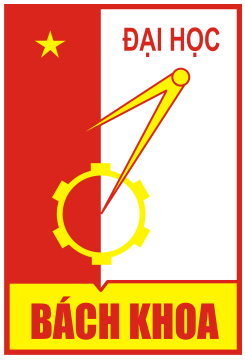
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Toán ứng dụng và Tin học



BÁO CÁO CUỐI KỲ
HỆ HỖ TRỢ QUYẾT ĐỊNH

PHÂN TÍCH CẢM XÚC KHÁCH HÀNG

GVHD: TS. Lê Hải Hà

Sinh viên: Nguyễn Anh Minh 20194117

Hà Nội tháng 7 năm 2022

Mục lục

Giới thiệu

Tổng quan đề tài

Phân tích cảm xúc là một ứng dụng trí tuệ nhân tạo, nó sử dụng các thuật toán phức tạp để xử lý ngôn ngữ tự nhiên của con người (NLP) và xác định các đặc điểm cảm xúc tiêu cực/tích cực tại một thời điểm thông qua văn bản hoặc lời nói. Các mô hình có độ chính xác cao trong việc dự đoán cảm xúc của khách hàng với sản phẩm sẽ giúp các công ty thương mại dự đoán được thị hiếu của khách hàng từ đó sản xuất các mặt hàng phù hợp giúp thu lợi nhuận cao. Trong bài báo cáo này, tác giả sẽ xây dựng mô hình Word2Vec và mạng LSTM kết hợp việc sử dụng nguồn dữ liệu bình luận về phim của trang đánh giá phim Imdb để phân tích cảm xúc của người bình luận thành 2 nhãn: Tích cực hoặc Tiêu cực.

Mục tiêu của báo cáo này là xây dựng một mô hình phân tích cảm xúc khác hàng có khả năng dự đoán tốt và cung cấp các kiến thức cơ bản về Học sâu, mạng Recurrent Neural cũng như mô hình NLP Word2Vec.

Cấu trúc bài báo cáo

Bài báo cáo sẽ được chia thành 3 phần chính với nội dung như sau:

- **Phần 1: Giới thiệu mạng Neural và mạng Recurrent Neural** sẽ cung cấp các kiến thức liên quan về Học sâu và giới thiệu các mô hình Học sâu cơ bản như mạng Neural và mạng Recurrent Neural.
- **Phần 2: Phương pháp luận** sẽ nêu ra lý thuyết xây dựng mô hình, cung cấp lý thuyết về các phương pháp xử lý từ Word Embedding mà cụ thể là Word2Vec và nêu cách thiết kế mô hình mà tác giả xây dựng để phân tích cảm xúc khách hàng bình luận về phim.
- **Phần 3: Kết quả thực nghiệm** sẽ trình bày cách cài đặt mô hình, đưa ra kết quả và đánh giá kết quả.

Trong báo cáo này, tác giả sử dụng một số tên viết tắt cho các thuật ngữ như sau:

DL	Deep Learning
NLP	Natural Language Processing
LSTM	Long-short Term Memory
RNN	Recurrent Neural Network

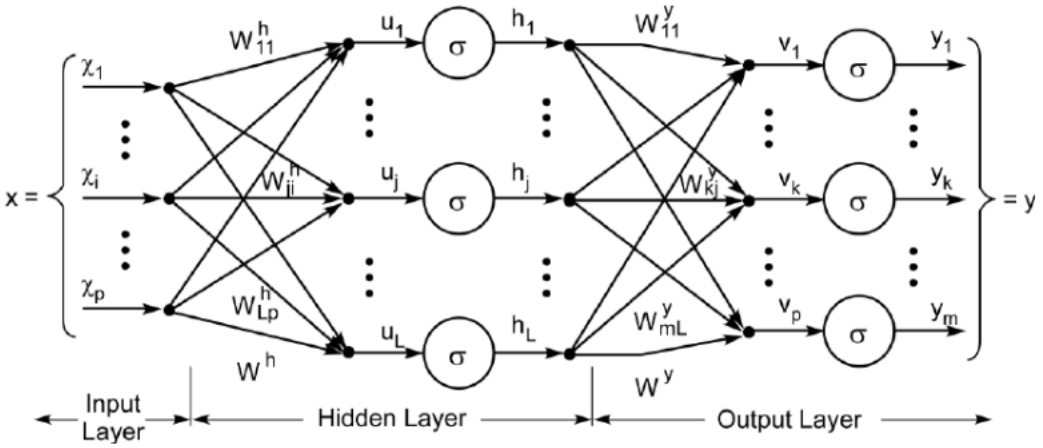
Bảng 1: Bảng thuật ngữ viết tắt

1 Giới thiệu mạng Neural và mạng Recurrent Neural

Trong phần này, tác giả sẽ trình bày các kiến thức cơ bản về mạng Neural và kiến trúc của mạng Recurrent Neural. Các kiến thức trong phần này được tham khảo chính trong sách [1]. Các ký hiệu in đậm như $\boldsymbol{v}$ *biểu thị cho các vector.*

1.1 Tổng quan mạng Neural

Mô hình mạng Neural được xây dựng dựa trên mô hình tổng quát Multilayer Perceptrion-MLP (theo [1]). Một MLP là ánh xạ $y = f * (x, \theta)$ với input là dữ liệu và output là giá trị dự đoán. Hàm f được xây dựng từ các hàm tuyến tính, hàm tanh, ... Bằng việc lựa chọn các hàm thích hợp và dữ liệu đầu vào mà mô hình MLP có thể xấp xỉ được tham số θ và từ đó giúp máy tính xác định đầu ra.



Hình 1: Kiến trúc cơ bản của mạng MLP

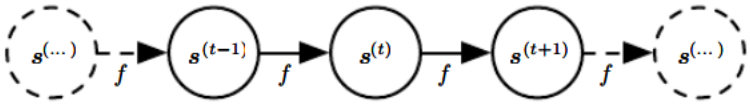
Kiến trúc của mạng Neural được xây dựng tương tự như ở Hình 1 với 3 thành phần chính là:

1. Input Layer: Lớp input dữ liệu.
2. Hidden Layer: Lớp tính toán, lưu trữ thông tin trung gian gồm nhiều các *node* mạng với các *activation function* f khác nhau.
3. Output Layer: Lớp đầu ra dự đoán của mô hình.

Mỗi *layer* là một tầng trong mạng Neural, một *Hidden Layer và Output Layer* có thể gồm nhiều *node* (tương ứng các σ trong Hình 1) là thành phần lưu trữ thông tin trong mạng, mỗi *node* sẽ liên kết với toàn bộ các *node* của *layer* trước đó với các hệ số w riêng và ở mỗi *node* sẽ diễn ra 2 bước

tính toán: tính tổng tuyến tính với các hệ số w và sau đó sử dụng *activation function* riêng để tính toán ra đầu ra dùng cho *layer* tiếp theo.

Việc mô hình hóa tính toán của mạng Neural được biểu diễn thông qua *computational graph* như sau:



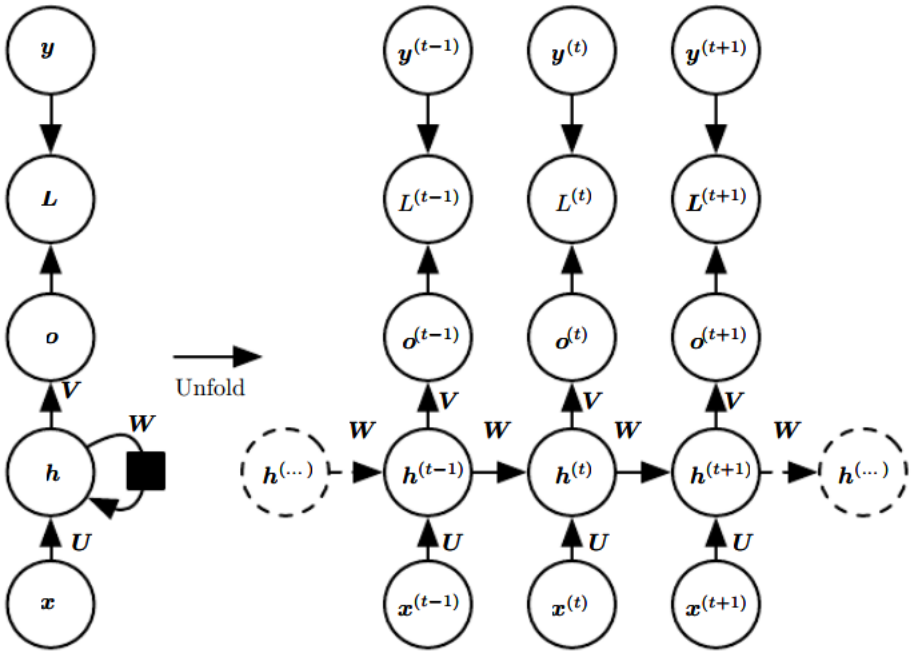
Hình 2: Computational Graph

Mỗi một *node* có một *activation function* f riêng (ví dụ như hàm tuyến tính, hàm *relu*, hàm *tanh*, hàm *softmax*,...) và được minh họa trong *computational graph* như trên. Ví dụ trong hình 2, ta có biểu diễn cách tính giá trị *state* $\boldsymbol{s}^{(t)}$ của *layer* sau được tính thông qua *layer* trước:

$$\boldsymbol{s}^{(t)} = f\left(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right)$$

1.2 Sequence Modeling: Mạng LSTM và GRU

Mạng *Long-short term memory LSTM* và mạng *Gated Recurrent Unit GRU* đều có kiến trúc chung là mạng *Recurrent Neural RNN*, chúng được sử dụng rộng rãi trong các mô hình *Natural Language Processing NLP*, cụ thể là dịch và xử lý các đoạn văn bản. Kiến trúc mạng RNN tổng quát có thể được biểu diễn như sau:



Hình 3: Computational graph biểu diễn RNN

Việc tính toán *forward* trong mạng RNN được tổng quát thông qua hàm sau:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, x^{(t)}; \boldsymbol{\theta})$$

Như trong hình 3 biểu diễn, mạng RNN sẽ gồm các ánh xạ nối chuỗi input $\boldsymbol{x}$ đến các giá trị output tương ứng của từng *node* là $\boldsymbol{o}$. Một hàm mất mát *loss* L sẽ tính toán độ sai khác giữa mỗi $\boldsymbol{o}$ với giá trị *training* $\boldsymbol{y}$ thông qua đại lượng dự đoán của mạng là $\hat{\boldsymbol{y}} = softmax(\boldsymbol{o})$. Mạng RNN gồm các ma trận trọng số sau: ma trận trọng số $\boldsymbol{U}$ tính toán từ input $\boldsymbol{x}$ tạo ra các giá trị ẩn *hidden* $\boldsymbol{h}$ của mạng, các giá trị $\boldsymbol{h}$ sẽ được dùng làm input cho bước lặp tiếp theo trong *forward* mạng RNN và các giá trị $\boldsymbol{h}$ đó được kết nối với nhau thông qua ma trận trọng số $\boldsymbol{W}$, việc tính toán từ $\boldsymbol{h}$ tới các giá trị output $\boldsymbol{o}$ của mạng thông qua ma trận trọng số $\boldsymbol{V}$. Cụ thể, quá trình tính toán trong mạng RNN sẽ tiến hành như sau:

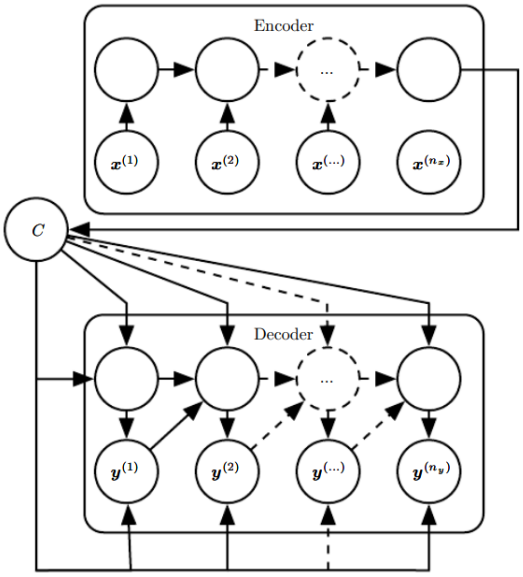
$$\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\
\boldsymbol{h}^{(t)} &= \tanh\left(\boldsymbol{a}^{(t)}\right) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}\left(\boldsymbol{o}^{(t)}\right)
\end{aligned}$$

Các tham số *bias* của mạng được biểu diễn thông qua vectors $\boldsymbol{b}$ và $\boldsymbol{c}$. Mô hình trên được sử dụng trong trường hợp chuỗi input $\boldsymbol{x}$ và chuỗi output $\boldsymbol{y}$ có cùng độ dài. Tổng giá trị hàm mất mát khi biết trước chuỗi input $\boldsymbol{x}$ và chuỗi output $\boldsymbol{y}$ là tổng các hàm mất mát của tất cả các bước lặp. Ví dụ $L^{(t)}$ là hàm mất mát *negative log-likelihood* của $y^{(t)}$ cho trước chuỗi $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}$ thì:

$$\begin{aligned}
& L\left(\left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\right\}, \left\{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\right\}\right) \\
=& \sum_t L^{(t)} \\
=& -\sum_t \log p_{\text{model}}\ \left(y^{(t)} \mid \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\right\}\right)
\end{aligned}$$

với $p_{\text{model}}\ \left(y^{(t)} \mid \left\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\right\}\right)$ được tính thông qua mô hình khi so sánh giá trị của $y^{(t)}$ với giá trị dự đoán của mô hình là $\hat{\boldsymbol{y}}^{(\boldsymbol{t})}$.

Kiến trúc Encoder-Decoder Sequence-to-Sequence

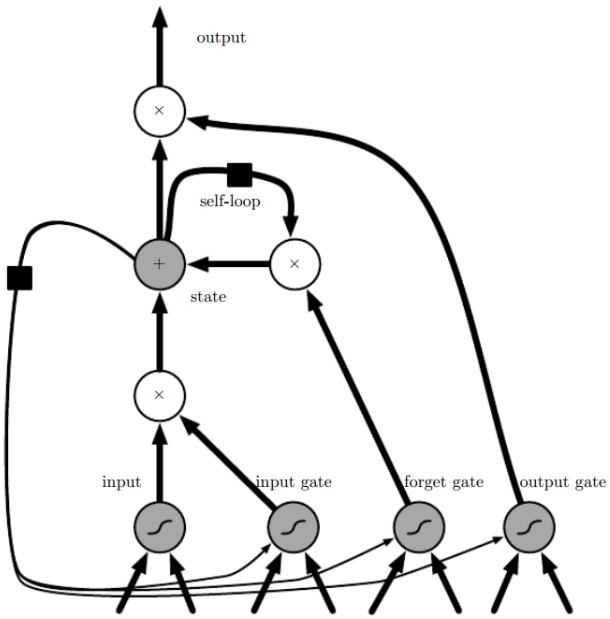


Hình 4: Kiến trúc Encoder-Decoder

Hình 4 miêu tả kiến trúc kiểu sequence-to-sequence của mạng RNN trong việc học tạo ra chuỗi output $\left(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_y)}\right)$ khi biết trước chuỗi input $\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n_x)}\right)$. Nó là sự kết hợp của một khối Encoder RNN đọc chuỗi dữ liệu đầu vào và một khối Decoder RNN tạo ra chuỗi output đầu ra (hoặc tính xác suất xảy ra một sự kiện nào đó (bài toán phân loại) khi cho trước chuỗi đầu vào). Lớp ẩn cuối cùng của Encoder RNN được dùng để tính một biến "ngữ cảnh" (context) cố định C biểu diễn thông tin được trích xuất ra từ chuỗi đầu vào để là đầu vào cho khối Decoder RNN.

Mạng LSTM

Về mặt lý thuyết mạng RNN có thể mang thông tin từ các *layer* trước đến các *layer* sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng *state* nhất định, sau đó thì sẽ xảy ra hiện tượng *vanishing gradient* (tức đạo hàm hàm *loss* theo tham số của mạng xấp xỉ bằng 0) hay nói cách khác là mạng RNN chỉ học được từ các state gần nó, đây là hiện tượng *short term memory.* Với bài toán dịch văn bản trong xử lý ngôn ngữ tự nhiên thì input của mạng là những câu văn dài và ta cần thông tin của những câu văn trước đó để output đầu ra dịch đúng ngữ cảnh nhất. Chính vì vậy, để khắc phục hiện tượng *vanishing gradient* một mạng RNN cải tiến ra đời chính là mạng *Long-short term memory* LSTM.



Hình 5: Kiến trúc tế bào của mạng LSTM

Điểm đặc biệt trong cấu trúc tế bào của mạng LSTM là nó có cơ chế *self-loop* để làm giảm hiện tượng *vanishing gradient*, tế bào LSTM sẽ quyết định tỷ lệ lưu giữ thông tin của các *state* trước đó, các tham số của *self-loop* được điều khiển thông qua cổng "quên" (forget gate) $f_i^{(t)}$ (ứng với bước lặp thứ t và tế bào thứ i), cổng này sẽ điều chỉnh tỷ lệ quên thông tin trước trong khoảng từ 0 tới 1 thông qua hàm *sigmoid*:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right)$$

trong đó $\boldsymbol{x}^{(t)}$ là vector input hiện tại và $\boldsymbol{h}^{(t)}$ là vector *hidden layer* hiện tại chứa output của toàn bộ tế bào LSTM trước đó. $b^f, \boldsymbol{U}^f, \boldsymbol{W}^f$ lần lượt là tham số *bias* của mạng, trọng số của dữ liệu đầu vào và trọng số cho tầng cổng quên. Thành phần quan trọng nhất của tế bào là *state unit* $s_i^{(t)}$ lưu trữ thông tin của mạng. Khi đó quá trình cập nhật giá trị trong mạng diễn ra như sau:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right)$$

với $\boldsymbol{b}, \boldsymbol{U}$ và $\boldsymbol{W}$ lần lượt ký hiệu cho hệ số *bias*, tham số của dữ liệu input và tham số *recurrent* của tế bào LSTM. Ta có thể thấy giá trị của s_i^t được cập nhật kết hợp với giá trị của tầng cổng quên $f_i^{(t)}$. Giá trị *external input gate* $g_i^{(t)}$ được tính toán tương tự như tầng cổng quên với các tham số riêng để tính toán tỷ lệ ảnh hưởng của dữ liệu đầu vào ở bước thứ t là $x_j^{(t)}$ và thông tin từ *state* trước đó là $h_j^{(t-1)}$:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right).$$

Giá trị output dự đoán $h_i^{(t)}$ của tế bào LSTM có thể bị bỏ qua thông qua cổng *output* $q_i^{(t)}$ cũng được tính thông qua hàm *sigmoid*:

$$h_i^{(t)} = \tanh\left(s_i^{(t)}\right) q_i^{(t)}$$
$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right)$$

có chứa các vector tham số $\boldsymbol{b}^o, \boldsymbol{U}^o, \boldsymbol{W}^o$ lần lượt ứng với *bias*, trọng số với dữ liệu input, trọng số của tầng cổng quên đối với kết quả output dự đoán của các *state* trước đó.

Mạng GRU

Mạng GRU *Gated Recurrent Unit* là một loại mạng RNN cũng được thiết kế để khắc phục hiện tượng *vanishing gradient*. Mạng GRU có kiến trúc tương tự mạng LSTM nhưng có sự thay đổi trong cách thiết kế tế bào *cell*, cụ thể là thay đổi cách thiết lập hàm để "quên" thông tin từ các *state* trước đó cũng như cách tính trọng số trong việc sử dụng thông tin từ các *state* trước đó:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + \left(1 - u_i^{(t-1)}\right) \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t-1)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)}\right)$$

với $\boldsymbol{u}$ biểu thị cho *cổng cập nhật (update gate)* và $\boldsymbol{r}$ biểu thị cho *cổng reset*. Các cổng này được thiết lập tính toán như sau:

$$u_i^{(t)} = \sigma\left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)}\right)$$

và

$$r_i^{(t)} = \sigma\left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)}\right)$$

Cổng *reset* cũng như cổng *cập nhật* có thể bỏ qua thông tin của các *state* trước đó, cổng cập nhật có cơ chế tương tự như tầng cổng quên, cổng reset sẽ điều khiển phần thông tin nào của *state* trước đó được dùng để sử dụng trong việc tính toán của *state* này.

2 Phương pháp luận

Trong phần này, tác giả sẽ trình bày phương pháp luận để xây dựng mô hình phân tích cảm xúc khách hàng. Mô hình sẽ gồm 2 khối xử lý chính: Một là khối Word Embedding giúp chuyển dữ liệu dạng văn bản text sang dạng số vector; Hai là khối mạng Neural gồm các khối DropOut, LSTM, Dense Layer.

2.1 Word Embedding

Để có thể đưa text vào mô hình DL, việc đầu tiên chúng ta cần làm là số hóa các từ đầu vào hay nói cách khác đây là quá trình Embedding (nhúng từ). Có rất nhiều phương pháp Word Embedding từ nguyên thủy cơ bản như One hot encoding, co-ocurrence matrix đến các phương pháp hiện đại hơn như Word2Vec, CBOW, Skip-gram [2], Glove [3].

One hot encoding:

Phương pháp này là phương pháp đơn giản nhất để đưa từ về dạng số hóa vector với chiều bằng với kích thước bộ từ điển. Mỗi từ sẽ được biểu diễn bởi 1 vector mà giá trị tại vị trí của từ đó trong từ điển bằng 1 và giá trị tại các vị trí còn lại đều bằng 0.

Ví dụ: Dữ liệu gồm 3 câu đầu vào: "Tôi đang đi học", "Mình đang bận nhé", "Tôi sẽ gọi lại sau". Ta xây dựng bộ từ điển gồm các từ: "Tôi, đang , đi, học, Mình, bận, nhé, sẽ , gọi, lại, sau". Tương ứng, ta có các biểu diễn one hot encoding của từng từ như sau:

Tôi: $[1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]$;

đang: $[0; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]$;

Mình: $[0; 0; 0; 0; 1; 0; 0; 0; 0; 0; 0; 0]$;

sau: $[0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1], \ldots$

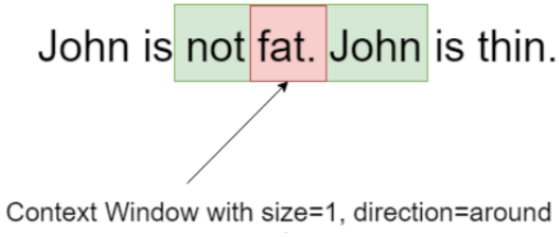
Cách biểu diễn này tui đơn giản nhưng có những hạn chế rất lớn sau đây:

- **Chi phí tính toán lớn**: Nếu data có 1000 từ, độ dài của vector one-hot là 1000 .Nếu data có 10000 từ, độ dài của vector one-hot là 10000. Tuy nhiên, để mô hình có độ khái quát cao thì trong thực tế dữ liệu có thể bùng nổ và lên đến hàng triệu từ, lúc đó độ dài vector one-hot sẽ phình to gây khó khăn cho việc tính toán, lưu trữ.
- **Mang ít giá trị thông tin:** Các vector hầu như toàn số 0, kém về mặt biểu diễn ngữ nghĩa dữ liệu. Một cách Embedding tốt sẽ cho 2 vector biểu diễn của 2 từ đồng nghĩa hoặc sát nghĩa gần nhau như vector biểu diễn từ "man" phải gần với vector biểu diễn từ "boy". One-hot vector không thể biểu diễn điều đó vì nó chỉ đánh index theo thứ tự từ điển đầu vào chứ không phải vị trí các từ trong một ngữ cảnh cụ thể.

- **Độ khái quát yếu:** Ví dụ ta có ba từ cùng chỉ người mẹ : mẹ, má, bầm. Tuy nhiên, từ bầm tương đối hiếm gặp trong tiếng Việt. Khi biểu diễn bằng one-hot encoding, khi đưa vào model train thì từ bầm mặc dù cùng nghĩa so với hai từ kia nhưng lại bị phân vào class khác nhau do cách biểu diễn khác nhau.

Co-occurrence Matrix

Năm 1957, nhà ngôn ngữ học J.R. Firth đã chỉ ra rằng: Nghĩa của một từ có thể được biểu thị qua nghĩa của các từ đi kèm với nó. Ví dụ nhắc đến Việt Nam, người ta thường có các cụm từ quen thuộc như "Chiến tranh Việt Nam", "Áo dài Việt Nam",... dựa vào những từ xung quanh ta có thể hiểu hoặc mường tượng ra được "Việt Nam" là gì, như thế nào. Co-occurrence Matrix được xây dựng dựa trên nhận xét trên, ma trận sẽ đảm bảo quan hệ ngữ nghĩa giữa các từ, dựa trên số lần xuất hiện của các cặp từ trong một ngữ cảnh *(context window)*. Một ngữ cảnh được xác dựa trên kích thước và hướng của nó, ví dụ của context window:



Hình 6: Ví dụ về context window

Co-ocurrence matrix là một ma trận vuông đối xứng, mỗi hàng, mỗi cột sẽ làm vector đại diện cho từ tương ứng. Từ ví dụ trên ta tiếp tục xây dựng co-ocurrence matrix:

	John	is	not	fat	thin
John	0	2	0	1	0
is	2	0	1	0	1
not	0	1	0	1	0
fat	1	0	1	0	0
thin	0	1	0	0	0

Hình 7: Ví dụ về Co-ocurrence matrix

Trong đó, giá trị tại ô $[i, j]$ là số lần xuất hiện của từ i nằm trong context window của từ j. Cách biểu diễn trên mặc dù đã giữ được thông tin về ngữ nghĩa của một từ, tuy vẫn còn các hạn chế như sau:

- Khi kích thước bộ từ điển tăng, chiều vector cũng tăng theo.
- Lưu trữ Co-ocurrence matrix cần rất nhiều tài nguyên về bộ nhớ.
- Các mô hình phân lớp bị gặp vấn đề với biểu diễn thưa (có rất nhiều giá trị 0 trong ma trận).

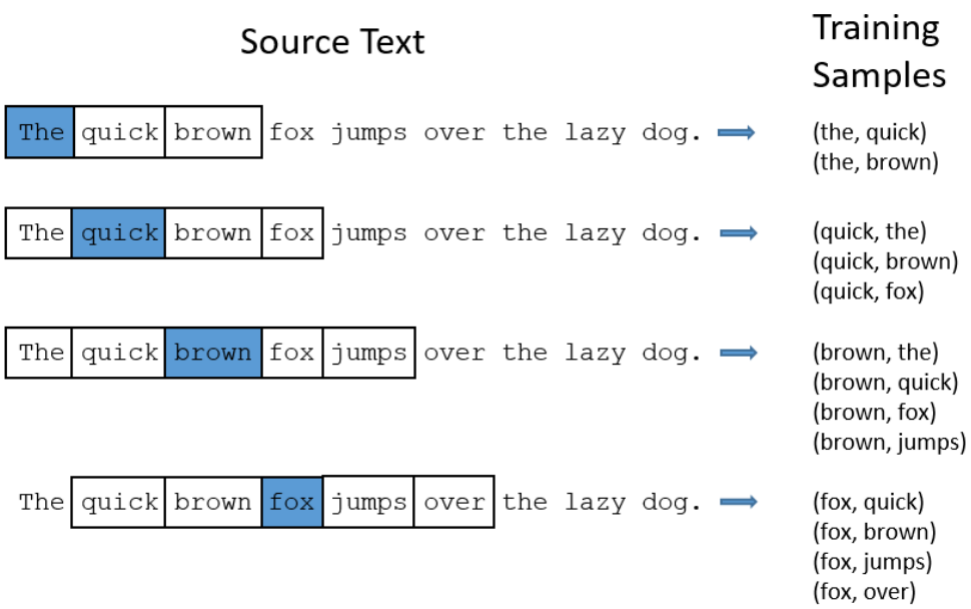
Để làm giảm kích thước của co-ocurrence matrix người ta thường sử dụng phép SVD (Singular Value Decomposition) để giảm chiều ma trận. Ma trận thu được sau SVD có chiều nhỏ hơn, dễ lưu trữ hơn và ý nghĩa của từ cũng cô đọng hơn. Tuy nhiên, SVD có độ phức tạp tính toán cao, tăng nhanh cùng với chiều của ma trận ($O(mn^2)$ với m là chiều của ma trận trước SVD, n là chiều của ma trận sau SVD và $n < m$) , ngoài ra phương pháp này cũng gặp khó khăn khi thêm các từ vựng mới vào bộ từ điển. Do vậy, ta cần phương pháp khác lưu trữ được nhiều thông tin và vector biểu diễn nhỏ.

2.2 Phương pháp Word2Vec

Word2vec là một mô hình đơn giản và nổi tiếng giúp tạo ra các biểu diễn Embedding của từ, chuyển từ trong một không gian có số chiều thấp hơn nhiều lần so với số từ trong từ điển. Ý tưởng cơ bản của word2vec có thể được gói gọn trong 2 ý sau:

- Hai từ xuất hiện trong những ngữ cảnh giống nhau thường có ý nghĩa gần với nhau.
- Ta có thể đoán được một từ nếu biết các từ xung quanh nó trong câu. Ví dụ, với câu "Hà Nội là ... của Việt Nam" thì từ trong dấu ba chấm khả năng cao là "thủ đô". Với câu hoàn chỉnh "Hà Nội là thủ đô của Việt Nam", mô hình Word2vec sẽ xây dựng ra embeding của các từ sao cho xác suất để từ trong dấu ba chấm là "thủ đô" là cao nhất.

Trong một context window sẽ có target word hay từ đích, những từ xung quanh nó được gọi là context words hay từ ngữ cảnh. Với mỗi từ đích trong một câu của cơ sở dữ liệu, các từ ngữ cảnh được định nghĩa là các từ trong cùng câu có vị trí cách từ đích một khoảng không quá $C/2$ với C là một số tự nhiên dương. Như vậy, với mỗi từ đích, ta sẽ có một bộ không quá C từ ngữ cảnh. Xét ví dụ sau đây với câu tiếng Anh: "The quick brown fox jump over the lazy dog" với C=4.



Hình 8: Ví dụ về từ đích và từ ngữ cảnh

Khi "the" là từ đích, ta có cặp dữ liệu huấn luyện là (the, quick) và (the, brown). Khi "brown" là từ đích, ta có cặp dữ liệu huấn luyện là (brown, the), (brown, quick), (brown, fox) và (brown, jumps).

Word2vec định nghĩa hai embedding vector cùng chiều cho mỗi từ w trong từ điển. Khi nó là một từ đích, embedding vector của nó là u; khi nó là một từ ngữ cảnh, embedding của nó là v, việc xây dựng 2 vector như trên do ý nghĩa của từ đó khi nó là từ đích và từ ngữ cảnh là khác nhau. Tương ứng với đó, ta có hai ma trận embedding U và V cho các từ đích và các từ ngữ cảnh. Có hai cách khác nhau xây dựng mô hình Word2vec:

- **Skip-gram:** Dự đoán những từ ngữ cảnh nếu biết trước từ đích.
- **CBOW (Continuous Bag of Words):** Dựa vào những từ ngữ cảnh để dự đoán từ đích.

Mỗi cách có những ưu nhược điểm khác nhau và áp dụng với những loại dữ liệu khác nhau.

Skip-gram

Ứng với một từ ngữ cảnh sẽ có một hàm mất mát. Hàm mất mát tổng cộng sẽ là tổng của hàm mất mát tại mỗi từ ngữ cảnh. Việc tối ưu hàm mất mát có thể được thực hiện thông qua Gradient Descent trên từng từ ngữ cảnh hoặc một batch các từ ngữ cảnh. Xét ví dụ bên trên với từ đích là "fox" và các từ ngữ cảnh là "quick", "brown", "jumps" và "over". Việc dự đoán xác suất xảy ra các từ ngữ cảnh khi biết từ đích được mô hình hóa bởi:

$$P(\text{"}quick\text{"},\text{"}brown\text{"},\text{"}jumps\text{"},\text{"}over\text{"}|\text{"}fox\text{"})$$

Ta có thể giả sử rằng sự xuất hiện của một từ ngữ cảnh khi biết từ đích độc lập với các từ ngữ cảnh khác để xấp xỉ xác suất trên đây bởi:

$$P(\text{"}quick\text{"}|\text{"}fox\text{"})P(\text{"}brown\text{"}|\text{"}fox\text{"})P(\text{"}jumps\text{"}|\text{"}fox\text{"})P(\text{"}over\text{"}|\text{"}fox\text{"})$$

Giả sử từ đích có chỉ số t trong từ điển $\mathcal{V}$ và tập hợp các chỉ số của các từ ngữ cảnh tương ứng là $\mathcal{C}_{\sqcup}$. Số lượng phần tử của $\mathcal{C}_{\sqcup}$ dao động từ $C/2$ (nếu w_t đứng đầu hoặc cuối câu) tới C (nếu w_t đứng ở giữa câu và có đủ $C/2$ từ ngữ cảnh ở mỗi phía). Từ dữ liệu đã có, ta cần một mô hình sao cho xác suất dưới đây càng lớn càng tốt với mỗi từ ngữ cảnh w_t :

$$\prod_{c \in \mathcal{C}_t} P\left(w_c \mid w_t\right)$$

Để tránh các sai số tính toán khi nhân các số nhỏ hơn 1 với nhau, bài toán tối ưu này thường được đưa về bài toán tối thiểu đối số của log (thường được gọi là negative log loss):

$$-\sum_{c \in \mathcal{C}_t} \log P\left(w_c \mid w_t\right)$$

Xác suất có điều kiện $P\left(w_c \mid w_t\right)$ được định nghĩa bởi:

$$P\left(w_c \mid w_t\right) = \frac{\exp\left(\mathbf{u}_t^T \mathbf{v}_c\right)}{\sum_{i=1}^{N} \exp\left(\mathbf{u}_t^T \mathbf{v}_i\right)} \tag{1}$$

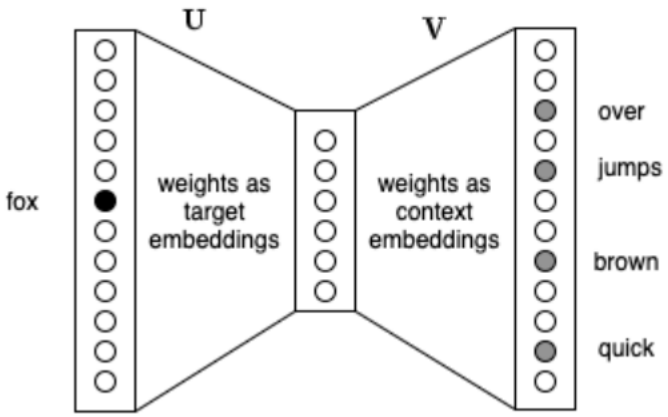
với N là số phần tử của từ điển $\mathcal{V}$. Ở đây $\exp\left(\mathbf{u}_t^T \mathbf{v}_c\right)$ thể hiện mỗi quan hệ giữa từ đích w_t và từ ngữ cảnh w_c. Biểu thức này càng cao thì xác suất thu được càng lớn. Tích vô hướng $\mathbf{u}_t^T \mathbf{v}_c$ cũng thể hiện sự tương tự giữa hai vector. Việc định nghĩa xác suất như biểu thức ở trên đảm bảo rằng

$$\sum_{w \in \mathcal{V}} P\left(w \mid w_t\right) = 1$$

Tóm lại, hàm mất mát ứng với từ đích w_t theo $\mathbf{U}, \mathbf{V}$ được cho bởi

$$\mathcal{L}\left(\mathbf{U}, \mathbf{V}; w_t\right) = -\sum_{c \in \mathcal{C}_t} \log \frac{\exp\left(\mathbf{u}_t^T \mathbf{v}_c\right)}{\sum_{i=1}^{N} \exp\left(\mathbf{u}_t^T \mathbf{v}_i\right)}$$

Biểu diễn Skip Gram dưới dạng mạng Neural đơn giản gồm một tầng ẩn và không có hàm kích hoạt *(activation function)*.



Hình 9: Biểu diễn Skip Gram dưới dạng mạng Neural

Như trên Hình 9, $\mathbf{u}_t$ chính là kết quả của phép nhân vector one-hot tương ứng với w_t với ma trận trọng số $\mathbf{U}$, vì vậy đây chính là giá trị đầu ra của của tầng ẩn ở giữa khi xét từ đích w_t. Tiếp theo, đầu ra của tầng ẩn không hàm kích hoạt này được nhân trực tiếp với ma trận trọng số đầu ra $\mathbf{V}$ để được $\mathbf{u_t^T V}$, đây chính là giá trị vector logit trước khi đi vào hàm kích hoạt softmax như trong biểu thức (2.2). Kiến trúc đơn giản này giúp Word2vec hoạt động tốt ngay cả khi số lượng từ trong từ điển là cực lớn (có thể lên tới nhiều triệu từ).

Việc tối ưu hai ma trận trọng số $\mathbf{U}$ và $\mathbf{V}$ được thực hiện thông qua các thuật toán Gradient Descent. Các thuật toán tối ưu dạng này yêu cầu tính gradient cho từng ma trận. Xét riêng số hạng:

$$\log P\left(w_c \mid w_t\right) = \log\left(\frac{\exp\left(\mathbf{u}_t^T \mathbf{v}_c\right)}{\sum_{i=1}^{N} \exp\left(\mathbf{u}_t^T \mathbf{v}_i\right)}\right) = \mathbf{u}_t^T \mathbf{v_c} - \log\left(\sum_{i=1}^{N} \exp\left(\mathbf{u}_t^T \mathbf{v}_i\right)\right)$$

Đạo hàm theo $\mathbf{u}_t$:

$$\frac{\partial \log P\left(w_c \mid w_t\right)}{\partial \mathbf{u_t}} = \mathbf{v}_c - \sum_{j=1}^{N}\left(\frac{\exp\left(\mathbf{u}_t^T \mathbf{v}_j\right)\mathbf{v}_j}{\sum_{i=1}^{N} \exp\left(\mathbf{u}_t^T \mathbf{v}_i\right)}\right) = \mathbf{v}_c - \sum_{j=1}^{N} P\left(w_j \mid w_t\right)\mathbf{v}_j$$

Như vậy, mặc dù gradient này rất đẹp, chúng ta vẫn cần phải tính toán các xác suất $P\left(w_j \mid w_t\right)$. Mỗi xác suất này phụ thuộc toàn bộ ma trận trọng số $\mathbf{V}$ và vector $\mathbf{u}_t$. Như vậy ta cần cập nhập tổng cộng $N * d + d$ trọng số. Đây rõ ràng là một con số rất lớn với N lớn.

2.2.1 CBOW- Continuous Bag of Words

Ngược với Skip-gram, CBOW đi tìm xác suất xảy ra từ đích khi biết các từ ngữ cảnh xung quanh. Ta cần mô hình hóa dữ liệu sao cho xác suất sau đây đạt giá trị lớn:

$$P(\text{" fox " | " quick ", "brown "," jumps ", " over"})$$

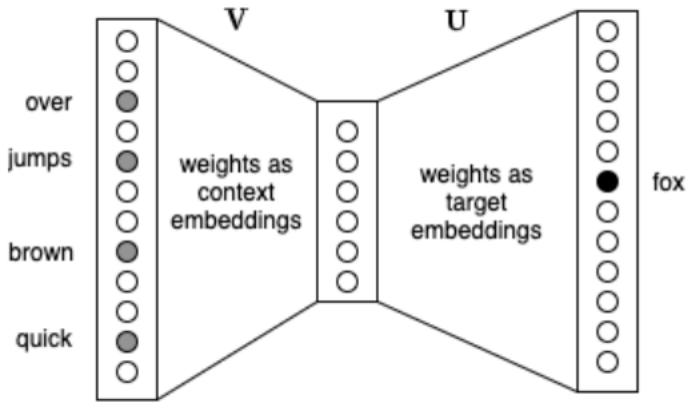
Vì có nhiều từ ngữ cảnh trong điều kiện, chúng thường được đơn giản hóa bằng cách lấy một từ "trung bình" làm đại diện.

$$P\left(w_t \mid \bar{w}_{\mathcal{C}_t}\right)$$

với $\bar{w}_{\mathcal{C}_t}$ là trung bình cộng của các từ trong ngữ cảnh của từ đích w_t. Embedding của từ trung bình này là trung bình của embedding các từ ngữ cảnh. Xác xuất này cũng được định nghĩa tương tự như trong Skipgram:

$$P\left(w_t \mid \bar{w}_{\mathcal{C}_t}\right) = \frac{\exp\left(\mathbf{u}_t^T \frac{1}{C} \sum_{c \in \mathcal{C}_t} \mathbf{v}_c\right)}{\sum_{i=1}^{N} \exp\left(\mathbf{u}_i^T \frac{1}{C} \sum_{c \in \mathcal{C}_t} \mathbf{v}_c\right)}$$

Biểu diễn mạng neural cho CBOW được thể hiện như trong Hình (10) dưới đây:



Hình 10: Biểu diễn mạng Neural của CBOW

2.3 Mô hình kết hợp Word2Vec và mạng LSTM trong việc phân tích cảm xúc khách hàng

Mô hình mạng LSTM là một mô hình thường được sử dụng trong các mô hình NLP với khả năng giảm được hiện tượng short-term memory và trích xuất được thông tin của dữ liệu dạng chuỗi *(sequence)*. Trong báo cáo này, tác giả sẽ xây dựng mô hình gồm hai khối chính:

- **Khối Word Embedding:** Tác giả sẽ sử dụng mô hình Word2Vec để tiến hành chuyển dữ liệu dạng text sang các vector số với tính chất các vector biểu diễn các từ cùng nghĩa thì sẽ có khoảng cách gần nhau theo chuẩn Euclide.
- **Khối mạng Neural:** Mạng Neural sẽ được thiết kế gồm: một khối mạng LSTM để tổng hợp dữ liệu dạng chuỗi kết hợp với 1 lớp Dense - lớp mạng Neural cổ điển có hàm kích hoạt là *Relu* để trích xuất thông tin cho lớp cuối là lớp Dense với hàm kích hoạt *Softmax* để tính xác suất phân loại 2 nhãn: Positive và Negative với dữ liệu là câu bình luận của khách hàng.

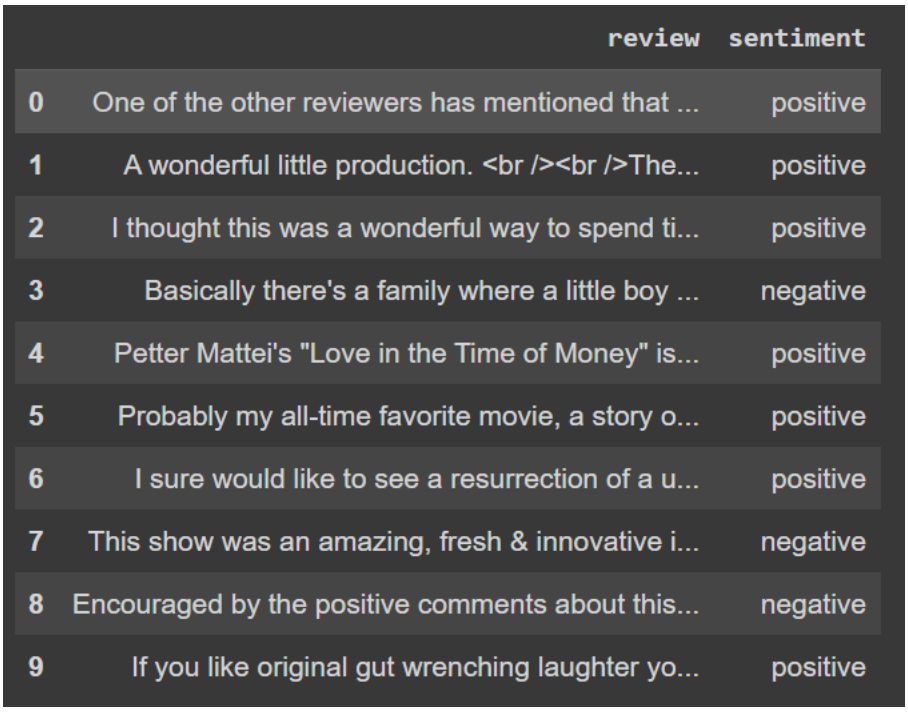
Kết hợp 2 khối trên sẽ giúp mô hình xử lý được dữ liệu dạng chuỗi cũng như mô hình sẽ thuộc loại bài toán phân loại với kết quả đầu ra là xác suất phân loại vào 2 nhãn.

3 Kết quả thực nghiệm

Trong phần này tác giả sẽ trình bày về kết quả thực nghiệm và đánh giá kết quả sau huấn luyện của mô hình.

3.1 Dữ liệu sử dụng

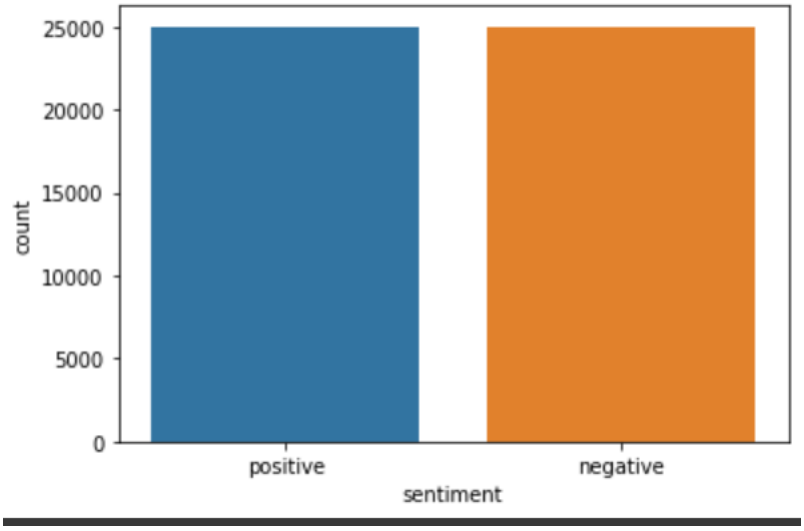
Bộ dữ liệu được sử dụng là bình luận của người xem về các bộ phim được trang đánh giá Imdb thu thập. Dữ liệu sẽ gồm 50000 bộ gồm 2 cột "review": chứa bình luận của khách hàng và cột "sentiment": chứa cảm xúc của khách hàng với hai nhãn: tích cực *Positive* hoặc tiêu cực *Negative*. Dữ liệu sẽ có dạng như sau:



	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
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

Hình 11: Tổng quan dữ liệu

Bộ dữ liệu sẽ có số lượng bình luận có nhãn tiêu cực và nhãn tích cực bằng nhau và bằng 25000.



Hình 12: Tổng quan dữ liệu

Một câu bình luận gốc trong bộ dữ liệu sẽ có dạng như hình (13) với các chữ in hoa, một số ký tực đặc biệt, các chữ tiếng Anh:

```
'One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. They are right, as this is exactly what
h me.<br /><br />The first thing that struck me about Oz was its brutality and unflinching scenes of violence, which set in right from th
ust me, this is not a show for the faint hearted or timid. This show pulls no punches with regards to drugs, sex or violence. Its is hard
classic use of the word.<br /><br />It is called OZ as that is the nickname given to the Oswald Maximum Security State Penitentary. It fo
on Emerald City, an experimental section of the prison where all the cells have glass fronts and face inwards, so privacy is not high on
m City is home to many..Aryans, Muslims, gangstas, Latinos, Christians, Italians, Irish and more....so scuffles, death stares, dodgy deal
y agreements are never far away.<br /><br />I would say the main appeal of the show is due to the fac...'
```

Hình 13: Tổng quan dữ liệu

Do đó, dữ liệu cần được chuyển về dạng chuẩn bằng cách bỏ đi những từ *StopWord* nghĩa là những từ không có quá nhiều ảnh hưởng đến nghĩa của câu trong tiếng Anh như: "in, once, some,...", cũng như bỏ đi các ký tự đặc biệt và chuyển hết các từ ngữ về dạng chính tắc. Sau quá trình chuẩn hóa, dữ liệu sẽ có dạng như sau:

```
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
print(stop_words)
print()
df['review']=df['review'].apply(lambda cw: ' '.join(word for word in cw.split() if word not in stop_words))
df['review'].head(10)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
{'under', 'he', 'but', 'had', 'm', 'off', 'once', 'couldn', 'shan', 'ours', 'of', 'through', 'we', 'out', 'with', 'as', 'few', 'has', 'itself', 'too',

0    one reviewers mentioned watching 1 oz episode ...
1    wonderful little production br br filming tech...
2    thought wonderful way spend time hot summer we...
3    basically family little boy jake thinks zombie...
4    petter mattei love time money visually stunnin...
5    probably time favorite movie story selflessnes...
6    sure would like see resurrection dated seahunt...
7    show amazing fresh innovative idea 70 first ai...
8    encouraged positive comments film looking forw...
9    like original gut wrenching laughter like movi...
```

Hình 14: Dữ liệu sau chuẩn hóa

Cuối cùng do nhãn dữ liệu chưa ở dạng số nên ta dùng phương pháp Label Encoder để chuyển nhãn của dữ liệu về dạng 1-0 và thu được dạng sau:

```
reviews = df['review'].values
labels = df['sentiment'].values
print(labels)
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
print(encoded_labels)
train_sentences, test_sentences, train_labels, test_labels=\
train_test_split(reviews,encoded_labels , test_size=0.2,random_state=16)
```

```
['positive' 'positive' 'positive' ... 'negative' 'negative' 'negative']
[1 1 1 ... 0 0 0]
```

Hình 15: Label Encoder

Qua quá trình import và chuẩn hóa dữ liệu ta sẽ đến với bước xây dựng mô hình.

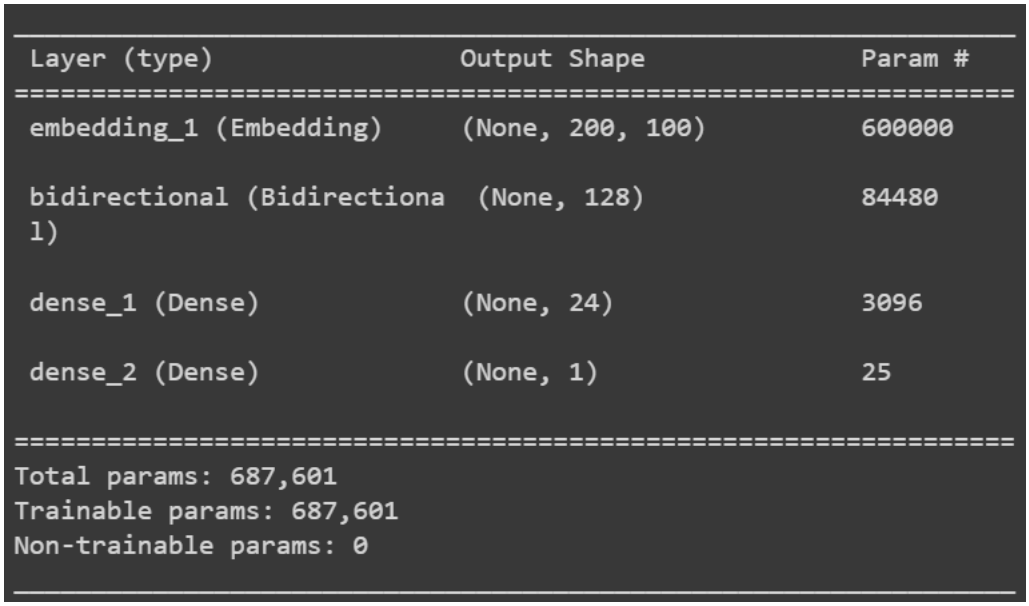
3.2 Cài đặt mô hình

Mô hình được cài đặt bằng ngôn ngữ Python tại `https://bit.ly/3S9yGP2`.

Tác giả tiến hành xây dựng 2 mô hình phân tích cảm xúc khách hàng:

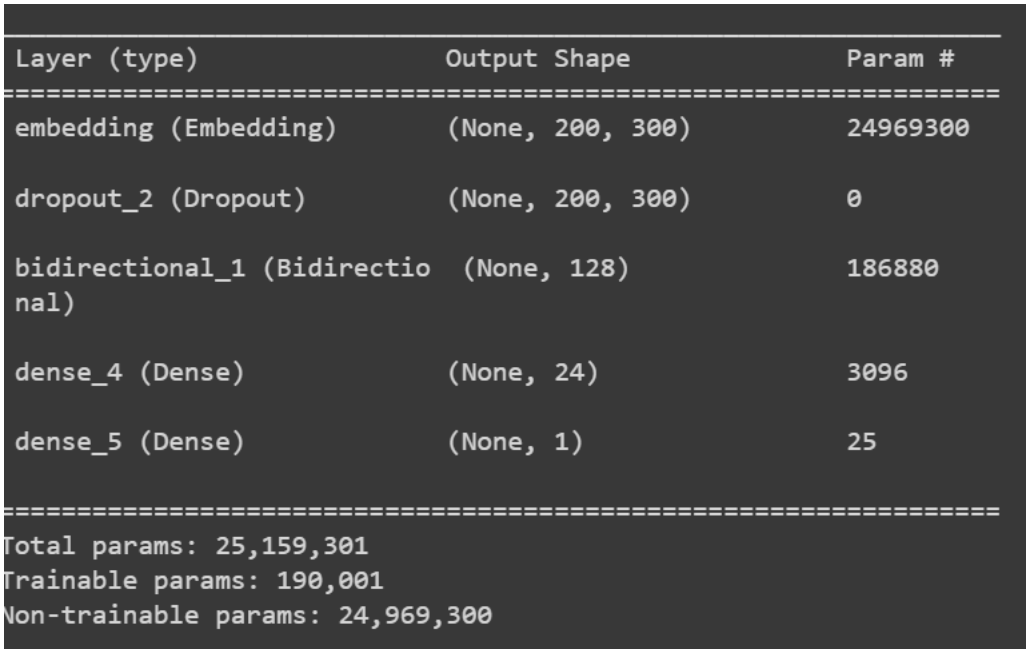
1. **Mô hình 1:** Sử dụng lớp Embedding Layer của Framework Keras có tác dụng như một ma trận trọng số được huấn luyện trong quá trình luyện mô hình giúp cô đọng dữ liệu chuyển dữ

liệu từ không gian có số chiều lớn sang không gian có số chiều nhỏ hơn để tối ưu việc tính toán. Sau đó mô hình vẫn sử dụng các lớp LSTM và Dense Layer như đã trình bày ở phần 2.3.



Hình 16: Mô hình I

2. **Mô hình 2:** Sử dụng phương pháp Word2Vec để khởi tạo ma trận trọng số cho lớp Embedding và không tiến hành cập nhật ma trận trọng số trong quá trình huấn luyện. Sau đó ta tiếp tục sử dụng các lớp LSTM và lớp Dense Layer như đã trình bày.



Hình 17: Mô hình II

Sau quá trình huấn luyện kết quả thu được và đánh giá mô hình sẽ được trình bày ở phần 3.3

3.3 Kết quả và đánh giá

Kết quả ứng với mô hình I như sau:

```
Accuracy of prediction on test set :  0.8829
```

```
from sklearn.metrics import classification_report
print(classification_report(test_labels, pred_labels))
```

	precision	recall	f1-score	support
0	0.90	0.87	0.88	4983
1	0.87	0.90	0.89	5017
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Hình 18: Kết quả mô hình I

Kết quả dự đoán trên tập Test đạt 0.8829 một kết quả dự đoán tốt vớ mô hình đơn giản này. Đồng thời F1-score của mô hình xấp xỉ 0.88 cũng là một kết quả tốt. Tương ứng với kết quả của mô hình II như sau:

```
Accuracy of prediction on test set :  0.8799
```

```
from sklearn.metrics import classification_report
print(classification_report(test_labels, pred_labels))
```

	precision	recall	f1-score	support
0	0.90	0.85	0.88	4983
1	0.86	0.91	0.88	5017
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Hình 19: Kết quả mô hình II

Mô hình II cho kết quả dự đoán trên tập Test là 0.8799, một kết quả tốt xong thấp hơn so với mô hình I. F1-score của mô hình II xấp xỉ với mô hình I và có ưu thế hơn ở 1 số thông số về precision và recall rate.

Mô hình sau khi huấn luyện cho kết quả dự đoán bình luận tốt, ví dụ như sau:

```
The movie was very touching and heart whelming
Predicted sentiment :  Positive
I have never seen a terrible movie like this
Predicted sentiment :  Negative
the movie plot is terrible but it had good acting
Predicted sentiment :  Negative
```

Hình 20: Ví dụ dự đoán

Kết luận

Kết quả đạt được

Qua quá trình xây dựng mô hình và kiểm thử với bộ dữ liệu Imdb, báo cáo này đã đạt được một số kết quả như sau:

- Cung cấp kiến thức cơ bản về Học sâu, mạng Neural và mạng Recurrent Neural.
- Giới thiệu về các phương pháp Word Embedding và trình bày chi tiết phương pháp Word2Vec.
- Cài đặt, kiểm thử 2 mô hình phân tích cảm xúc khách hàng và đạt được tỷ lệ dự đoán cao trên tập Test và các thông số đánh giá khác như F1-score đạt kết quả tốt.

Hướng phát triển trong tương lai

Trong tương lai tác giả sẽ tiếp tục tiến hành xây dựng các mô hình phân tích cảm xúc khách hàng phức tạp hơn, đánh giá trên nhiều bộ dữ liệu để đưa ra mô hình hoạt động tốt nhất.

Lời cảm ơn

Báo cáo này được thực hiện và hoàn thành tại Trường Đại học Bách Khoa Hà Nội, nằm trong nội dung học phần *Hệ hỗ trợ quyết định* của kì học 2021-2.

Tác giả xin được dành lời cảm ơn chân thành tới TS. Lê Hải Hà, là giảng viên đã trực tiếp hướng dẫn và gợi ý cho tác giả đề tài rất thú vị này, đồng thời thầy cũng đã giúp đỡ tận tình và có những đóng góp bổ ích để tác giả có thể hoàn thành báo cáo này một cách tốt nhất.

Tài liệu tham khảo

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.