

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

----- oOo -----



ĐỒ ÁN III

ĐỀ TÀI: PHÂN TÍCH CẢM XÚC TRONG TIẾNG VIỆT

Giảng viên hướng dẫn: Ts. Lê Chí Ngọc

Sinh viên thực hiện: Nguyễn Thị Hằng

MSSV: 20161384

Lớp: Toán Tin - K61

Hà Nội, 2020

LỜI NÓI ĐẦU

Thu thập thông tin phản hồi của khách hàng là một cách giúp cho các doanh nghiệp hiểu được điểm mạnh, điểm yếu trong sản phẩm, dịch vụ của mình; đồng thời nhanh chóng nắm bắt được tâm lý và nhu cầu khách hàng nhằm phát triển sản phẩm phù hợp thị hiếu, mang đến cho khách hàng sản phẩm, dịch vụ hoàn hảo nhất.

Ngày nay, với sự phát triển vượt bậc của khoa học và công nghệ, đặc biệt là sự bùng nổ của Internet với các phương tiện truyền thông, mạng xã hội, thương mại điện tử,... việc chia sẻ thông tin, thể hiện thái độ, quan điểm của mình đối với các sản phẩm, dịch vụ và các vấn đề xã hội khác ngày càng trở nên dễ dàng, phong phú dưới nhiều hình thức khác nhau. Vì vậy Internet trở thành nguồn cung cấp một lượng thông tin vô cùng lớn và quan trọng.

Thông qua những dữ liệu được cung cấp qua Internet:

- Người dùng sử dụng nó để tìm kiếm, tham khảo trước khi đưa ra quyết định về sử dụng một sản phẩm hay dịch vụ nào đó.
- Các nhà cung cấp dịch vụ cũng có thể sử dụng những nguồn thông tin này để đánh giá về sản phẩm của mình, từ đó có thể đưa ra những cải tiến phù hợp hơn với người dùng, mang lại lợi nhuận cao hơn, tránh các rủi ro đáng tiếc xảy ra. Đặc biệt, khi một doanh nghiệp có một sản phẩm mới ra mắt thị trường thì việc lấy ý kiến phản hồi là vô cùng cần thiết.
- Các cơ quan chức năng có thể sử dụng những thông tin này để tìm hiểu xem quan điểm và thái độ của cộng đồng để có thể kịp thời sửa đổi, ban hành các chính sách cho hợp lý hơn.

Đứng trước sự phát triển mạnh mẽ của công nghệ, thương mại điện tử việc phân tích, bóc tách dữ liệu là việc cần thiết và vô cùng quan trọng. Vì vậy em quyết định lựa chọn đề tài “Phân tích cảm xúc trong Tiếng Việt” để tìm hiểu, nghiên cứu

làm đồ án III với mong muốn phần nào áp dụng bài toán vào thực tế. Đề tài đặt ra một số yêu cầu cần giải bài toán như:

1. Thu thập và phân tích đặc điểm dữ liệu.
2. Tiền xử lý dữ liệu.
3. Vector hóa dữ liệu.
4. Xây dựng và huấn luyện mô hình.
5. Hướng phát triển bài toán.

Với những yêu cầu đặt ra ở trên cấu trúc đồ án sẽ bao gồm:

- **GIỚI THIỆU VỀ BÀI TOÁN PHÂN TÍCH CẢM XÚC TRONG TIẾNG VIỆT**

Giới thiệu tổng quan về bài toán phân tích cảm xúc trong Tiếng Việt.

- **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

Một số cơ sở lý thuyết về xử lý ngôn ngữ tự nhiên.

- **CHƯƠNG 2: MỘT SỐ MÔ HÌNH ĐỀ XUẤT**

Phân tích một số mô hình đề xuất áp dụng trong phạm vi đề tài.

Giới thiệu mô hình hóa bài toán, thiết kế dữ liệu luyện mạng, mô hình đề xuất huấn luyện mạng, xây dựng chương trình.

- **CHƯƠNG 3: KẾT QUẢ**

Trình bày môi trường cài đặt, các yêu cầu liên quan, đánh giá kết quả, ưu điểm, khuyết điểm và đưa ra hướng phát triển tương lai.

Hà nội, tháng 03 năm 2020

Người thực hiện đồ án

Hằng

Nguyễn Thị Hằng

MỤC LỤC

| | |
|--|-----------|
| LỜI NÓI ĐẦU | 2 |
| DANH SÁCH HÌNH VẼ..... | 6 |
| GIỚI THIỆU VỀ BÀI TOÁN PHÂN TÍCH CẢM XÚC TRONG TIẾNG VIỆT | 8 |
| Giới thiệu bài toán phân tích cảm xúc trong Tiếng Việt | 8 |
| Ứng dụng bài toán trong thực tế..... | 9 |
| CHƯƠNG 1: CƠ SỞ LÝ THUYẾT | 10 |
| 1.1 Giới thiệu mạng neural nhân tạo | 11 |
| 1.2 Mạng neural hồi tiếp (Recurrent Neural Network) | 25 |
| 1.2.1 Cấu trúc mạng neural hồi tiếp..... | 26 |
| 1.2.2 Phương pháp đào tạo mạng Neural hồi tiếp | 30 |
| CHƯƠNG 2: MỘT SỐ MÔ HÌNH ĐỀ XUẤT..... | 36 |
| 2.1 Giới thiệu mô hình hóa bài toán | 36 |
| 2.2 Thiết kế dữ liệu huấn luyện mạng | 36 |
| 2.3 Mô hình đề xuất huấn luyện mạng và xây dựng mô hình | 39 |
| 2.3.1 Mô hình Naïve Bayes | 39 |
| 2.3.2 Mô hình Support Vector Machine (SVM)..... | 44 |
| 2.3.3 Mô hình FastText+CNN+LSTM | 49 |
| 2.3.4 Lựa chọn mô hình | 66 |
| CHƯƠNG 3: KẾT QUẢ..... | 68 |
| 3.1 Môi trường cài đặt thuật toán và các vấn đề liên quan..... | 68 |
| 3.2 Kết quả đánh giá mô hình..... | 68 |
| 3.3 Xây dựng chương trình..... | 69 |

| | | |
|---------------------------------|--|-----------|
| 3.4 | Định hướng phát triển trong tương lai..... | 73 |
| CHỈ MỤC..... | | 75 |
| TÀI LIỆU THAM KHẢO | | 77 |

DANH SÁCH HÌNH VẼ

| | |
|---|----|
| Hình 1 Natural Language Processing Pipeline | 10 |
| Hình 2 Mô hình mạng neural tuyến tính | 11 |
| Hình 3 Mô hình mạng neural perceptron | 12 |
| Hình 4 Mô hình mạng neural sigmoid | 12 |
| Hình 5 Mô hình chung của mạng neural | 13 |
| Hình 6 Đồ thị hàm tuyến tính | 15 |
| Hình 7 Đồ thị hàm bước nhị phân | 16 |
| Hình 8 Đồ thị hàm Sigmoid | 16 |
| Hình 9 Đồ thị loss function – logistic (trường hợp: $y_i=1$) | 17 |
| Hình 10 Đồ thị loss function – logistic (trường hợp $y_i=0$) | 18 |
| Hình 11 Đồ thị hàm sigmoid lưỡng cực | 19 |
| Hình 12 Đồ thị hàm tanh | 20 |
| Hình 13 Đồ Thị Hàm ReLU | 21 |
| Hình 14 Mô hình mạng lan truyền tiến | 22 |
| Hình 15 Mạng neural Elman | 26 |
| Hình 16 Mạng RNN với self-feedback ở hidden layer | 27 |
| Hình 17 Mạng RNN kết nối đầy đủ self-feedback | 27 |
| Hình 18 Mạng neural hồi tiếp kết nối đầy đủ với một lớp có 2 neural theo bước thời gian (giai đoạn FFNNs) | 30 |
| Hình 19 Mô hình bài toán phân tích cảm xúc tiếng việt | 36 |
| Hình 20 Lệ của hai classes là bằng nhau và lớn nhất có thể. | 46 |
| Hình 21 Phân tích bài toán SVM. | 46 |
| Hình 22 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn. | 48 |
| Hình 23 Mô hình CBOW và Skip-gram | 50 |
| Hình 24 Kiến trúc mạng Word Embedding | 51 |
| Hình 25 Kiến trúc mạng CBOW | 53 |
| Hình 26 Kiến trúc mạng skip-gram | 54 |
| Hình 27 Mô phỏng hoạt động của Convolution | 56 |

| | |
|---|----|
| Hình 28 Mô tả hạt nhân hoạt động Conv2D | 57 |
| Hình 29 Mô tả hạt nhân hoạt động Conv1D | 57 |
| Hình 30 Mô phỏng quá trình hoạt động của convolution | 58 |
| Hình 31 Max - Pooling | 59 |
| Hình 32 Kiến trúc LSTM tiêu chuẩn | 61 |
| Hình 33 Mô hình Word_Embedding+CNN+LSTM | 65 |
| Hình 34 Kiến trúc mô hình | 65 |
| Hình 35 Cấu trúc xây dựng chương trình | 69 |
| Hình 36 Trang login | 70 |
| Hình 37 Trang chủ hệ thống | 71 |
| Hình 38 Trang thu thập dữ liệu | 71 |
| Hình 39 Trang lựa chọn dữ liệu có sẵn hoặc từ bên thứ 3 | 72 |
| Hình 40 Trang thông tin tiền xử lý giao diện | 72 |
| Hình 41 Trang xuất dữ liệu cần phân tích | 73 |

GIỚI THIỆU VỀ BÀI TOÁN PHÂN TÍCH CẢM XÚC TRONG TIẾNG VIỆT

Giới thiệu bài toán phân tích cảm xúc trong Tiếng Việt

Phân tích cảm xúc là bài toán cơ bản trong xử lý ngôn ngữ tự nhiên (natural language processing). Đây là bài toán đánh giá sắc thái tình cảm yêu, thích, ghét giận dữ ... của mọi người đối với một đối tượng được xem xét (sản phẩm, dịch vụ, vấn đề chính trị xã hội...)

Bài toán phân tích cảm xúc thuộc dạng bài toán phân tích ngữ nghĩa văn bản. Vì vậy, ta cần phải xây dựng một mô hình để hiểu được ý nghĩa của câu văn, đoạn văn để quyết định xem câu văn đó hoặc đoạn văn đó mang màu sắc cảm xúc chủ đạo nào.

Phát biểu theo góc nhìn của máy học (Machine Learning) thì phân tích cảm xúc là bài toán phân lớp cảm xúc chính là bài toán xử lý ngôn ngữ tự nhiên với đầu vào của bài toán là một câu hay một đoạn văn bản, còn đầu ra là các giá trị xác suất (điểm số) của N lớp cảm xúc mà ta cần xác định.

Trong phạm vi đề tài đồ án em sẽ nghiên cứu ở mức độ đơn giản là phân tích cảm xúc với hai lớp là tích cực (positive) và tiêu cực (negative):

- ✓ Đầu vào bài toán: Là một file chứa tất cả các câu bình luận của khách hàng được thu thập trên website (có thể là của một công ty, đơn vị, tổ chức... nào đó).
- ✓ Đầu ra bài toán: Là dự đoán một câu, một đoạn văn là tiêu cực hay tích cực và sau đó đưa ra một bảng hiển thị tỷ lệ bình luận tích cực và tiêu cực của khách hàng về đối tượng được xem xét.

Ứng dụng bài toán trong thực tế

Việc phân tích cảm xúc trong văn bản được ứng dụng trong hàng loạt các vấn đề như: Quản trị thương hiệu doanh nghiệp, thương hiệu sản phẩm, quản trị quan hệ khách hàng, khảo sát ý kiến xã hội học, phân tích trạng thái tâm lý con người...

❖ Ví dụ cụ thể:

- ✓ Điều cốt yếu của giải pháp này chính là phân tích cảm xúc của các dòng trạng thái trên mạng xã hội (facebook, zalo, Instagram...) nhằm lọc ra các thông tin bất lợi để xử lý.
- ✓ Phân tích mức độ hưởng ứng sản phẩm của khách hàng thông qua các bình luận để đưa ra các chiến lược kinh doanh phù hợp...

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

Xử lý ngôn ngữ tự nhiên - Natural Language Processing (NLP): Được xây dựng dựa trên ngôn ngữ học phức tạp, các nguyên lý thống kê, và thuật toán mạng neural (neural network algorithms). Chương trình NLP có khả năng đọc và hiểu được văn bản với tốc độ cao. Do đó, dù bạn có 1000 tài liệu hay thậm chí hàng tỉ văn bản, chương trình NLP có thể “tổng hợp” nhanh chóng tất cả các thông tin này, từ đó có thể rút trích ra được những tri thức (knowledge) đáng giá cho doanh nghiệp của bạn như: tri thức về các khách hàng, tri thức về những đối thủ cạnh tranh, tri thức về các hoạt động trong doanh nghiệp như điều hành, marketings, sales, kỹ thuật, và sản phẩm.

Bài toán xử lý ngôn ngữ tự nhiên đang được quan tâm:

❖ Bài toán mức độ đơn giản:

- ✓ Kiểm tra chính tả (Spell checking)
- ✓ Tìm kiếm từ khóa (keyword search)
- ✓ Tìm từ đồng nghĩa (Finding synonyms)

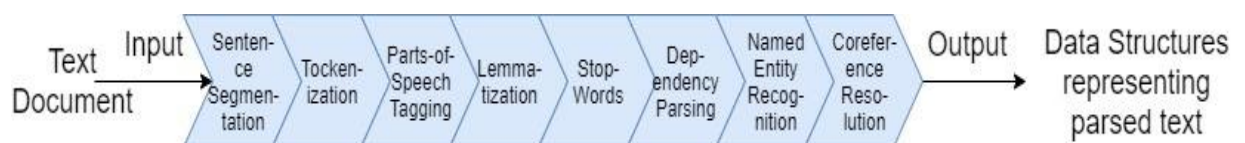
❖ Bài toán mức độ trung bình:

- ✓ Phân tích thông tin từ các tài liệu, websites, ... (Parsing information from websites, documents, etc.)

❖ Bài toán mức độ khó:

- ✓ Dịch máy (Machine translation)
- ✓ Phân tích ngữ nghĩa (Semantic analysis)
- ✓ Coreference resolution
- ✓ Trả lời câu hỏi, chatbot, ... (Questions answering)

Sơ đồ mức độ bài toán xử lý ngôn ngữ tự nhiên



Hình 1 Natural Language Processing Pipeline

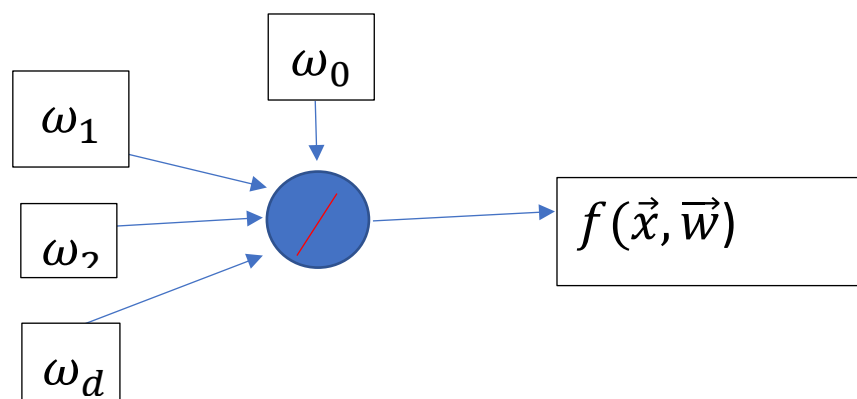
1.1 Giới thiệu mạng neural nhân tạo

Mạng neural nhân tạo hay thường gọi ngắn gọn mạng neural (tiếng Anh là Artificial Neural network - ANN hay Neural Network) là một mô hình toán học hay mô hình tính toán được sử dụng phổ biến trong lĩnh vực machine learning và deep learning. Nó gồm có một nhóm các neural nhân tạo được nối với nhau thông qua các liên kết (trọng số liên kết) và làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Việc xử lý thông tin bằng cách truyền theo các kết nối và tính giá trị mới tại các nút. Trong nhiều trường hợp, mạng neural nhân tạo là một hệ thống thích ứng (*adaptive system*) tự thay đổi cấu trúc của mình dựa trên các thông tin bên ngoài hay bên trong chảy qua mạng trong quá trình học [1] [2].

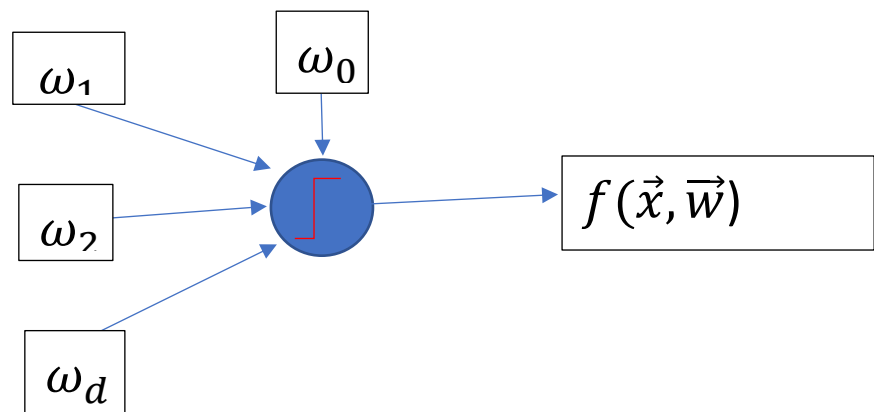
Trong thực tế sử dụng, nhiều mạng neural là các công cụ mô hình hóa dữ liệu thống kê phi tuyến. Chúng có thể được dùng để mô hình hóa các mối quan hệ phức tạp giữa dữ liệu vào và kết quả hoặc để tìm kiếm các dạng mẫu trong dữ liệu.

Một mạng neural nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu...) thông qua một quá trình học từ tập các mẫu huấn luyện. Một mạng neural nhân tạo thường được kết hợp từ ba thành phần: đơn vị xử lý, hàm kết hợp, hàm kích hoạt [3].

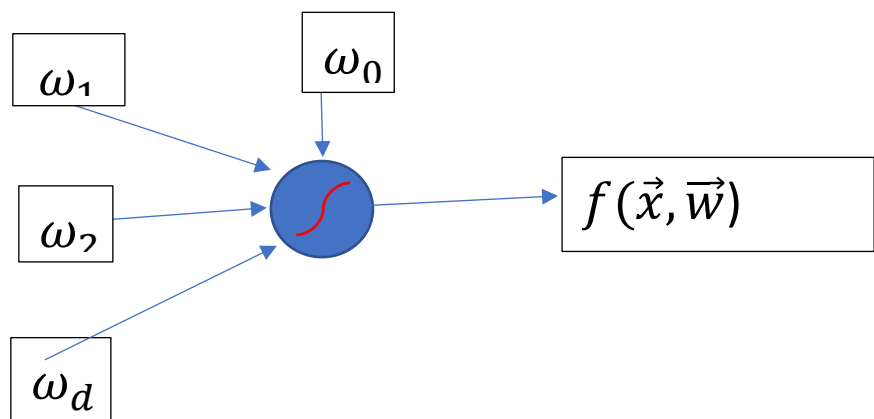
Ví dụ một số mạng neural cơ bản:



Hình 2 Mô hình mạng neural tuyến tính



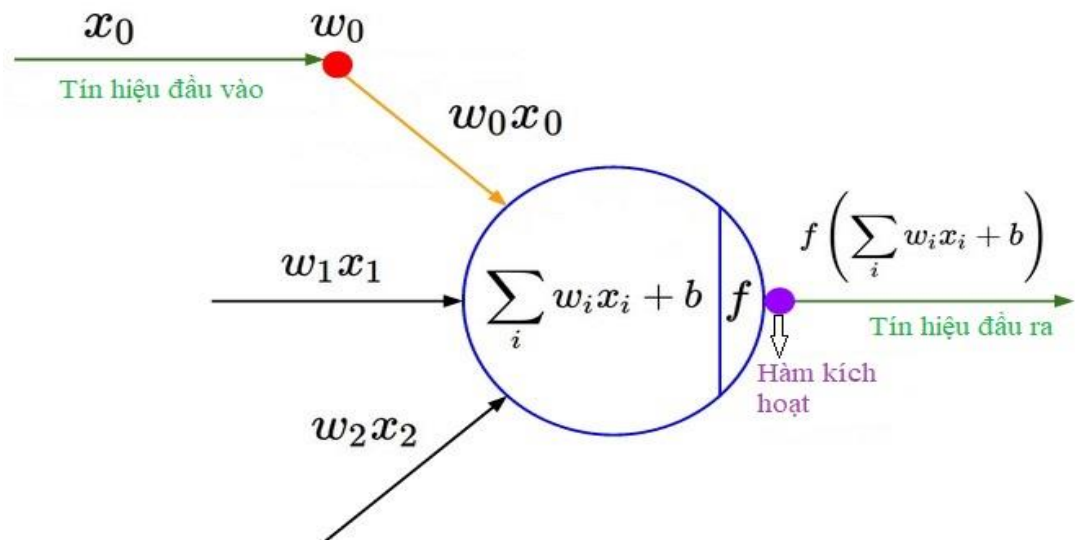
Hình 3 Mô hình mạng neural perceptron



Hình 4 Mô hình mạng neural sigmoid

Đơn vị xử lý:

Còn được gọi là một neural hay một nút (node), thực hiện một công việc rất đơn giản: Nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu đầu ra sẽ được truyền sang một đơn vị khác. Cấu trúc của một neural được minh họa như sau:



Hình 5 Mô hình chung của mạng neural

Trong đó:

- ✓ x_i là các tín hiệu đầu vào
- ✓ w_i là các trọng số. Tương ứng với x_i có trọng số $w_i x_i$
- ✓ b là hệ số bias
- ✓ $a_i = \sum_i w_i x_i + b$ là đầu vào mạng (net input)
- ✓ $z_i = f(\sum_i w_i x_i + b)$ là đầu ra của neural
- ✓ f hàm kích hoạt

Một mạng neural có ba kiểu đơn vị:

- ✓ Các đơn vị đầu vào (input units), nhận tín hiệu từ bên ngoài
- ✓ Các đơn vị đầu ra (output units), gửi tín hiệu ra bên ngoài
- ✓ Các đơn vị ẩn (hidden units) các tín hiệu vào (input) và ra (output) của nó nằm trong mạng

Mỗi đơn vị i có thể có một hoặc nhiều đầu vào nhưng chỉ có một đầu ra. Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc là đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

Hàm kết hợp:

Mỗi một đơn vị trong mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi luật lan truyền cụ thể. Trong phần lớn các mạng neural, chúng ta giả sử rằng mỗi một đơn vị cung cấp một bộ cộng như là đầu vào cho đơn vị mà nó có liên kết. tổng đầu vào đơn vị i đơn giản chỉ là trọng số của các đầu ra riêng lẻ từ các đơn vị kết nối cộng thêm ngưỡng hay độ lệch b (*bias*):

$$a_i = \sum_i w_i x_i + b$$

Trường hợp $w_i > 0$, neural được coi là đang ở trong trạng thái kích thích. tương tự nếu như $w_i < 0$, neural ở trạng thái kiềm chế.

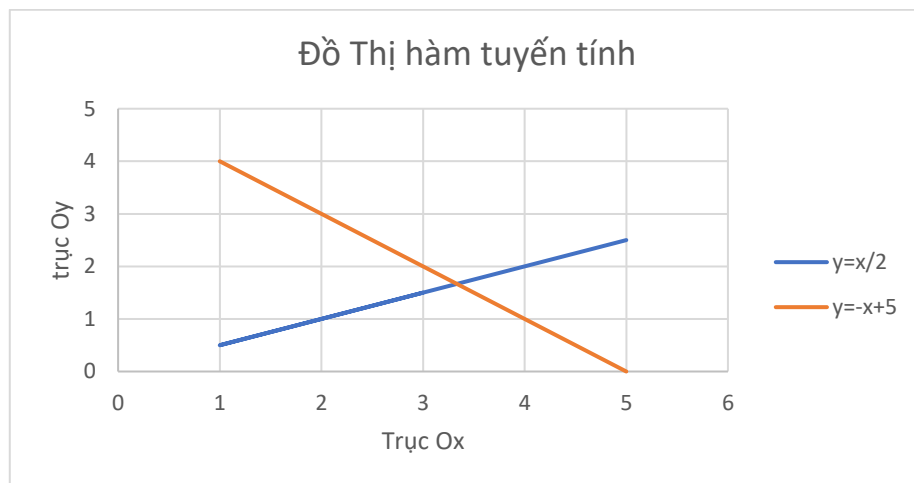
Hàm kích hoạt:

Phần lớn các đơn vị trong mạng neural chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt [4], kết quả của hàm này là một giá trị gọi là mức kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm ép (squashing). Các hàm kích hoạt được sử dụng là:

Hàm đồng nhất (Linear function , Identity function):

$$l(x) = x$$

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



Hình 6 Đồ thị hàm tuyến tính

Công thức hàm mất mát:

$$J = \frac{1}{2} * \frac{1}{N} * \left(\sum_{i=1}^N (\hat{y}_i - y_i)^2 \right)$$

$$\hat{y}_i = \omega_i * x_i + \omega_0$$

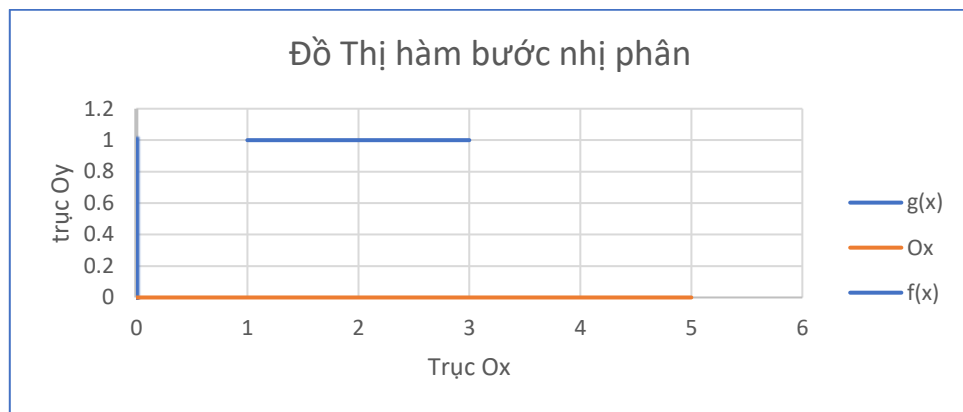
- ✓ J không âm
 - ✓ J càng nhỏ thì đường thẳng càng gần điểm dữ liệu. Nếu j=0 thì đường thẳng đi qua tất cả các điểm dữ liệu
- => Bài toán đặt ra là cực tiểu hàm mất mát để thu được kết quả tốt nhất có thể

Hàm bước nhị phân (Binary step function, Hard limit function):

Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function) [4]. Đầu ra của hàm này được giới hạn vào một trong hai giá trị:

$$g(x) = \begin{cases} 1, & \text{nếu } (x > \theta) \\ 0, & \text{nếu } (x < \theta) \end{cases}$$

Dạng hàm này được sử dụng trong các mạng chỉ có một lớp. Trong hình vẽ sau, b được chọn bằng một.



Hình 7 Đồ thị hàm bước nhị phân

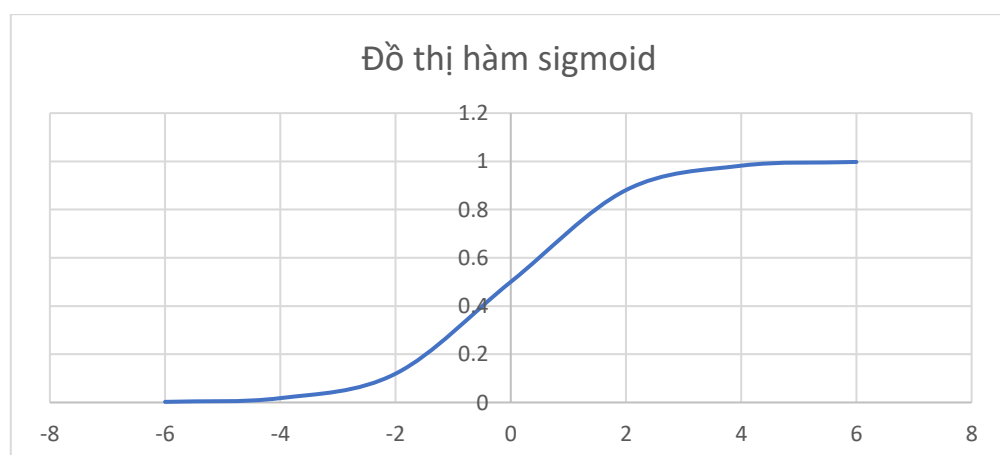
Hàm sigmoid (Sigmoid function (logsig)) [2]:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Quan hệ linear function với logistic function:

$$\hat{y} = \sigma(\omega_0 + \omega_1 * x_1^{(i)} + \omega_2 * x_2^{(i)}) = \frac{1}{1 + e^{-(\omega_0 + \omega_1 * x_1^{(i)} + \omega_2 * x_2^{(i)})}}$$

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện bởi thuật toán Lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng (0,1).



Hình 8 Đồ thị hàm Sigmoid

Hàm Sigmoid bão hòa và triệt tiêu gradient: Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương) thì giá trị hàm mất mát sẽ không thay đổi, gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với đơn vị đang xét sẽ gần như không được cập nhật (còn được gọi là *vanishing gradient*).

Đạo hàm logistic function:

Hàm logistic có đạo hàm tại mọi điểm và dễ tính toán:

$$\frac{d}{dx}f(x) = f(x)(1 - f(x))$$

Công thức hàm mất mát - Loss function:

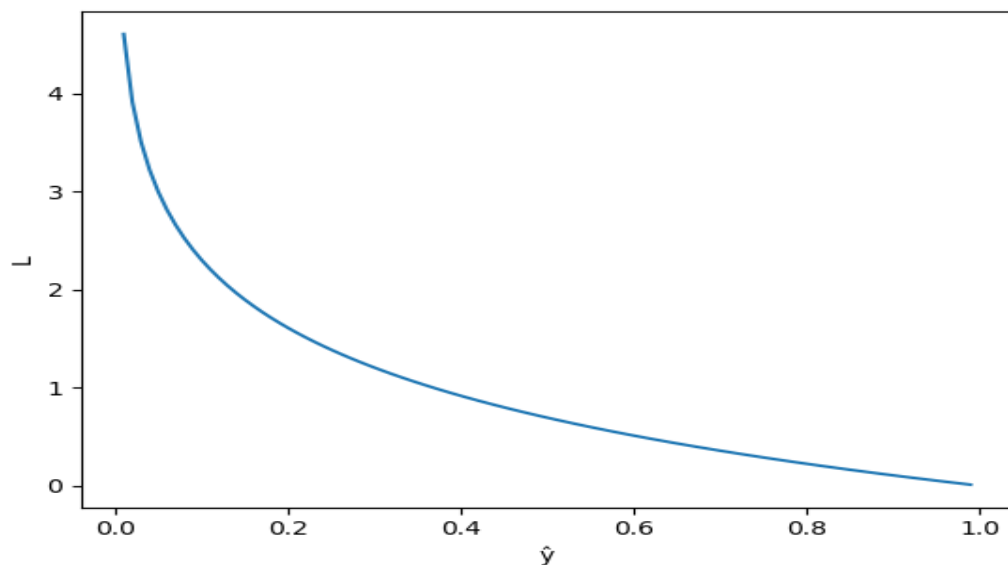
Đây là hàm đánh giá độ tốt của mô hình. Như vậy \hat{y} càng gần y càng tốt:

$$L = -(y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i))$$

(Chú ý đây là trường hợp đặc biệt của hàm mất mát *cross entropy*)

Đánh giá L nếu:

$$y_i = 1 \Rightarrow L = -\log(\hat{y}_i)$$

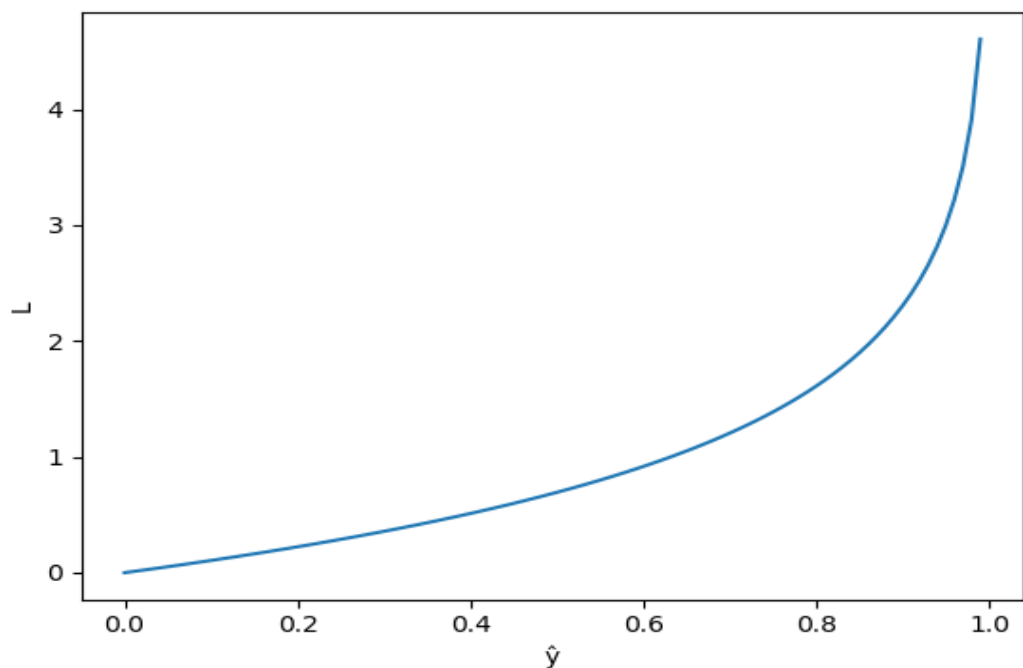


Hình 9 Đồ thị loss function – logistic (trường hợp: $y_i=1$)

- ✓ Hàm L giảm dần từ 0 đến 1.
- ✓ Khi mô hình dự đoán \hat{y}_i gần 1, tức giá trị dự đoán gần với giá trị thật y_i thì L nhỏ, xấp xỉ 0.
- ✓ Khi mô hình dự đoán \hat{y}_i gần 0, tức giá trị dự đoán ngược lại giá trị thật y_i thì L rất lớn.

Ngược lại nếu:

$$y_i = 0 \Rightarrow L = -\log(1 - \hat{y}_i)$$



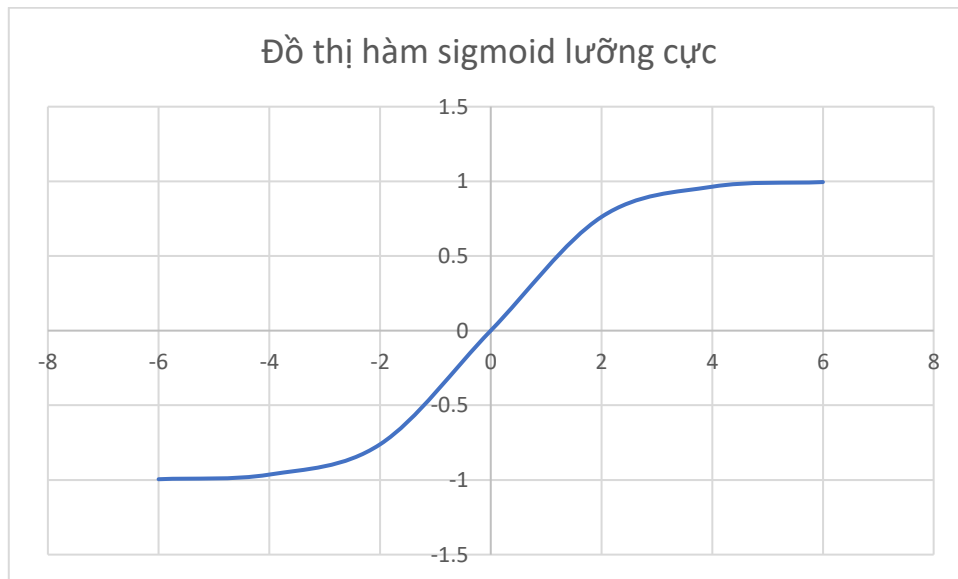
Hình 10 Đồ thị loss function – logistic (trường hợp $y_i=0$)

- ✓ Hàm L tăng dần từ 0 đến 1.
- ✓ Khi mô hình dự đoán \hat{y}_i gần 0, tức giá trị dự đoán gần với giá trị thật y_i thì L nhỏ, xấp xỉ 0.
- ✓ Khi mô hình dự đoán \hat{y}_i gần 1, tức giá trị dự đoán ngược lại giá trị thật y_i thì L rất lớn.

Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig)):

$$\sigma_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng $(-1,1)$.



Hình 11 Đồ thị hàm sigmoid lưỡng cực

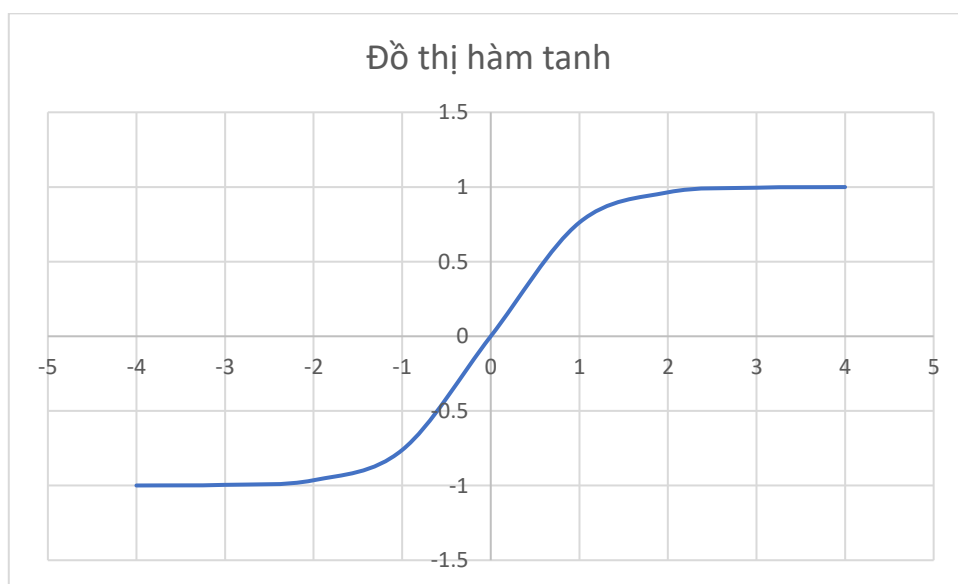
Các hàm chuyển của các đơn vị ẩn (hidden units) là cần thiết để biểu diễn sự phi tuyến vào trong mạng. Lý do là hợp thành của các hàm đồng nhất là một hàm đồng nhất. Mặc dù vậy nhưng nó mang tính chất phi tuyến (nghĩa là, khả năng biểu diễn các hàm phi tuyến) làm cho các mạng nhiều tầng có khả năng rất tốt trong biểu diễn các ánh xạ phi tuyến. Tuy nhiên, đối với luật học lan truyền ngược, hàm phải khả vi (differentiable) và sẽ có ích nếu như hàm được gắn trong một khoảng nào đó. Do vậy, hàm sigmoid là lựa chọn thông dụng nhất.

Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng $(0,1)$, hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng vẫn có ích, nó có thể cho ta các giá trị ra hay giá trị đích được căn trong một khoảng của hàm kích hoạt

đầu ra. Nhưng nếu các giá trị đích không được biết trước khoảng xác định thì hàm hay được sử dụng nhất là hàm đồng nhất (identity function). Nếu giá trị mong muốn là dương nhưng không biết cận trên thì nên sử dụng một hàm kích hoạt dạng mũ (exponential output activation function).

Hàm tanh (Tanh function):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Hình 12 Đồ thị hàm tanh

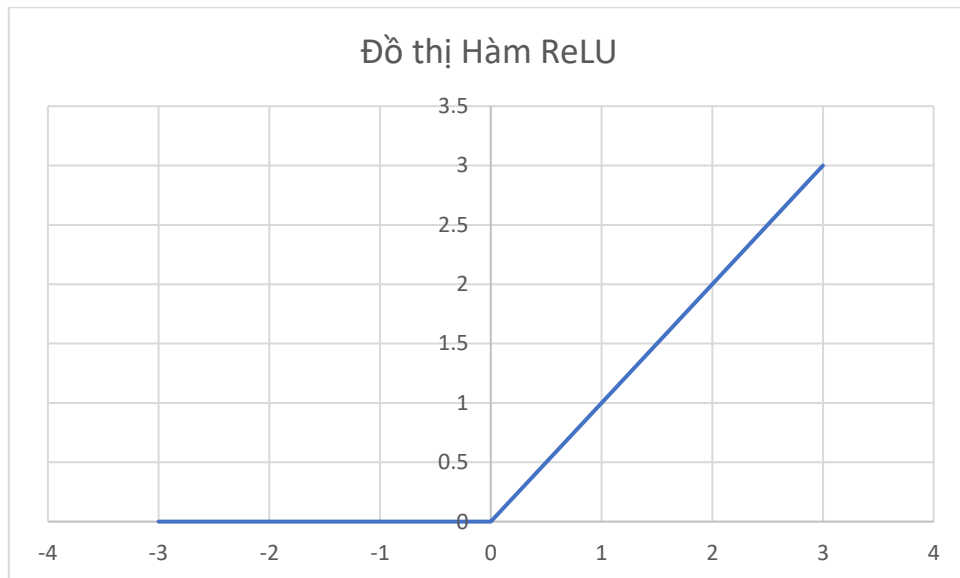
Hàm tanh nhận đầu vào là một số thực và chuyển thành một giá trị trong khoảng $(-1; 1)$, tương tự hàm Sigmoid lưỡng cực. Cũng như Sigmoid, hàm Tanh bị bão hoà ở hai đầu (gradient thay đổi rất ít ở hai đầu). Tuy nhiên hàm Tanh lại đối xứng qua 0 nên khắc phục được nhược điểm của Sigmoid (khó khăn trong việc hội tụ).

Hàm tanh còn được biểu diễn bằng hàm sigmoid như sau:

$$\tanh(x) = \sigma(2x) - 1$$

Hàm ReLU (ReLU function):

$$f(x) = \max(0, x)$$



Hình 13 Đồ Thị Hàm ReLU

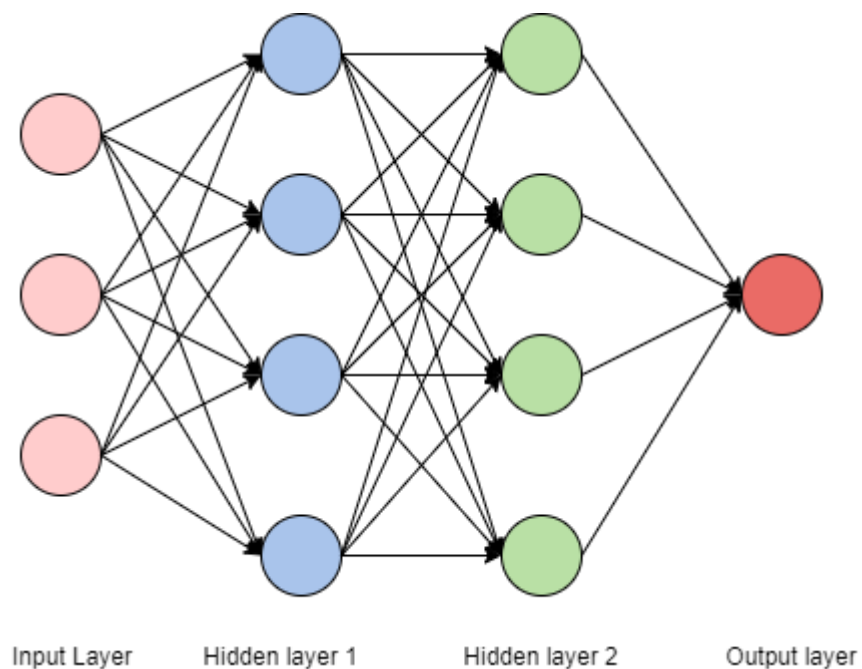
Hàm ReLU đang được sử dụng lọc các giá trị nhỏ hơn không.

- ✓ (+) ReLU có tốc độ hội tụ nhanh điều này do ReLU không bị bão hoà ở 2 đầu.
- ✓ (+) Tính toán nhanh hơn, công thức đơn giản tiết kiệm chi phí tính toán.
- ✓ (-) Tuy nhiên ReLU cũng có một nhược điểm: Với các nút có giá trị nhỏ hơn không, qua ReLU activation sẽ thành không, hiện tượng này gọi là “Dying ReLU”. Nếu các nút bị chuyển thành không thì sẽ không có ý nghĩa với bước linear activation ở lớp tiếp theo và các hệ số tương ứng từ nút này cũng không được cập nhật với gradient descent.
- ✓ (-) Khi tốc độ học - learning rate lớn, các trọng số (weights) có thể thay đổi theo cách làm tắt cả neural dừng việc cập nhật.

Phân loại theo cách truyền tín hiệu

Dựa vào cách truyền tín hiệu trong mạng neural ta có thể phân loại thành một số cách truyền tín hiệu:

Mạng lan truyền tiến – Feed-Forward Neural Networks (FFNNs): Gồm các mạng perceptron một lớp, mạng perceptron nhiều tầng và mạng RBF [2].



Hình 14 Mô hình mạng lan truyền tiến

Như bạn thấy thì tất cả các nút mạng được kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào tới tầng ra. Tức là mỗi nút ở một tầng nào đó sẽ nhận đầu vào là tất cả các nút ở tầng trước đó mà không suy luận ngược lại. Hay nói cách khác, việc lan truyền trong mạng neural network là lan truyền tiến (feed-forward):

$$z_i^{(l+1)} = \sum_{j=1}^{n^l} w_{ij}^{(l+1)} a_j^{(l)} + b_i^{(l+1)}$$

$$a_i^{(l+1)} = f(z_i^{(l+1)})$$

Trong đó, n^l số lượng nút ở tầng l tương ứng và $a_j^{(l)}$ là nút mạng thứ j của tầng l . Còn $w_{ij}^{(l+1)}$ là tham số trọng lượng của đầu vào $a_j^{(l)}$ đối với nút mạng thứ i của tầng $l+1$ và $b_i^{(l+1)}$ là độ lệch (*bias*) của nút mạng thứ i của tầng $l+1$. Đầu ra của nút mạng này được biểu diễn bằng $a_j^{(l+1)}$ ứng với hàm kích hoạt $f(z_i)$ tương ứng.

Riêng với tầng vào, thông thường $a^{(l)}$ cũng chính là các đầu vào X tương ứng của mạng.

Để tiện tính toán, ta coi $a_0^{(l)}$ là một đầu vào và $w_{i0}^{(l+1)} = b_i^{(l+1)}$ là tham số trọng lượng của đầu vào này. Lúc đó ta có thể viết lại công thức trên dưới dạng vector:

$$\begin{aligned} z_i^{(l+1)} &= w_i^{(l+1)} \cdot a^{(l)} \\ a_i^{(l+1)} &= f(z_i^{(l+1)}) \end{aligned}$$

Nếu nhóm các tham số của mỗi tầng thành một ma trận có các cột tương ứng với tham số mỗi nút mạng thì ta có thể tính toán cho toàn bộ các nút trong một tầng bằng vector:

$$\begin{aligned} z^{(l+1)} &= w^{(l+1)} \cdot a^{(l)} \\ a^{(l+1)} &= f(z^{(l+1)}) \end{aligned}$$

Mạng lan truyền ngược – backpropagation:

Mạng lan truyền ngược hay còn được gọi là mạng phản hồi (truy hồi) được sử dụng khá phổ biến trong các mô hình của AI hiện nay như DeepID-X hay CNNs và đã được ứng dụng trong thực tế như: dùng làm bộ nhớ địa chỉ hóa nội dung; dùng làm các bộ tối ưu ... [2]

Quá trình này diễn ra theo hai chiều, đầu tiên quá trình đào tạo diễn ra giống feed-forward (tuy nhiên không có bước cập nhật lại trọng số trong quá trình lan truyền). Cho đến khi đơn vị đầu ra được đưa ra tại output layer thì quá trình sẽ chuyển qua giai đoạn back-wards các trọng số đào tạo hệ số bias sẽ được cập nhật ngược trở lại thông qua việc làm tối thiểu tổng số lỗi – error. [2]

Lỗi e_o sẽ được tính toán theo công thức: $e_o = d_o - y_o$

Trong đó: d_o là đầu ra mong muốn, y_o là đầu ra thu được tại output layer.

Khái niệm về lỗi tổng thể của mạng (overall error of the network):

$$E = \frac{1}{2} \sum_{o \in O} e_o^2$$

Cập nhật trọng số w_{ij} :

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}}, \alpha \text{ là hệ số learning rate.}$$

$$\Rightarrow \Delta w_{ij} = -\alpha \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial s_j} \frac{\partial s_j}{\partial w_{ij}}, y_j \text{ đầu ra của đơn vị } j$$

$$s_j = z_j + b_j, z_j = \sum_i w_{ij} x_{ij} \text{ là trạng thái của đơn vị } j.$$

Trong đó x_{ij} là thông tin i truyền vào làm đầu vào cho j

i là đơn vị có trước j

Đạo hàm của lỗi theo đơn vị đầu ra:

$$\frac{\partial E}{\partial y_o} = -(d_o - y_o)$$

Đạo hàm của đơn vị đầu ra theo trạng thái tương ứng:

$$\frac{\partial y_o}{\partial s_o} = y_o(1 - y_o)$$

Đạo hàm của trạng thái theo trọng số kết nối đơn vị ẩn h với đơn vị đầu ra o :

$$\frac{\partial s_j}{\partial w_{ij}} = y_h, y_h \text{ là đầu ra tại hidden layer}$$

$$\text{Đặt } v_o \text{ là lỗi signal: } v_o = \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial s_o} \Rightarrow v_o = (d_o - y_o) y_o(1 - y_o)$$

Cập nhật trọng số giữa đơn vị ẩn h và đơn vị đầu ra o :

$$\Delta w_{ij} = \alpha v_o y_h$$

Bây giờ, đối với một đơn vị ẩn h , nếu chúng ta xem xét rằng khái niệm lỗi của nó có liên quan đến việc nó đã đóng góp bao nhiêu vào việc sản xuất một đầu ra bị lỗi, thì chúng ta có thể truyền lại lỗi từ các đơn vị đầu ra mà h gửi tín hiệu đến; chính xác hơn, đối với một đơn vị đầu vào i , chúng ta cần mở rộng phương trình [5]:

$$\Delta w_{ih} = -\alpha \frac{\partial E}{\partial w_{ih}}$$

Thành:

$$\Delta w_{ih} = -\alpha \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial s_o} \frac{\partial s_o}{\partial y_h} \frac{\partial y_h}{\partial s_h} \frac{\partial s_h}{\partial w_{ih}}$$

Với $o \in \text{Suc}(h)$, $\text{Suc}(h)$ là tập các đơn vị succed h (tập các đơn vị kết nối từ h). nghĩa là, các đơn vị được cung cấp với đầu ra của h là một phần của đầu vào của chúng. Bằng cách giải các đạo hàm riêng, chúng ta thu được:

$$\begin{aligned} \Delta w_{ih} &= -\alpha \sum_o (v_o w_{ho}) \frac{\partial y_h}{\partial s_h} \frac{\partial s_h}{\partial w_{ih}} \\ &= \alpha \sum_o (v_o w_{ho}) y_h (1 - y_h) y_i \end{aligned}$$

Tín hiệu lỗi - error signal của đơn vị ẩn h :

$$v_h = \sum_o (v_o w_{ho}) y_h (1 - y_h), o \in \text{Suc}(h)$$

Sau đó công thức tính trọng số thay đổi: $\Delta w_{ji} = \alpha v_h y_j$

Quá trình back-wards tính toán Δw_{ji} được thực hiện lặp lại cho đến khi tất cả đầu ra của mạng nằm trong phạm vi chấp nhận được hoặc điều kiện kết thúc được đưa ra.

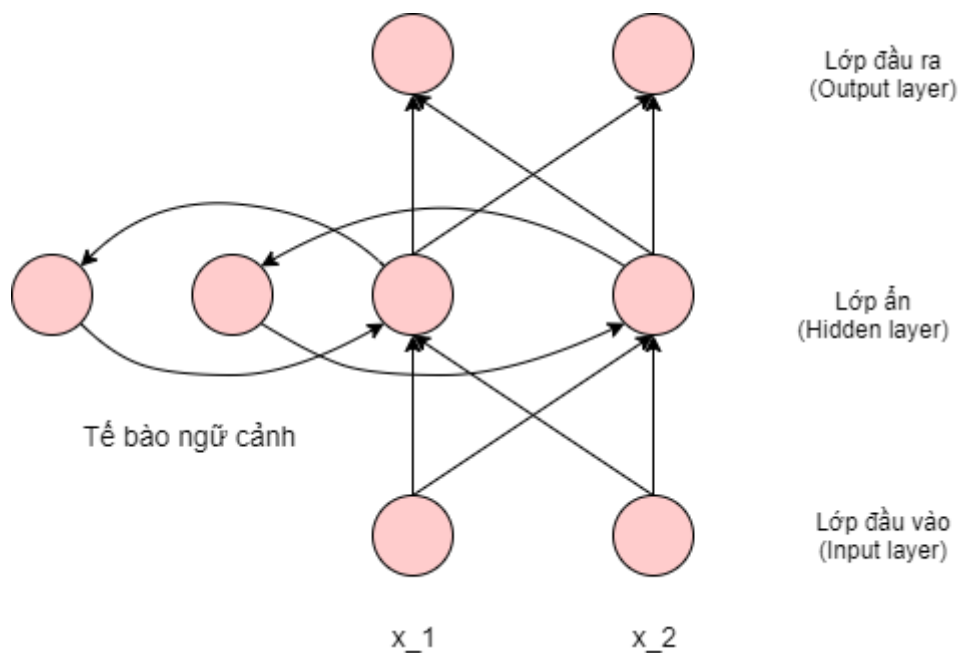
1.2 Mạng neural hồi tiếp (Recurrent Neural Network)

Mạng neural hồi tiếp-RNN là các hệ thống động (dynamic systems) [5] và chúng có một trạng thái bên trong mỗi bước của quá trình phân loại. Điều này là do

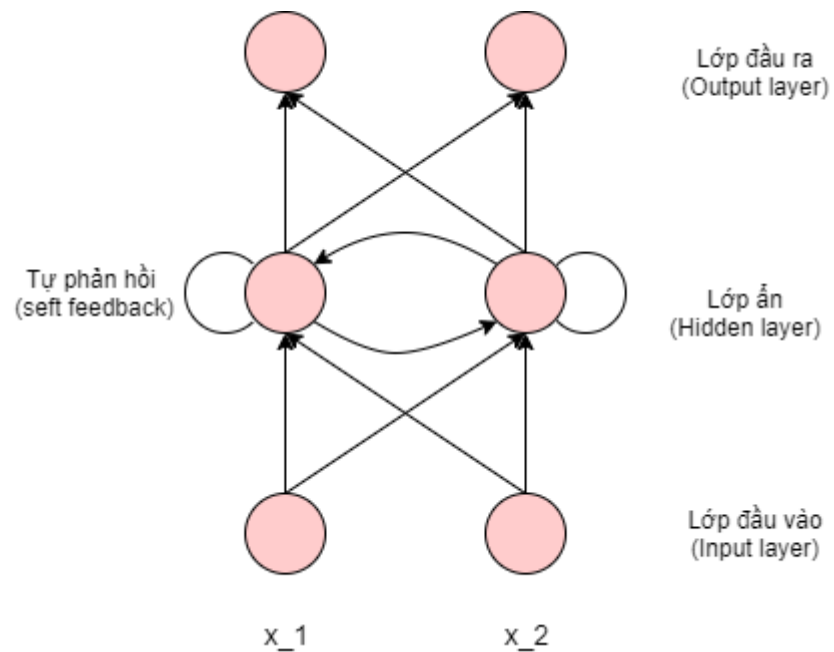
các kết nối giữa neural ở lớp cao hơn với neural ở lớp thấp hơn và các kết nối tự phản hồi chính nó (self-feedback). Các kết nối phản hồi (feedback) cho phép RNN truyền dữ liệu từ các sự kiện trước đó đến bước xử lý hiện tại. do đó RNN thường được sử dụng cho các bài toán liên quan đến chuỗi thời gian [6].

1.2.1 Cấu trúc mạng neural hồi tiếp

Có hai mô hình RNN đơn giản được đề xuất bởi Jodan và Elman. Trong đó mạng Elman tương tự như mạng neural có ba lớp, nhưng ngoài ra, đầu ra của các lớp ẩn sẽ được lưu lại ở ‘tế bào ngữ cảnh - context cell’. Đầu ra của một neural tế bào ngữ cảnh được cung cấp tuần hoàn trở lại neural ẩn cùng với tín hiệu gốc. Mỗi neural ẩn có một tế bào ngữ cảnh riêng và nhận đầu vào của cả lớp đầu vào và tế bào ngữ cảnh. Mạng elman có thể được đào tạo tương tự với backpropagation tiêu chuẩn với đầu ra từ tế bào ngữ cảnh được coi là đầu vào bổ sung.

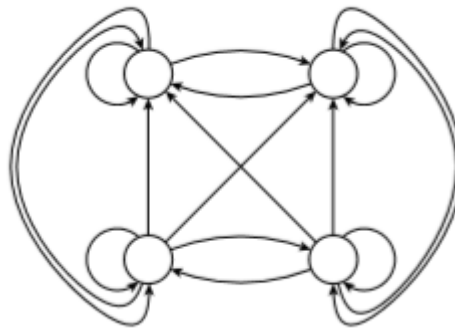


Hình 15 Mạng neural Elman



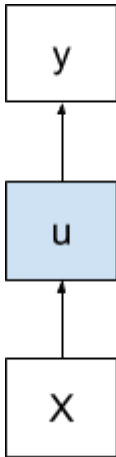
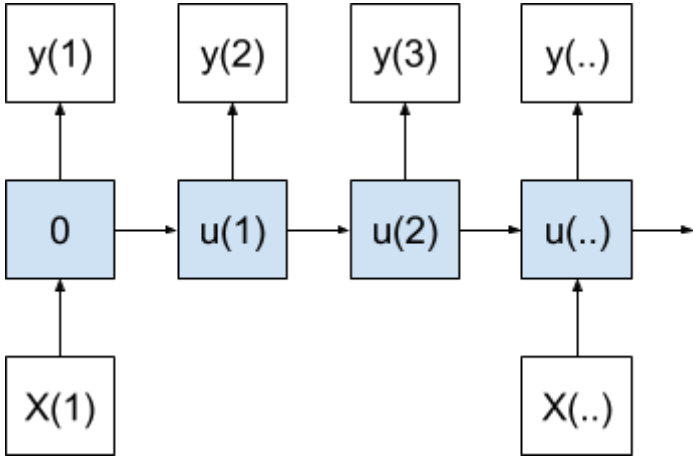
Hình 16 Mạng RNN với self-feedback ở hidden layer

Mạng Jordan có cấu trúc tương tự như mạng Elman, nhưng tế bào ngữ cảnh được thay thế bởi lớp đầu ra. Mạng Jordan có kết nối đầy đủ ở tất cả các nút mạng thay vì chỉ kết nối tại lớp ẩn.



Hình 17 Mạng RNN kết nối đầy đủ self-feedback

Phân loại Mạng neural hồi tiếp dựa theo đầu vào và đầu ra của mạng:

| Mạng neural hồi tiếp | Cấu trúc |
|--|--|
| One to one $T_i = T_o = 1$ |  |
| One to many $T_i = 1 \& T_o > 1$ |  |
| Many to one $T_i > 1 \& T_o = 1$ | |

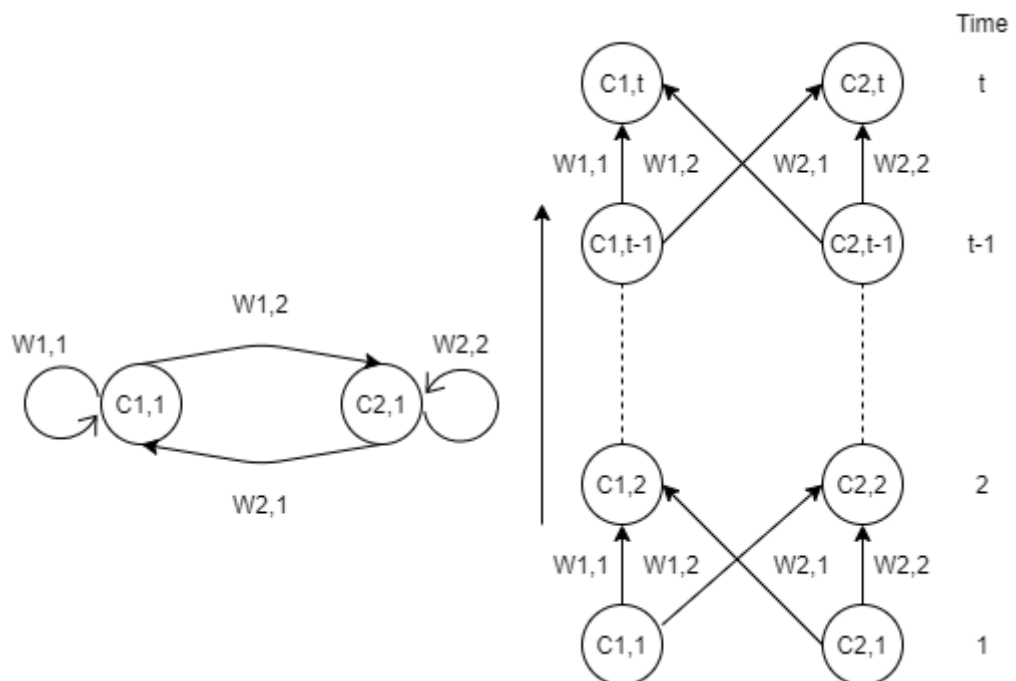
| | |
|--|--|
| | |
| Many to many $T_i = T_o$ Trường hợp: $T_i = T_o = 1$ Là trường hợp đặc biệt của many to many | |
| Many to many $T_i \neq T_o$ Trường hợp tổng quát nhất | |

Trong phạm vi đề tài loại mạng ‘Many to one’ được sử dụng để phân lớp tình cảm của văn bản.

1.2.2 Phương pháp đào tạo mạng Neural hồi tiếp

1.2.2.1 Phương pháp Backpropagation Throught time - BPTT

BPTT được sử dụng chủ yếu trong thực tế để đào tạo mạng neural hồi tiếp. Trong một giai đoạn huấn luyện mô hình sẽ có một mạng FFNNs dùng chung cho toàn quá trình huấn luyện. Sau quá trình huấn luyện với mạng FFNNs thì các trọng số huấn luyện mô hình được tính toán, tại bước này sẽ sử dụng thuật toán backpropagation để tính toán cập nhật trọng số cho tất cả các bước thời gian được tính toán thông qua lỗi.



Hình 18 Mạng neural hồi tiếp kết nối đầy đủ với một lớp có 2 neural theo bước thời gian (giai đoạn FFNNs)

Tính toán tín hiệu lỗi cho mỗi đơn vị tại tất cả các bước thời gian trong mỗi lần chạy. Chúng ta xem xét các bước thời gian rời rạc $1, 2, 3, \dots$ được gọi là đơn vị thời gian ký hiệu: τ . Mạng bắt đầu đào tạo tại thời điểm t' và chạy cho đến thời gian cuối cùng t . Khung thời gian từ t' đến t được gọi là epoch. Đặt U là tập hợp tất cả các đơn vị không phải đơn vị đầu vào (đơn vị sinh ra trong quá trình đào tạo) và đặt f_u là hàm băm của đơn vị u tại thời điểm τ được cho bởi công thức [2]:

$$y_u(\tau) = f_u(z_u(\tau))$$

Với trọng số đầu vào:

$$\begin{aligned} z_u(\tau + 1) &= \sum_l W_{[u,l]} X_{[l,u]}(\tau + 1) \text{ trong đó } l \in Pre(u) \\ &= \sum_v W_{[u,v]} y_v(\tau) + \sum_i W_{[u,i]} y_i(\tau + 1) \end{aligned}$$

Với $v \in U \cap Pre(u)$; $Pre(u)$ là tập các đơn vị có kết nối đến u , $i \in I$, I là tập các đơn vị đầu vào. Lưu ý rằng các đầu vào cho đơn vị u tại thời điểm $\tau + 1$ có 2 loại: đầu vào môi trường đến đúng lúc tại thời điểm $\tau + 1$ thông qua đơn vị đầu vào và đầu ra hồi tiếp từ tất cả các đơn vị được sinh ra tại thời điểm τ . Nếu mạng là kết nối đầy đủ (fully connected), thì $U \cap Pre(u) = u$. Đặt $T(\tau)$ là tập các đơn vị u thỏa mãn tại thời điểm τ các đơn vị đầu ra $y_u(\tau)$ phải khớp với ít nhất một giá trị đích $d_u(\tau)$. Hàm chi phí là tổng lỗi $E_{total}(t', t)$ cho từng epoch $t', t' + 1, \dots, t$, chúng ta cần tối thiểu hóa hàm chi phí [2].

$$E_{total}(t', t) = \sum_{\tau=t'}^t E(\tau)$$

Với $E(\tau)$ được xác định bởi:

$$E(\tau) = \frac{1}{2} \sum_{u \in U} (e_u(\tau))^2$$

Và lỗi $e_u(\tau)$ của đơn vị u tại thời điểm (τ) là:

$$e_u(\tau) = \begin{cases} d_u(\tau) - y_u(\tau) & , u \in T(\tau) \\ 0 & \end{cases}$$

Để điều chỉnh trọng số, chúng ta sử dụng tín hiệu lỗi $v_u(\tau)$ của đơn vị u tại thời điểm τ được xác định:

$$v_u(\tau) = \frac{\partial E(\tau)}{\partial z_u(\tau)}$$

$$= \begin{cases} f'_u(z_u(\tau))e_u(\tau), & \tau = t \\ f'_u(z_u(\tau)) \sum_{k \in U} W_{[k,u]} v_k(\tau + 1), & t' \leq \tau < t \end{cases}$$

Sau bước tính toán backpropagation được thực hiện đến thời điểm t' , chúng ta tính toán cập nhật trọng số $\Delta W_{[u,v]}$ trong mạng hồi tiếp bằng các tổng hợp các trọng số cập nhật tương ứng theo tất cả bước thời gian:

$$\Delta W_{[u,v]} = -\alpha \frac{\partial E(t', t)}{\partial W_{[u,v]}}$$

$$\frac{\partial E(t', t)}{\partial W_{[u,v]}} = \sum_{\tau=t'}^t v_u(\tau) \frac{\partial z_u(\tau)}{\partial W_{[u,v]}} = \sum_{\tau=t'}^t v_u(\tau) X_{[u,v]}(\tau)$$

Với $X_{[u,v]}(\tau)$ là đầu vào của đơn vị u đến từ đơn vị v

1.2.2.2 Học hồi tiếp thời gian thực - Real-Time Recurrent Learning (RTRL)

RTRL là một thuật toán cũng khá phổ biến để huấn luyện mạng RNN. Thuật toán không yêu cầu lỗi lan truyền trong mạng mà tất cả các thông tin cần thiết để tính toán gradient được thu thập từ luồng dữ liệu đầu vào được đưa vào mạng. Điều này làm cho mạng bị lỗi thời sau một thời gian đào tạo nhất định. RTRL tốn một khoảng chi phí tính toán đáng kể cho mỗi chu kỳ cập nhật và thông tin được lưu trữ là không cục bộ. Vì vậy chúng ta cần một đại lượng được gọi là độ nhạy của đầu ra để đo lường lỗi. Bộ nhớ được yêu cầu trong quá trình đào tạo chỉ phụ thuộc vào kích thước của mạng mà không phải phụ thuộc vào kích thước của đầu vào.

Đặt $v \in I \cup U$; $u, k \in U$; $t' \leq \tau \leq t$. Không giống như BPTT, đối với RTRL, giả sử nhãn $d_k(\tau)$ tồn tại tại mỗi thời điểm τ (được coi là một thuật toán online) cho mọi đơn vị k . Bởi vậy, mục tiêu tối thiểu lỗi chung của mạng, tại thời điểm τ được cho bởi công thức:

$$E(\tau) = \frac{1}{2} \sum_{k \in U} (d_k(\tau) - y_k(\tau))^2$$

Gradient của tổng số lỗi là tổng của các gradient của các bước thời gian trước đó và bước thời gian hiện tại:

$$\nabla_W E_{total}(t', t+1) = \nabla_W E_{total}(t', t) + \nabla_W E(t+1)$$

Trong khi trình bày một mạng của chuỗi thời gian, chúng ta cần tích lũy gradient tại mỗi bước thời gian. Do đó chúng ta có thể theo dõi sự thay đổi của các trọng số $\Delta W_{[u,v]}(\tau)$. Sự thay đổi trọng số tổng thể cho toàn mạng $W_{[u,v]}$ được tính toán:

$$W_{[u,v]} = \sum_{\tau=t'+1}^t \Delta W_{[u,v]}(\tau)$$

Trong đó:

$$\begin{aligned} \Delta W_{[u,v]} &= -\alpha \frac{\partial E(\tau)}{\partial W_{[u,v]}} = -\alpha \frac{\partial E(\tau)}{\partial y_k(\tau)} \frac{\partial y_k(\tau)}{\partial W_{[u,v]}} \\ &= -\alpha \sum_{k \in U} (d_k(\tau) - y_k(\tau)) \frac{\partial y_k(\tau)}{\partial W_{[u,v]}} \end{aligned}$$

$e_k(\tau) = d_k(\tau) - y_k(\tau)$ luôn được xác định, do đó chúng ta chỉ cần tính giá trị của biểu thức:

$$p_{uv}^k(\tau) = \frac{\partial y_k(\tau)}{\partial W_{[u,v]}}$$

Khi xem xét hiệu quả của sự thay đổi trọng số trên toàn bộ mạng từ thời điểm t' đến t , ta sẽ đo lường độ nhạy của đầu ra của đơn vị k tại thời điểm τ đến một thay đổi nhỏ của $W_{[u,v]}$. Chú ý trọng số $W_{[u,v]}$ không có kết nối đến đơn vị k , do đó thuật toán là không cục bộ, do đó những thay đổi cục bộ trong mạng có thể diễn ra bất kỳ ở đâu trong mạng.

Trong RTRL, thông tin về gradient được lan truyền tiến (forward-propagates).
Đầu ra tại thời điểm $t + 1$:

$$y_k(t + 1) = f_k(z_k(t + 1))$$

Với

$$\begin{aligned} z_k(t + 1) &= \sum_l W_{[k,l]} X_{[k,l]}(t + 1) \text{ trong đó } l \in \text{Pre}(k) \\ &= \sum_v W_{[k,v]} y_v(t) + \sum_i W_{[k,i]} y_i(t + 1) \end{aligned}$$

Tính toán kết quả cho toàn bộ các bước thời gian $\geq t + 1$ với:

$$\begin{aligned} p_{uv}^k(t + 1) &= \frac{\partial y_k(t + 1)}{\partial W_{[u,v]}} = \frac{\partial}{\partial W_{[u,v]}} \left[f_k \left(\sum_l W_{[k,l]} X_{[k,l]}(t + 1) \right) \right] \\ &= f'_k(z_k(t + 1)) \left[\frac{\partial}{\partial W_{[u,v]}} \left(\sum_l W_{[k,l]} X_{[k,l]}(t + 1) \right) \right] \\ &= f'_k(z_k(t + 1)) \left[\left(\sum_l \frac{\partial W_{[k,l]}}{\partial W_{[u,v]}} X_{[k,l]}(t + 1) \right) + \left(\sum_l \frac{\partial X_{[k,l]}(t + 1)}{\partial W_{[u,v]}} W_{[k,l]} \right) \right] \\ &= f'_k(z_k(t + 1)) \left[(\delta_{uk} X_{[k,l]}(t + 1)) \right. \\ &\quad \left. + \left(\sum_l \frac{\partial y_l(t)}{\partial W_{[u,v]}} W_{[k,l]} + \sum_{i \in I} \frac{\partial y_i(t + 1)}{\partial W_{[u,v]}} W_{[k,i]} \right) \right] \\ &= f'_k(z_k(t + 1)) \left[(\delta_{uk} X_{[k,l]}(t + 1)) + \sum_l p_{uv}^l(t) W_{[k,l]} \right] \end{aligned}$$

Trong đó:

$$\delta_{uk} = \begin{cases} 1, & u = k \\ 0, & u \neq k \end{cases}$$

Giả sử trạng thái ban đầu của mạng không phụ thuộc vào hàm trọng số, đạo hàm cho bước thời gian đầu tiên:

$$p_{uv}^k(t') = \frac{\partial y_k(t')}{\partial W_{[u,v]}} = 0$$

Như vậy ta có thể thấy $p_{uv}^k(t+1)$ có thể được tính dựa trên $p_{uv}^k(t)$. Như vậy đối với RTRL, chúng ta có thể học đồng thời khi chúng ta nhận thêm giá trị đầu vào mới (trong thời gian thực) mà không cần phải thực hiện BPTT. Biết giá trị $p_{uv}^k(t')$ chúng ta có thể tính toán đệ quy các giá trị $p_{uv}^k(\tau)$ bước thời gian tiếp theo. Lưu ý trong công thức tính $p_{uv}^k(\tau)$ thì giá trị trọng số $W_{[u,v]}$ là giá trị tại thời điểm t' chứ không phải giá trị trọng số trong khoảng thời gian t' đến t' . Kết hợp với giá trị vector $e(\tau)$ tại bước thời gian đó chúng ta có thể tính chính xác giá trị $\nabla WE(\tau)$. Cuối cùng là cập nhật trọng số $W_{[u,v]}$. [7] [8]

CHƯƠNG 2: MỘT SỐ MÔ HÌNH ĐỀ XUẤT

2.1 Giới thiệu mô hình hóa bài toán

Bài toán phân tích cảm xúc tiếng việt trình bày trong báo cáo này được mô hình hóa theo mô hình dưới đây:



Hình 19 Mô hình bài toán phân tích cảm xúc tiếng việt

Đầu vào của bài toán sẽ là một đoạn văn bản tiếng việt bất kỳ, sau đó cho qua mô hình SAV-Sentiment Analysis Vietnamese sẽ thu được đầu ra là tính chất của đoạn văn bản ở một trong hai loại là tích cực-Positive hoặc tiêu cực-Negative.

Trong đó mô hình phân tích cảm xúc tiếng Việt (SAV) được xây dựng dựa trên các thuật toán deep learning (Vd: Naïve Bayes, Support Vector Machine (SVM), Long short term memory + Convolution neural network (CNN+LSTM), Word2vec + BiLSTM, Bidirectional Encoder Representations from Transformers (BERT) ...).

2.2 Thiết kế dữ liệu huấn luyện mạng

Đây là bộ dữ liệu nhận xét (review) đã được xử lý tách từ và được gán nhãn với 2 loại cảm xúc tích cực và tiêu cực.

Link dataset: (<https://github.com/congngghia0609/ntc-scv>)

Data được chia làm 2 phần: train_data (40000 mẫu), test_data (10000). Tuy nhiên em sẽ tách tập train_data ra thành 2 tập mới là: train_data (30000), và val_data (10000). (train_data: dùng để huấn luyện đào tạo xác minh qua mỗi epoch huấn luyện

mô hình. `val_data`: được sử dụng để điều chỉnh các tham số của mô hình huấn luyện. `test_data`: dùng để đánh giá hiệu suất hoạt động của mô hình).

`Train_data`: gồm 2 file `train_pos.txt` (tập bình luận tích cực), `train_neg.txt` (tập bình luận tiêu cực). tương đương với mỗi bình luận thuộc `train_pos.txt` được gán nhãn tích cực và mỗi bình luận thuộc `train_neg.txt` được gán nhãn tiêu cực.

`Test_data`, `val_data`: cũng tương tự được chia là `test_pos.txt`, `test_neg.txt`, `val_pos.txt`, `val_neg.txt`

Ví dụ `train_neg.txt`:

- deal la k đọc review nhieu khen mua thử đi tầm h quán vắng món quá dở giá_cả k đĩ bạn_trai đi xog đau bụng suốt đêm keu món nghêu cát chết nước_chấm k ngon ốc k tươi phục_vụ k thân_thiện đc quán sạch_sẽ sợ lun hên xài deal thay tiếc

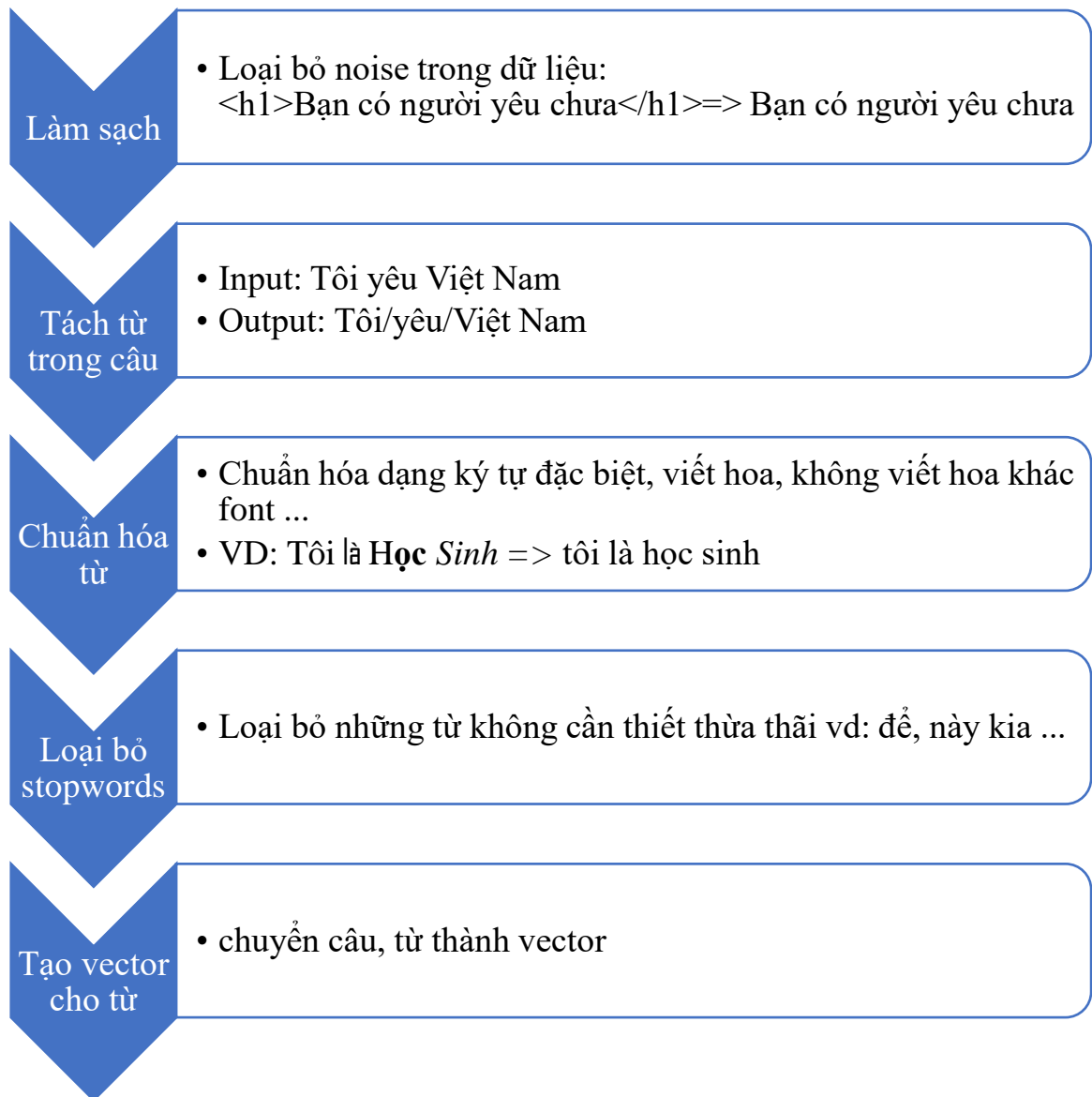
Ví dụ `train_pos.txt`:

- xe_đẩy com_chiên nằm đầu đường khu dân_cư metro chạy vô xe thứ_hai mua com_lắm com_chiên mềm nóng_ăn trứng chiên lap_xưởng thịt heo chà_lắm dưa_leo cà_chua ăn_không ngán xong hợp com bao no trưa giá chủ_hộ buổi_sáng đông ghé mua ngon rẻ no đông lắm nói_chuyện vui_về lịch_sự

=> Tất cả các dữ liệu đều được xử lý tách từ và loại bỏ stopwords (nên chỉ cần đưa xử lý nhúng từ và đưa vào huấn luyện.

Tuy nhiên sau khi huấn luyện thì các dữ liệu được đưa vào để sử dụng mô hình đều là các dữ liệu thô, được crawl trên các twitter, facebook, blog, ... Nên cần tiền xử lý dữ liệu trước khi đưa vào mô hình để dự đoán, ...

Tiền xử lý dữ liệu:



2.3 Mô hình đề xuất huấn luyện mạng và xây dựng mô hình

2.3.1 Mô hình Naïve Bayes

Naïve Bayes Classifier:

Xét bài toán classification, phân lớp tập dạng $\Omega = \{X \in R^d \mid d \in N\}$ thành C lớp với $C = \{C_i, i = \overline{1, c}\}$ là một phân hoạch của tập dạng Ω . Khi đó cho điểm dữ liệu $X \in \Omega$, việc tính xác suất để điểm dữ liệu rơi vào lớp C_i dựa vào công thức bayes [9]:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Trong đó: $P(X) = \sum_{i=1}^c P(X|C_i)P(C_i)$ (Công thức xác suất đầy đủ).

Biểu thức trên nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi lớp. Từ đó có thể giúp xác định lớp của điểm dữ liệu đó bằng cách chọn ra lớp có xác suất cao nhất:

$$\begin{aligned} c &= \operatorname{argmax}(P(C_i|X)), i = \overline{1, c} \\ &= \operatorname{argmax}\left(\frac{P(X|C_i)P(C_i)}{P(X)}\right) \\ &= \operatorname{argmax}(P(X|C_i)P(C_i)) \end{aligned}$$

(Vì $P(X)$ là một hằng số không phụ thuộc vào C_i nên trong quá trình tính toán có thể bỏ qua $P(X)$).

- ✓ $P(C_i)$ là tần suất điểm dữ liệu trong tập huấn luyện rơi vào lớp C_i , hay còn được tính bằng tỉ lệ số điểm dữ liệu trong tập huấn luyện rơi vào lớp C_i chia cho tổng số điểm dữ liệu trong tập huấn luyện.
- ✓ Việc tính toán $P(X|C_i)$ thường khá là phức tạp nên để đơn giản hóa việc tính toán người ta thường coi các thuộc tính của vector X là độc lập tuyến tính khi đó ta có công thức:

$$P(X|C_i) = \prod_{k=1}^d P(x_k|C_i) = P(x_1|C_i)P(x_2|C_i) \dots P(x_d|C_i)$$

Việc tính toán $P(x_k|C_i), k = \overline{1, d}; i = \overline{1, c}$ phụ thuộc vào loại dữ liệu, thông thường sẽ có ba quy tắc phân lớp thường dùng phổ biến: Gaussian Naïve Bayes, Multinomial Naïve Bayes, và Bernoulli Naïve Bayes.

Quy tắc phân lớp Gaussian Naive Bayes [10]:

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(X|C_i) = g(x_k, \mu_{c_i}, \sigma_{c_i})$$

Bộ tham số $\theta = (\mu_{c_i}, \sigma_{c_i})$ được xác định là kỳ vọng μ_{c_i} và phương sai $\sigma_{c_i}^2$ của các thuộc tính x_k tương ứng.

Tuy nhiên trong thư viện sklearn bộ tham số $\theta = (\mu_{c_i}, \sigma_{c_i})$ được tính toán dựa trên công thức:

$$\theta = (\mu_{c_i}, \sigma_{c_i}) = \underset{i}{\operatorname{argmax}} \prod_{k=1}^d P(x_k^{(c_i)} | \mu_{c_i}, \sigma_{c_i})$$

$$i = \overline{1, c}$$

Quy tắc phân lớp Multinomial Naïve Bayes [10]:

Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà vector đặc trưng được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ k trong mỗi vector chính là số lần từ thứ k xuất hiện trong văn bản đó.

Khi đó, $P(x_k|C_i)$ tỉ lệ với tần suất từ thứ k (hay feature thứ k cho trường hợp tổng quát) xuất hiện trong các văn bản của lớp C_i . Giá trị này có thể được tính bằng cách:

$$\lambda_{c_i}^k = P(x_k|C_i) = \frac{N_{c_i}^k}{N_{c_i}}$$

Trong đó:

- ✓ $N_{c_i}^k$ là tổng số từ thứ k xuất hiện trong lớp C_i , nó còn được tính bằng tổng các thành phần thứ k của các vector đặc trưng ứng với lớp C_i .
- ✓ N_{c_i} là tổng số từ xuất hiện trong lớp C_i kể cả lặp.

$$N_{c_i} = \sum_{k=1}^d N_{c_i}^k \Rightarrow \sum_{k=1}^d \lambda_{c_i}^k = 1$$

Cách tính này có một hạn chế là nếu có một từ mới chưa bao giờ xuất hiện trong lớp C_i thì giá trị $\lambda_{c_i}^k = 0$, điều này dẫn đến $P(X|C_i) = 0$ bất kể các giá trị còn lại có lớn thế nào. Việc này sẽ dẫn đến kết quả không chính xác, để giải quyết việc này, một kỹ thuật được gọi là *Laplace smoothing* được áp dụng:

$$\widehat{\lambda}_{c_i}^k = \frac{N_{c_i}^k + \alpha}{N_{c_i} + d\alpha}$$

Với α là một số dương tùy ý, thông thường $\alpha = 1$, để tránh trường hợp $N_{c_i}^k = 0$, và mẫu số được cộng thêm một giá trị $d\alpha$ để đảm bảo $\sum_{k=1}^d \widehat{\lambda}_{c_i}^k = 1$. Như vậy mỗi lớp C_i sẽ được mô tả bằng bộ các số dương có tổng bằng 1:

$$\widehat{\lambda}_{c_i} = \{\widehat{\lambda}_{c_i}^1, \widehat{\lambda}_{c_i}^2, \dots, \widehat{\lambda}_{c_i}^d\}$$

Quy tắc phân lớp Bernoulli Naïve Bayes [10]:

Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary - bằng 0 hoặc 1. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không. Khi đó $P(x_k|C_i)$ được tính bằng công thức:

$$P(x_k|C_i) = P(k|C_i)^{x_k} (1 - P(k|C_i))^{1-x_k}$$

Trong đó $P(k|C_i)$ xác suất từ thứ k xuất hiện trong các văn bản của lớp C_i .

Xây dựng mô hình Naïve Bayes

❖ Túi từ - Bag of Words (BOW)

Mô hình thường dùng trong các tác vụ phân lớp văn bản (Text Classification). Thông tin sẽ được biểu diễn thành tập các từ đi kèm với tần xuất xuất hiện của mỗi từ này trong văn bản. Bag of Words được dùng như đặc trưng để huấn luyện cho bài toán phân lớp.

| Đoạn văn | Phân tích | |
|--|-----------|-----|
| Đại học bách khoa Hà Nội là trường đại học hàng đầu Việt Nam về khoa học kỹ thuật. | Đại | 2 |
| | Học | 3 |
| | Bách | 1 |
| | ... | ... |
| | Thuật | 1 |

Bảng mô tả túi từ

Các bước xây dựng mô hình túi từ:

✓ Bước 1: Thu thập dữ liệu và xử lý dữ liệu

Xây dựng một mảng chứa toàn bộ dữ liệu văn bản trong tập huấn luyện. Các dữ liệu thu thập được sẽ được đem tiền xử lý (các bước tiền xử lý phần thiết kế dữ liệu huấn luyện mạng)

✓ Bước 2: Xây dựng bộ từ điển

Trong bước này ta sẽ xây dựng một bộ danh sách các từ có xuất hiện trong mảng thu thập ở bước 1. Trong đó các từ xuất hiện trong từ điển là duy nhất.

✓ Bước 3: Tạo vector cho câu

Bước tiếp theo là chấm điểm các từ trong mỗi tài liệu. Mục tiêu là biến mỗi tài liệu văn bản thành một vector mà chúng ta có thể sử dụng làm đầu vào cho một mô hình học máy. Sử dụng thứ tự các từ trong bộ từ điển, chúng ta có thể chuyển đoạn văn bản thành một vector nhị phân (có n thành phần với mỗi thành phần có thể nhận giá trị 0 nếu từ không

xuất hiện trong văn bản, hoặc nhận giá trị 1 nếu từ đó có xuất hiện trong văn bản) hoặc chuyển thành một vector có n thành phần với các thành phần là biểu thị tần suất xuất hiện của từ.

❖ Tf-idf

✓ Tf

$$tf_i = \frac{n_i}{N_i}$$

Trong đó: $i = \overline{1, n}$,

n là tổng số văn bản trong tập dữ liệu huấn luyện,

n_i tần suất xuất hiện của từ trong văn bản i ,

N_i tổng số từ trong văn bản i .

✓ Idf

$$idf_i = \log \frac{n}{d}$$

Trong đó: $i = \overline{1, n}$,

n là tổng số văn bản trong tập dữ liệu huấn luyện,

d số văn bản có sự xuất hiện của từ.

✓ Tf-idf

Tf-idf thể hiện trọng số của mỗi từ theo ngữ cảnh văn bản. tf-idf sẽ có giá trị tăng tỷ lệ thuận với số lần xuất hiện của từ trong văn bản và số văn bản có chứa từ đó trên toàn bộ tập tài liệu.

$$tfidf_i = tf_i \times idf_i$$

Xây dựng mô hình Naïve Bayes ta sẽ thực hiện vector hóa các câu đánh giá, bình luận dựa trên kỹ thuật BOW + Tf-idf và mỗi câu tương ứng với mỗi điểm dữ liệu là vector có d thành phần, tương ứng với số từ trong bộ từ điển. Trong bài báo cáo này, sử dụng quy tắc phân lớp Multinomial Naïve Bayes để tính toán phân lớp điểm dữ liệu.

| Đánh giá mô hình: | | Dự đoán | |
|-------------------|------------|--------------|-----------------|
| | | Âm tính | Dương tính |
| Thực tế | Âm tính | Âm tính thật | Dương tính giả |
| | Dương tính | Âm tính giả | Dương tính thật |

$$Precision = \frac{Dương\ tính\ thật}{Dương\ tính\ thật + Dương\ tính\ giả}$$

$$Recall = \frac{Dương\ tính\ thật}{Dương\ tính\ thật + Âm\ tính\ giả}$$

$$Chỉ\ số\ F1 = 2 \frac{Precision * Recall}{Precision + Recall}$$

Kết quả huấn luyện mô hình:

| | |
|---------------------|-------------------|
| Độ chính xác | 84.19 |
| Precision | 82.89397729459303 |
| Recall (macro) | 84.19000000000001 |
| Recall (micro) | 84.19 |
| Chỉ số - F1 (macro) | 84.18386191497058 |
| Chỉ số - F1 (micro) | 84.19000000000001 |

2.3.2 Mô hình Support Vector Machine (SVM)

2.3.2.1 Khoảng cách từ một điểm tới một siêu phẳng

Trong không gian 2 chiều, khoảng cách từ một điểm có tọa độ (x_0, y_0) tới đường thẳng có phương trình $\omega_0 + \omega_1 x_1 + \omega_2 x_2 = 0$ được xác định bởi:

$$\frac{|\omega_0 + \omega_1 x_0 + \omega_2 y_0|}{\sqrt{\omega_1^2 + \omega_2^2}}$$

Trong không gian 3 chiều, khoảng cách từ một điểm có tọa độ (x_0, y_0, z_0) tới một mặt phẳng có phương trình $\omega_0 + \omega_1 x_1 + \omega_2 x_2 + \omega_3 x_3 = 0$ được xác định bởi:

$$\frac{|\omega_0 + \omega_1 x_0 + \omega_2 y_0 + \omega_3 z_0|}{\sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2}}$$

Hơn nữa, nếu bỏ dấu trị tuyệt đối ở tử số, chúng ta có thể xác định được điểm đó nằm về phía nào của đường thẳng hay mặt phẳng đang xét. Những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu dương nằm về cùng một phía (có thể gọi đây là *phía dương*), những điểm làm cho biểu thức trong dấu giá trị tuyệt đối mang dấu âm nằm về phía còn lại (có thể gọi đây là *phía âm*). Những điểm nằm trên *đường thẳng* hoặc *mặt phẳng* sẽ làm cho tử số có giá trị bằng 0, tức khoảng cách bằng 0.

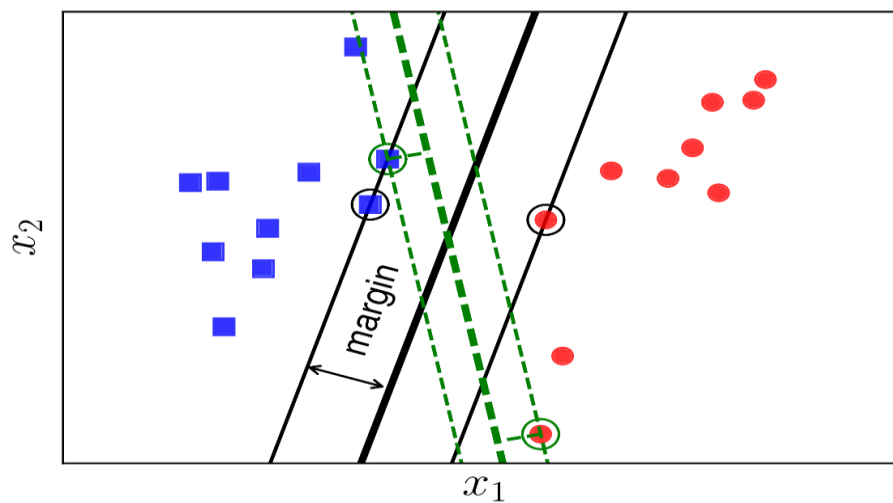
Tổng quát, với không gian n chiều ta có khoảng cách từ một điểm (vector) có tọa độ X_0 đến siêu phẳng có phương trình $\omega_0 + W^T X = 0$ được xác định bởi:

$$\frac{|\omega_0 + W^T X_0|}{\|W\|_2}$$

Trong đó: $\|W\|_2 = \sqrt{\sum_{i=1}^n \omega_i^2}$ là chuẩn của W theo trọng hàm $u = 2$

2.3.2.2 Mô hình phân chia hai lớp sử dụng Support Vector Machine

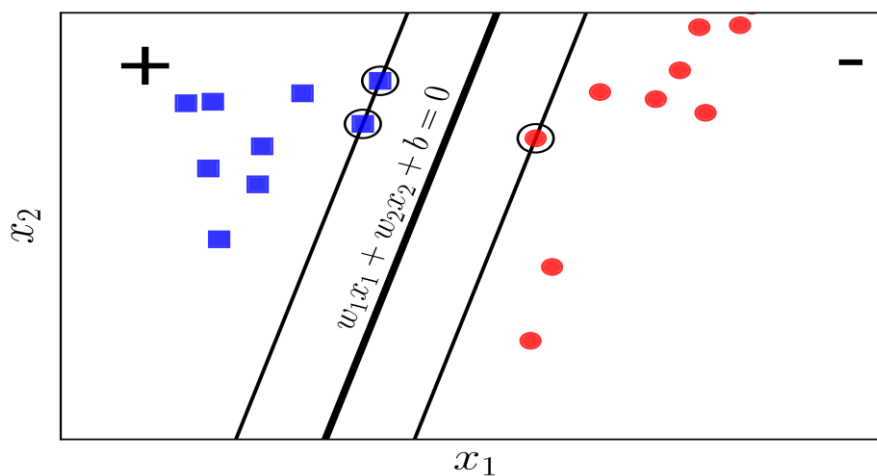
Giả sử rằng có hai lớp khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai lớp này là phân biệt tuyến tính, tức tồn tại một siêu phẳng phân chia chính xác hai lớp đó. Việc tìm một siêu mặt phẳng phân chia hai lớp đó, sao cho tất cả các điểm thuộc một lớp nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc lớp còn lại. (Hoặc một lớp nằm trọn vẹn về phía dương của siêu phẳng hoặc nằm trọn vẹn về phía âm của siêu phẳng).



Hình 20 Lề của hai classes là bằng nhau và lớn nhất có thể.

Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi lớp (các điểm được khoanh tròn) tới đường phân chia là như nhau, khoảng cách như nhau này được gọi là *Lề (Margin)*. Lề của cả hai lớp càng lớn thì việc phân lớp càng tốt (càng rạch ròi).

Bài toán tối ưu trong *Support Vector Machine (SVM)* giúp giải quyết vấn đề tìm ra đường phân chia sao cho lề của hai lớp là lớn nhất [11].



Hình 21 Phân tích bài toán SVM.

Các cặp dữ liệu trong tập huấn luyện là $(X_1, y_1); (X_2, y_2); \dots; (X_n, y_n)$; với vector (X_i) thể hiện là điểm dữ liệu trong tập huấn luyện và (y_i) thể hiện nhãn của điểm dữ liệu đó.

Khoảng cách từ điểm $(X_i, y_i), \forall i \in [1, n]$ tới mặt phân chia có phương trình $\omega_0 + W^T X = 0$ được xác định bởi:

$$\frac{y_i(\omega_0 + W^T X_i)}{\|W\|_2}$$

Ta có y_i luôn cùng dấu với phía của X_i , như vậy y_i luôn cùng dấu với $\omega_0 + W^T X_i$ và giá trị biểu thức $y_i(\omega_0 + W^T X_i) > 0$. [12] [11]

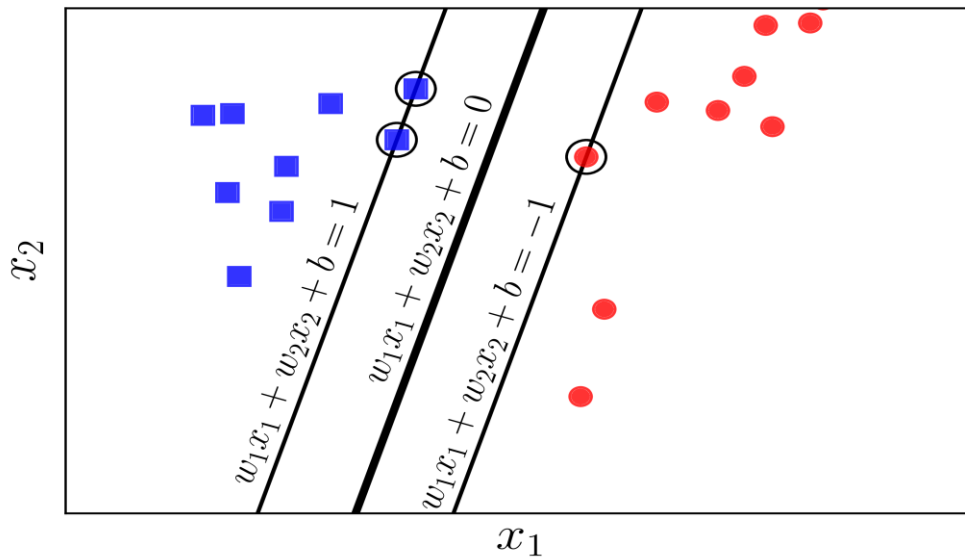
Với mặt phân chia như trên, lề được tính bằng khoảng cách gần nhất từ một điểm dữ liệu tới mặt phân chia đó (điểm dữ liệu được chọn là điểm bất kỳ thuộc một trong hai lớp):

$$margin = \min_i \frac{y_i(\omega_0 + W^T X_i)}{\|W\|_2}, \forall i \in [1, n]$$

Bài toán tối ưu trong SVM chính là bài toán tìm bộ giá trị tham số (W, ω_0) sao cho lề đạt giá trị lớn nhất:

$$\begin{aligned} (W, \omega_0) &= \arg \max_{W, \omega_0} \left\{ \min_i \frac{y_i(\omega_0 + W^T X_i)}{\|W\|_2} \right\} \\ &= \arg \max_{W, \omega_0} \left\{ \frac{1}{\|W\|_2} \min_i y_i(\omega_0 + W^T X_i) \right\} \end{aligned}$$

Giả sử $y_i(\omega_0 + W^T X_i) = 1$ khi đó ta có:



Hình 22 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn.

Như vậy $\forall i \in [1, n]$ ta có: $y_i(\omega_0 + W^T X_i) \geq 1$ Bài toán trở thành bài toán quy hoạch tuyến tính được phát biểu như sau:

$$(W, \omega_0) = \arg \max_{(W, \omega_0)} \frac{1}{\|W\|_2}$$

$$v.đ.k: y_i(\omega_0 + W^T X_i) \geq 1, \forall i \in [1, n]$$

Kết quả huấn luyện mô hình:

| | |
|---------------------|-------------------|
| Độ chính xác | 87.16000000000001 |
| Precision | 85.92420726991493 |
| Recall (macro) | 87.16000000000001 |
| Recall (micro) | 87.16000000000001 |
| Chỉ số - F1 (macro) | 87.1562002902939 |
| Chỉ số - F1 (micro) | 87.16000000000001 |

2.3.3 Mô hình FastText+CNN+LSTM

Mô hình FastText

Vector từ (Word vectors) – phương pháp word embedding cổ điển

Xây dựng một không gian vector có N chiều đủ để mã hóa tất cả các ngữ nghĩa của từ (câu). Các từ (câu) mã hóa sẽ thể hiện ý nghĩa của nó thông qua các thông số của vector mã hóa, ví dụ kích thước vector có thể biểu diễn trạng thái từ (câu): quá khứ, hiện tại hay tương lai, trọng số vector có thể thể hiện tần xuất từ xuất hiện hoặc vị trí của nó trong ngôn ngữ, ...

Thông thường vector được sử dụng nhiều nhất là one-hot vector (các vector mã hóa thuộc không gian $R^{|V|}$). Vector chứa tất cả các phần tử 0 và duy nhất có một phần tử 1 để chỉ vị trí của từ được sắp xếp trong túi từ. Trong đó ký hiệu $|V|$ là kích thước của bộ từ điển.

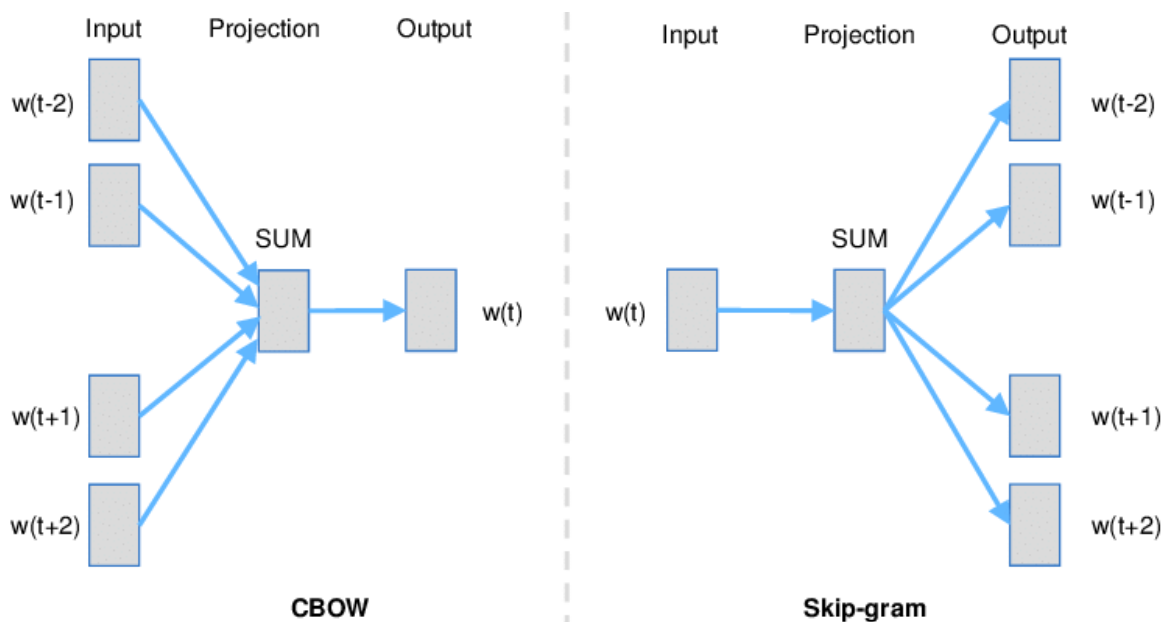
$$\text{Ex: } w^k = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Giới thiệu Phương pháp nhúng từ - Word Embeddings:

Một khái niệm chung nhất để định nghĩa cho nhúng từ là dùng một số (dãy số) biểu diễn cho một từ và đặc biệt hơn nữa mỗi vector đại diện cho một từ chính là biểu diễn mối quan hệ tương quan, tương đồng với các từ có ngữ nghĩa tương đồng với nó. Các mô hình nhúng từ như vậy còn được gọi là mô hình phân phối ngữ nghĩa.

Như đã giới thiệu về vector one hot, chúng ta có thể biểu diễn một cách ngây thơ một từ dưới dạng one hot trong không gian R^V trong đó V là kích thước của bộ từ điển (số lượng từ trong bộ từ điển). Tuy nhiên cách biểu diễn này mắc phải một nhược điểm vô cùng lớn là chiều dài các vector sẽ vô cùng lớn, ngoài ra đối với những bài toán cần đến ngữ nghĩa văn bản các vector one hot sẽ không thể biểu diễn được mối quan hệ của các từ trong câu.

Do đó, người ta tìm cách ánh xạ các vector one hot lên một không gian vector khác sao cho các từ có ngữ nghĩa tương đồng nhau nằm gần nhau. Vậy làm thế nào để tìm được một ánh xạ như thế? Ý tưởng giải quyết vấn đề này là xây dựng một mạng neural hai lớp với duy nhất một lớp ẩn và một lớp đầu ra để mạng học được một bộ trọng số có thể sinh một không gian vector biểu diễn mức độ tương đồng ngữ nghĩa của từ. Đề xuất hai mô hình phổ biến dùng những từ là mô hình Continuous Bag of Words (CBOW) và Skip-gram.



Hình 23 Mô hình CBOW và Skip-gram

Mô hình Continuous Bag of Words (CBOW): Cho ngữ cảnh (các từ xung quanh) và đoán ra xác suất xuất hiện từ đích.

Mô hình Skip-gram: Cho từ hiện tại và đoán xác suất của các từ ngữ cảnh (các từ xung quanh)

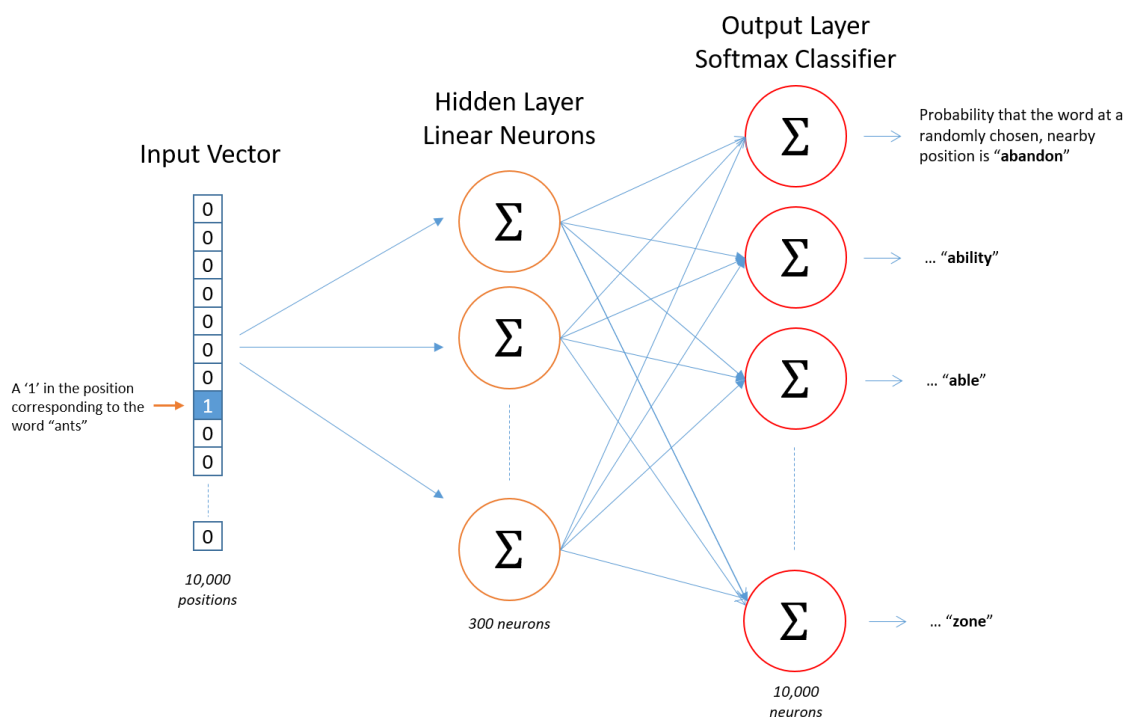
Điểm chung của hai mô hình này đều sử dụng một mạng neural gồm hai lớp với duy nhất một lớp ẩn và một lớp đầu ra (mạng neural đặc biệt không phải mạng CNN).

Xây dựng mô hình:

Như phần xây dựng mô hình Naïve Bayes trước đó, chúng ta cần xây dựng một kho ngữ liệu (hay còn được gọi là corpus) để có được một mô hình tốt thì chúng ta cần xây dựng được một bộ từ điển đủ tốt (vốn từ vựng phong phú và được sắp xếp trong nhiều ngữ cảnh khác nhau).

Sau khi xây dựng bộ từ điển đủ tốt chúng ta sẽ nhúng các từ có trong kho ngữ liệu dưới dạng vector one-hot với số chiều là số từ trong kho ngữ liệu và thành phần nhận giá trị '1' chính là biểu thị cho vị trí của từ trong kho ngữ liệu.

Chúng ta có kiến trúc mạng như sau:



Hình 24 Kiến trúc mạng Word Embedding

Giả sử kho ngữ liệu có W từ, khi đó đầu vào của mạng sẽ là một từ được nhúng thành một vector one-hot W chiều. Lớp đầu ra của mô hình có v neural (tương ứng với một vector đầu ra W chiều và mỗi thành phần của vector sẽ biểu diễn xác suất xuất hiện tiếp theo của từ với điều kiện từ trước đó (hoặc sau đó) là từ đầu vào). Lớp ẩn có d số neural (d - chính là số chiều vector nhúng từ mà chúng ta mong muốn).

Hàm mục tiêu của mô hình:

$$\frac{1}{T} \sum_{t=1}^T \sum_{j=-c}^c \log p(\omega_{t+j} | \omega_t)$$

Mục đích chính là tìm ra các trọng số để giá trị trung bình logarit xác suất của từ thứ $t+j$ với điều kiện có từ thứ t .

Trong đó xác suất $p(\omega_{t+j} | \omega_t)$ được tính theo công thức hàm softmax [13]:

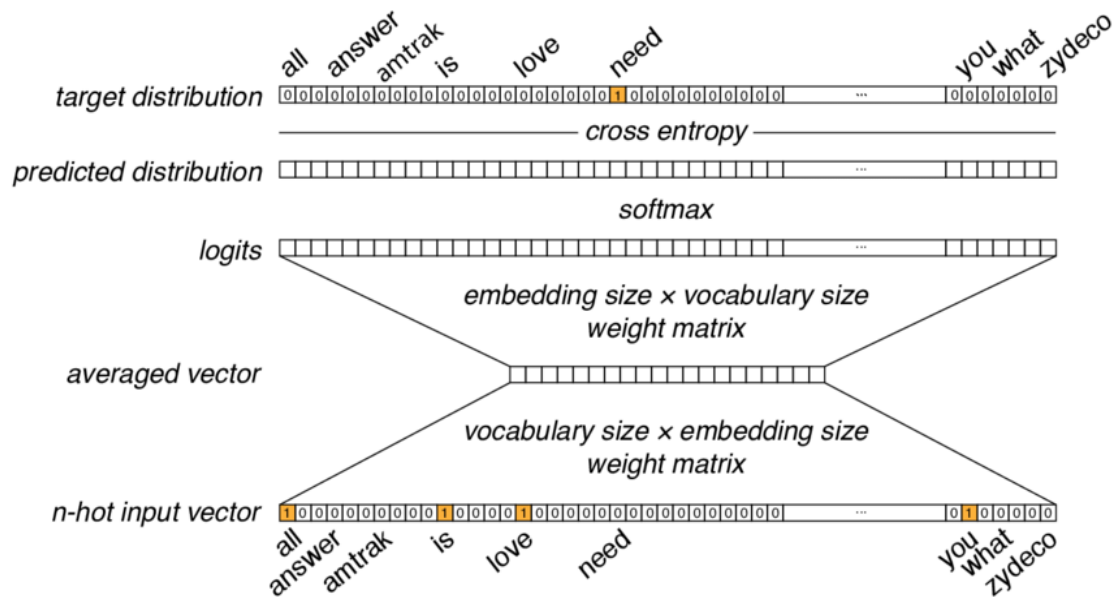
$$p(\omega_o | \omega_l) = \frac{\exp((v'_{\omega_o})^T v_{\omega_l})}{\sum_{\omega=1}^W \exp((v'_{\omega})^T v_{\omega_l})}$$

Trong đó: v_{ω}, v'_{ω} là các vector tương ứng đại diện cho từ đầu vào và từ đầu ra ω . W là số từ trong kho ngữ liệu. Tuy nhiên có thể thấy chi phí tính toán hàm này tương đối lớn và tỉ lệ thuận với W : $\text{cost}(\log(p(\omega_o | \omega_l))) \sim W$. Negative Sampling được đề xuất để giải quyết vấn đề đó, thay vì tính toàn bộ, ta chọn ngẫu nhiên một số từ trong kho ngữ liệu để làm từ negative, các từ context xung quanh ω được coi là positive. Việc negative sampling làm là chuyển bài toán của ta từ multi-class classification thành bài toán binary classification. Giả sử ta có một cặp từ context (ω, c) , điều ta cố gắng thực hiện là tăng xác suất xuất hiện của cặp "positive" (ω, c) và giảm xác suất xuất hiện của cặp "negative" (ω, s) được chọn bất kì, với s là từ không hề nằm trong bất kì window nào của center word ω . Từ đó, ta cố gắng maximum hàm sau [13]:

$$\log \sigma((v'_{\omega_o})^T v_{\omega_l}) + \sum_{i=1}^k E_{\omega_i \sim P_n}(\omega) [\sigma(-(v'_{\omega_i})^T v_{\omega_l})]$$

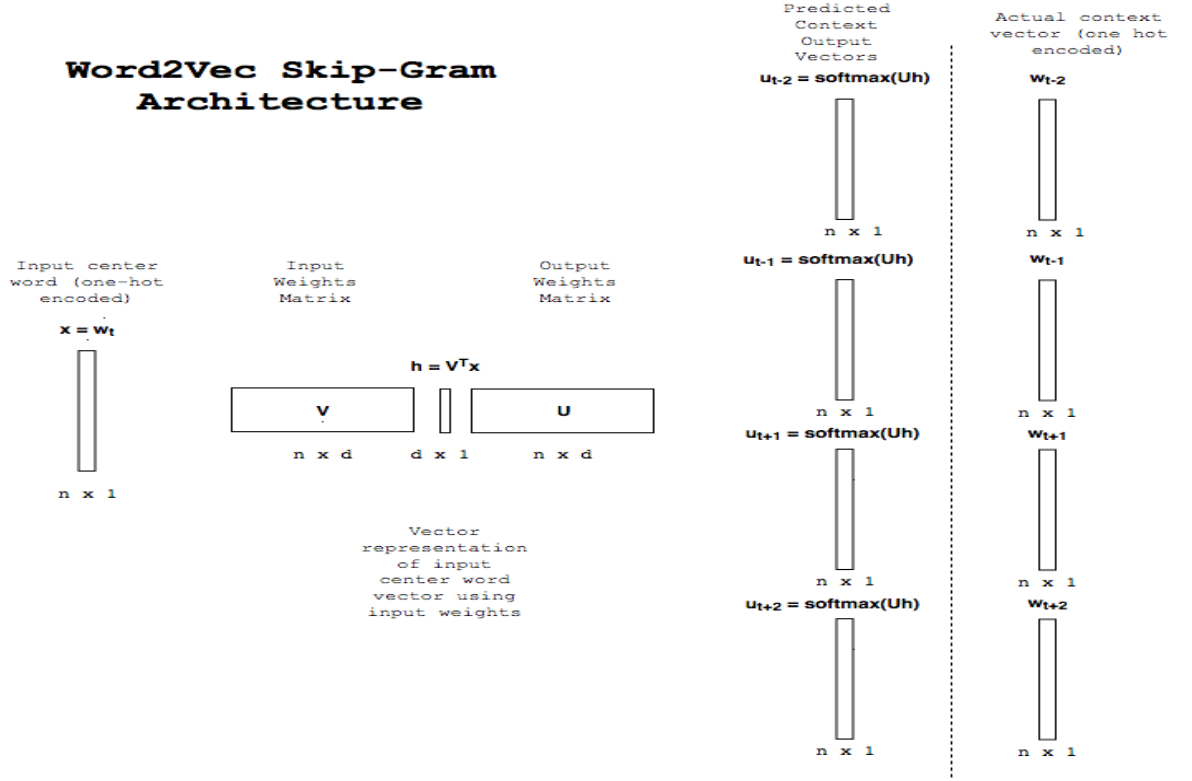
Trong đó v_{ω}, v'_{ω} là các vector tương ứng đại diện cho từ đầu vào và từ đầu ra ω . $\sigma(z) = \frac{1}{1+e^{-z}}$ là hàm kích hoạt sigmoid (cũng có thể coi là hàm tham số mạng). P_n là xác suất nhiễu được sử dụng để lấy mẫu các từ ngẫu nhiên. k là window (số từ xung quanh center word ω được chọn là positive).

Kiến trúc mạng mô hình CBOW:



Hình 25 Kiến trúc mạng CBOW

Kiến trúc mạng mô hình skip-gram:



Hình 26 Kiến trúc mạng skip-gram

Kiến trúc mô hình FastText:

Cốt lõi của mô hình FastText sử dụng mô hình CBOW tuy nhiên có phần cải tiến và nâng cấp hơn để tăng tốc thời gian đào tạo mà vẫn giữ được độ tốt của mô hình. FastText thay thế mức tiêu dự đoán một từ là một danh mục. và FastText thay thế hàm softmax của mô hình CBOW bằng hàm hierarchical softmax [13]:

$$P(\omega | \omega_I) = \prod_{j=1}^{L(\omega)-1} \sigma([n(\omega, j+1) = ch(n(\omega, j))]) (v'_{n(\omega, j)})^T v_{\omega_I}$$

Trong đó v_{ω}, v'_{ω} là các vector tương ứng đại diện cho từ đầu vào và từ đầu ra ω . $\sigma(z) = \frac{1}{1+e^{-z}}$ là hàm kích hoạt sigmoid. $n(\omega, j)$ là nút thứ j trên đường dẫn từ gốc đến ω , $L(\omega)$ độ dài của đường dẫn. $n(\omega, 1) = root$; $n(\omega, L(\omega)) = \omega$; $ch(n(\omega, j))$ là nút con tùy ý của nút $n(\omega, j)$; $z = [n(\omega, j+1) = ch(n(\omega, j))]$.

$$\text{Đặt } [z] = \begin{cases} 1 & \text{nếu } z = \text{True} \\ -1 & \text{nếu } z = \text{False} \end{cases}$$

Mạng neural tích chập

Convolutional Neural Networks (CNNs – Mạng neural tích chập) là một trong những mô hình học sâu tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động.

CNNs được sử dụng nhiều trong các bài toán nhận dạng, phân lớp các đối tượng trong ảnh. Để tìm hiểu tại sao thuật toán này được sử dụng rộng rãi cho việc nhận dạng (detection).

Định nghĩa mạng neural tích chập

Mô hình mạng neural nhân tạo truyền thẳng ra đời đã được áp dụng rất nhiều trong các bài toán nhận dạng. Tuy nhiên mạng neural truyền thẳng không thực hiện tốt lắm với các bài toán dữ liệu nhận dạng hình ảnh. Ví dụ như một hình ảnh có kích thước 32x32 pixel sẽ cho ra vector đặc trưng có 1024 chiều, còn đối với ảnh màu có cùng kích thước sẽ là 3072 chiều. Điều này có nghĩa là ta sẽ cần đến 3072 trọng số nối giữa lớp ẩn kế tiếp, dẫn đến mô hình quá đồ sộ. Điều này càng khó khăn hơn khi thao tác với các ảnh kích thước lớn hơn nữa. Chính vì vậy mạng Convolution Neural Networks (CNNs- mạng neural tích chập) ra đời. Mạng neural tích chập (CNNs hoặc ConvNet) là một trong thuật toán học hình ảnh và video phổ biến nhất. Giống như các mạng neural khác, CNNs bao gồm một lớp đầu vào, một lớp đầu ra và nhiều lớp ẩn ở giữa [3].

Các lớp trong mô hình neural tích chập

Tầng trích xuất vector đặc trưng: Khối này bao gồm 3 lớp được sử dụng kết hợp với nhau nhằm mục đích trích xuất đặc trưng, và lựa chọn đặc trưng.

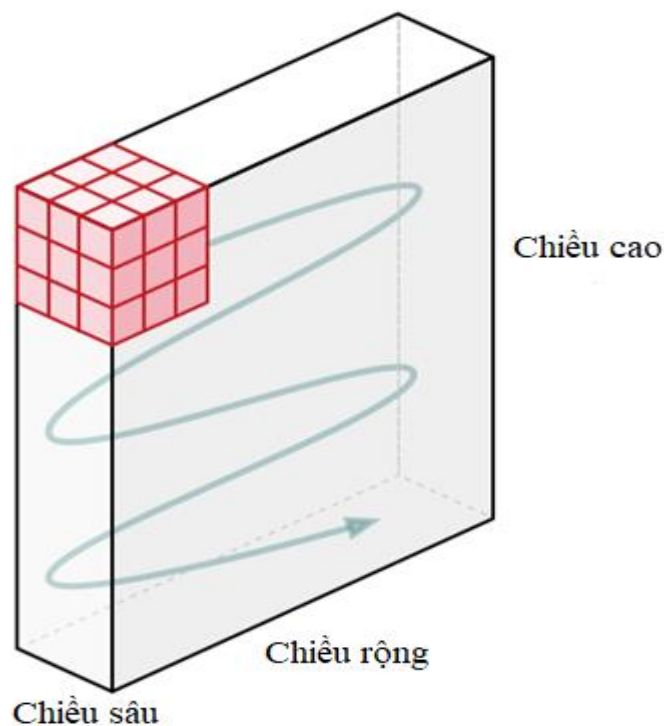
Tích chập: Đây là lớp quan trọng nhất trong cấu trúc của CNNs, đây chính là nơi trích xuất các đặc trưng thể hiện tư tưởng ban đầu của mạng neural tích chập [3]. Thay vì kết nối toàn bộ điểm, lớp này sẽ sử dụng một bộ hạt nhân có kích thước nhỏ hơn so với ảnh (thường là 2×2 cho đến 5×5) áp vào một vùng trong ảnh và tiến hành tích chập giữa bộ hạt nhân này và giá trị điểm ảnh trong vùng cục bộ đó. Hạt nhân sẽ lần lượt được dịch chuyển theo một giá trị bị trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.

| | | | | |
|-----|-----|-----|---|---|
| 1x1 | 1x0 | 1x1 | 0 | 0 |
| 0x0 | 1x1 | 1x0 | 1 | 0 |
| 0x1 | 0x0 | 1x1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

| | | |
|---|--|--|
| 4 | | |
| | | |
| | | |

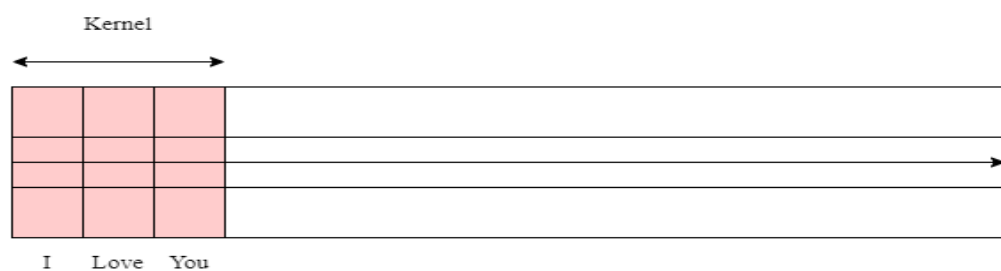
Hình 27 Mô phỏng hoạt động của Convolution

Tuy nhiên ảnh màu có tới 3 channels red, green, blue nên khi biểu diễn ảnh dưới dạng tensor 3 chiều. Nên ta cũng sẽ định nghĩa hạt nhân là 1 tensor 3 chiều kích thước $k \times k \times 3$. Hạt nhân trượt theo 2 chiều của dữ liệu là chiều rộng và chiều sâu, quá trình trượt được mô tả như sau:



Hình 28 Mô tả hạt nhân hoạt động Conv2D

Đối với các dữ liệu dưới dạng chuỗi thời gian, văn bản thì thông thường dữ liệu sẽ được biểu diễn dưới dạng tensor 2 chiều. tại đây hạt nhân sẽ trượt theo một chiều duy nhất của dữ liệu và độ rộng của hạt nhân sẽ bằng độ rộng của dữ liệu, quá trình trượt được mô tả như sau:



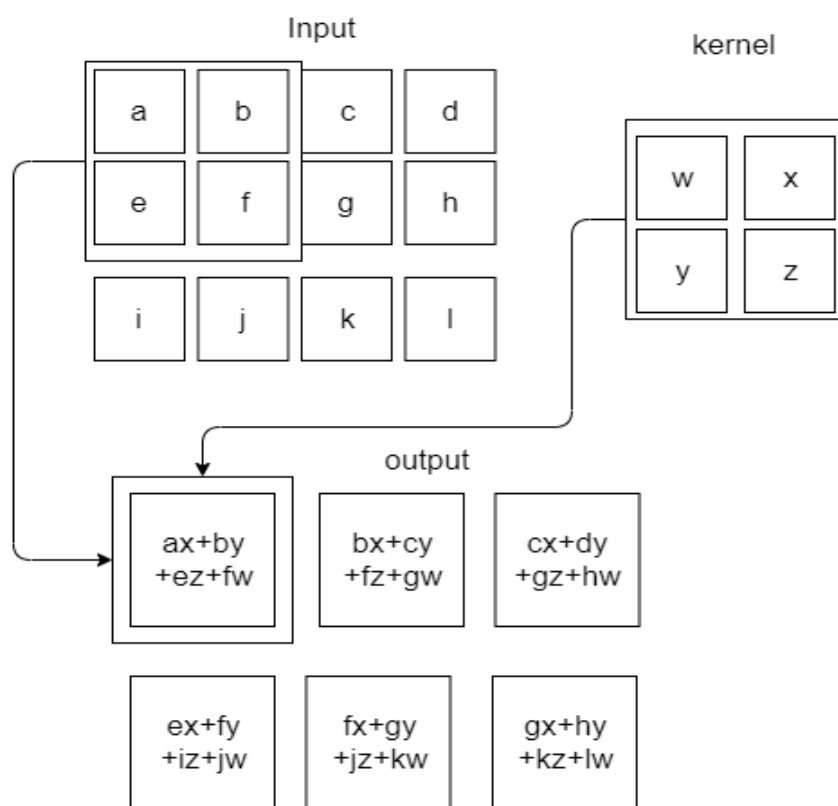
Hình 29 Mô tả hạt nhân hoạt động Conv1D

Như vậy sử dụng tích chập sẽ có những ưu điểm sau:

Giảm số lượng tham số: Ở ANN truyền thống, các neural ở lớp trước sẽ kết nối tất cả các neural ở lớp sau (full connected) gây nên tình trạng quá nhiều tham số cần học. Đây là nguyên nhân chính gây nên tình trạng overfitting cũng như tang thời

gian huấn luyện. Việc sử dụng tích chập trong đó cho phép chia sẻ sử dụng local receptive fields giúp giảm tham số.

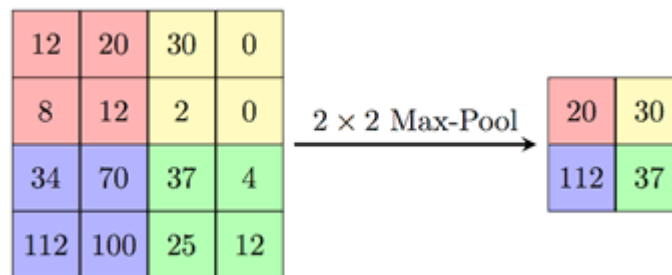
Các tham số trong quá trình sử dụng tích chập hay giá trị của các filter-hạt nhân sẽ được học trong quá trình huấn luyện. Như giới thiệu ở phần trên các thông tin này biểu thị thông tin giúp rút trích ra được các đặc trưng như góc, cạnh, đốm màu trong ảnh ... như vậy việc sử dụng Conv sẽ giúp xây dựng mô hình tự học rất đặc trưng.



Hình 30 Mô phỏng quá trình hoạt động của convolution

Lớp pooling sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Tuy nhiên, khác với lớp Convolution, lớp pooling không tính tích chập mà tiến hành lấy mẫu (subsampling) [3]. Khi cửa sổ trượt trên ảnh chỉ có một giá trị được xem là giá trị đại diện cho thông tin ẩn tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong lớp pooling là maxpooling (lấy giá trị lớn nhất), minpooling (lấy giá trị nhỏ nhất) và averagepooling

(lấy giá trị trung bình). Xét một ảnh có kích thước 32x32 và lớp pooling sử dụng là filter có kích thước 2x2 và bước trượt stride = 2, phương pháp sử dụng maxpooling. Filter sẽ lần lượt duyệt qua ảnh, với mỗi lần duyệt chỉ có giá trị lớn nhất trong bốn giá trị nằm trong vùng cửa sổ 2x2 của filter được giữ lại và đưa ra đầu ra. Như vậy sau khi qua pooling, ảnh sẽ giảm kích thước xuống 16x16 (mỗi chiều giảm 2 lần).



Hình 31 Max - Pooling

Lớp pooling có vai trò làm giảm kích thước dữ liệu. Với mỗi bức ảnh kích thước lớn qua nhiều lớp pooling sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc cần cho việc nhận dạng (thông tin cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting) [14].

Lớp ReLU – Rectified Linear Unit: về cơ bản tích chập là một phép biến đổi tuyến tính. Nếu tất cả các neural được tổng hợp bởi các phép biến đổi tuyến tính thì một mạng neural đều có thể đưa về dạng một hàm tuyến tính. Khi đó mạng ANN sẽ đưa các bài toán về logistic regression. Do đó tại mỗi neural cần có một hàm truyền dưới dạng phi tuyến [14].

Lớp này sử dụng hàm kích hoạt $f(x) = \max(0, x)$. Nói một cách đơn giản, lớp này có nhiệm vụ chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp tích chập thành giá trị 0. Ý nghĩa của cách cài đặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng đa tầng đa lớp trở nên vô nghĩa. Có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt sigmoid, tanh....

Tầng phân lớp: Sau khi phát hiện kiến trúc của CNNs chuyển sang quá trình phân loại.

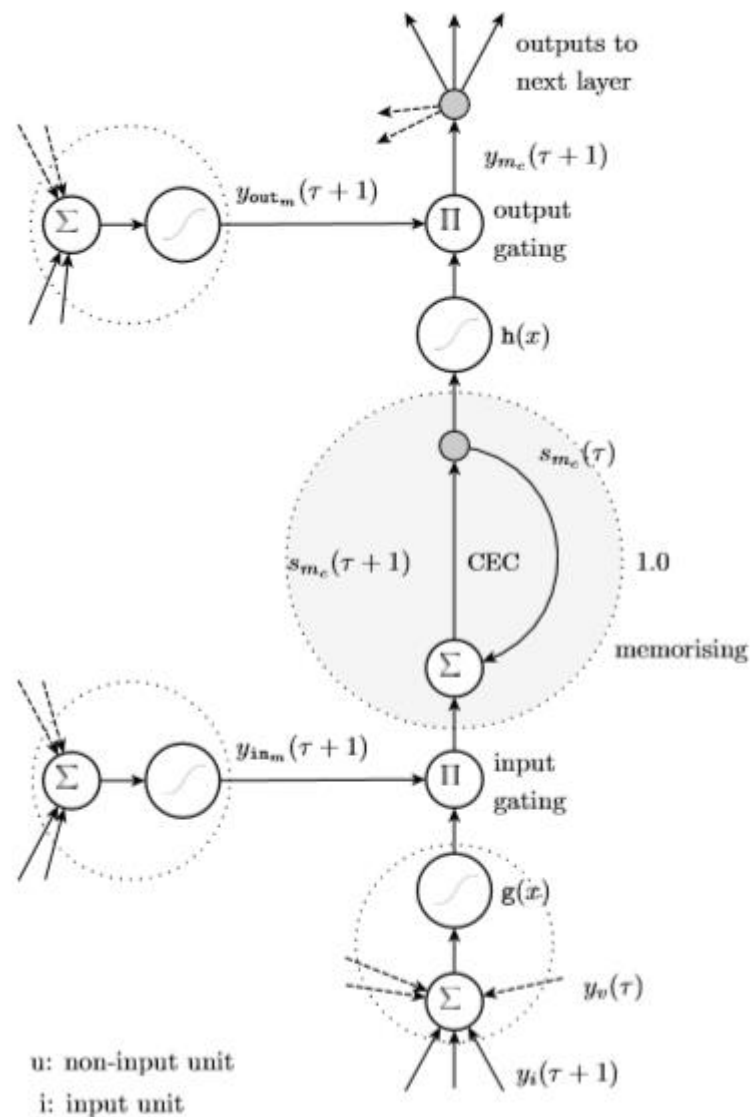
Lớp kết nối đầy đủ: lớp này tương tự với lớp trong mạng neural truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào nút trong lớp tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ lớp trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng. Tóm lại, lớp kết nối đầy đủ đóng vai trò như một mô hình phân lớp và tiến hành dựa trên dữ liệu đã được xử lý ở các lớp trước đó.

2.3.3.1 Long Short Term Memory – LSTM

RNN tiêu chuẩn không thể lưu trữ được thông tin vượt quá 5-10 bước thời gian [15]. Vì thế qua nhiều bước thời gian hơn thì lỗi có thể bị bùng nổ hoặc triệt tiêu. Các tín hiệu lỗi bùng nổ sẽ dẫn thẳng đến các trọng số dao động (bị thay đổi), trong khi đó vấn đề lỗi bị triệt tiêu thì việc học sẽ mất một khoảng thời gian không thể chấp nhận được (học vô hạn thời gian) hoặc là hoàn toàn không hoạt động. Trong thuật toán BPTT, càng ngược về các bước thời gian trước đó thì giá trị gradient được tính toán càng giảm dần và sau một số bước thời gian nhất định thì gradient sẽ bị triệt tiêu, do đó làm giảm tốc độ hội tụ của các trọng số do sự thay đổi trọng số vô cùng nhỏ hoặc không thể hội tụ được. LSTM là một phiên bản mở rộng của RNN giải quyết được nhược điểm lỗi bị triệt tiêu của RNN, và nó có thể ghi nhớ lưu trữ thông tin lên đến hơn 1000 bước thời gian rời rạc.

LSTM mở rộng với các cổng đầu vào và đầu ra được kết nối đến lớp đầu vào và các tế bào nhớ (memory cells) khác. Điều này dẫn đến một đơn vị LSTM phức tạp hơn, được gọi là khối nhớ (memory block). LSTM đã sử dụng kết hợp hai thuật toán học: BPTT để huấn luyện các thành phần mạng nằm sau các cell và RTRL để huấn luyện các thành phần mạng nằm trước và bao gồm các cell. Các đơn vị sau hoạt động với RTRL vì có một số đạo hàm riêng (liên quan đến trạng thái tế bào – cell state) cần được tính toán trong mỗi bước, bất kể giá trị đích có được đưa ra hay không ở bước đó [2];

Kiến trúc tiêu chuẩn của LSTM:



Hình 32 Kiến trúc LSTM tiêu chuẩn

Kiến trúc LSTM bao gồm ít nhất một cell có tự kết nối hồi tiếp và được đánh trọng số '1'. Trạng thái tế bào – cell state được ký hiệu s_c , truy cập đọc và ghi dữ liệu được quy định bởi các cổng đầu vào – input gate: y_{in} và cổng đầu ra – output gate: y_{out} . [2]

Ý tưởng cốt lõi của LSTM:

LSTM có kiến trúc tương tự như RNN tuy nhiên các module trong LSTM lại khác thay vì chỉ có một tầng mạng neural như RNN thì chúng có bốn tầng mạng tương tác với nhau rất đặc biệt.

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. Vì vậy mà 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 bào, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate). LSTM có 3 cổng: cổng quên - forget gate, cổng đầu vào - input gate, cổng đầu ra - output 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.

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 “cổng quên” (forget gate). Giá trị đầu ra của cổng quên được tính theo công thức:

$$y_{\varphi m} = f_{\varphi m}(z_{\varphi m}(\tau + 1) + b_{\varphi m})$$

Trong đó $f(s) = \frac{1}{1+e^{-s}}$ là hàm kích hoạt sigmoid có giá trị đầu ra trong khoảng $[0, 1]$, $b_{\varphi m}$ là hệ số bias của cổng quên.

$$\begin{aligned} z_{\varphi m}(\tau + 1) &= \sum_u W_{[\varphi m, u]} X_{[u, \varphi m]}(\tau + 1) \text{ trong đó } u \in Pre(\varphi m) \\ &= \sum_{v \in U} W_{[\varphi m, v]} y_v(\tau) + \sum_{i \in I} W_{[\varphi m, i]} y_i(\tau + 1) \end{aligned}$$

$b_{\varphi m}$ ban đầu thường có giá trị bằng 0 tuy nhiên theo [16] thiết lập giá trị $b_{\varphi m} = 1$ thì hiệu suất hoạt động LSTM sẽ được cải thiện đáng kể.

Cập nhật trạng thái tế bào:

$$s_{mc}(\tau + 1) = s_{mc}(\tau)y_{\phi m}(\tau + 1) + y_{in_m}(\tau + 1)g(z_{mc}(\tau + 1))$$

$$y_{\phi m}(\tau + 1) = 1 \text{ nếu không có cổng quên}$$

$$s_{mc}(0) = 0, g(z) = \frac{4}{1 + e^{-z}} - 2$$

Các trọng số bias của các cổng đầu vào và đầu ra được khởi tạo với các giá trị âm và các trọng số của cổng quên được khởi tạo với các giá trị dương. Từ đó, khi bắt đầu đào tạo, kích hoạt cổng quên sẽ gần với ‘1.0’. Tế bào nhớ sẽ hoạt động giống như một ô nhớ LSTM tiêu chuẩn mà không có cổng quên. Điều này ngăn không cho tế bào bộ nhớ LSTM quên.

Các cổng đầu vào có hàm kích hoạt sigmoid với phạm vi hàm kích hoạt trong khoảng $[0,1]$, điều khiển các tín hiệu từ mạng đến tế bào nhớ; khi cổng được đóng, giá trị đầu ra của cổng đầu vào gần bằng không. Ngoài ra, chúng có thể học cách bảo vệ nội dung được lưu trữ trong đơn vị u khỏi sự xáo trộn bởi các tín hiệu không liên quan.

Các cổng đầu ra có hàm kích hoạt tanh có thể tìm hiểu cách kiểm soát truy cập vào nội dung của tế bào nhớ, giúp bảo vệ các tế bào nhớ khác khỏi các nhiễu loạn gây ra từ đơn vị u. Vì vậy, chúng ta có thể thấy rằng chức năng cơ bản của các đơn vị cổng nhân là cho phép hoặc từ chối truy cập vào thông tin trạng thái tế bào nhớ.

Đầu vào của cổng đầu vào:

$$\begin{aligned} z_{in_m}(\tau + 1) &= \sum_u W_{[in_m, u]} X_{[u, in_m]}(\tau + 1) \text{ trong đó } u \in Pre(in_m) \\ &= \sum_{v \in U} W_{[in_m, v]} y_v(\tau) + \sum_{i \in I} W_{[in_m, i]} y_i(\tau + 1) \end{aligned}$$

Hàm kích hoạt của cổng đầu ra out_m

$$y_{out_m} = f_{out_m}(z_{out_m}(\tau + 1))$$

Với đầu vào của cổng đầu ra

$$\begin{aligned} z_{out_m}(\tau + 1) &= \sum_u W_{[out_m, u]} X_{[u, out_m]}(\tau + 1) \text{ trong đó } u \in Pre(out_m) \\ &= \sum_{v \in U} W_{[out_m, v]} y_v(\tau) + \sum_{i \in I} W_{[out_m, i]} y_i(\tau + 1) \end{aligned}$$

Do $f(s) = \frac{1}{1+e^{-s}}$ là hàm sigmoid có giá trị nằm trong phạm vi $[0, 1]$ nên đầu vào từ tế bào nhớ được phép cho qua nếu giá trị tín hiệu thu được từ cổng đầu vào là gần bằng 1 hoặc bằng 1.

Cuối cùng, ta cần quyết định xem ta muốn 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 ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bào qua một hàm tanh để co 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.

$$y_{mc}(\tau + 1) = y_{out_m}(\tau + 1)h(s_{mc(\tau+1)})$$

$$h(z) = \frac{2}{1 + e^{-z}} - 1$$

Sơ đồ mô hình FastText+CNN+LSTM

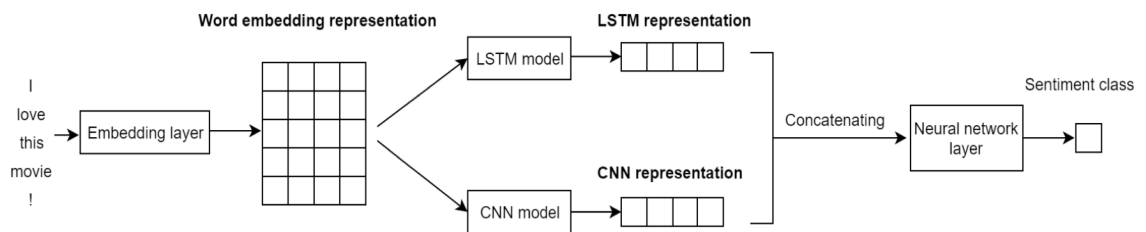


Figure 1. Multi-channel LSTM-CNN model

Hình 33 Mô hình Word_Embedding+CNN+LSTM

| Layer (type) | Output Shape | Param # | Connected to |
|-----------------------------|-------------------|---------|--|
| main_input (InputLayer) | (None, 1055) | 0 | |
| embedding_1 (Embedding) | (None, 1055, 150) | 3000000 | main_input[0][0] |
| conv1d_4 (Conv1D) | (None, 1053, 150) | 67650 | embedding_1[0][0] |
| conv1d_5 (Conv1D) | (None, 1051, 150) | 112650 | embedding_1[0][0] |
| conv1d_6 (Conv1D) | (None, 1049, 150) | 157650 | embedding_1[0][0] |
| lambda_4 (Lambda) | (None, 150) | 0 | conv1d_4[0][0] |
| lambda_5 (Lambda) | (None, 150) | 0 | conv1d_5[0][0] |
| lambda_6 (Lambda) | (None, 150) | 0 | conv1d_6[0][0] |
| lstm_2 (LSTM) | (None, 128) | 142848 | embedding_1[0][0] |
| concatenate_3 (Concatenate) | (None, 450) | 0 | lambda_4[0][0] lambda_5[0][0] lambda_6[0][0] |
| concatenate_4 (Concatenate) | (None, 578) | 0 | lstm_2[0][0] concatenate_3[0][0] |
| dense_3 (Dense) | (None, 200) | 115800 | concatenate_4[0][0] |
| dropout_2 (Dropout) | (None, 200) | 0 | dense_3[0][0] |
| dense_4 (Dense) | (None, 2) | 402 | dropout_2[0][0] |
| Total params: 3,597,000 | | | |
| Trainable params: 3,597,000 | | | |
| Non-trainable params: 0 | | | |

Hình 34 Kiến trúc mô hình

Kết quả huấn luyện mô hình:

| | |
|---------------------|-------------------|
| Độ chính xác | 87.3 |
| Precision | 87.3 |
| Recall (macro) | 87.3 |
| Recall (micro) | 87.3 |
| Chỉ số - F1 (macro) | 87.29428954549331 |
| Chỉ số - F1 (micro) | 87.29999999999998 |

2.3.4 Lựa chọn mô hình

| Mô hình Naïve Bayes | Mô hình Support Vector Machine (SVM) | Mô hình FastText+CNN+LSTM |
|---|---|--|
| <ul style="list-style-type: none"> Có thời gian huấn luyện và test rất nhanh (thời gian huấn luyện: 2 phút). Điều này có được là do giả sử về tính độc lập giữa các thành phần (tuy nhiên trong thực tế tính độc lập giữa các thành phần gần như là không tồn tại) Khả năng phân lớp khá tốt với bộ dữ liệu nhỏ và giả sử được tính độc lập của dữ liệu | <ul style="list-style-type: none"> Thời gian huấn luyện nhanh tuy nhiên chậm hơn so với Naïve Bayes (thời gian huấn luyện: 9 phút) SVM không yêu cầu khắt khe về mặt dữ liệu như Naïve Bayes Đối với bài toán phân tích cảm xúc SVM cũng có nhược điểm tương tự với Naïve Bayes đều không biểu diễn được ngữ nghĩa văn bản dẫn đến nếu với dữ liệu | <ul style="list-style-type: none"> Thời gian huấn luyện tương đối lâu (2 giờ 12 phút) Ngoài ra còn thời gian huấn luyện mô hình FastText (mô hình word embedding 10 phút). Tuy nhiên thời gian test khá nhanh 1 phút (cho 10000 câu review) Mô hình cho kết quả tương đối tốt, nó có thể khắc phục được nhược điểm về ngữ nghĩa so với các mô |

| | | |
|---|---|--|
| <ul style="list-style-type: none"> • Đối với bài toán phân tích cảm sử dụng mô hình này cho kết quả đánh giá tương đối tốt. Tuy nhiên mô hình không học được ngữ nghĩa của văn bản, không thể hiện được mối liên kết của các từ trong văn bản nếu dữ liệu đưa vào là một dữ liệu mới không có quá nhiều liên hệ với dữ liệu huấn luyện thì kết quả mô hình sẽ không ổn định (hoặc rất tồi) | <p>mới chưa từng xuất hiện trong tập huấn luyện thì kết quả cũng không quá tốt)</p> | <p>hình Naïve Bayes và SVM. Ngoài ra chỉ cần xây dựng được mô hình word embedding đủ tốt thì việc phân lớp một review mới đối với mô hình này sẽ khá tốt do các câu tuy chưa từng xuất hiện trong tập train nhưng các từ đã có xuất hiện trong mô hình word embedding đều được nhúng thành một vector và có liên kết về mặt ngữ nghĩa với các từ khác => Mô hình vẫn có khả năng đưa ra kết quả tương đối tốt</p> |
|---|---|--|

Như đã phân tích ở trên mô hình FastText+CNN+LSTM tuy tồn đọng một số nhược điểm nhất định về mặt thời gian đào tạo mô hình, nhưng nó lại phù hợp với bài toán có phân tích về mặt ngữ nghĩa (phân tích cảm xúc tiếng Việt) và cho kết quả đánh giá tốt hơn hai mô hình còn lại là Naïve Bayes và SVM. Do đó em quyết định lựa chọn mô hình FastText+CNN+LSTM để xây dựng chương trình.

CHƯƠNG 3: KẾT QUẢ

3.1 Môi trường cài đặt thuật toán và các vấn đề liên quan

Môi trường huấn luyện mô hình: sử dụng google colab (bản free); sử dụng GPU của colab; RAM: 12,72 GB; Disk: 68,40 GB; môi trường python 3 và sử dụng thư viện TensorFlow version: 2.2.0

Chương trình được cài đặt trên ngôn ngữ python (phiên bản 3.7.4) và được thử nghiệm trên hệ điều hành Windows 10, 64 bit, máy tính laptop lenovo ThinkPad T450s, sử dụng chip Intel® Core™ i7-5600U CPU @ 2.60GHz 2.59GHz, RAM 8.00 GB.

Dữ liệu đem phân tích đánh giá: Các bình luận được thu thập từ website: <https://www.thegioiiddong.com/dtdd/samsung-galaxy-a50/danh-gia>

IDE sử dụng: Visual studio code.

Các thư viện Python sử dụng: tensorflow, pandas, numpy, matplotlib, gensim, keras, Flask, Pygal.

3.2 Kết quả đánh giá mô hình

Kết quả mô hình naïve bayes:

| | |
|---------------------|-------------------|
| Độ chính xác | 84.19 |
| Precision | 82.89397729459303 |
| Recall (macro) | 84.19000000000001 |
| Recall (micro) | 84.19 |
| Chỉ số - F1 (macro) | 84.18386191497058 |
| Chỉ số - F1 (micro) | 84.19000000000001 |

Kết quả mô hình Support Vector Machine:

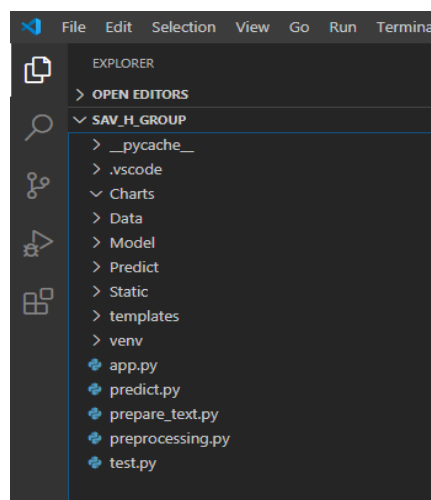
| | |
|---------------------|-------------------|
| Độ chính xác | 87.16000000000001 |
| Precision | 85.92420726991493 |
| Recall (macro) | 87.16000000000001 |
| Recall (micro) | 87.16000000000001 |
| Chỉ số - F1 (macro) | 87.1562002902939 |
| Chỉ số - F1 (micro) | 87.16000000000001 |

Kết quả mô hình FastText+CNN+LSTM:

| | |
|---------------------|-------------------|
| Độ chính xác | 87.3 |
| Precision | 87.3 |
| Recall (macro) | 87.3 |
| Recall (micro) | 87.3 |
| Chỉ số - F1 (macro) | 87.29428954549331 |
| Chỉ số - F1 (micro) | 87.29999999999998 |

3.3 Xây dựng chương trình

Cấu trúc xây dựng chương trình:



Hình 35 Cấu trúc xây dựng chương trình

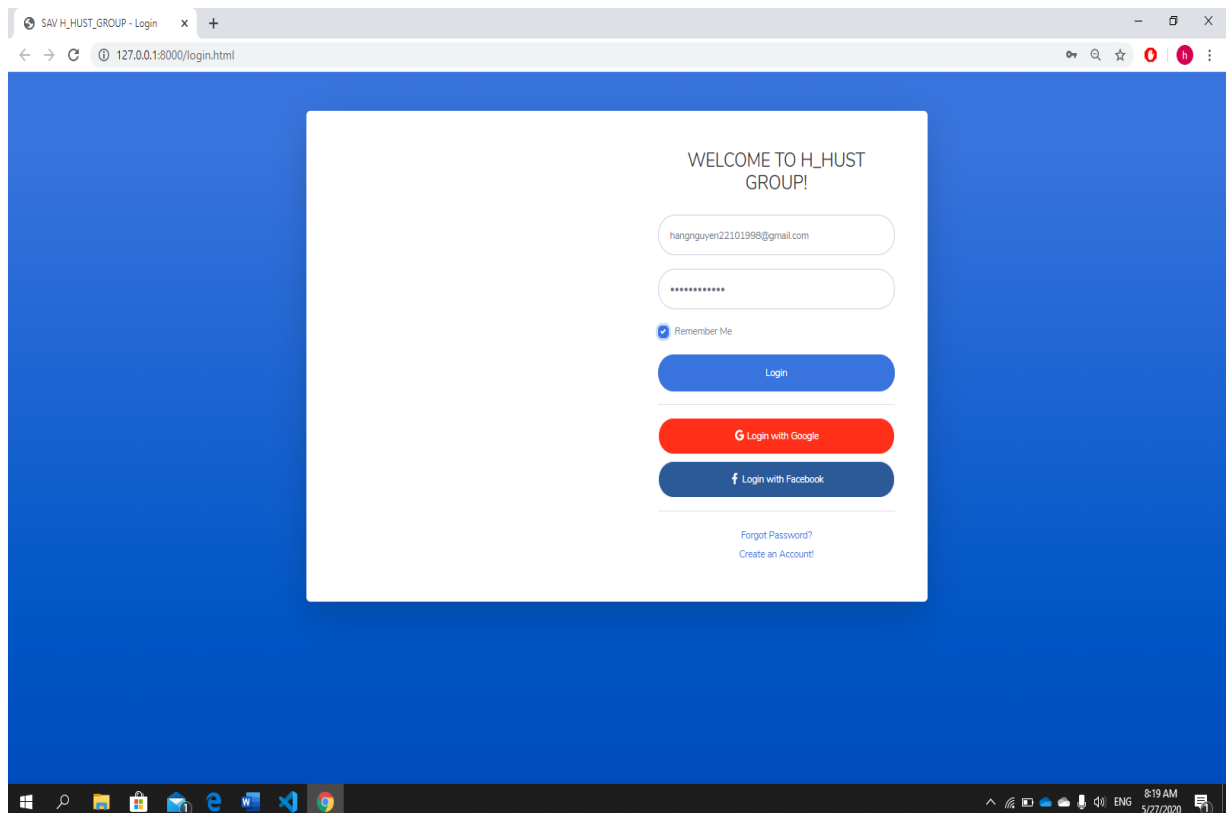
Trong đó chức năng chính của các module như sau:

`preprocessing.py`: Đảm nhiệm chức năng tiền xử lý dữ liệu thô loại bỏ các stopwords, tách từ,...

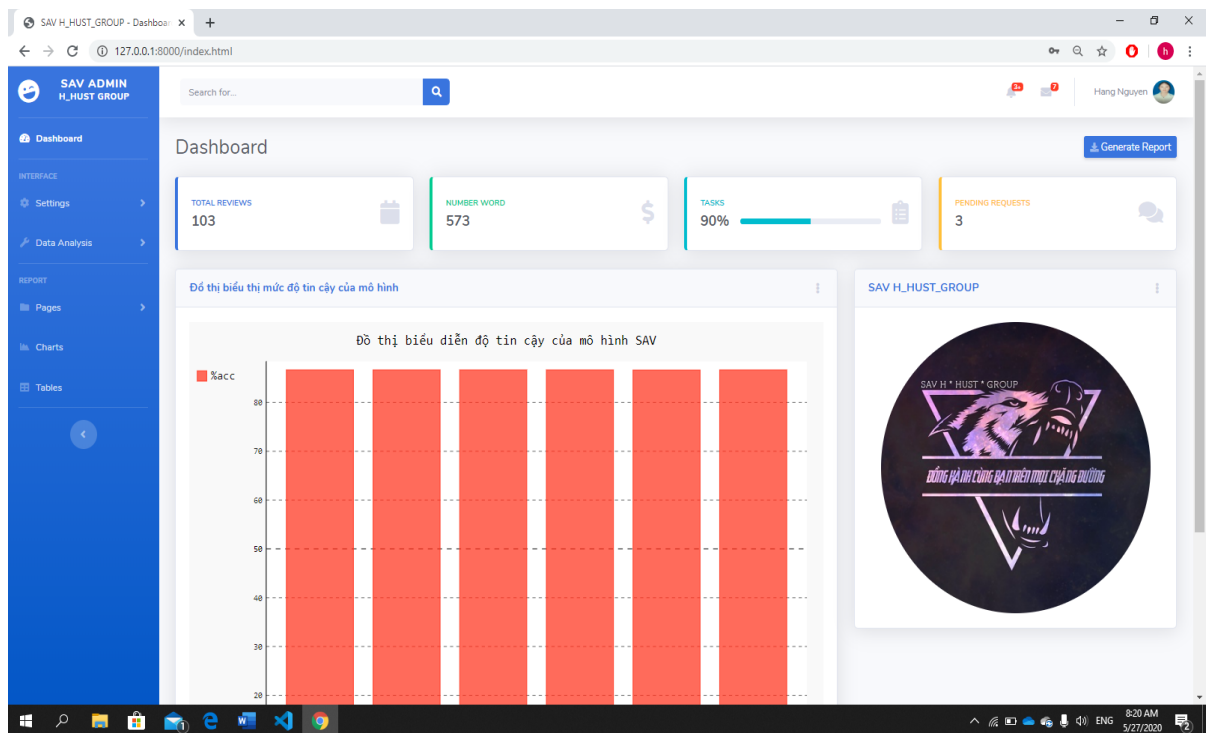
`prepare_text.py`: Sau khi dữ liệu được qua xử lý thô thì tại đây các từ, các câu sẽ được vector hóa trước khi đem đi phân tích, dự đoán.

`predict.py`: Đưa ra dự đoán từng câu trong bộ dữ liệu là tích cực hoặc tiêu cực.

Giao diện chương trình:



Hình 36 Trang login



Hình 37 Trang chủ hệ thống

Crawl Data

Crawl_data export to json

Generate JSON Data

Crawl_data export to csv

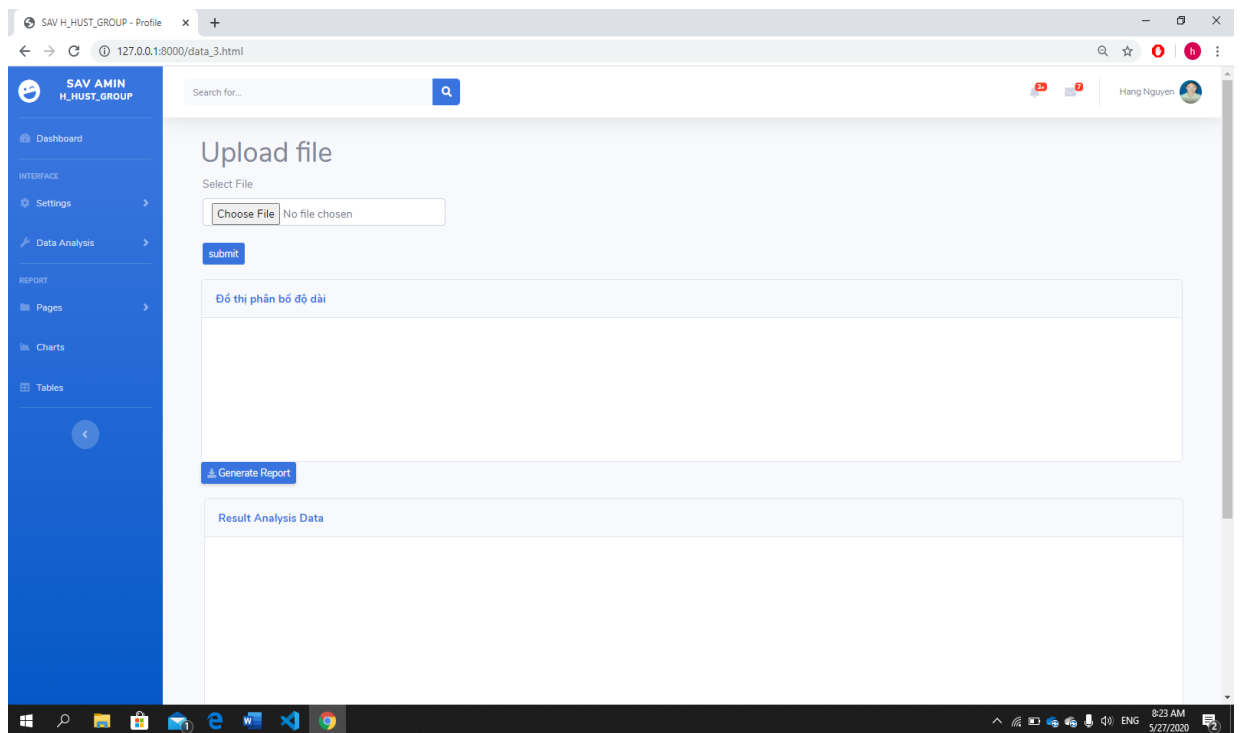
Generate CSV Data

Select File

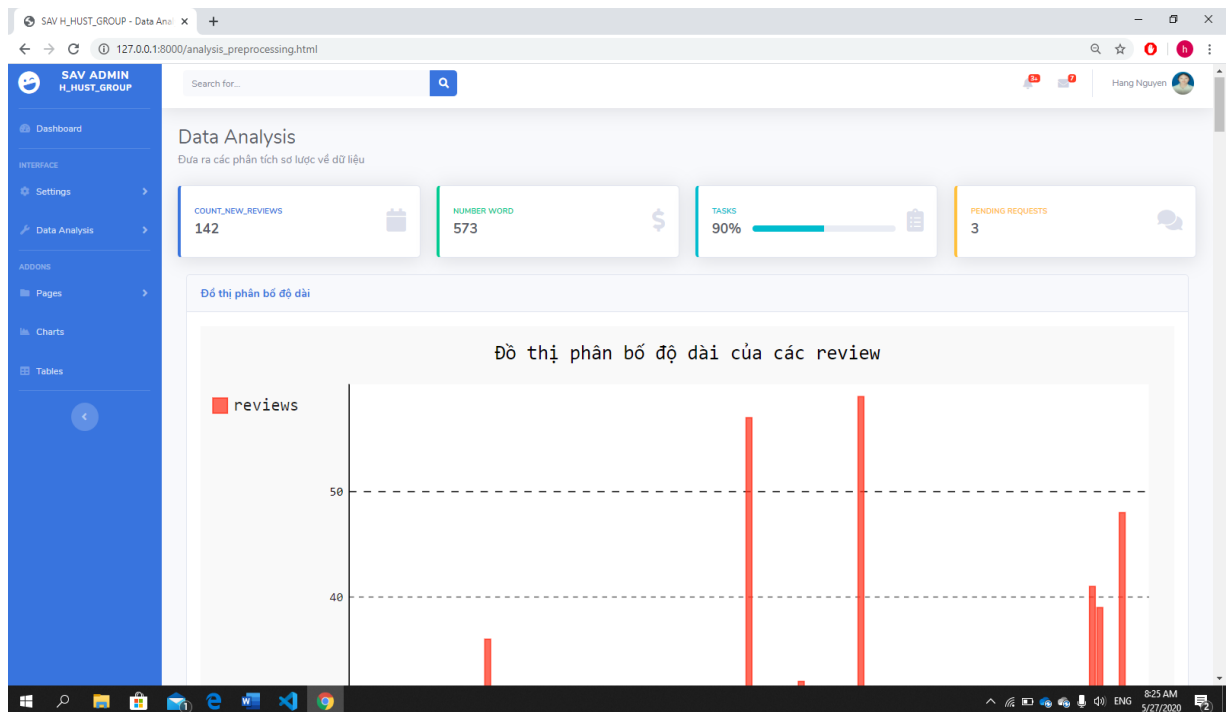
Select File

Copyright © Your Website 2019

Hình 38 Trang thu thập dữ liệu



Hình 39 Trang lựa chọn dữ liệu có sẵn hoặc từ bên thứ 3



Hình 40 Trang thông tin tiền xử lý giao diện

SAV ADMIN
HUST GROUP

Search for...

Tables

DataTables is a third party plugin that is used to generate the demo table below. For more information about DataTables, please visit the [official DataTables documentation](#).

DataTables Example

Show 10 entries

Search:

| User | Comment | Time |
|---------------|--|--------------|
| Anh Nhut | Có cái điện thoại mà sóng sánh làm không nên, bắt sóng yếu, không ổn định thua cả điện thoại tàu. Ai có chơi game hay cần làm việc gì cần sóng ổn định thì ko nên mua. Vote 0 sao chứ 1 sao cũng không đáng | 29/03/2020 |
| Anh Nhut | Tình trạng rất mạng, mất sóng 4g xuống h+ thường xuyên khi chơi game. Cực kì ức chế -1* luôn | 03/04/2020 |
| Anh tuyến | Mình dùng 4 tháng rồi cũng thấy ổn bin dùng được cam tạm được nói chung là được | 22/12/2019 |
| Buianhtu | Mình mua dùng được vài ngày phát hiện trong camera có rất nhiều bụi đã đổi mới nhưng cái mới cũng bị bụi lọt vào phải nói chất lượng quá tệ mình đem ra đổi ko được nữa vì họ nói họ không phát hiện có bụi | 05/12/2019 |
| Bùi Văn Hoàng | Mua 2 máy a50 máy mới dùng lượng pin 4000mAh mà so vs samsung j7 pro pin chỉ có 3600mAh mà k bằng j7 PRO thua về mọi mặt,mất lòng tin bên hãng,và k thích chính sách đổi trả bên tgdd,tôi đã tin tưởng và mua hàng bên tgdd và dmxc,nhưng giờ tôi phải nghĩ lại trước khi mua hàng bên tgdd,và dmxc... | 17/01/2020 |
| Bảo Khanh | Máy hơi nóng là có hiện tượng loạn cảm ứng , chơi game hay nhập bàn phím thì không ăn nhạy loạn xa, tác vụ hằng ngày thì lâu lâu bị đơng r văng..... | 03/03/2020 |
| Cao tuần | Nâng cấp lên 1/2/2020 thì tịt hần luôn ý | 18/03/2020 |
| Châu | Các bạn vào cài đặt, sau đó vào phần màn hình, sau đó kéo xuống là thấy phần tăng độ nhạy cảm ứng là oke. | 4 tuần trước |
| Cúc Huỳnh | Sản phẩm mua về mới sai thì ổn sau khi cập nhật thì có hiện tượng treo logo làm sao để khắc phục ạ ?? | 4 tuần trước |
| Dương | Làm sao để tăng được ạ? | 16/04/2020 |

Hình 41 Trang xuất dữ liệu cần phân tích

3.4 Định hướng phát triển trong tương lai

Trong đồ án III, hiện tại hệ thống phân hỗ trợ phân tích cảm xúc trong tiếng việt có khá nhiều khuyết điểm:

Về mặt mô hình bài toán: Độ chính xác mô hình vẫn chưa thực sự tối ưu nên trong tương lai em sẽ cố gắng cải thiện mô hình để đưa ra kết quả tốt hơn. Ngoài ra tốc độ xử lý của mô hình vẫn còn khá chậm sắp tới em sẽ nghiên cứu và khắc phục nhược điểm này.

Về mặt giao diện và chức năng hệ thống:

- ✓ Xây dựng được giao diện đẹp hơn, dễ dàng sử dụng.
- ✓ Về phần thu thập dữ liệu sẽ cải tiến để cho phép người dùng đưa tên miền website, facebook, zalo,... để có thể crawl dữ liệu bất cứ nguồn nào mong muốn thay vì mặc định như hiện tại.
- ✓ Phần xuất báo cáo vẫn chưa hoàn thiện nên sẽ hoàn thiện phần xuất báo cáo

- ✓ Phần thông tin người sử dụng vẫn chưa được linh động chưa cho phép người dùng được chỉnh sửa: thêm sửa xóa thông tin.

Trước khi hoàn thành báo cáo đồ án này em muốn gửi lời cảm ơn chân thành đến Ts. Lê Chí Ngọc đã tận tình dạy dỗ và truyền đạt cho em những kiến thức quý giá trong suốt quá trình thực hiện đồ án và cả những kỹ năng để có thể sàng lọc nắm bắt những kiến thức đúng và loại bỏ những kiến thức sai lệch trong quá trình tìm hiểu từ kho tri thức rộng lớn trên mạng internet thay vì tiếp thu chúng một cách thụ động không có sàng lọc.

CHỈ MỤC

B

| | |
|---|----|
| Backpropagation Throught time – BPTT | 30 |
| Bài toán tối ưu trong Support Vector Machine (SVM) | 46 |

C

| | |
|-------------------------------|----|
| cổng đầu ra | 63 |
| cổng đầu vào | 63 |
| cổng quên | 62 |
| Convolutional Neural Networks | 55 |

D

| | |
|---------------|----|
| Đơn vị xử lý: | 12 |
|---------------|----|

F

| | |
|-----------|----|
| F1-Scores | 44 |
|-----------|----|

H

| | |
|-------------------|----|
| Hàm bước nhị phân | 15 |
| Hàm đồng nhất | 14 |
| Hàm kết hợp | 14 |
| Hàm kích hoạt | 14 |
| Hàm ReLu | 21 |

| | |
|-----------------------------|----|
| Hàm sigmoid | 16 |
| Hàm sigmoid lưỡng cực | 19 |
| Hàm tanh | 20 |
| Học hồi tiếp thời gian thực | 32 |

K

| | |
|---|----|
| Khoảng cách từ một điểm tới một siêu phẳng | 44 |
|---|----|

L

| | |
|-------------------------------|--------|
| Linear function | 11, 14 |
| Long Short Term Memory – LSTM | 60 |
| Lớp kết nối đầy đủ | 60 |
| Lớp pooling | 58 |
| Lớp ReLU | 59 |

M

| | |
|---------------------------------|----|
| Mạng Jodan | 27 |
| Mạng lan truyền ngược | 23 |
| Mạng lan truyền tiến | 22 |
| Mạng neural hồi tiếp | 25 |
| Mạng neural nhân tạo | 11 |
| Mô hình Continuous Bag of Words | 50 |
| Mô hình Fasttext | 54 |
| Mô hình Skip-gram | 50 |

| | |
|--|----|
| <i>N</i> | |
| Naïve Bayes Classifier _____ | 39 |
| <i>P</i> | |
| Precision _____ | 44 |
| <i>Q</i> | |
| Quy tắc phân lớp Bernoulli Naïve Bayes _____ | 41 |
| Quy tắc phân lớp Gaussian Naïve Bayes _____ | 40 |
| Quy tắc phân lớp Multinomial Naïve Bayes _____ | 40 |
| <i>R</i> | |
| Recall _____ | 44 |

| | |
|---|----|
| <i>T</i> | |
| Tầng phân lớp _____ | 60 |
| Tầng trích xuất vector đặc trưng: _____ | 56 |
| tế bào nhớ (memory cells) _____ | 60 |
| TF-IDF _____ | 43 |
| Tích chập _____ | 56 |
| Trạng thái tế bào – cell state _____ | 61 |
| Túi từ - Bag of Words (BOW) _____ | 42 |
| <i>V</i> | |
| Vector từ (Word vectors) _____ | 49 |
| <i>X</i> | |
| Xử lý ngôn ngữ tự nhiên _____ | 10 |

TÀI LIỆU THAM KHẢO

- [1] Prabhu, "Understanding of Convolutional Neural Network (CNN) — Deep Learning," 4 Mar 2018. [Online]. Available: <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>.
- [2] E. R. M. Ralf C. Staudemeye, "– Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks," September 23, 2019.
- [3] A. C. M. a. S. Guido, Introduction to Machine Learning with Python, FILL IN PRODUCTION EDI-, June 2016.
- [4] T. T. T. Minh Quang Nhat Pham, "A Lightweight Ensemble Method for Sentiment," FPT Technology Research Institute (FTRI), 2016.
- [5] P. J. Werbos, Backpropagation through time: What it does and how to do it., 1990.
- [6] R. J. W. a. D. Zipser., A learning algorithm for continually running fully recurrent neural networks. Neural Computation, jun 1989.
- [7] R. J. W. a. D. Zipser, "A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1(2):270– 280," jun 1989.
- [8] R. J. W. a. D. Zipser, " Gradient-based learning algorithms for recurrent networks and their computational complexity. In Back-propagation: Theory, Architectures and Applications, pages 1–45," L. Erlbaum Associates Inc, jan 1995.
- [9] T. C. a. N. Bayes, Stanford.

- [10] N. R. Kavya Suppala, "Sentiment Analysis Using Naïve Bayes Classifier," June, 2019.
- [11] C. M. Bishop, Pattern recognition and Machine Learning, Springer (2006).
- [12] R. O. P. E. H. a. D. G. S. J. W. & S. Duda, "Pattern classification," 2012.
- [13] I. S. K. c. G. C. J. D. Tomas Mikolov, "Distributed Representations of Words and Phrases and their Compositionality," 16, oct, 2013.
- [14] P. Punyawiwat, "Interns Explain CNN," 8 Feb 2018. [Online]. Available: <https://blog.datawow.io/interns-explain-cnn-8a669d053f8b>.
- [15] J. S. a. F. C. Felix A. Gers, "Learning to Forget: Continual Prediction with LSTM. Neural Computation," oct 2000.
- [16] W. Z. a. I. S. Rafal Jozefowicz, " An empirical exploration of Recurrent Network Architectures. In Proc. of the 32nd Int. Conf on Machine Learning, pages 2342—2350," 2015.
- [17] Keras-team, "Keras Documentation," [Online]. Available: <https://keras.io/layers/convolutional/>.
- [18] M. I. Jordan., Attractor dynamics and parallelism in a connectionist sequential machine. In Proc. of the Eighth Annual Conf. of the Cognitive Science Society, pages 531–546. IEEE Press., jan 1986.
- [19] J. L. Elman., Finding structure in time. Cognitive Science, 14(2):179– 211, mar 1990., mar 1990.