ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÁO CÁO

HỌC PHẦN PROJECT II

Đề tài: Phân loại văn bản đối với phim được bình luận

Đỗ thành Đức

Duc.dt200159@sis.hust.edu.vn

Giảng viên hướng dẫn: Nguyễn Thị Thu Hương

Bộ môn: Project II

Hà Nội, Tháng 06/2023

Mục lục

Chương 1. Phát biểu bài toán

1.1. Mô tả bài toán

- Hiện nay có rất nhiều comment được người dùng đưa ra nhằm đánh giá về một sản phẩm phim nào đó. Việc phân tích thống kê lại những sản phẩm đó là tích cực hay tiêu cực sẽ giúp các nhà làm phim, nhà sản xuất và các chuyên gia điện ảnh hiểu được phản hồi của khán giả và đánh giá được của sản phẩm của mình. Dựa trên các đánh giá này, họ có thể cải thiện sản phẩm và tạo ra những bộ phim tốt hơn trong tương lai.
- Việc phân loại các bình luận của người dùng thành các nhãn tích cực, tiêu cực đóng vai trò quan trọng trong việc cải thiện các sản phẩm phim trong tương lai đồng thời giúp cho khán giả có cái nhìn tổng quan về một bộ phim trước khi xem, cung cấp các thông tin hữu ích cho khán giả xem có nên xem hay không nhằm tiết kiệm thời gian và chi phí bằng cách tránh các bộ phim không phù hợp với họ. Tuy nhiên, việc phân tích một số lượng lớn các bình luận này tốn nhiều công sức nên cần được thực hiện tự động. Vì vậy, bài toán phân tích cảm xúc bình luận (Sentiment Analysis) ra đời nhằm giải quyết nhu cầu trên, giúp cho việc nắm bắt trải nghiệm, đánh giá của khách hàng được thực hiện nhanh chóng, từ đó những người quản lý có thể cải thiện hơn các sản phẩm tiếp theo của mình.

- Trong đề tài lần này, em nghiên cứu xây dựng các bộ dự đoán quan điểm trong các câu bình luận dựa trên nguồn dữ liệu được gán nhãn trên kaggle: IMDB Dataset of 50K Movie Reviews.

1.2. Dữ liệu, Input và Output của bài toán

- Dữ liệu: gồm 50000 review được lấy trên cơ sở dữ liệu phim ảnh IMDB đã được gán nhãn được đăng trên Kaggle.
- Thông tin dữ liệu: IMDB Dataset of 50K Movie Reviews | Kaggle
- Đầu vào: Các review phim đã được gán nhãn được viết bằng tiếng anh trong tập IMDB Dataset of 50K Movie Reviews.
- Đầu ra: Sắc thái tích cực, tiêu cực của review đó.

Chương 2. Động lực lựa chọn phương pháp giải quyết.

- Hiện nay có nhiều cách tiếp cận để giải quyết bài toán phân loại (classification), cụ thể trong bài toán mà em thực hiện là bài toán phân loại 2 lớp. Mỗi phương pháp, cách tiếp cận lại có những ưu, nhược điểm khác nhau, phụ thuộc vào kiến thức và dữ liệu thu thập được.
- Phương pháp tiếp cận của em là áp dụng một số mô hình và so sánh kết quả để tìm ra mô hình phù hợp để giải quyết bài toán. Từ dữ liệu đã được gán nhãn mà tác giả đã thu thập được, chúng em sẽ thực hiện bài toán này theo hai phương pháp là phương pháp Học máy (Support Vector Machine) và phương pháp Học sâu (Convolution Neural Network)

Chương 3. Phương pháp giải quyết bài toán

Sau đây là các phương pháp em thực hiện để giải quyết bài toán

3.1. Thuật toán Học máy: Support Vector Machine

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm siêu phẳng (hyper-plane) phân chia các lớp:

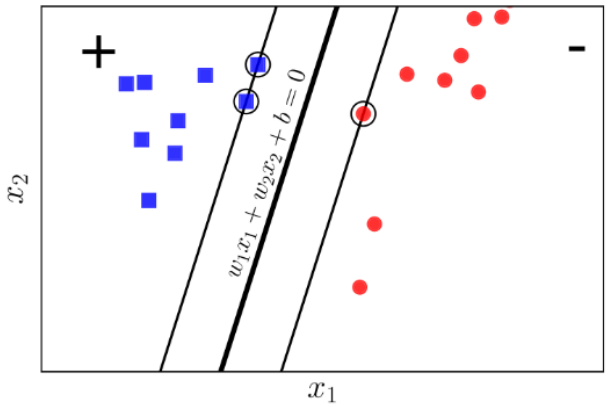
SVM xác định một hàm phân tách tuyến tính:

$$F(x) = \langle w \cdot x \rangle + b$$

Với mỗi x_i:

$$y_i = \begin{cases} 1 & if \quad \langle \mathbf{w} \times \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \quad \langle \mathbf{w} \times \mathbf{x}_i \rangle + b < 0 \end{cases}$$

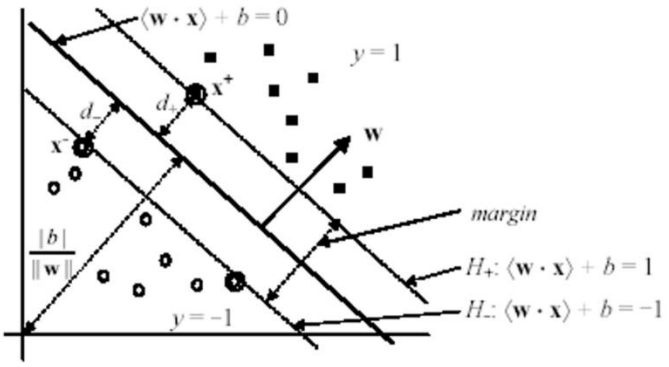
Siêu phẳng phân tách các quan sát thuộc lớp dương và âm: **<w . x> + b = 0**



điểm vuông xanh thuộc class 1, tròn đỏ thuộc class -1 và mặt

$\mathbf{w}^T\mathbf{x} + b = w_1x_1 + w_2x_2 + b = 0$ là mặt phân chia giữa hai classes.

Xác định các siêu phẳng và các thông số:



Mức lề là khoảng cách giữa 2 siêu phẳng lề H_+ và H_-

$$margin = d_+ + d_- = \frac{2}{\|\mathbf{w}\|}$$

Ta cần tìm giá trị cực đại của margin khi đó ta cần giải bài toán:

Tìm w và b sao cho: $margin = \frac{2}{\|w\|}$ đạt cực đại

Với điều kiện

$$\begin{cases} \langle \mathbf{w} \times \mathbf{x_i} \rangle + b \geq 1, \text{if } y_i = 1 \\ \langle \mathbf{w} \times \mathbf{x_i} \rangle + b \leq -1, \text{if } y_i = -1 \end{cases}$$

Với mọi ví dụ huấn luyện $\mathbf{x_i}$ (i=1..r)

Đưa về bài toán:

Cực tiểu hóa: $\frac{\langle w \times w \rangle}{2}$

Với điều kiện: y_i (<w · x_i> + b) ≥ 1, "i=1..r"

Áp dụng phương pháp lagrange với điều kiện KKT để giải vài toán trên.

Tập điều kiện KKT:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{r} \alpha_i y_i \mathbf{x_i} = 0$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} \alpha_i y_i = 0$$

$$y_i(\langle \mathbf{w} \times \mathbf{x_i} \rangle + b) - 1 \geq 0, \ \forall \mathbf{x_i} \ (i = 1..r)$$

$$\alpha_i \geq 0$$

$$\alpha_i(y_i(\langle \mathbf{w} \times \mathbf{x_i} \rangle + b) - 1) = 0$$

Hàm Lagrange:

$$L_P(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\langle \mathbf{w} \times \mathbf{w} \rangle - \sum_{i=1}^{r} \alpha_i [y_i(\langle \mathbf{w} \times \mathbf{x}_i \rangle + b) - 1]$$

trong đó $\alpha_i (\geq 0)$ là các hệ số nhân Lagrange

Đưa về bài toán tối ưu đối ngẫu:

Cực đại hóa: $L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \times \mathbf{x_j} \rangle$

Với điều kiện: $\begin{cases} \sum_{i=1}^{r} \alpha_i y_i = 0 \\ \alpha_i \geq 0, \forall i = 1..r \end{cases}$

Từ điều kiện KKT có thể tính được w*:

$$\mathbf{w}^* = \sum_{i=1}^{r} \alpha_i y_i \mathbf{x}_i = \sum_{\mathbf{x}_i \in SV} \alpha_i y_i \mathbf{x}_i; \qquad \text{bởi vì } \forall \boldsymbol{x_i} \notin SV: \ \alpha_i = 0$$

(bất kỳ) một vectơ hỗ trợ $\boldsymbol{xk}$, ta có

$\alpha_k [y_k(\langle \boldsymbol{w}^*, \boldsymbol{x}_k \rangle + b^*) - 1] = 0$

Nhớ rằng $\alpha_k > 0$ đối với mọi vectơ hỗ trợ $\boldsymbol{x}_k$

Vì vậy: $y_k(\langle \boldsymbol{w}^*, \boldsymbol{x}_k \rangle + b^*) - 1 = 0$

Từ đây, ta tính được giá trị $b^* = y_k - \langle \boldsymbol{w}^*, \boldsymbol{x}_k \rangle$

***) Linear SVM: không phân tách được**

Tức là khi dữ liệu có nhiễu và lỗi thì ta không tìm được lời giải w* và b* như trên. Khi đó ta cần nơi lỏng các điều kiện lề bằng việc sử dụng biến slack >=0

$\langle \mathbf{w} \times \mathbf{x}_i \rangle + b \geq 1 - \xi_i$ đối với các ví dụ có giá trị $y_i = 1$

$\langle \mathbf{w} \times \mathbf{x}_i \rangle + b \leq -1 + \xi_i$ đối với các ví dụ có giá trị $y_i = -1$

Đối với một ví dụ nhiễu/lỗi: $\xi_i > 1$

Các điều kiện mới đối với trường hợp (phân lớp tuyến tính) không thể phân tách được:

$$y_i(\langle \boldsymbol{w} \cdot \boldsymbol{x}_i \rangle + b) \geq 1 - \xi_i, \quad \forall i = 1..r$$
$$\xi_i \geq 0, \quad \forall i = 1..r$$

Cần phải tích hợp lỗi trong hàm tối ưu mục tiêu.

Bằng cách gán giá trị chi phí (cost) cho các lỗi, và tích hợp chi phí này trong hàm mục tiêu mới:

Cực tiểu hóa:

$$\frac{\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle}{2} + C \sum_{i=1}^{r} \xi_{\mathrm{i}}$$

trong đó C (>0) là tham số xác định mức độ phạt (penalty degree) đối với các lỗi

→ Giá trị C càng lớn, thì mức độ phạt càng cao đối với các lỗi

Khi đó bài toán trở thành:

$$\frac{\langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle}{2} + C \sum_{i=1}^{r} \xi_{\mathrm{i}}$$

Cực tiểu hóa:

Với điều kiện:

$$\begin{cases} y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, & \forall i = 1..r \\ \xi_i \geq 0, & \forall i = 1..r \end{cases}$$

Bài toán tối ưu mới này được gọi là **Soft-margin SVM**

Giải bài toán này tương đương với việc cực tiểu hóa hàm:

$$\left[\frac{1}{r}\sum_{i=1}^{r} \max(0, 1 - y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b))\right] + \lambda \|\boldsymbol{w}\|_2^2$$

Trong đó: $\max(0, 1 - y_i (w \cdot x_i + b)) + \lambda \|w\|_2^2$ thường được gọi là *Hinge loss* với λ là tham số

Biểu thức tối ưu lagrange:

$$L_P = \frac{1}{2}\langle \mathbf{w} \cdot \mathbf{w} \rangle + C\sum_{i=1}^{r} \xi_i - \sum_{i=1}^{r} \alpha_i [y_i(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{r} \mu_i \xi_i$$

trong đó α_i (≥0) và μ_i (≥0) là các hệ số nhân Lagrange

Tập điều kiện KKT:

$$\frac{\partial L_P}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{r} \alpha_i y_i \mathbf{x_i} = 0$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{r} \alpha_i y_i = 0$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i = 0, \quad "i = 1..r$$

$$y_i\left(\langle \mathbf{w} \times \mathbf{x_i} \rangle + b\right) - 1 + \xi_i \geq 0, \quad "i = 1..r$$

$$\xi_i \geq 0$$

$$\alpha_i \geq 0$$

$$\mu_i \geq 0$$

$$\alpha_i\left(y_i\left(\langle \mathbf{w} \times \mathbf{x_i} \rangle + b\right) - 1 + \xi_i\right) = 0$$

$$\mu_i \xi_i = 0$$

Vì $C - \alpha_i - \mu_i = 0,$ mà $\mu_i \geq 0$ nên $\alpha_i \leq C$

Biểu thức đối ngẫu:

Cực đại hóa: $L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j y_i y_j \langle \mathbf{x_i} \cdot \mathbf{x_j} \rangle$

Với điều kiện: $\begin{cases} \sum_{i=1}^{r} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \quad \forall i = 1..r \end{cases}$

Từ các biểu thức ta suy ra kết luận:

If	$\alpha_i = 0$	then $y_i\left(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b\right) \geq 1,$	and $\xi_i = 0$
If	$0 < \alpha_i < C$	then $y_i\left(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b\right) = 1,$	and $\xi_i = 0$
If	$\alpha_i = C$	then $y_i\left(\langle \mathbf{w} \cdot \mathbf{x_i} \rangle + b\right) < 1,$	and $\xi_i > 0$

Lời giải được dựa trên rất ít (**sparse**) các giá trị α_i

+ Rất nhiều ví dụ học nằm ngoài khoảng lề (margin area), và chúng có giá trị α_i bằng 0

+ Các ví dụ nằm trên lề ($y_i (w \cdot x_i + b)) = 1$ - chính là các vector hỗ trợ), thì có giá trị α_i khác 0 ($0 < \alpha_i < C$)

+ Các ví dụ nằm trên lề ($y_i (w \cdot x_i + b)) < 1$ - là các ví dụ nhiễu/lỗi), thì có giá trị α_i = C

Nếu không có đặc điểm thưa thới (sparsity) này, thì phương pháp SVM không thể hiệu quả đối với các tập dữ liệu lớn.

Ranh giới quyết định phân lớp chính là siêu phẳng:

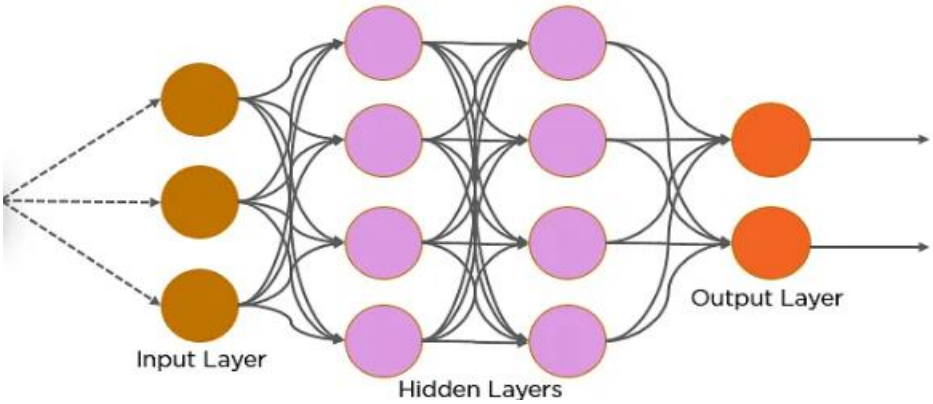
$$\langle \mathbf{w}^* \cdot \mathbf{x} \rangle + b^* = \sum_{i=1}^{r} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b^* = 0$$

SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu. Nó là một công cụ đắc lực cho các bài toán về xử lý ảnh, phân loại văn bản, phân tích quan điểm. Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng Kernel function khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn. Với rất nhiều những ưu điểm như: Xử lý trên không gian có số chiều cao, tiết kiệm bộ nhớ, linh hoạt. Bên cạnh đó SVM cũng có những nhược điểm như: Với bài toán có số chiều dữ liệu lớn (trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tồi), chưa thể hiện rõ tính xác suất xuất hiện của dữ liệu.

3.2. Thuật toán Học sâu Convolution Neural Network

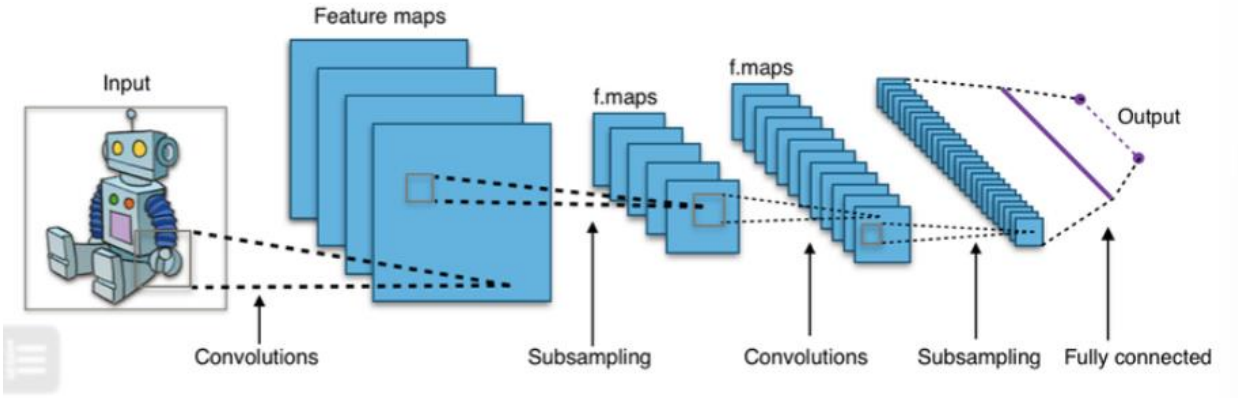
Là kiến trúc mạng neural sâu được thiết kế để xử lý dữ liệu có cấu trúc lưới, sử dụng một phép toán convolution thay cho phép nhân ma trận tổng quát được ứng dụng nhiều trong phân loại hình ảnh, phân tích hình ảnh hay xử lý ngôn ngữ tự nhiên, …

Cấu trúc của CNN tương tự gồm 3 phần chính: Input layer, Hidden layer và Output layer



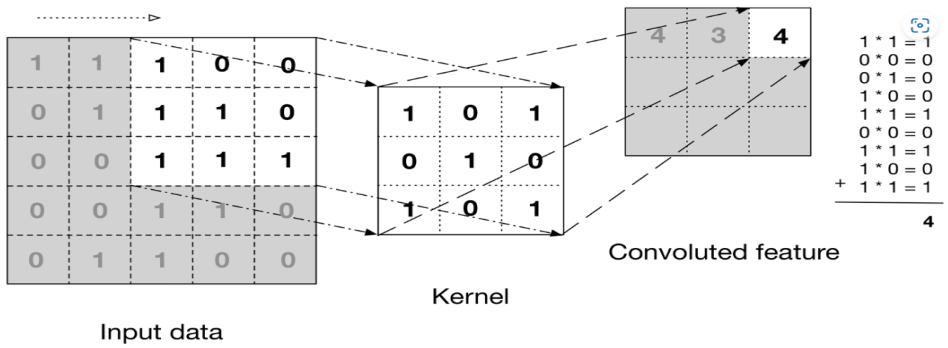
Cấu trúc mạng CNN: là một tập hợp các lớp convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Các layer liên kết với nhau thông qua cơ chế convolution, các layer tiếp theo là kết quả convoluton từ layer trước đó tạo nên các kết nối cục bộ. Mỗi lớp sử dụng một filter khác nhau và kết hợp lại.

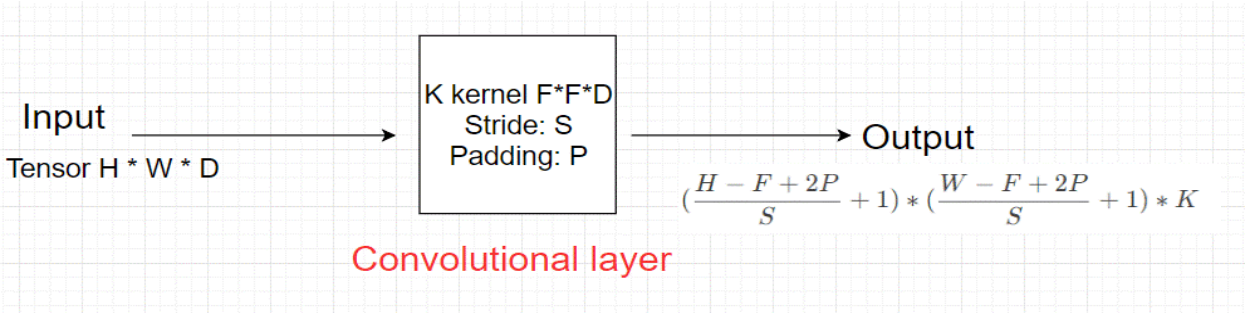


Các lớp trong CNN:

Lớp convolution: áp dụng phép tích chập để trích xuất đặc trưng của ảnh hoặc dữ liệu đầu vào. Mỗi nơ-ron tích chập kết nối cục bộ với dữ liệu đầu vào được trượt từ trái sang phải và từ trên xuống dưới để sinh ra bản đồ kích hoạt (activation map).

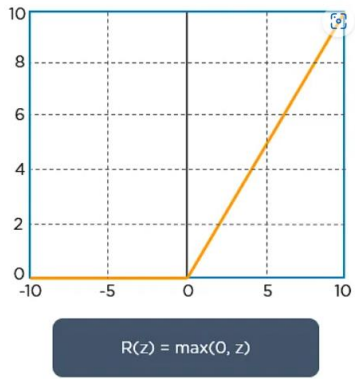


- Đầu ra của phép convolution là cũng một matrix, với mỗi kernel khác nhau sẽ tạo ra các đặc trưng khác nhau cho dữ liệu. Có thể dùng nhiều kernel để có thể tạo ra nhiều đầu ra cho lớp này.
- muốn bảo toàn kích thước dữ liệu ta cần thêm viền (padding) bởi các số 0 vào ma trận đầu vào.
- Trong trường hợp tổng quát với tensor (kích thước H*W*D)và kernel (F*F*D), bước nhảy stride: S, đệm padding: P. khi đó:

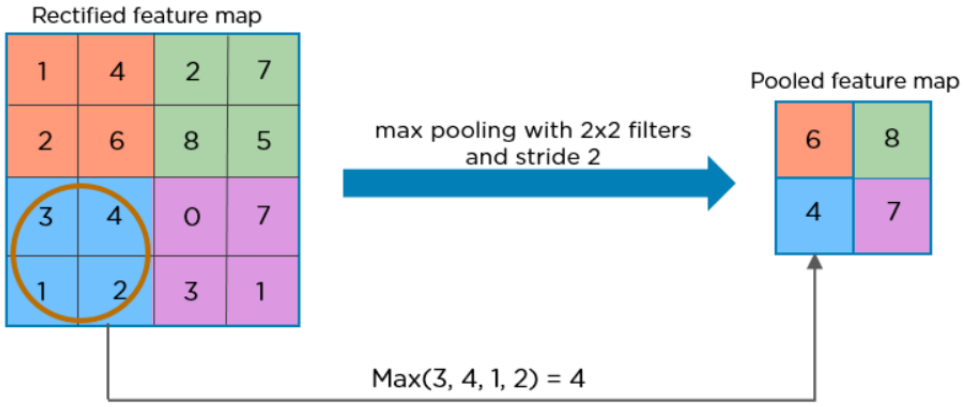


Mỗi kernel có kích thước F*F*D và có 1 hệ số bias, nên tổng parameter của 1 kernel là F*F*D + 1. Mà convolutional layer áp dụng K kernel => Tổng số parameter trong layer này là K * (F*F*D + 1).

Lớp activation: là một phần quan trọng để tạo tính phi tuyến tính và khả năng học các đặc trưng phức tạp được sử dụng sau mỗi convolution layer. Một số hàm kích hoạt phổ biến như ReLU, sigmoid, tanh, softmax,… Ví dụ áp dụng hàm kích hoạt (như ReLU) để tạo độ dốc cho mạng.



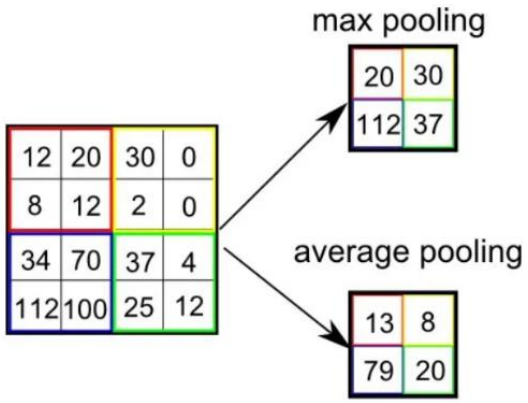
Lớp pooling: thường dùng giữa các convolution layer để giảm kích thước của đầu vào trong một cửa sổ nhưng vẫn giữ được các thuộc tính quan trọng.



Hoạt động độc lập trên từng bản đồ kích hoạt, lớp này biểu diễn bất biến đối với các thay đổi tịnh tiến hoặc biến dạng của dữ liệu vào.

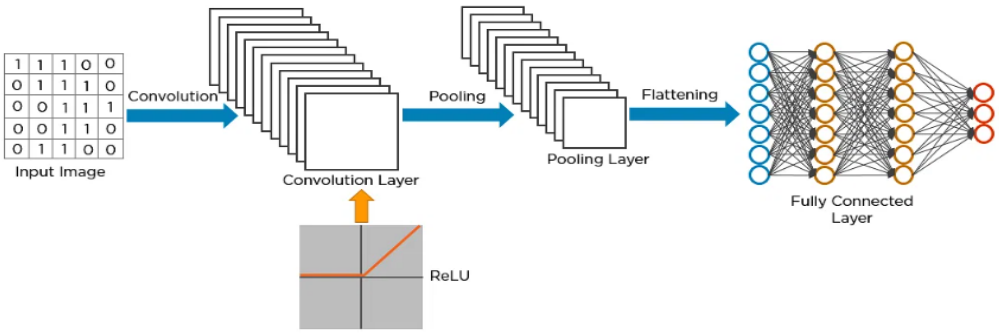
Pooling size kích thước K*K. Input của pooling layer có kích thước H*W*D, ta tách ra làm D ma trận kích thước H*W. Với mỗi ma trận, trên vùng kích thước K*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả

Có hai loại phổ biến cho lớp này là max pooling và average pooling:



Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

Lớp fully connected: nhận đầu vào là một ma trận đặc trưng đã được duỗi và thực hiện các phép tính toán tuyến tính để đưa ra kết quả cuối cùng.



Chương 4. Mô tả tập dữ liệu đánh giá

- Dữ liệu: bao gồm 50000 review được chia thành Training dataset gồm 35000 review đã được gắn nhãn, bộ Testing dataset gồm 15000 đoạn review. Do dữ liệu không được phân chia ngay từ ban đầu nên sẽ trích xuất 1 phần dữ liệu thuộc dataset làm test set để đánh giá.

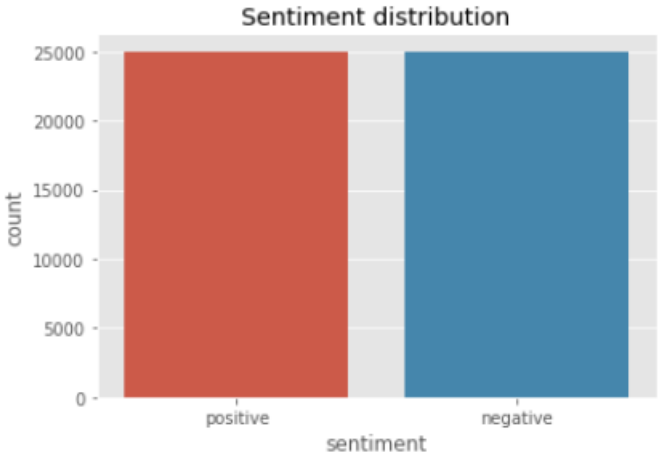
Dataset gồm bình luận và nhãn của bình luận.

- Bình luận tích cực ví dụ: "If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it.
Great Camp!!! " được gắn nhãn Positive.
- Bình luận tiêu cực ví dụ: "Besides being boring, the scenes were oppressive and dark. The movie tried to portray some kind of moral, but fell flat with its message. What were the redeeming qualities?? On top of that, I don't think it could make librarians look any more unglamorous than it did." được gắn nhãn negative.
- Thông tin dữ liệu: IMDB Dataset of 50K Movie Reviews | Kaggle
- Dữ liệu ban đầu ở dạng CSV với 2 trường dữ liệu review và sentiment:

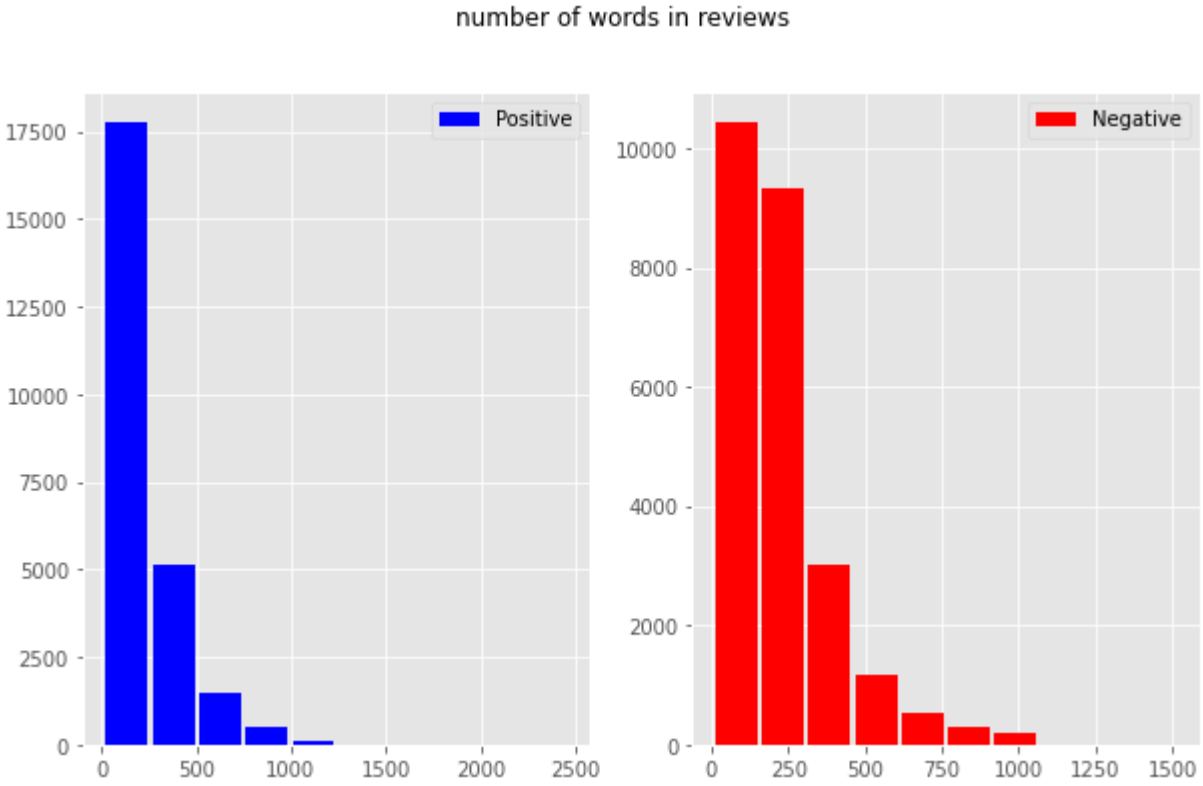
	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

- Phân bố nhãn của tập dữ liệu: Nhận thấy dataset gồm 2 class cân bằng nên ta không cần sinh thêm dữ liệu nữa.

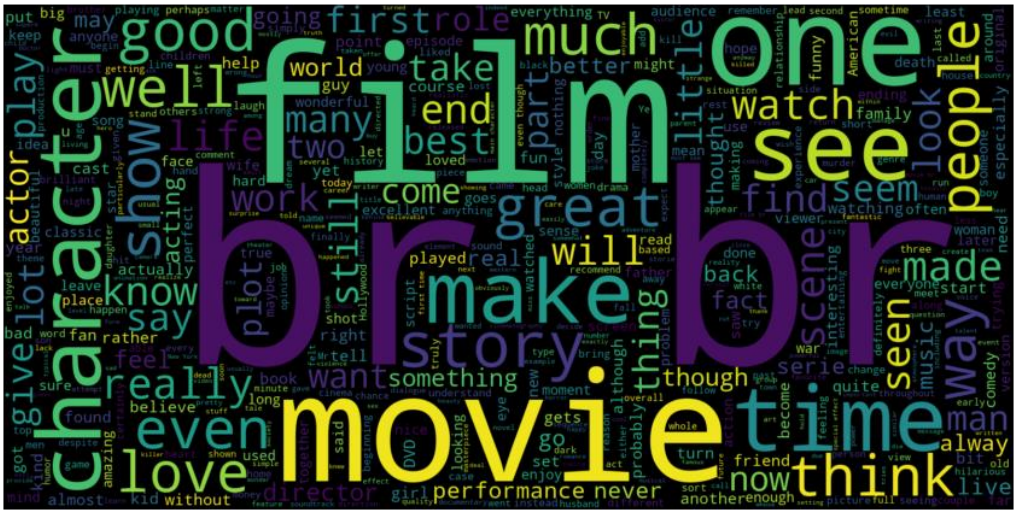


- Phân bố từ trong dữ liệu:

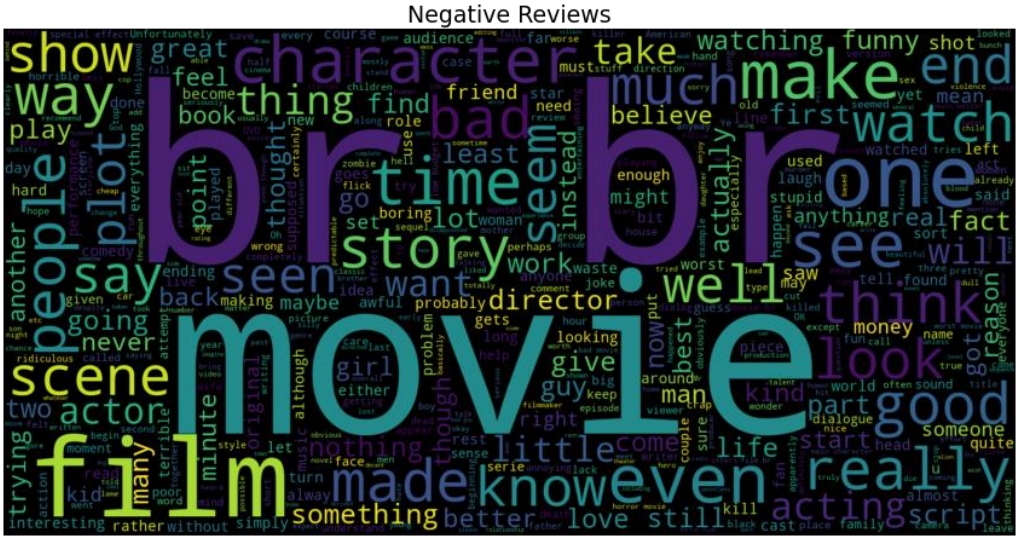


- Các từ vựng xuất hiện thường xuyên trong tập dữ liệu positive và negative

Positive Reviews



Negative Reviews



- Hiện trạng dữ liệu vẫn còn nhiều nhiễu, chưa được làm sạch như chứa các thẻ html, các ký tự đặc biệt, … Trước khi sử dụng model huấn luyện, ta cần làm sạch dữ liệu trước.

Chương 5. Mô tả thiết lập thí nghiệm, các độ đo đánh giá, giá trị các siêu tham số

5.1 Tiền xử lý dữ liệu

- Chuyển các nhãn Positive thành 1 và negative thành 0

```
df.sentiment.replace({'positive': 1, 'negative': 0}, inplace=True)
df.head()
```

	review	sentiment	word count
0	One of the other reviewers has mentioned that ...	1	307
1	A wonderful little production. The...	1	162
2	I thought this was a wonderful way to spend ti...	1	166
3	Basically there's a family where a little boy ...	0	138
4	Petter Mattei's "Love in the Time of Money" is...	1	230

- Chuyển đổi về chữ thường: Chuyển đổi toàn bộ văn bản trong bộ dữ liệu về chữ thường giúp cho việc xử lý và phân loại được dễ dàng hơn.
- Loại bỏ các ký tự đặc biệt, số và chữ cái viết tắt: loại bỏ các đoạn văn bản có thẻ HTML ‘
’, các đường link tới trang web, các ký tự đặc biệt như @, # ; loại bỏ các ký tự không phải chữ và số. Ở đây ta áp dụng thư viện re với các biểu thức chính quy.
- Tách văn bản thành các từ đơn (tokenize) và loại bỏ stop words: Stop words là những từ thường xuất hiện trong văn bản nhưng không mang ý nghĩa quan trọng, ví dụ như "a", "an", "the", "and", "in",... Loại bỏ các stop words giúp giảm kích thước của bộ dữ liệu và tăng độ chính xác của mô hình phân loại. Các stop word được lấy từ thư viện nltk với ngôn ngữ là tiếng anh:

```
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

- Chuyển về nguyên dạng (stemming, lemmatization) áp dụng từ nltk.stem

Để làm các bước trên ta sử dụng các thư viện sẵn có như **re** để loại bỏ các từ, **nltk** để loại bỏ stop word hay đưa về nguyên dạng của từ, từ đó ta xây dựng

được hàm data_processing nhận dữ liệu đầu vào là các câu review và trả về là các review ddax qua xử lý như sau:

```
def data_processing(text):
    text = text.lower()
    text = re.sub('<br />', '', text)
    text = re.sub(r"https\S+|www\S+|http\S+", '', text, flags=re.MULTILINE)
    text = re.sub(r'\@\w+|\#', '', text)
    text = re.sub(r'[^\w\s]', ' ', text)
    text_tokens = word_tokenize(text)
    filtered_text = [word for word in text_tokens if word not in stop_words]
    stemmer = PorterStemmer()
    stemmed_text = [stemmer.stem(word) for word in filtered_text]
    lemmatizer = WordNetLemmatizer()
    lemmatized_text = [lemmatizer.lemmatize(word) for word in stemmed_text]
    return " ".join(lemmatized_text)
```

- Áp dụng hàm trên ta được một tập dữ liệu mới đã được xử lý:

```
df.review = df['review'].apply(data_processing)
df['word count'] = df['review'].apply(lambda x: len(str(x).split(" ")))
df.head(20)
```

	review	sentiment	word count
0	one review mention watch 1 oz episod hook righ...	1	162
1	wonder littl product film techniqu unassum old...	1	86
2	thought wonder way spend time hot summer weeke...	1	85
3	basic famili littl boy jake think zombi closet...	0	66
4	petter mattei love time money visual stun film...	1	125
5	probabl time favorit movi stori selfless sacri...	1	56
6	sure would like see resurrect date seahunt ser...	1	76
7	show amaz fresh innov idea 70 first air first ...	0	83
8	encourag posit comment film look forward watch...	0	64
9	like origin gut wrench laughter like movi youn...	1	17
10	phil alien one quirki film humour base around ...	0	49

- Tiếp theo ta loại bỏ các review có sự trùng lặp và xóa hàng đó đi:

```
duplicated_count = df.duplicated().sum()
print("Number of duplicated rows: ", duplicated_count)
```

```
Number of duplicated rows:  428
```

```
df=df.drop_duplicates('review')
```

Qua các bước, em đã thu được dữ liệu được chuẩn hóa, em tiến hành trực quan hóa và xem xét lại dữ liệu sau khi đã làm sạch đồng thời lưu dữ liệu đã xử lý để xử lý cho các lần tiếp theo.

5.2 Huấn luyện mô hình Machine Learning với SVM

- Sau khi trải qua các bước tiền xử lý dữ liệu, em tiến hành chia dữ liệu thành tập train chiếm 70% và tập test chiếm 30%

```
X = df['review']
Y = df['sentiment']
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
print("Training set has {} examples.".format(x_train.shape[0]))
print("Testing set has {} examples.".format(x_test.shape[0]))
```

✓ 0.0s

```
Training set has 34700 examples.
Testing set has 14872 examples.
```

- Dữ liệu trước khi được sử dụng cho huấn luyện sẽ được thực hiện chuyển đổi các câu comment thành vector với kỹ thuật **tf-idf (**frequency – inverse document frequency). Tf-idf là trọng số của một từ trong văn bản thu được qua thống kê thể hiện mức độ quan trọng của từ này trong một văn bản, mà bản thân văn bản đang xét nằm trong một tập hợp các văn bản.
- Đầu tiên sử dụng CountVectorizer để transform comment thành vector. Mỗi string sẽ chuyển đổi thành 1 vector có độ dài d (số từ xuất hiện ít nhất 1 lần trong corpus), giá trị của thành phần thứ i trong vector chính là số lần từ đó xuất hiện trong string. Rồi sử dụng TfidfTransformer.

```
steps.append(('CountVectorizer', CountVectorizer(ngram_range=(1,5) ,max_df=0.5, min_df=5)))
steps.append(('tfidf', TfidfTransformer(use_idf=False, sublinear_tf = True,norm='l2',smooth_idf=True)))
```

CountVectorizer: sử dụng từ scikit-learn nhằm chuyển văn bản thành ma trận tần suất của các từ

- ngram_range=(1, 5): Cho phép tạo ra các từ đơn và các n-gram từ 1 đến 5 từ.
- max_df=0.5: Loại bỏ các từ xuất hiện trong hơn 50% các văn bản.
- min_df=5: Loại bỏ các từ xuất hiện trong ít hơn 5 văn bản.

TfidfTransformer: bước này nhằm chuyển ma trận tần suất thành ma trận TF-IDF

Các tham số được cung cấp ở đây:

- use_idf=False: Không sử dụng IDF (Inverse Document Frequency).
- sublinear_tf=True: Áp dụng chế độ sublinear TF, trong đó tần suất xuất hiện của một từ được biểu diễn bằng logarit của nó.
- norm='l2': Chuẩn hóa ma trận TF-IDF theo chuẩn L2.
- smooth_idf=True: Áp dụng kỹ thuật smoothing cho IDF.

Việc xây dựng mô hình áp ụng LinearSVC từ thư viện scikit learn:

```
steps.append(('classifier',  LinearSVC(dual=False, penalty="l2", loss='squared_hinge')))
```

- dual=False: Tối ưu hóa mô hình với số lượng biến độc lập lớn hơn số lượng quan sát (thích hợp cho kích thước dữ liệu lớn).
- penalty="l2": Sử dụng hồi quy Ridge (L2) để giảm overfitting.

  L2 regularization = C * $\|w\|^2$

  Trong đó: C là hệ số điều chỉnh thường được lựa chọn trước huấn luyện và w là vector các tham số mô hình
- loss='squared_hinge': Sử dụng squared hinge loss trong quá trình tối ưu hóa mô hình. Sử dụng hàm mất mát squared hinge: max(0, 1 – y * (w * x + b) (phiên bản của hinge loss với mất mát bình phương), có thể mang lại kết quả tốt hơn so với hinge (do nó tạo một đường phân chia tốt hơn và nhạy hơn đối với nhiễu).

Mô hình xây dựng một pipeline chứa khởi tạo các bước trên:

```
steps = []
steps.append(('CountVectorizer', CountVectorizer(ngram_range=(1,5) ,max_df=0.5, min_df=5)))
steps.append(('tfidf', TfidfTransformer(use_idf=False, sublinear_tf = True,norm='l2',smooth_idf=True)))
steps.append(('classifier',  LinearSVC(dual=False, penalty="l2", loss='squared_hinge')))
clf = Pipeline(steps)
start = time.time()
clf.fit(x_train, y_train)
end = time.time()
y_pred = clf.predict(x_test)
report = metrics.classification_report(y_test, y_pred, labels=[1,0], digits=3)
```

- Dữ liệu huấn luyện (x_train và y_train) được sử dụng để huấn luyện mô hình.
- Thời gian bắt đầu và kết thúc huấn luyện được ghi lại để đo thời gian huấn luyện.

- Mô hình được sử dụng để dự đoán nhãn cho dữ liệu kiểm tra (x_test).
- Các kết quả dự đoán (y_pred) được sử dụng để tính toán các độ đo đánh giá mô hình như classification_report.
- Classification_report tính toán precision, recall, f1-score và support cho từng nhãn.

```
print('Train', report)
print('Time train: ', end - start)
```

```
Train              precision    recall  f1-score   support

           1      0.889     0.911     0.900      7427
           0      0.909     0.887     0.898      7445

    accuracy                          0.899     14872
   macro avg      0.899     0.899     0.899     14872
weighted avg      0.899     0.899     0.899     14872

Time train:  59.158746004104614
```

Đánh giá với phương pháp cross-validation với hàm cross_val_score thực hiện trên mô hình clf.

```
cross_score = cross_val_score(clf, x_train, y_train, cv=5)
print("CROSSVALIDATION 5 FOLDS: %0.4f (+/- %0.4f)" %
      (cross_score.mean(), cross_score.std() * 2))
```

```
CROSSVALIDATION 5 FOLDS: 0.8952 (+/- 0.0068)
```

Trong trường hợp này sử dụng cross-validation với 5 folds và được lưu trữ trong biến cross_score. cross_score.mean() là chỉ số đánh giá trung bình của mô hình và cross_score.std() là tính độ lệch chuẩn đánh giá độ biến thiên của mô hình, khi độ lệch chuẩn nhân 2 ta được khoảng tin cậy và in kết quả.

5.3. Huấn luyện mô hình Deep learning với CNN

- Nhóm tiến hành chia dữ liệu thành ba tập train, test và val theo tỉ lệ 63%, 30% và 7%

```
x_train, x_test, y_train, y_test = train_test_split(
    df['review'], df['sentiment'], test_size=0.3, random_state=42)
x_train, x_valid, y_train, y_valid = train_test_split(
    x_train, y_train, test_size=0.1, random_state=42)
```

✓ 0.1s

```
print("Training set has {} examples.".format(x_train.shape[0]))
print("Validation set has {} examples.".format(x_valid.shape[0]))
print("Testing set has {} examples.".format(x_test.shape[0]))
```

✓ 0.3s

```
Training set has 31230 examples.
Validation set has 3470 examples.
Testing set has 14872 examples.
```

- Các bước áp dụng model của nhóm vào bài toán là:
  - Vector hóa từ
  - Xây dựng mô hình
  - Huấn luyện mô hình
- Nhóm sử dụng Tokenizer để tách comment thành các từ, xây dựng bộ từ điển với vocab_size = 72243. Mỗi comment sẽ được vector hóa thành 1 vector có độ dài maxlen = 500.

```
from keras.preprocessing import text
tokenizer = text.Tokenizer(num_words=config.vocab_size, split=' ')
tokenizer.fit_on_texts(x_train)
X_train = tokenizer.texts_to_sequences(x_train)
X_valid = tokenizer.texts_to_sequences(x_valid)
X_test = tokenizer.texts_to_sequences(x_test)
X_train = pad_sequences(X_train, maxlen=config.maxlen)
X_valid = pad_sequences(X_valid, maxlen=config.maxlen)
X_test = pad_sequences(X_test, maxlen=config.maxlen)

X_train.shape, X_valid.shape, X_test.shape
```

✓ 51.9s

```
((31230, 500), (3470, 500), (14872, 500))
```

Ví dụ 1 vector comment

```
array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
...
         154,   338,    17, 11662,   338,   680,   919,   672,    52,
         216,   437,     8,    75,  7572,   503,    20,  5454,     3,
          90,   421,   494,   521,  5050,    20,  1536,    13,   256,
          61,   102,    27,   735,  1310,   109,    29,   121,   256,
         112,     1,   527,   628,   788])
```

Xây dựng mô hình: Tạo một mô hình Sequential: `model = Sequential()` tạo một đối tượng mô hình tuần tự (Sequential model) bằng cách khởi tạo đối tượng `Sequential()`.

```
model = Sequential()
model.add(Embedding(config.vocab_size,
          config.embedding_dims, input_length=config.maxlen))
model.add(Conv1D(config.filters, config.kernel_size,
          padding='valid', activation='relu', strides=1))
model.add(GlobalMaxPooling1D())
model.add(Dropout(0.2))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam', metrics=['accuracy'])
model.summary()
time_start = time.time()
model.fit(X_train, y_train, batch_size=config.batch_size, epochs=config.epochs,
          validation_data=(X_valid, y_valid), callbacks=[WandbCallback()])
time_end = time.time()
model.save('CNN1.h5')
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_2 (Embedding)      (None, 500, 300)          21672900

 conv1d_2 (Conv1D)            (None, 498, 64)           57664

 global_max_pooling1d_2 (Glo  (None, 64)               0
 balMaxPooling1D)

 dropout_1 (Dropout)          (None, 64)                0

 dense_4 (Dense)              (None, 16)                1040

 dense_5 (Dense)              (None, 1)                 17

=================================================================
Total params: 21,731,621
Trainable params: 21,731,621
Non-trainable params: 0
```

- Kiến trúc của mô hình bao gồm các tầng chính là:

● Embedding layer: `model.add(Embedding(config.vocab_size, config.embedding_dims, input_length=config.maxlen))`: Thêm lớp nhúng (Embedding layer) vào mô hình. Lớp này được sử dụng để chuyển đổi các số nguyên đại diện cho từng từ thành một vectơ nhúng. `config.vocab_size` là kích thước từ điển từ vựng, `config.embedding_dims` là số chiều của vectơ nhúng, và `config.maxlen` là độ dài tối đa của câu.

● Convolution: `model.add(Conv1D(config.filters, config.kernel_size, padding='valid', activation='relu', strides=1))`: Thêm lớp Conv1D (Convolutional layer) vào mô hình. Lớp này thực hiện việc áp dụng bộ lọc (filter) và hoạt động tích chập trên dữ liệu đầu vào. `config.filters` là số lượng bộ lọc, `config.kernel_size` là kích thước của bộ lọc, `padding='valid'` chỉ định không sử dụng padding, `activation='relu'` là hàm kích hoạt ReLU được áp dụng sau khi tích chập và `strides=1` là bước di chuyển của bộ lọc.

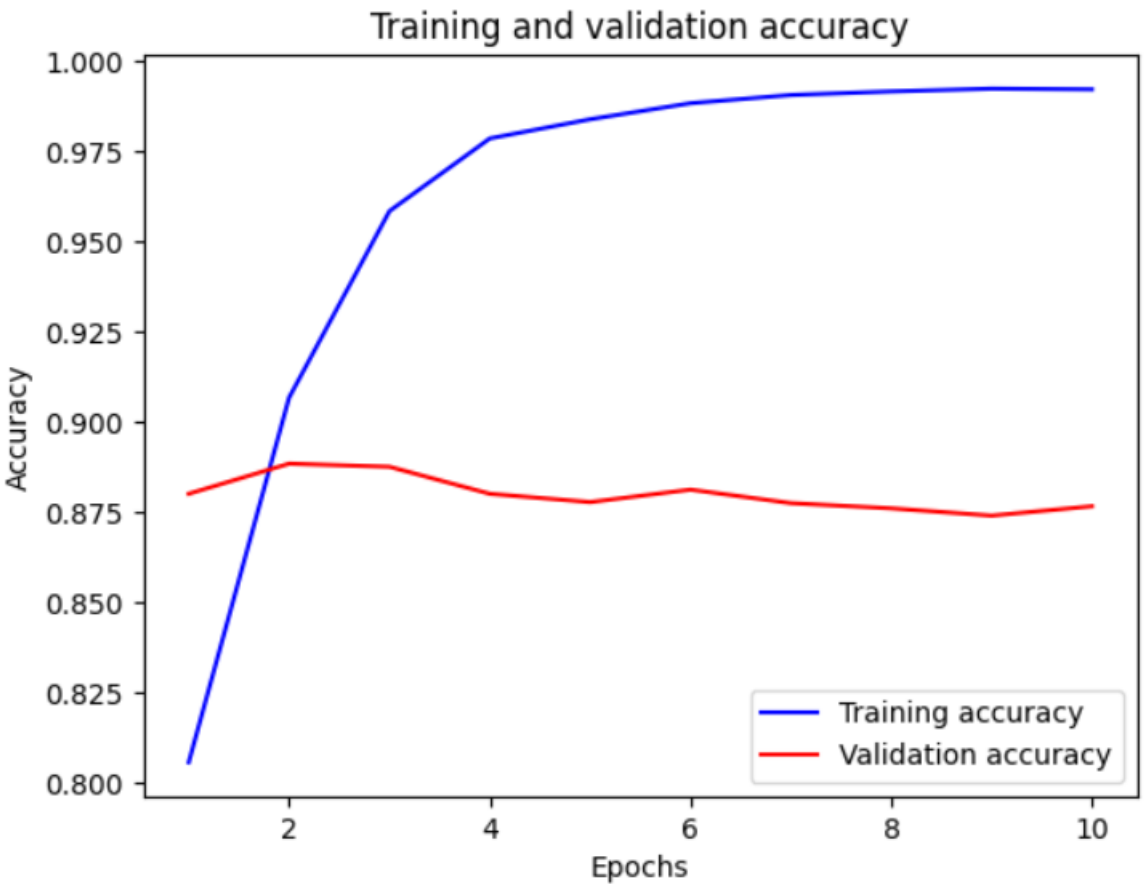
● Max_pooling: `model.add(GlobalMaxPooling1D())`: Thêm lớp GlobalMaxPooling1D vào mô hình. Lớp này thực hiện phép pooling theo chiều dọc trên các đặc trưng (features) đã được tính toán từ lớp Conv1D để giảm kích thước dữ liệu.

- Fully Connected: `model.add(Dense())`: ở đây em thêm 2 lớp Fully Connected , lớp đầu sử dụng hàm Relu với 16 đầu ra và lớp sau với 1 đơn vị đầu ra và hàm kích hoạt sigmoid để dự đoán xác suất cho lớp nhãn positive (1) hoặc negative (0).
- Dropout: ở đây em sử dụng thêm dropout để ngăn overfitting tron mạng neural bằng cách loại bỏ ngẫu nhiên một số node và trọng số tương ứng trong quá trình huấn luyện.
    - Compile mô hình: `model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])` định nghĩa hàm mất mát (loss function) là binary cross-entropy, trình tối ưu hóa là Adam và các chỉ số đánh giá là accuracy.
    - Hiển thị thông tin về mô hình: `model.summary()` in ra thông tin tổng quan về mô hình, bao gồm kiến trúc của từng lớp và số lượng tham số trong mô hình.
    - Huấn luyện mô hình: `model.fit(X_train, y_train, batch_size=config.batch_size, epochs=config.epochs, validation_data=(X_valid, y_valid), callbacks=[WandbCallback()])` thực hiện quá trình huấn luyện mô hình trên dữ liệu huấn luyện `X_train` và nhãn `y_train`. `config.batch_size` là kích thước của từng batch trong quá trình huấn luyện, `config.epochs` là số lượng epoch, `validation_data=(X_valid, y_valid)` chỉ định dữ liệu validation để đánh giá mô hình sau mỗi epoch. `callbacks=[WandbCallback()]` là một danh sách các callbacks được sử dụng trong quá trình huấn luyện.
    - Lưu mô hình: `model.save('CNN1.h5')` lưu mô hình đã được huấn luyện vào file có tên 'CNN1.h5'.

- Huấn luyện mô hình với
- epoch: 10
- batch_size: 32

- loss_function: binary_crossentropy
- optimizer: adam
- Quá trình huấn luyện:

```
Epoch 1/10
974/976 [============================>.] - ETA: 0s - loss: 0.4162 - accuracy: 0.8055WARNING:absl:Found untraced functions such as _jit_
wandb: Adding directory to artifact (/content/wandb/run-20230609_041902-fon4g32f/files/model-best)... Done. 8.0s
976/976 [==============================] - 139s 141ms/step - loss: 0.4158 - accuracy: 0.8057 - val_loss: 0.2869 - val_accuracy: 0.8801
Epoch 2/10
976/976 [==============================] - ETA: 0s - loss: 0.2315 - accuracy: 0.9068WARNING:absl:Found untraced functions such as _jit_
wandb: Adding directory to artifact (/content/wandb/run-20230609_041902-fon4g32f/files/model-best)... Done. 12.7s
976/976 [==============================] - 61s 63ms/step - loss: 0.2315 - accuracy: 0.9068 - val_loss: 0.2710 - val_accuracy: 0.8885
Epoch 3/10
976/976 [==============================] - 26s 27ms/step - loss: 0.1160 - accuracy: 0.9585 - val_loss: 0.2934 - val_accuracy: 0.8876
Epoch 4/10
976/976 [==============================] - 17s 17ms/step - loss: 0.0644 - accuracy: 0.9786 - val_loss: 0.3602 - val_accuracy: 0.8801
Epoch 5/10
976/976 [==============================] - 14s 14ms/step - loss: 0.0457 - accuracy: 0.9840 - val_loss: 0.3895 - val_accuracy: 0.8778
Epoch 6/10
976/976 [==============================] - 14s 14ms/step - loss: 0.0338 - accuracy: 0.9884 - val_loss: 0.4305 - val_accuracy: 0.8813
Epoch 7/10
976/976 [==============================] - 13s 14ms/step - loss: 0.0273 - accuracy: 0.9907 - val_loss: 0.4891 - val_accuracy: 0.8775
Epoch 8/10
976/976 [==============================] - 12s 13ms/step - loss: 0.0257 - accuracy: 0.9917 - val_loss: 0.5159 - val_accuracy: 0.8761
Epoch 9/10
976/976 [==============================] - 10s 10ms/step - loss: 0.0223 - accuracy: 0.9924 - val_loss: 0.5347 - val_accuracy: 0.8741
Epoch 10/10
976/976 [==============================] - 11s 11ms/step - loss: 0.0224 - accuracy: 0.9923 - val_loss: 0.5551 - val_accuracy: 0.8767
```



Độ chính xác trên tăng lên đến 88% sau 6 epochs! Tuy nhiên sau đó model liên tục giảm dần, còn độ chính xác tập train thì vẫn tiếp tục tăng. Do vậy có thể model đã bị "overfitting" sau epoch thứ 6.

Model hoạt động tốt nhất ở epoch thứ 6

5.4. Độ đo

- Để đánh giá hiệu quả mô hình, nhóm sử dụng chỉ số Accuracy và các chỉ số liên quan như Precision, Recall, F1
- Precision được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm mô hình dự đoán là Positive. Recall được định nghĩa là tỉ lệ số điểm Positive mô hình dự đoán đúng trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu).
- Precision càng cao, tức là số điểm mô hình dự đoán là positive đều là positive càng nhiều. Precision = 1, tức là tất cả số điểm mô hình dự đoán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.
- Recall càng cao, tức là số điểm là positive bị bỏ sót càng ít. Recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.
- F1-score là trung bình điều hòa (harmonic mean) của precision và recall (giả sử hai đại lượng này khác 0). F1-score được tính theo công thức:

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}}$$

Chương 6. Kết quả thực nghiệm

STT	Mô hình	Accuracy	Time Training
1	SVM	0.899	41.933
2	CNN	0.866	317.869

Chương 7. Kết luận và hướng phát triển

7.1 Kết luận

- Các mô hình càng phức tạp thì có độ hiệu quả càng cao, nhưng kèm theo đó thời gian training và thời gian dự đoán lâu hơn.
- Nhìn kết quả thực nghiệm, ta có thể thấy khi sử dụng mô hình phức tạp như CNN cũng không nâng cao được độ chính xác. Điều đó nói rằng nếu muốn tăng độ chính xác cho bài toán, ta cần tập trung vào cải thiện chất lượng của dữ liệu hơn là mô hình, bằng cách tăng cường thêm dữ liệu hay cải tiến tiền xử lý dữ liệu,...

7.2 Hướng phát triển trong tương lai

- Áp dụng thêm một số mô hình phân loại khác như BiLSTM, PhoBert,...
- Làm sạch dữ liệu hơn nữa

Tài liệu tham khảo

- https://machinelearningcoban.com/
- https://en.wikipedia.org/wiki/Convolutional_neural_network
- https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/
- https://nttuan8.com/bai-6-convolutional-neural-network/