum_{c=1}^C e_c \quad (3.11)$$

Trong đó:

- h: Giá trị của hidden layer đã tính trước đó ( $W_{input}^T x$ ) .

- $\sum_{c=1}^C e_c$ : Tổng sai số dự đoán (sum prediction error).

Hình 3.12, là một ví dụ về tính toán gradients của ma trận trọng số đầu ra  $\nabla W_{output}$ .

h	Sum_error	grad_W_output
0.068	0.256	0.017 0.044 -0.028
0.17	0.251	0.017 0.043 -0.027
-0.109	0.513	0.035 0.087 -0.056
	0.248	0.017 0.042 -0.027
	0.245	0.017 0.042 -0.027
	0.253	0.017 0.043 -0.028
	-0.741	-0.050 -0.126 0.081
	-0.759	-0.052 -0.129 0.083
(1X3)	(1X8)	(8 X 3)

Hình 3.12: Tính toán gradient của ma trận trọng số đầu ra  $\nabla W_{output}$

### Cập nhật hai ma trận $W_{input}$ và $W_{output}$

Sau khi chúng tôi tính được gradients cho hai ma trận  $\nabla W_{input}$  và  $\nabla W_{output}$ . Chúng tôi tiến hành cập nhật hai ma trận  $W_{input}$  và  $W_{output}$  lần lượt bằng hai công thức được định nghĩa như sau:

$$W_{input}^{(new)} = W_{input}^{(old)} - \eta \cdot x \cdot \left( W_{output}^T \sum_{c=1}^C e_c \right) \quad (3.12)$$

$$W_{output}^{(new)} = W_{output}^{(old)} - \eta \cdot h \cdot \sum_{c=1}^C e_c \quad (3.13)$$

Trong đó:

- $\eta$ : Hệ số học (learning rate) [15].

- $\nabla W_{input}$ : Ma trận trọng số đầu vào.
- $\nabla W_{output}$ : Ma trận trọng số đầu ra.

Hệ số học (learning rate) là một siêu tham số do người dùng xác định. Hệ số này nếu quá cao thì mô hình nhanh đến đích nhưng khó có thể hội tụ được, ngược lại nếu hệ số này quá thấp thì mô hình sẽ hội tụ chậm ảnh hưởng đến hiệu suất của thuật toán. Ở đây chúng tôi chọn hệ số học (learning rate) là  $\eta=0.05$ .

Lần lượt cập nhật hai ma trận  $W_{input}$  và  $W_{output}$  như hình 3.13 và hình 3.14.

The diagram illustrates the update of the input weight matrix  $W_{input}$ . It shows the old matrix  $W_{input} (old)$  (8x3), the learning rate  $\eta = 0.05$ , the gradient matrix  $grad\_W_{input}$  (8x3), and the new matrix  $W_{input} (new)$  (8x3). The update is performed as  $W_{input} (new) = W_{input} (old) - \eta \times grad\_W_{input}$ .

$W_{input} (old)$			Learning R.	$grad\_W_{input}$			=	$W_{input} (new)$		
-0.078	0.018	0.033	0.05	0.000	0.000	0.000	=	-0.078	0.018	0.033
0.068	0.170	-0.109		0.046	0.049	0.069		0.066	0.168	-0.112
-0.158	-0.081	-0.151		0.000	0.000	0.000		-0.158	-0.081	-0.151
0.150	0.064	0.145		0.000	0.000	0.000		0.150	0.064	0.145
-0.097	-0.055	0.188		0.000	0.000	0.000		-0.097	-0.055	0.188
0.036	0.071	0.059		0.000	0.000	0.000		0.036	0.071	0.059
0.168	-0.060	-0.058		0.000	0.000	0.000		0.168	-0.060	-0.058
0.098	0.015	0.096		0.000	0.000	0.000		0.098	0.015	0.096

Hình 3.13: Cập nhật ma trận  $W_{input}$

The diagram illustrates the update of the output weight matrix  $W_{output}$ . It shows the old matrix  $W_{output} (old)$  (8x3), the learning rate  $\eta = 0.05$ , the gradient matrix  $grad\_W_{output}$  (8x3), and the new matrix  $W_{output} (new)$  (8x3). The update is performed as  $W_{output} (new) = W_{output} (old) - \eta \times grad\_W_{output}$ .

$W_{output} (old)$			Learning R.	$grad\_W_{output}$			=	$W_{output} (new)$		
0.192	0.176	0.012	0.05	0.017	0.044	-0.028	=	0.191	0.174	0.013
0.070	0.061	-0.046		0.017	0.043	-0.027		0.069	0.059	-0.045
-0.066	0.117	0.083		0.035	0.087	-0.056		-0.068	0.113	0.086
0.014	0.006	-0.044		0.017	0.042	-0.027		0.013	0.004	-0.043
-0.012	0.067	0.147		0.017	0.042	-0.027		-0.013	0.065	0.148
0.013	0.111	-0.097		0.017	0.043	-0.028		0.012	0.109	-0.096
0.016	0.175	-0.198		-0.050	-0.126	0.081		0.019	0.181	-0.202
-0.028	-0.016	0.148		-0.052	-0.129	0.083		-0.025	-0.010	0.144

Hình 3.14: Cập nhật ma trận  $W_{output}$

Quá trình này sẽ thực hiện lặp đi lặp lại cho đến khi tổng sai số dự đoán (sum prediction error) dần hội tụ về 0 (giá trị tối ưu), khi đó quá trình huấn



luyện mô hình sẽ kết thúc. Cuối cùng, chúng tôi sử dụng ma trận trọng số  $W_{input}$  làm ma trận biểu diễn vector từ dùng trong các ứng dụng xử lý ngôn ngữ tự nhiên.

## 3.3 Mô hình continuous bag of words (CBOW)

### 3.3.1 Khái niệm

Mô hình continuous bag of words (CBOW) tương tự như mô hình skip-gram. Khác biệt lớn nhất là mô hình CBOW giả định rằng từ đích trung tâm được tạo ra dựa trên các từ ngữ cảnh phía trước và sau nó trong một chuỗi văn bản. Mô hình này quan tâm đến xác suất có điều kiện tạo ra từ đích trung tâm dựa trên các từ ngữ cảnh cho trước [3].

Ví dụ, giả sử chúng tôi có chuỗi văn bản gồm các từ là “*the*”, “*man*”, “*loves*”, “*his*” và “*son*” được đề cập như ở trên. Chúng tôi sẽ sử dụng từ “*loves*” làm từ đích trung tâm và đặt kích thước cửa sổ ngữ cảnh (window size) là 2. Khi đó, xác suất có điều kiện sinh ra từ đích trung tâm “*loves*” dựa trên các từ ngữ cảnh (“*the*”, “*man*”, “*his*” và “*son*”) được mô tả như sau:

$$P(\text{"loves"} | \text{"the"}, \text{"man"}, \text{"his"}, \text{"son"}).$$

Do có nhiều từ ngữ cảnh trong mô hình CBOW, chúng tôi sẽ lấy trung bình các vector từ ngữ cảnh và sau đó sử dụng phương pháp tương tự như trong mô hình skip-gram để tính xác suất có điều kiện. Giả sử chúng tôi có  $\mathbf{v}_i \in \mathbb{R}^d$  và  $\mathbf{u}_i \in \mathbb{R}^d$  là vector từ ngữ cảnh và vector từ đích trung tâm của từ có chỉ số  $i$  trong từ điển (lưu ý rằng các ký hiệu này ngược với các ký hiệu trong mô hình skip-gram). Gọi  $c$  là chỉ số của từ đích trung tâm  $w_c$ , và  $o_1, \dots, o_{2m}$  là chỉ số các từ ngữ cảnh  $w_{o_1}, \dots, w_{o_{2m}}$  trong từ điển.

Từ đó, xác suất có điều kiện sinh ra từ đích trung tâm dựa vào các từ ngữ cảnh cho trước bằng softmax trên tích vô hướng của vector được định nghĩa như sau:

$$P(w_c | w_{o_1}, \dots, w_{o_{2m}}) = \frac{\exp\left(\frac{1}{2m} \mathbf{u}_c^\top (\mathbf{v}_{o_1} + \dots, +\mathbf{v}_{o_{2m}})\right)}{\sum_{i \in \mathcal{V}} \exp\left(\frac{1}{2m} \mathbf{u}_i^\top (\mathbf{v}_{o_1} + \dots, +\mathbf{v}_{o_{2m}})\right)} \quad (3.14)$$

Để dễ hiểu chúng tôi sẽ ký hiệu  $\mathcal{W}_o = \{w_{o_1}, \dots, w_{o_{2m}}\}$  và  $\bar{\mathbf{v}}_o = (\mathbf{v}_{o_1} + \dots, +\mathbf{v}_{o_{2m}}) / (2m)$ .

Phương trình sẽ được đơn giản hóa thành:

$$P(w_c | \mathcal{W}_o) = \frac{\exp(\mathbf{u}_c^\top \bar{\mathbf{v}}_o)}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)} \quad (3.15)$$

Cho một chuỗi văn bản có độ dài  $T$ , chúng tôi giả định rằng từ xuất hiện tại bước thời gian  $t$  là  $w^{(t)}$ , và kích thước của cửa sổ ngữ cảnh là  $m$ . Hàm likelihood của mô hình CBOW là xác suất sinh ra bất kỳ từ đích trung tâm nào dựa vào các từ ngữ cảnh xung quanh. Hàm này được định nghĩa như sau:

$$\prod_{t=1}^T P(w^{(t)} | w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)}) \quad (3.16)$$

### 3.3.2 Huấn luyện mô hình

Quá trình huấn luyện mô hình CBOW gần giống với quá trình huấn luyện mô hình skip-gram. Ước lượng hợp lý cực đại của mô hình CBOW tương đương với việc cực tiểu hóa hàm mất mát (loss function) như sau:

$$- \sum_{t=1}^T \log P(w^{(t)} | w^{(t-m)}, \dots, w^{(t-1)}, w^{(t+1)}, \dots, w^{(t+m)}) \quad (3.17)$$

Chúng tôi có logarit xác suất có điều kiện của mô hình CBOW như sau:

$$\log P(w_c | \mathcal{W}_o) = \mathbf{u}_c^\top \bar{\mathbf{v}}_o - \log \left( \sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o) \right) \quad (3.18)$$

Thông qua phép đạo hàm, chúng tôi có thể tính log của xác suất có điều kiện của gradient của bất kỳ vector từ ngữ cảnh nào  $\mathbf{v}_{o_i} (i = 1, \dots, 2m)$  từ công thức trên như sau:

$$\frac{\partial \log P(w_c | \mathcal{W}_o)}{\partial \mathbf{v}_{o_i}} = \frac{1}{2m} \left( \mathbf{u}_c - \sum_{j \in \mathcal{V}} \frac{\exp(\mathbf{u}_j^\top \bar{\mathbf{v}}_o) \mathbf{u}_j}{\sum_{i \in \mathcal{V}} \exp(\mathbf{u}_i^\top \bar{\mathbf{v}}_o)} \right) = \frac{1}{2m} \left( \mathbf{u}_c - \sum_{j \in \mathcal{V}} P(w_j | \mathcal{W}_o) \mathbf{u}_j \right) \quad (3.19)$$

Sau đó, chúng tôi cũng sử dụng phương pháp SGD (stochastic gradient descent) [13] để tính gradient cho các vector của từ khác. Không giống như mô hình skip-gram, trong mô hình CBOW chúng tôi sử dụng vector từ ngữ cảnh làm vector biểu diễn một từ.

### 3.4 Đánh giá mô hình

Về cơ bản thì hai mô hình skip-gram và mô hình continuous bag of words (CBOW) có cách hoạt động và cách thức huấn luyện mô hình gần giống nhau. Tùy vào mục đích sử dụng và kích thước tập dữ liệu để chọn mô hình tối ưu nhất.

Dưới đây là một số khác biệt cơ bản của hai mô hình:

- Việc huấn luyện mô hình CBOW là tương đối nhanh hơn việc huấn luyện mô hình Skip-gram. Vì mô hình CBOW chỉ tính một hàm softmax cho một từ trung tâm còn mô hình skip-gram thì tính cho tất cả các từ ngữ cảnh.
- Mô hình CBOW biểu diễn tốt hơn cho các từ thường xuyên xuất hiện trong kho ngữ liệu. Bởi vì nếu một từ thường xuyên xuất hiện thì nó sẽ có nhiều dữ liệu để đào tạo hơn.

- Mô hình skip-gram hoạt động với tập dữ liệu nhỏ. Biểu diễn tốt hơn các từ hiếm, ít khi xuất hiện trong kho ngữ liệu hơn so với mô hình CBOW.

## Chương 4

# MÔ HÌNH GLOBAL VECTOR

### 4.1 Khái niệm Global Vector

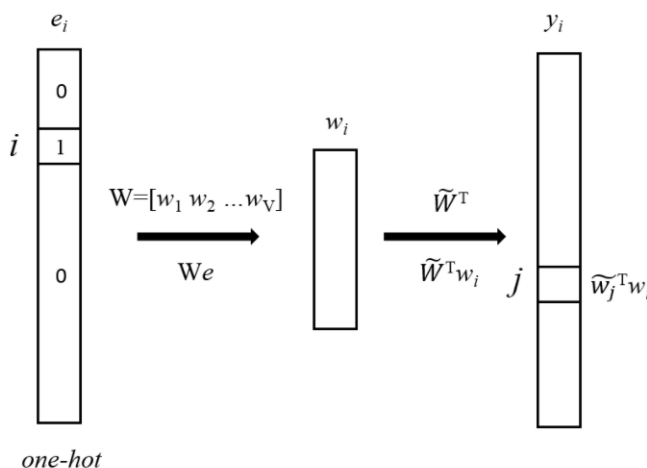
Global Vector (glove) là một thuật toán học tập không giám sát để biểu diễn phân tán các vector do đại học Stanford công bố vào năm 2014. Nó dựa trên kỹ thuật phân tích ma trận thành nhân tử (matrix factorization) dựa trên ma trận từ - ngữ cảnh (word - context) [16].

Việc đào tạo được thực hiện dựa trên số liệu thống kê đồng xuất hiện từ-từ được tổng hợp từ một text corpus (các dữ liệu văn bản, ngôn ngữ đã được số hoá có thể được xem như là một kho ngữ liệu, được sử dụng như đầu vào của các thuật toán trong NLP).

Mô hình glove tính toán xác suất từ dựa trên toàn bộ tập dữ liệu còn word2vec tính toán xác suất dựa trên các ngữ cảnh đơn lẻ. Đầu vào của mô

hình này được biểu diễn dưới dạng one-hot. Ma trận nhúng từ đóng vai trò là ma trận trọng số của mô hình. Do đó đầu ra của mô hình này là một vector sản phẩm bên trong của vector từ.

Hình 4.1, Mô tả tổng quát mô hình kiến trúc của mô hình glove.



Hình 4.1: Mô hình kiến trúc của glove

## 4.2 Ý tưởng triển khai mô hình

Ý tưởng chính của mô hình glove là từ việc quan sát thấy rằng những từ có mối quan hệ về ý nghĩa, ngữ nghĩa và ngữ cảnh sẽ có xác suất xuất hiện đồng thời cao hơn.

Đầu tiên, chúng tôi thiết lập một số ký hiệu (notation). Hãy để ma trận đồng xuất hiện từ - từ (word-word co-occurrence) được biểu thị bằng  $X$ , có mục nhập  $X_{ij}$  lập bảng số lần từ  $j$  xảy ra trong ngữ cảnh của từ  $i$ . Cho  $X_i = \sum_k X_{ik}$  là số lần xuất hiện của từ  $i$  trong ngữ cảnh của toàn bộ các từ còn lại ngoại trừ  $i$ . Cuối cùng, cho  $P_{ij} = P(j | i) = X_{ij}/X_i$  là xác suất để từ  $j$  xuất hiện trong ngữ

cảnh của từ  $i$ .

Hình 4.2, mô tả xác suất đồng xuất hiện (co-occurrence probabilities) đối với các từ mục tiêu băng và hơi với các từ ngữ cảnh được chọn từ kho ngữ liệu 6 tỷ từ.

.	k = rắn	k = khí	k = nước	k = áo
P (k   băng)	$1.9 \times 10^{-4}$	$6.6 \times 10^{-5}$	$3.0 \times 10^{-3}$	$1.7 \times 10^{-5}$
P (k   hơi)	$2.2 \times 10^{-5}$	$7.8 \times 10^{-4}$	$2.2 \times 10^{-3}$	$1.8 \times 10^{-5}$
P (k   băng) / P (k   hơi)	8.9	$8.5 \times 10^{-2}$	1.36	0.96

Hình 4.2: Xác suất đồng xuất hiện

Từ Bảng 4.1, chúng tôi nhận thấy mối quan hệ của những từ này có thể được kiểm tra bằng cách nghiên cứu tỷ lệ xác suất đồng xuất hiện của chúng với các từ thăm dò  $k$  khác nhau. Cho các từ  $k$  liên quan đến *băng* nhưng không liên quan đến *hơi* như  $k = \text{rắn}$ , giả sử  $i = \text{băng}$  và  $j = \text{hơi}$ , chúng tôi kỳ vọng tỷ lệ  $P_{ik}/P_{jk}$  sẽ lớn. Tương tự, đối với các từ  $k$  liên quan đến *hơi* nhưng không liên quan đến *băng* như  $k = \text{khí}$ , tỷ lệ được kỳ vọng sẽ nhỏ. Đối với những từ  $k$  như *nước* hoặc *áo*, cùng quan đến cả *băng* và *hơi*, hoặc cùng không liên quan đến cả hai, tỷ lệ sẽ gần với giá trị 1.

Thực tế, xác suất xuất hiện gần nhau của cặp từ [*“băng”, “rắn”*] là cao hơn 2.8 lần khi so với xác suất của cặp từ [*“băng”, “khí”*]. Trong một trường hợp khác xác suất của cặp từ [*“băng”, “áo”*] là rất thấp đúng như kỳ vọng.

Điều này phù hợp với các quan điểm như “băng ở thể rắn”, “băng trái ngược với khí”, “băng không liên quan tới áo quần”.

Vì vậy, nếu chúng tôi có thể tìm ra cách kết hợp tỷ lệ  $\frac{P_{ik}}{P_{jk}}$  để tính toán vector từ, chúng tôi sẽ đạt được mục tiêu thống kê toàn cục (global statistic) khi sử dụng phương pháp học vector từ (wordvector learning). Giả sử rằng có một hàm  $F$  nhận các vector từ của  $i, j$  và  $k$  để xuất ra tỷ lệ mà chúng tôi quan tâm [17]:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \quad (4.1)$$

Trong đó:

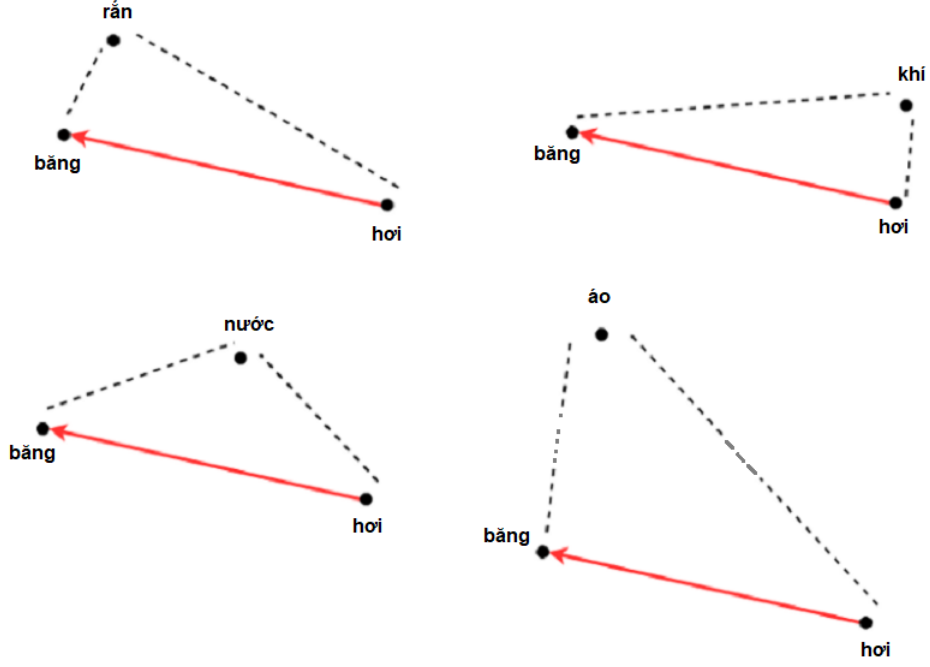
- $w \in \mathbb{R}^d$ : là vector từ.
- $\tilde{w} \in \mathbb{R}^d$ : là các vector từ có ngữ cảnh riêng biệt.

Ở công thức (4.1), vế bên phải được trích xuất từ kho ngữ liệu. Vế bên trái là sự tham gia của hai lớp nhúng (embedding layers) là  $w$  và  $\tilde{w}$ , về bản chất hai lớp nhúng này có sự tương đồng và chỉ khác nhau bởi các lần khởi tạo ngẫu nhiên khác nhau. Việc có hai lớp nhúng này sẽ giúp mô hình giảm đi hiện tượng *Overfitting*.

Đầu tiên, chúng tôi muốn  $F$  mã hóa tỷ lệ  $P_{ik}/P_{jk}$  trong không gian vector từ (word vector space). Vì không gian vector vốn là cấu trúc tuyến tính (linear structures), cách tự nhiên nhất để làm điều này là sử dụng phương pháp chênh lệch vector (vector difference) - là kết quả của việc trừ một vector từ một vector khác (Hình 4.3). Từ đó, chúng tôi biến đổi công thức (4.1) thành:

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \quad (4.2)$$





Hình 4.3: Mô phỏng khoảng cách vector ứng với  $w_i - w_j$

Tiếp theo, lưu ý rằng các đối số của hàm  $F$  trong công thức (4.2) là các vector trong khi vế bên phải là một đại lượng vô hướng.

Để giải quyết vấn đề này, trước tiên chúng tôi có thể lấy tích vô hướng (dot product) của đối số.

$$F \left( (w_i - w_j)^T \tilde{w}_k \right) = \frac{P_{ik}}{P_{jk}} \quad (4.3)$$

Tiếp theo, lưu ý rằng đối với ma trận đồng xuất hiện từ - từ, sự phân biệt giữa một từ và một từ theo ngữ cảnh là tùy ý và rằng chúng tôi có thể tự do trao đổi hai vai trò của chúng. Làm như vậy sẽ đảm bảo tính nhất quán, chúng tôi không chỉ phải trao đổi  $w \leftrightarrow \tilde{w}$ , mà còn  $X \leftrightarrow X^T$  [17].

$$F \left( (w_i - w_j)^T \tilde{w}_k \right) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \quad (4.4)$$

Mà, công thức (4.3) được chúng tôi giải quyết như sau:

$$F\left(w_i^T \tilde{w}_k\right)=P_{ik}=\frac{X_{ik}}{X_i} \quad (4.5)$$

Giải pháp cho công thức (4.4), chúng tôi sẽ tiến hành log hóa cho hai vế:

$$w_i^T \tilde{w}_k=\log \left(P_{ik}\right)=\log \left(X_{ik}\right)-\log \left(X_i\right) \quad (4.6)$$

Tiếp theo, chúng tôi lưu ý rằng ở công thức (4.6) sẽ thể hiện sự trao đổi đối xứng (symmetry) nếu không có  $\log \left(X_i\right)$  về bên phải. Tuy nhiên, số hạng này độc lập với  $k$  vì vậy nó có thể được gộp vào giá trị *bias*  $b_i$  cho  $w_i$ . Cuối cùng, chúng tôi thêm một *bias* bổ sung  $\tilde{b}_k$  cho  $\tilde{w}_k$  nhằm khôi phục sự đối xứng [17].

$$w_i^T \tilde{w}_k+b_i+\tilde{b}_k=\log \left(X_{ik}\right) \quad (4.7)$$

Công thức (4.7) được chúng tôi đơn giản hóa mạnh mẽ so với công thức (4.1), nhưng nó không thật sự đúng vì logarit sẽ phân kỳ (diverges) bất cứ khi nào đối số của nó bằng 0. Một giải pháp cho vấn đề này, chúng tôi sẽ tiến hành cộng thêm một giá trị cho hàm  $\log \left(X_{ik}\right) \rightarrow \log \left(1+X_{ik}\right)$ , nhằm duy trì độ thưa thớt (sparsity) của  $X$  trong khi tránh các phân kỳ (diverges). Một nhược điểm chính của mô hình này là nó cân bằng tất cả các trường hợp đồng xuất hiện, ngay cả những trường hợp hiếm khi xảy ra hoặc không bao giờ. Các đồng xuất hiện hiếm như vậy thường bị nhiễu và mang ít thông tin hơn những loại thường xuyên. Tuy nhiên, thậm chí các mục nhập với giá trị 0 (zero entries) cũng chiếm 75–95% dữ liệu trong  $X$ , tùy thuộc vào kích thước từ vựng và kho ngữ liệu.

Chúng tôi đề xuất một mô hình hồi quy bình phương có trọng số nhỏ nhất để giải quyết những vấn đề này. Công thức (4.7) như một bài toán bình phương

nhỏ nhất và đưa một hàm trọng số  $f(X_{ij})$  vào hàm mất mát (loss function) cho chúng tôi mô hình (4.8)

$$J = \sum_{i,j=1}^V f(X_{ij}) \left( w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij} \right)^2 \quad (4.8)$$

Trong đó:

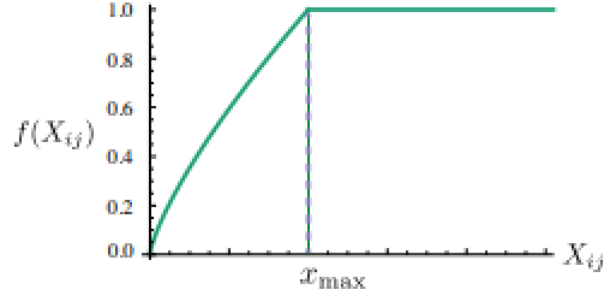
- $V$ : Kích thước tập từ vựng.
- $b_i(|V| \times 1)$  column vector (bias) là hệ số điều chỉnh thể hiện mức độ sai lệch trung bình ứng với cột.
- $b_j(1 \times |V|)$  row vector (bias) là hệ số điều chỉnh thể hiện mức độ sai lệch trung bình ứng với hàng.

Hàm trọng số phải tuân theo các thuộc tính sau:

- $f(0) = 0$ . Nếu xem  $f$  là một hàm liên tục, nó sẽ bị triệt tiêu dưới dạng  $x \rightarrow 0$  đủ nhanh để  $\lim_{x \rightarrow 0} f(x) \log^2 x$  là hữu hạn.
- $f(x)$  không giảm để các đồng xuất hiện hiếm không bị quá tải.
- $f(x)$  phải tương đối nhỏ đối với các giá trị lớn của  $x$ , để các trường hợp đồng xuất hiện thường xuyên không bị quá tải.

Tất nhiên, một số lượng lớn các hàm thỏa mãn các thuộc tính này, nhưng một lớp hàm (4.9) mà chúng tôi nhận thấy hoạt động tốt có thể được tham số hóa là [17].

$$f(X_{ij}) = \begin{cases} \left( \frac{X_{ij}}{x_{max}} \right)^\alpha & \text{if } X_{ij} < x_{max} \\ 1 & \text{otherwise} \end{cases} \quad (4.9)$$



Hình 4.4: Hàm trọng số  $f$  với  $\alpha = 3/4$

Hình 4.4. cho thấy hiệu suất của mô hình phụ thuộc yếu vào điểm cắt, giá trị  $x_{max}$  được tinh chỉnh đến 100 trong tất cả các thí nghiệm. Chúng tôi nhận thấy rằng giá trị  $\alpha = 3/4$  mang lại sự cải thiện khiêm tốn hơn so với phiên bản tuyến tính có giá trị  $\alpha = 1$ .

Qua những bước phân tích trên, chúng tôi có thể kết luận mục đích cuối cùng của việc triển khai mô hình glove là học ra các vector từ ngữ để tích vô hướng của chúng sẽ bằng xác suất xuất hiện đồng thời của chúng trong tập dữ liệu, đồng thời độ lỗi phát sinh trong quá trình huấn luyện sẽ được giải quyết bằng hàm mất mát.

## 4.3 Huấn luyện mô hình

**Bước 1:** Từ vector được biểu diễn dưới dạng one-hot, chúng tôi sẽ xây dựng ma trận đồng xuất hiện  $(X_{ij} |V|x|V|)$ , Hình 4.5. minh họa cụ thể cho một ma trận đồng xuất hiện.

Tập dữ liệu = {"Tôi là học sinh",  
"Tôi là sinh viên",  
"Tôi không phải là giáo viên"}

Kích thước cửa sổ = 2

		Ngữ cảnh							
		Tôi	là	học	sinh	viên	không	phải	giáo
Từ mục tiêu	Tôi	0	2	1	1	0	1	1	0
	là	2	0	1	2	2	1	1	1
	học	1	1	0	1	0	0	0	0
	sinh	1	2	1	0	1	0	0	0
	viên	0	2	0	1	0	0	0	0
	không	1	1	0	0	0	0	1	0
	phải	1	1	0	0	0	1	0	1
	giáo	0	1	0	0	0	0	1	0

Hình 4.5: Minh họa cho ma trận đồng xuất hiện

$X_{ij}$  là số lần mà từ  $j$  sẽ xuất hiện trong ngữ cảnh của  $i$ , hay nói đơn giản hơn là số lần mà  $i$  và  $j$  cùng xuất hiện gần nhau. Mô hình glove chọn ngữ cảnh dựa trên cửa sổ ngữ cảnh nên kết quả thu được là một ma trận vuông đối xứng qua đường chéo chính.

Ví dụ: Số lần từ “là” và “sinh” cùng xuất hiện gần nhau ( $X(\text{là} \mid \text{sinh})$ ) = 2 do từ “là” xuất hiện trong phạm vi kích thước cửa sổ ngữ cảnh bằng 2 ở hai câu “Tôi là học sinh” và “Tôi là sinh viên”.

**Bước 2:** Chuyển thành ma trận xác suất đồng xuất hiện P (Hình 4.6).

Tập dữ liệu = {"Tôi là học sinh",  
"Tôi là sinh viên",  
"Tôi không phải là giáo viên"}.

Kích thước cửa sổ = 2

		Ngữ cảnh							
Từ mục tiêu		Tôi	là	học	sinh	viên	không	phải	giáo
	Tôi	0/6	2/6	1/6	1/6	0/6	1/6	1/6	0/6
	là	2/6	0/6	1/6	2/6	2/6	1/6	1/6	1/6
	học	1/3	1/3	0/3	1/3	0/3	0/3	0/3	0/3
	sinh	1/5	2/5	1/5	0/5	1/5	0/5	0/5	0/5
	viên	0/4	2/4	0/4	1/4	0/4	0/4	0/4	0/4
	không	1/3	1/3	0/3	0/3	0/3	0/3	1/3	0/3
	phải	1/4	1/4	0/4	0/4	0/4	1/4	0/4	1/4
	giáo	0/3	1/3	0/3	0/3	0/3	0/3	1/3	0/3

Hình 4.6: Ma trận xác suất đồng xuất hiện P

Ví dụ:

Số từ xuất hiện trong ngữ cảnh của từ “*tôi*” bằng 6, gồm [“*là*”, “*học*”, “*là*”, “*sinh*”, “*không*”, “*phải*”]. Từ đó, chúng tôi được  $X_i = 6$ .

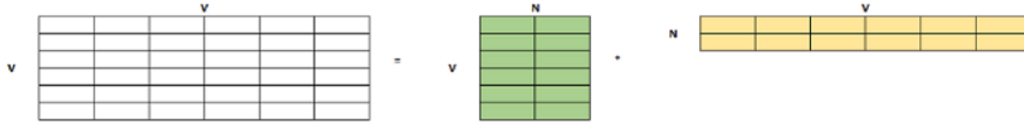
Số lần xuất hiện của từ “*tôi*” trong ngữ cảnh của từ “*là*”:  $X_{ij} = 2$ .

Như vậy, chúng tôi được xác suất xuất hiện của  $P_{ij}$  của từ “*là*” trong ngữ cảnh của từ “*tôi*” là:  $P_{ij} = \frac{X_{ij}}{X_i} = 2/6$ .

Tiếp tục Log hóa  $P_{ij}$  chúng tôi sẽ tính được tích vô hướng của hai vector từ  $i, j$  (“*tôi*”, “*là*”) dựa theo công thức (4.6).

Mục đích của chúng tôi ở bước này nhằm xác định ràng buộc, cụ thể là sai số *bias* khi tính tích vô hướng giữa hai vector từ  $i$  và  $j$ , cách tính được dựa theo công thức (4.7).

**Bước 3:** Từ ma trận đồng xuất hiện [VxV] ban đầu, ta phân rã thành hai ma trận con [VxN] và [NxV] như Hình 4.7.



Hình 4.7: Phân rã ma trận đồng xuất hiện  $[V \times V]$  ban đầu thành hai ma trận con  $[V \times N]$  và  $[N \times V]$

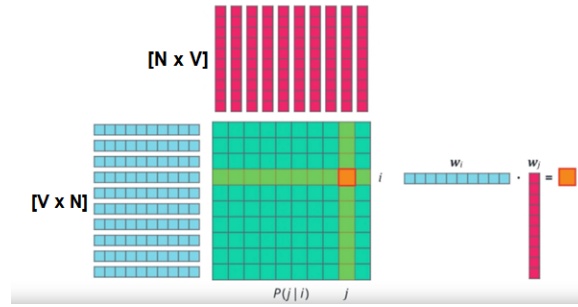
Mục tiêu của chúng tôi là dựa vào hệ số *bias* ở trên để tối ưu hóa nhằm tạo ra hai ma trận từ trung tâm (word\_vector)  $W[V \times N]$  và từ ngữ cảnh (context\_vector)  $\tilde{W}[N \times V]$  sao cho tích vô hướng của chúng sẽ bằng xác suất đồng xuất hiện  $P[V \times V]$  đã được tính ở **Bước 2**. Với  $W_i$  và  $W_j$  đại diện cho từ  $i^{th}$  &  $j^{th}$  cho hai ma trận tương ứng  $W$  và  $\tilde{W}$ .

**Bước 4:** Tính toán và thu lấy ma trận nhúng từ.

Sau khi thực hiện việc tối ưu hóa thì kết quả của tích vô hướng hai ma trận  $[V \times N]$  và  $[N \times V]$  thường vẫn sẽ xuất hiện sai số so với xác suất đồng xuất hiện  $P[V \times V]$ , chúng tôi gọi nó là độ lỗi, mục tiêu đề ra là giảm thiểu độ lỗi này ở mức thấp nhất có thể, chúng tôi sẽ giải quyết bài toán này với hàm mất mát (loss function), công thức (4.8).

Quá trình huấn luyện mô hình sẽ kết thúc khi độ lỗi của tích vô hướng hai ma trận  $[V \times N]$  và  $[N \times V]$  đạt giá trị nhỏ nhất.

**Output:** Kết quả cuối cùng mà chúng tôi thu được là một ma trận nhúng từ word\_vector  $[V \times N]$  tối ưu (Hình 4.8)



Hình 4.8: Cách thức hoạt động để thu được ma trận nhúng từ  $[V \times N]$

Mô hình glove thực hiện việc phân rã ma trận xác suất đồng xuất hiện thành hai ma trận nhỏ hơn. Trong mô hình glove chúng tôi sử dụng vector từ trong ma trận nhúng từ  $[V \times N]$  để biểu diễn một từ.

## 4.4 Đánh giá mô hình

**Ưu điểm:**

- Thời gian huấn luyện của mô hình glove nhanh hơn word2vec (do không phụ thuộc vào kích thước của kho ngữ liệu).
- Glove bổ sung một số ý nghĩa thiết thực hơn vào các vector từ bằng cách xem xét các mối quan hệ giữa cặp từ và cặp từ hơn là từ và từ.
- Glove giảm giá trị trọng số cho các cặp từ có tần suất cao để tránh các từ dừng (stop words - các từ được lọc ra trước hoặc sau quá trình xử lý dữ liệu văn bản) vô nghĩa như “the”, “an” sẽ không chi phối tới tiến trình huấn luyện mô hình.

**Khuyết điểm:**



- Mô hình được huấn luyện dựa trên ma trận đồng xuất hiện của các từ, vì thế cần rất nhiều bộ nhớ để lưu trữ. Đặc biệt, nếu chúng tôi thay đổi các siêu tham số liên quan đến ma trận đồng xuất hiện, chúng tôi phải xây dựng lại ma trận, điều này dẫn đến việc tiêu tốn rất nhiều thời gian.

## Chương 5

# XÂY DỰNG HỆ THỐNG PRODUCT RECOMMENDATION SỬ DỤNG WORD2VEC

### 5.1 Mục tiêu

Chúng tôi xây dựng một công cụ recommendation system đơn giản sử dụng mô hình word2vec để đề xuất các sản phẩm tương tự cho người khách hàng trên trang web thương mại điện tử dựa trên hành vi mua hàng trong quá khứ của người khách hàng.

## 5.2 Mô tả tập dữ liệu

Online retail là tập dữ liệu xuyên quốc gia chứa tất cả các giao dịch xảy ra từ ngày 01/12/2010 đến 09/12/2011 cho một trang web thương mại điện tử có trụ sở tại vương quốc Anh. Công ty chủ yếu bán quà tặng nhân dịp độc đáo.

## 5.3 Giải pháp

Quá trình xây dựng một công cụ recommendation system đơn giản gồm hai giai đoạn như sau:

- Xây dựng mô hình word2vec trong NLP (Xử lý ngôn ngữ tự nhiên) để tạo ra nhúng từ cho tên của tất cả các sản phẩm.
- Đề xuất các sản phẩm hàng đầu cho khách hàng dựa trên sự giống nhau của vector giữa các sản phẩm đã mua của khách hàng và toàn bộ sản phẩm trong trang web.

## 5.4 Thực hiện

**Đọc dữ liệu từ file excel**

Hình 5.1 , chúng tôi kiểm tra dữ liệu bán hàng từ file excel.

```
In [3]: df = pd.read_excel('Online_Retail.xlsx')
df.head()
```

Out[3]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

Hình 5.1: Kiểm tra dữ liệu bán hàng từ file excel

## Tiền xử lý dữ liệu

Hình 5.2, chúng tôi tiến hành tiền xử lý, loại bỏ các giá trị bị thiếu, các giá trị không hợp lệ.

```
In [44]: # kiểm tra các giá trị bị thiếu
df.isnull().sum()

# loại bỏ các giá trị không hợp lệ
df.dropna(inplace=True)

# chuyển Description sang kiểu string
df['Description'] = df['Description'].astype(str)
```

Hình 5.2: Tiền xử lý dữ liệu

## Chia tập dữ liệu thành train data and validation data

Hình 5.3, chúng tôi sắp xếp lại ngẫu nhiên các ID khách hàng và tiến hành chia tập dữ liệu thành train data (90%) và validation data (10%).

```
In [45]: customers = df['CustomerID'].unique().tolist()

# sắp xếp lại ngẫu nhiên các ID khách hàng
random.shuffle(customers)

# Chia tập dữ liệu gồm 90% train data và 10% validation data
customers_train = [customers[i] for i in range(round(0.9*len(customers)))]

# chia dữ liệu thành train data và validation data
train_df = df[df['CustomerID'].isin(customers_train)]
validation_df = df[~df['CustomerID'].isin(customers_train)]
#print(validation_df)
```

Hình 5.3: Chia tập dữ liệu thành train data and validation data

Tạo một chuỗi các sản phẩm đã mua của một khách hàng

Ở bước này, chúng tôi tạo ra một list để nắm bắt lịch sử khách hàng thực hiện trên tập train data được mô tả qua hình 5.4.

```
In [46]: #Tạo ra một List để nắm bắt Lịch sử khách hàng( thực hiện trên tập train data)
purchases_train = []

# Với mỗi CustomerID sẽ ghi danh sách các sản phẩm đã mua của họ
for i in tqdm(customers_train):
    temp = train_df[train_df["CustomerID"] == i]["Description"].tolist()
    purchases_train.append(temp)
```

100%|██████████| 3935/3935 [00:05<00:00, 723.76it/s]

Hình 5.4: Tạo một chuỗi các sản phẩm đã mua của một khách hàng trên tập train data

Tiếp theo, chúng tôi làm tương tự cho tập validation data như Hình 5.5.

```
In [47]: # Tạo ra một List để nắm bắt lịch sử khách hàng( thực hiện trên tập validation data)
purchases_val = []
# Với mỗi CustomerID sẽ ghi danh sách các sản phẩm đã mua của họ
for i in tqdm(validation_df['CustomerID'].unique()):
    temp = validation_df[validation_df["CustomerID"] == i]["Description"].tolist()
    purchases_val.append(temp)
#import csv
#with open('file1.csv', 'w', newline='') as f:
#    #writer = csv.writer(f)
#    #writer.writerows(purchases_val)
```

Hình 5.5: Tạo một chuỗi các sản phẩm đã mua của một khách hàng trên tập validation data

## Xây dựng Word2Vec Embeddings cho các sản phẩm

Hình 5.6, chúng tôi tiến hành xây dựng mô hình word2vec embeddings cho các sản phẩm.

```
In [48]: # train word2vec model
model = Word2Vec(window = 10, sg = 1, hs = 0,
                 negative = 10, # for negative sampling
                 alpha=0.03, min_alpha=0.0007,
                 seed = 14)

model.build_vocab(purchases_train, progress_per=200)

model.train(purchases_train, total_examples = model.corpus_count,
            epochs=10, report_delay=1)

Out[48]: (3626624, 3664250)

In [9]: model.init_sims(replace=True)
print(model)
model.wv.save_word2vec_format('model.bin')
model.wv.save_word2vec_format('model.txt', binary=False)

Word2Vec(vocab=3281, size=100, alpha=0.03)
```

Hình 5.6: Xây dựng word2vec embeddings cho các sản phẩm

## Các thông số của mô hình sau khi đã train

Hình 5.7, mô tả các thông số của mô hình sau khi kết thúc quá trình train. Mô hình gồm 3281 từ và được nhúng với số chiều là 100.

```
In [10]: # Model gồm 3281 từ với số chiều là 100
X = model[model.wv.vocab]

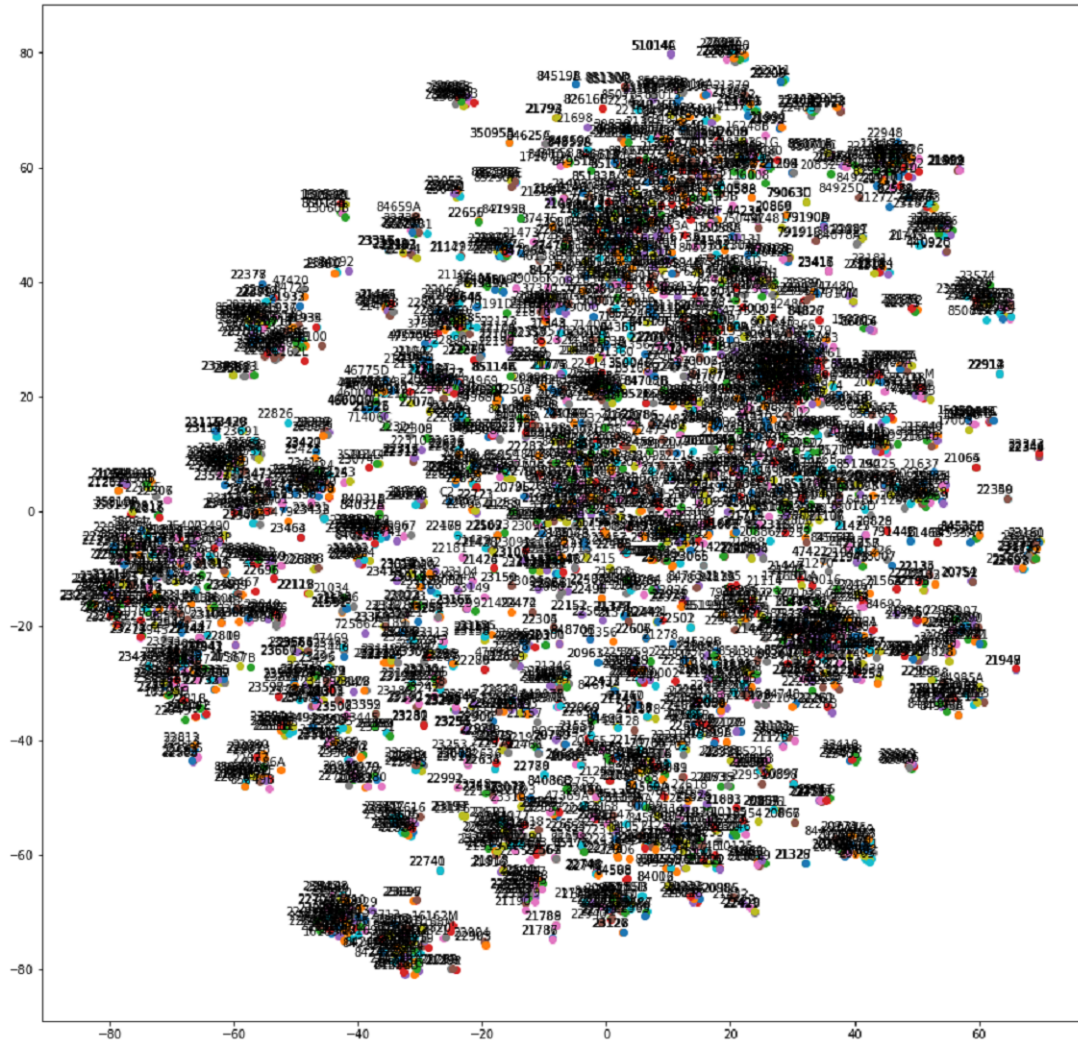
X.shape

Out[10]: (3281, 100)
```

Hình 5.7: Các thông số của mô hình sau khi đã train

Visualize word2vec embeddings bằng cách thu giảm số chiều sử dụng tSNE

Chúng tôi tiến hành visualize word2vec embeddings, với mỗi chấm là một khách hàng đã mua các sản phẩm. Hình 5.8, mô tả mã các sản phẩm mà khách hàng đó đã mua.



Hình 5.8: Các sản phẩm mà khách hàng đã mua

## Đề xuất sản phẩm

Chúng tôi tạo danh sách gồm mã sản phẩm-tên sản phẩm như hình 5.9.

```
In [14]: products = train_df[["Description", "StockCode"]]

# Loại bỏ các giá trị trùng
products.drop_duplicates(inplace=True, subset='Description', keep="last")

# ứng với mỗi sản phẩm là mã sản phẩm và tên sản phẩm đó
products_dict = products.groupby('Description')['Description'].apply(list).to_dict()
products_dict1 = products.groupby('Description')['StockCode'].apply(list).to_dict()
```

Hình 5.9: Tạo danh sách gồm mã sản phẩm-tên sản phẩm

Hình 5.10, chúng tôi viết một function sử dụng thư viện Gensim (similar\_by\_vector) để tìm sự tương đồng giữa các vector sản phẩm.

```
In [15]: def similar_products(v, n = 6):

# extract most similar products for the input vector
ms = model.similar_by_vector(v, topn=n+1)[1:]

# extract name and similarity score of the similar products
new_ms = []
for j in ms:
    pair = (products_dict[j[0]][0], products_dict1[j[0]][0], j[1])
    new_ms.append(pair)

return new_ms
```

Hình 5.10: Function tìm sự tương đồng giữa các vector sản phẩm

Sau đó, chúng tôi tìm các sản phẩm tương tự cho một sản phẩm có tên là ("YELLOW COAT RACK PARIS FASHION"). Kết quả thu được là 6 sản phẩm tương tự hàng đầu như hình 5.11.

```
In [23]: similar_products(model["YELLOW COAT RACK PARIS FASHION"])
Out[23]: [('BLUE COAT RACK PARIS FASHION', 22914, 0.905225396156311),
('RED COAT RACK PARIS FASHION', 22913, 0.8777114748954773),
('BLUE TILED TRAY', 20793, 0.592170000076294),
('CREAM CUPID HEARTS COAT HANGER', '84406B', 0.5742723941802979),
('FOUR HOOK WHITE LOVEBIRDS', 22212, 0.545958936214447),
('HELLO SAILOR BATHROOM SET', 84842, 0.5431439280509949)]
```

Hình 5.11: Đề xuất các sản phẩm tương tự cho sản phẩm có tên là ("YELLOW COAT RACK PARIS FASHION")

Để đưa ra đề xuất về các sản phẩm trong tương lai mà khách hàng có thể mua dựa vào lịch sử mua hàng trong quá khứ của họ. Chúng tôi tiến hành viết



một function trả về giá trị trung bình của các vector sản phẩm của một khách hàng đã mua.

Hình 5.12, function trả về giá trị trung bình của các vector sản phẩm của một khách hàng đã mua.

```
In [25]: def aggregate_vectors(products):  
         product_vec = []  
         #chuỗi các sản phẩm mà một khách hàng đã mua  
         for i in products:  
             try:  
                 product_vec.append(model[i])  
             except KeyError:  
                 continue  
  
         #tính giá trị trung bình cho product_vec  
         return np.mean(product_vec, axis=0)
```

Hình 5.12: Function trả về giá trị trung bình của các vector sản phẩm của một khách hàng đã mua

Cuối cùng, chúng tôi tiến hành đề xuất các sản phẩm trong tương lai cho khách hàng dựa vào toàn bộ lịch sử mua hàng của họ. Kết quả thu được là 6 sản phẩm hàng đầu như hình 5.13.

```
In [88]: #print(validation_df['CustomerID'].unique())
#Liệt kê các CustomerID (bỏ qua các giá trị trùng lặp)
#chọn ra 1 khách hàng có ID=16098
pp = validation_df[validation_df['CustomerID'] ==16098]
#Liệt kê thông tin sản phẩm mà khách hàng đó đã mua
print(pp[['CustomerID', 'StockCode', 'Description']])
#đưa ra các đề xuất về các sản phẩm trong tương lai
similar_products(aggregate_vectors(purchases_val[0]))
```

	CustomerID	StockCode	Description
142	16098.0	10002	INFLATABLE POLITICAL GLOBE
143	16098.0	21912	VINTAGE SNAKES & LADDERS
144	16098.0	21832	CHOCOLATE CALCULATOR
145	16098.0	22411	JUMBO SHOPPER VINTAGE RED PAISLEY
146	16098.0	22379	RECYCLING BAG RETROSPOT
...	...	...	...
337953	16098.0	21912	VINTAGE SNAKES & LADDERS
337954	16098.0	21888	BINGO SET
337955	16098.0	84754	S/15 SILVER GLASS BAUBLES IN BAG
337956	16098.0	23215	JINGLE BELL HEART ANTIQUE SILVER
337957	16098.0	23214	JINGLE BELL HEART ANTIQUE GOLD

```
[67 rows x 3 columns]
```

```
Out[88]: [('ALARM CLOCK BAKELIKE IVORY', 22730, 0.786604642868042),
('ALARM CLOCK BAKELIKE GREEN', 22726, 0.7797957062721252),
('ALARM CLOCK BAKELIKE PINK', 22728, 0.7644144296646118),
('ALARM CLOCK BAKELIKE CHOCOLATE', 22725, 0.7413551211357117),
('ALARM CLOCK BAKELIKE ORANGE', 22729, 0.738258957862854),
('BLUE DINER WALL CLOCK', 22192, 0.7034780979156494)]
```

Hình 5.13: Đề xuất các sản phẩm trong tương lai mà khách hàng có thể mua

Bên cạnh đó chúng tôi cũng thực hiện đề xuất dựa trên 10 sản phẩm gần đây nhất mà khách hàng mua. Kết quả là được 6 sản phẩm hàng đầu như hình 5.14.

```
In [87]: # Đề xuất sản phẩm dựa trên 10 sản phẩm mua gần đây nhất
print(purchases_val[0][-10:])
similar_products(aggregate_vectors(purchases_val[0][-10:]))
```

```
[('ALARM CLOCK BAKELIKE IVORY', 'ALARM CLOCK BAKELIKE CHOCOLATE', 'JUMBO SHOPPER VINTAGE RED PAISLEY', 'ALARM CLOCK BAKELIKE GRE
EN', 'ALARM CLOCK BAKELIKE GREEN', 'VINTAGE SNAKES & LADDERS', 'BINGO SET', 'S/15 SILVER GLASS BAUBLES IN BAG', 'JINGLE BELL HE
ART ANTIQUE SILVER', 'JINGLE BELL HEART ANTIQUE GOLD']
```

```
Out[87]: [('ALARM CLOCK BAKELIKE RED ', 22727, 0.7729189395904541),
('ALARM CLOCK BAKELIKE IVORY', 22730, 0.7670091986656189),
('ALARM CLOCK BAKELIKE CHOCOLATE', 22725, 0.7611282467842102),
('ALARM CLOCK BAKELIKE PINK', 22728, 0.7513300180435181),
('ALARM CLOCK BAKELIKE ORANGE', 22729, 0.6962610483169556),
('IVORY DINER WALL CLOCK', 22191, 0.6312854290008545)]
```

Hình 5.14: Đề xuất các sản phẩm dựa trên 10 sản phẩm mà khách hàng đã mua gần đây nhất

Như vậy là chúng tôi đã hoàn thành việc xây dựng hệ thống product recommendation đơn giản.

## Chương 6

# KẾT LUẬN

### 6.1 Đóng góp của tiểu luận

Về lý thuyết:

Đã hiểu được khái niệm cơ bản của kỹ thuật word embedding và cách hoạt động và huấn luyện của hai mô hình word2vec và mô hình glove.

Về ứng dụng:

Đã xây dựng được một hệ thống product recommendation (đề xuất sản phẩm) đơn giản cho trang bán hàng trực tuyến sử dụng mô hình word2vec.

### 6.2 Hạn chế của tiểu luận

Do xây dựng hệ thống product recommendation (đề xuất sản phẩm) trên tập dữ liệu nhỏ nên chưa thể hiện được quá tốt việc gợi ý các sản phẩm cho người dùng.

## 6.3 Phương hướng phát triển tiểu luận trong tương lai

### Về lý thuyết:

Trong tương lai, chúng tôi sẽ tiếp tục tìm hiểu cách cải tiến, tối ưu mô hình word2vec như: word2vec negative sampling và word2vec softmax phân cấp. Bên cạnh đó là tìm hiểu thêm các mô hình mới của kỹ thuật word embedding có tốc độ xử lý nhanh hơn, tối ưu hơn.

### Về ứng dụng:

Chúng tôi sẽ xây dựng một trang web product recommendation (đề xuất sản phẩm) với một lượng dữ liệu lớn hơn, mang tính thực tiễn cao hơn.

# Tài liệu tham khảo

- [1] Hironson. Why is word embeddings important for natural language processing. 21/03/2017. <https://medium.com/@Hironson/why-is-word-embeddings-important-for-natural-language-processing-6b69dd384a77>, Ngày truy cập: 21/12/2020.
- [2] Manjeet Singh. Word embedding. 15/10/2017. <https://medium.com/data-science-group-iitr/word-embedding-2d05d270b285>, Ngày truy cập: 21/12/2020.
- [3] Aston Zhang et al. *DIVE INTO DEEP LEARNING. 2020*, chapter 14. Ngày truy cập: 21/12/2020.
- [4] DATA SCIENCE. How to process textual data using tf-idf in python. 06/06/2018. <https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3>, Ngày truy cập: 01/01/2021.
- [5] NSS. An intuitive understanding of word embeddings: From count vectors to word2vec. 04/06/2017. <https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/>, Ngày truy cập: 02/01/2021.
- [6] Zeeshan Mulla. Word embeddings in natural language processing | nlp.

- 28/06/2020. <https://medium.com/@zeeshanmulla/word-embeddings-in-natural-language-processing-nlp-5be7d6fb1d73>, Ngày truy cập: 02/01/2021.
- [7] Will Koehrsen. Neural network embeddings explained. 02/10/2018. <https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526>, Ngày truy cập: 02/01/2021.
- [8] Tomas Mikolov et al. Efficient estimation of word representations in vector space. 07/09/2013. <https://arxiv.org/pdf/1301.3781.pdf>, Ngày truy cập: 03/01/2021.
- [9] Dipanjan(DJ)Sarkar. A hands-on intuitive approach to deep learning methods for text data — word2vec, glove and fasttext. 14/03/2018. <https://towardsdatascience.com/understanding-feature-engineering-part-4-deep-learning-methods-for-text-data-96c44370bbfa>, Ngày truy cập: 03/01/2021.
- [10] Thushan Ganegedara. Intuitive guide to understanding word2vec. 05/06/2018. <https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-understanding-word2vec-e0128a460f0f>, Ngày truy cập: 03/01/2021.
- [11] Chris McCormick. Word2vec tutorial - the skip-gram model. 19/04/2016. <http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/>, Ngày truy cập: 03/01/2021.
- [12] Machine Learning cơ Bản. Bài 13: Softmax regression. 17/02/2017. <https://machinelearningcoban.com/2017/02/17/softmax/>, Ngày truy cập: 04/01/2021.

- [13] Machine Learning cơ Bản. Bài 8: Gradient descent (phần 2/2). 16/01/2017. <https://machinelearningcoban.com/2017/01/16/gradientdescent2/>, Ngày truy cập: 04/01/2021.
- [14] Eric Kim. Demystifying neural network in skip-gram language modeling. 06/05/2019. <https://aegis4048.github.io>, Ngày truy cập: 03/01/2021.
- [15] Machine Learning cơ Bản. Bài 7: Gradient descent (phần 1/2). 12/01/2017. <https://machinelearningcoban.com/2017/01/12/gradientdescent/>, Ngày truy cập: 04/01/2021.
- [16] Hussain Mujtaba. What is word embedding | word2vec | glove. 12/07/2020. <https://www.mygreatlearning.com/blog/word-embedding/>, Ngày truy cập: 03/01/2021.
- [17] Jeffrey Pennington et al. Glove: Global vectors for word representation. 2014. <https://www.aclweb.org/anthology/D14-1162.pdf>, Ngày truy cập: 04/01/2021.