TRƯỜNG ĐẠI HỌC BÁCH KHOA
TPHCM KHOA KHOA HỌC VÀ KỸ
THUẬT MÁY TÍNH

SENTIMENT ANALYSIS

Môn học: Cách tiếp cận hiện đại trong xử lý ngôn ngữ tự nhiên

GVHD: PGS.TS Quản Thành Thơ

Sinh viên thực hiện:
Phạm Lê Ngọc Sơn 2470741
Email:plnson.sdh242@hcmut.edu.vn
Số điện thoại/ Zalo: 0334229371

TPHCM, ngày 9 tháng 12 năm 2025

MỤC LỤC

1. GIỚI THIỆU LONG SHORT-TERM MEMORY (LSTM)

1.1. Thuật toán LSTM là gì

LSTM (Long Short-Term Memory - Bộ nhớ dài-ngắn hạn) là một biến thể đặc biệt của mạng nơ-ron hồi quy (Recurrent Neural Network - RNN), được thiết kế để giải quyết vấn đề nan giải mà các mạng RNN truyền thống gặp phải: vấn đề triệt tiêu đạo hàm (vanishing gradient). Nếu RNN truyền thống giống như một người có "trí nhớ ngắn hạn" (chỉ nhớ được thông tin vừa xảy ra), thì LSTM là một chuyên gia có khả năng ghi nhớ các thông tin quan trọng từ rất xa trong quá khứ và bỏ qua những nhiễu loạn không cần thiết.

Về phần kỹ thuật, LSTM là một phần quan trọng của Deep Learning, đặc biệt thống trị trong các bài toán xử lý dữ liệu dạng chuỗi (sequence data) như: Xử lý ngôn ngữ tự nhiên (NLP), dịch máy, nhận dạng giọng nói và dự báo chuỗi thời gian. Khác với các mạng nơ-ron truyền thẳng (Feedforward) chỉ xử lý từng điểm dữ liệu độc lập, LSTM có các vòng lặp cho phép thông tin tồn tại bền vững.

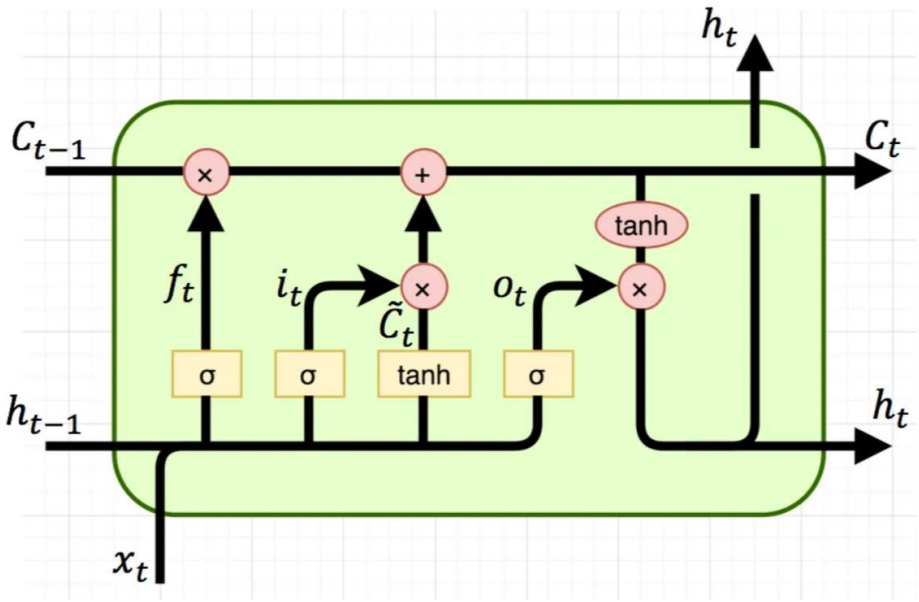
Trong quá trình training, LSTM học cách kiểm soát dòng chảy thông tin: cái gì nên nhớ, cái gì nên quên và cái gì nên chú ý tại thời điểm hiện tại. Điều này cho phép mô hình hiểu được ngữ cảnh dài hạn (long-term dependencies), ví dụ như hiểu chủ ngữ của một câu văn dài dù chủ ngữ đó nằm ở đầu đoạn văn.

1.2. Mô tả thuật toán LSTM

Các mẫu dữ liệu đầu vào của LSTM được mô tả dưới dạng chuỗi các vector theo thời gian (t). Thay vì chỉ có một tầng ẩn đơn giản như RNN truyền thống, mỗi đơn vị (unit) của LSTM sở hữu một cấu trúc phức tạp gọi là Cell State (Trạng thái tế bào) – đóng vai trò như một "đường cao tốc" giúp thông tin chạy suốt chuỗi mà ít bị biến đổi.

Để điều tiết dòng thông tin trên "đường cao tốc" này, LSTM sử dụng cơ chế Cổng (Gates). Mỗi cổng bao gồm một tầng mạng nơ-ron sử dụng hàm kích hoạt Sigmoid (đưa giá trị về khoảng 0 đến 1) và phép nhân từng điểm.

- Giá trị 0: "Đóng cửa" hoàn toàn, không cho thông tin đi qua (Quên).
- Giá trị 1: "Mở cửa" hoàn toàn, cho phép giữ lại toàn bộ thông tin (Nhớ).



Hình 1. Kiến trúc của LSTM cell

Các bước thực hiện của thuật toán LSTM tại một bước thời gian t như sau:

- Cổng Quên (Forget Gate) là bước đầu tiên trong quá trình xử lý của LSTM, nơi mạng quyết định phần thông tin nào từ trạng thái tế bào trước đó C_{t-1} không còn cần thiết và nên bị loại bỏ. Ở bước này, mô hình xem xét đồng thời đầu vào hiện tại x_t và trạng thái ẩn trước đó h_{t-1}, sau đó truyền chúng qua hàm Sigmoid. Giá trị đầu ra của Sigmoid nằm trong khoảng từ 0 đến 1, thể hiện mức độ giữ lại hay loại bỏ thông tin cũ: giá trị gần 0 tương ứng với việc thông tin bị xóa bỏ hoàn toàn, còn giá trị gần 1 cho phép giữ lại nguyên vẹn.
- Cổng Vào (Input Gate) là cơ chế thứ hai, quyết định thông tin mới nào sẽ được nạp vào bộ nhớ. Cổng này hoạt động qua hai bước nhỏ. Đầu tiên, một tầng Sigmoid sàng lọc các giá trị sẽ được cập nhật, xác định phần nào trong đầu vào là quan trọng. Tiếp theo, một tầng Tanh tạo ra một vector các giá trị mới tiềm năng C_t, đại diện cho thông tin mà mô hình có thể chọn đưa vào bộ nhớ dài hạn. Sự kết hợp giữa hai thành phần này giúp LSTM kiểm soát chặt chẽ việc tiếp nhận tri thức mới.
- Cập nhật Trạng thái Tế bào (Update Cell State) là bước then chốt trong kiến trúc LSTM, nơi bộ nhớ được điều chỉnh từ trạng thái cũ sang trạng thái mới. Trước tiên, trạng thái tế bào cũ C_{t-1} được nhân với kết quả của Cổng Quên nhằm loại bỏ những thông tin không còn cần thiết. Sau đó, mô

hình cộng thêm phần thông tin mới đã được lựa chọn bởi Cổng Vào, tạo nên trạng thái tế bào mới C_t. Kết quả này chính là "ký ức" hiện tại của mạng, đã qua quá trình chắt lọc và cập nhật phù hợp với ngữ cảnh mới.

- Cổng Ra (Output Gate) đảm nhận nhiệm vụ quyết định phần nào của ký ức sẽ được sử dụng để tạo ra trạng thái ẩn h_t tại thời điểm hiện tại—đây là thông tin mà mô hình truyền sang bước tiếp theo và cũng là cơ sở để dự đoán đầu ra. Đầu vào x_t cùng với trạng thái ẩn cũ h_{t-1} được đưa qua hàm Sigmoid để xác định mức độ ảnh hưởng đến đầu ra. Đồng thời, trạng thái tế bào C_t được chuẩn hóa qua hàm Tanh và nhân với kết quả từ Sigmoid, giúp lựa chọn thông tin phù hợp để xuất ra. Trạng thái ẩn h_t này không chỉ được truyền tiếp qua các bước thời gian mà còn có thể dùng trực tiếp cho các nhiệm vụ như phân loại cảm xúc hoặc dự đoán từ kế tiếp.
- Toàn bộ quy trình trên được lặp lại tuần tự cho từng phần tử trong chuỗi đầu vào, chẳng hạn như từng từ trong một câu. Đến cuối chuỗi, trạng thái ẩn cuối cùng sẽ chứa đựng phần lớn ý nghĩa và ngữ cảnh của toàn bộ đoạn văn bản, cho phép LSTM thực hiện các nhiệm vụ xử lý ngôn ngữ tự nhiên một cách hiệu quả.
- Lặp lại: Quá trình trên được lặp lại cho đến khi không có điểm nào trong tập gốc được thêm vào tập Z nữa (tập Z đã ổn định).

2. GIỚI THIỆU CONVOLUTIONAL NEURAL NETWORK (CNN)

2.1. Thuật toán CNN là gì

CNN (Convolutional Neural Network - Mạng Nơ-ron Tích chập) là một kiến trúc chuyên biệt của Deep Learning, được thiết kế mô phỏng theo cơ chế thị giác của sinh vật, và hiện là thuật toán thống trị trong lĩnh vực Thị giác máy tính (Computer Vision) và Xử lý ngôn ngữ tự nhiên (NLP). Khác với các mạng nơ-ron truyền thống phải xử lý toàn bộ dữ liệu đầu vào cùng một lúc, CNN sử dụng cơ chế chia sẻ trọng số để quét qua dữ liệu, giúp nó cực kỳ hiệu quả trong việc nhận diện các mẫu không gian (spatial patterns).

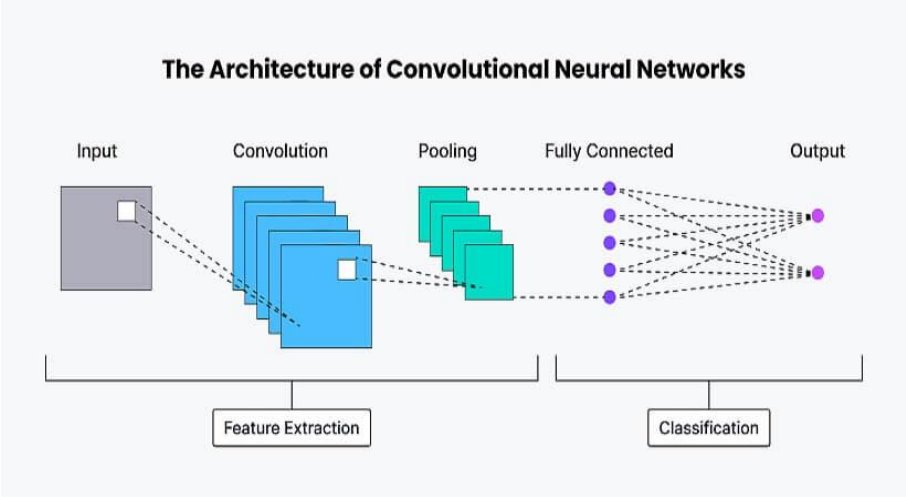
Về phần kỹ thuật, thuật toán CNN là mô hình được sử dụng chủ yếu để trích xuất

đặc trưng (feature extraction) từ dữ liệu dạng lưới (như ảnh hoặc chuỗi văn bản). Mỗi dữ liệu đầu vào sẽ đi qua một chuỗi các lớp xếp chồng lên nhau: các lớp Tích chập (Convolution) để phát hiện đặc điểm, các lớp Gộp (Pooling) để giảm tải tính toán, và cuối cùng là lớp Kết nối đầy đủ (Fully Connected) để tổng hợp thông tin.

Khi training, thuật toán này không học cách ghi nhớ từng điểm ảnh (pixel) hay từng từ, mà học các Bộ lọc (Filters/Kernels). Các bộ lọc này học cách phân biệt đâu là đường nét cơ bản, đâu là hình khối phức tạp, và đâu là đặc điểm nhận dạng quan trọng của đối tượng. Kết quả cuối cùng thường đi qua hàm Softmax để đưa ra xác suất phân loại

2.2. Mô tả thuật toán CNN

Các mẫu dữ liệu được mô tả bằng ma trận 3 chiều (Chiều cao x Chiều rộng x Độ sâu màu). Thay vì đưa nguyên ma trận khổng lồ này vào mạng nơ-ron thẳng tuột (sẽ gây bùng nổ số lượng tham số), CNN sẽ trượt các "cửa sổ nhỏ" qua toàn bộ hình ảnh để chắt lọc thông tin.



Hình 2. Kiến trúc của CNN

Các bước thực hiện của thuật toán CNN (trong một lượt lan truyền xuôi - forward pass) được mô tả như sau:

- Lớp Tích chập (Convolution Layer) là thành phần quan trọng nhất trong mô hình CNN, nơi quá trình trích xuất đặc trưng được thực hiện. Tại đây, một bộ lọc (kernel) – chẳng hạn kích thước 3x3 – sẽ trượt qua toàn bộ ma trận đầu vào. Ở mỗi vị trí mà bộ lọc đi qua, mô hình thực hiện phép nhân từng phần tử giữa kernel và vùng tương ứng trên ảnh, sau đó cộng tổng lại

để tạo ra một giá trị mới trong bản đồ đặc trưng. Khi quá trình này lặp lại trên toàn bộ vùng ảnh, CNN thu được các Feature Maps, giúp xác định những đặc điểm quan trọng như đường thẳng, đường cong hoặc các tín hiệu thị giác nổi bật trong từng khu vực của ảnh.

- Ngay sau lớp tích chập, dữ liệu được truyền qua hàm kích hoạt (Activation Function), thường sử dụng ReLU. Vai trò của bước này là đưa tính phi tuyến vào mô hình để giúp mạng học được các mẫu phức tạp trong dữ liệu thực tế. ReLU đặc biệt hữu ích vì nó loại bỏ các giá trị âm – những vị trí không mang đặc trưng quan trọng – trong khi giữ lại các tín hiệu dương có ý nghĩa, giúp mô hình tập trung hơn vào các đặc trưng nổi bật.
- Tiếp theo là Lớp Gộp (Pooling Layer), nơi thông tin được nén lại nhằm giảm kích thước dữ liệu và hạn chế nguy cơ overfitting. Phương pháp phổ biến nhất là Max Pooling, trong đó mạng lấy giá trị lớn nhất trong một vùng nhỏ, ví dụ 2x2. Mục đích chính của lớp gộp không phải xác định đặc trưng ở pixel chính xác nào, mà là xác nhận sự tồn tại của đặc trưng đó trong một khu vực.
- Sau khi trải qua nhiều lớp tích chập và gộp, dữ liệu được chuyển đến Lớp Kết nối đầy đủ (Fully Connected Layer) để đưa ra quyết định cuối cùng. Lúc này, các đặc trưng đã được cô đặc thành dạng trừu tượng hơn, chẳng hạn mô hình có thể nhận diện được các thành phần như mắt, tai hoặc lông. Dữ liệu 3 chiều từ các bản đồ đặc trưng được chuyển đổi thành một vector 1 chiều thông qua bước Flatten. Vector này được đưa vào mạng nơ-ron truyền thống, nơi các trọng số được kết hợp lại để tạo ra dự đoán cuối cùng.
- Toàn bộ quá trình trên được lặp lại nhiều lần trong giai đoạn huấn luyện. Nếu mô hình dự đoán sai, thuật toán Lan truyền ngược (Backpropagation) sẽ điều chỉnh lại các tham số của bộ lọc (Kernel) nhằm tối ưu hóa khả năng nhận diện trong những lần dự đoán tiếp theo. Nhờ cơ chế học liên tục này, CNN ngày càng cải thiện độ chính xác trong việc phân loại và nhận dạng đặc trưng.

3. CƠ SỞ TOÁN HỌC CỦA MÔ HÌNH

3.1. Phép tích chập 1 chiều (1D-Convolution) - Sử dụng trong TextCNN

Khác với xử lý ảnh (2D), trong xử lý ngôn ngữ tự nhiên, phép tích chập 1 chiều được sử dụng để trích xuất các đặc trưng cục bộ (n-grams) từ chuỗi văn bản. Giả sử $x_i \in R^k$ là vector biểu diễn từ thứ i trong câu có độ dài n.

Một bộ lọc (kernel) $w \in R^{hk}$ với kích thước cửa sổ h sẽ trượt qua chuỗi đầu vào. Đặc trưng c_i được tạo ra từ cửa sổ từ $x_{i:i+h-1}$ theo công thức:

$$c_i = f(w \cdot x_{i:i+h-1} + b)$$

Trong đó:

- $b \in R$ là hệ số bias.
- f là hàm kích hoạt phi tuyến tính (trong bài sử dụng hàm ReLU: f (x) = max(0,x)).
- Phép toán này tạo ra một bản đồ đặc trưng (feature map) $c=[c_1, c_2, \ldots, c_{n-h+1}]$.

Sau đó, lớp Global Max Pooling được áp dụng để lấy giá trị quan trọng nhất từ bản đồ đặc trưng:.

$$\hat{c} = \max(c)$$

3.2. Mạng bộ nhớ dài-ngắn hạn (LSTM) - Sử dụng trong BiLSTM

LSTM được thiết kế để giải quyết vấn đề phụ thuộc xa trong chuỗi văn bản thông qua cơ chế cổng (gates). Tại mỗi bước thời gian t, LSTM nhận đầu vào x_t và trạng thái ẩn trước đó h_{t-1}. Các công thức tính toán bên trong một tế bào LSTM như sau:

Tọa độ 2 điểm A(x_A, y_A) và B(x_B, y_B) khi đó khoảng cách Manhattan giữa 2 điểm như sau:

Các cổng kiểm soát thông tin:

Cổng quên (Forget gate): $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$

Cổng vào (Input gate): $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

Cổng ra (Output gate): $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Cập nhật trạng thái tế bào (Cell state):

Thông tin ứng viên: $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

Trạng thái tế bào mới: $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$

Đầu ra (Hidden state):

$$h_t = o_t * \tanh(C_t)$$

Trong mô hình Bi-LSTM (như trong Lab), quá trình này được thực hiện theo hai chiều: chiều xuôi (forward) và chiều ngược (backward), sau đó kết hợp vector đầu ra của cả hai chiều để nắm bắt ngữ cảnh toàn diện.

3.3. Hàm mất mát - Categorical Cross-Entropy

Bài toán phân loại đa lớp (3 lớp: Tiêu cực, Trung tính, Tích cực) sử dụng hàm Categorical Cross-Entropy để đo "khoảng cách" giữa phân phối xác suất dự đoán và nhãn thực tế.

Với y là vector one-hot của nhãn thực tế và là vector xác suất dự đoán từ mô hình (đầu ra của lớp Softmax), hàm mất mát cho một mẫu dữ liệu được tính như sau:

$$L(y, \hat{y}) = -\sum_{i=1}^{C} y_i \cdot \log(\hat{y}_i)$$

Trong đó:

- C là số lượng lớp (trong bài toán này C=3).
- Mục tiêu của quá trình huấn luyện là tối thiểu hóa giá trị L này thông qua thuật toán tối ưu hóa (trong bài sử dụng Adam Optimizer).

4. CHIẾN LƯỢC TỐI ƯU HÓA VÀ LỰA CHỌN SIÊU THAM SỐ

4.1. Kích thước bộ lọc (Kernel Size) trong TextCNN

Trong thuật toán TextCNN gốc, thay vì cố định một cửa sổ quan sát, mô hình sử dụng chiến lược "Đa kích thước bộ lọc" (Multi-kernel strategy). Mục tiêu là xây dựng một tập hợp các đặc trưng Z từ câu đầu vào sao cho việc kết hợp chúng có thể nắm bắt được cả ngữ nghĩa cục bộ ngắn hạn và dài hạn.

Cụ thể, trong bài thực hành này, thay vì chọn một giá trị cố định, mô hình sử dụng đồng thời 3 kích thước bộ lọc: k={3,4,5}.

- k=3 (3-grams): Bắt các cụm từ ngắn (ví dụ: "rất tốt", "quá tệ").
- k=4,5(4,5-grams): Bắt các cụm từ dài hơn hoặc cấu trúc phủ định (ví dụ: "không thể tin được", "chất lượng không tốt lắm").

Điều này giúp mô hình "nhất quán" trong việc phát hiện đặc trưng cảm xúc dù người dùng viết ngắn hay dài.

4.2. Cơ chế kiểm soát quá khớp và Mất cân bằng dữ liệu

Điều Trong quá trình xây dựng mô hình (Training), để đảm bảo mô hình không "học vẹt" (Overfitting) trên tập huấn luyện T, thuật toán áp dụng cơ chế Dropout.

Trong báo cáo này, mô hình cải tiến (v2) sử dụng SpatialDropout1D (với tỷ lệ 0.2 - 0.55). Khác với Dropout thường (tắt ngẫu nhiên các nơ-ron), SpatialDropout1D tắt toàn bộ các chiều của vector đặc trưng (feature map). Điều này buộc mô hình phải học các đặc trưng mạnh mẽ hơn, không phụ thuộc vào một vài từ khóa cụ thể, giúp làm mượt biên quyết định và tăng độ tổng quát hóa khi dự đoán trên tập dữ liệu thực tế.

Ngoài ra, vì dữ liệu VLSP có sự chênh lệch lớn giữa các nhãn (Imbalanced Data), phương pháp Class Weighting (Trọng số lớp) được áp dụng.

Nếu một điểm dữ liệu thuộc lớp thiểu số (ít mẫu) bị phân loại sai, hàm mất mát sẽ phạt nặng hơn so với lớp đa số.

Công thức điều chỉnh trọng số: $$W_{class} = \frac{N_{samples}}{N_{classes} \times N_{samples\_in\_class}}$$

Việc này đảm bảo mô hình không bị thiên vị (bias) về phía các nhãn xuất hiện nhiều (như nhãn Tiêu cực trong tập dữ liệu này), tương tự như việc "bổ sung mẫu quan trọng" vào tập nhớ trong các thuật toán cổ điển.

5. ĐÁNH GIÁ HIỆU SUẤT VÀ SỰ ĐÁNH ĐỔI

5.1. Độ chính xác và F1-Score

Đánh giá khả năng "hiểu" ngôn ngữ của mô hình.

Độ chính xác (Accuracy): Được tính bằng tỷ lệ số mẫu dự đoán đúng trên tổng

số mẫu kiểm thử.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Macro F1-Score: Do tập dữ liệu VLSP bị mất cân bằng (Imbalanced Data) – số lượng bình luận Tiêu cực (-1) chiếm đa số, nên độ chính xác đơn thuần có thể gây hiểu lầm. F1-Score (trung bình điều hòa giữa Precision và Recall) là chỉ số quan trọng hơn để đảm bảo mô hình nhận diện tốt cả những lớp thiểu số (như Tích cực hoặc Trung tính).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

5.2. Độ phức tạp và Chi phí huấn luyện

Số lượng tham số (Parameters): Tổng số trọng số (W) và bias (b) mà mạng cần học. Số lượng tham số càng lớn, mô hình càng mạnh nhưng càng tốn bộ nhớ và dễ bị học vẹt (Overfitting).

- Ví dụ trong bài: TextCNN (~3.7 triệu tham số) vs. BiLSTM (~3.7 - 4.0 triệu tham số).

Thời gian huấn luyện (Training Time): Thời gian để mô hình hoàn thành một vòng lặp (Epoch).

- Quan sát từ thực nghiệm: TextCNN thường huấn luyện rất nhanh (do tính toán song song) trong khi BiLSTM chậm hơn đáng kể (do phải xử lý tuần tự theo chuỗi thời gian).

5.3. Kết luận về sự đánh đổi

Một mô hình lý tưởng không chỉ là mô hình có độ chính xác cao nhất, mà phải cân bằng được chi phí.

- TextCNN: Phù hợp khi cần tốc độ xử lý nhanh, tài nguyên thấp, chấp nhận độ chính xác thấp hơn một chút ở các câu dài phức tạp.
- BiLSTM (Proposed Method): Trong bài báo cáo này, BiLSTM (v2) được chứng minh là tối ưu nhất. Dù chi phí tính toán cao hơn CNN, nhưng nó

đạt được Macro F1 và Accuracy cao nhất (~67.9%), giải quyết tốt vấn đề ngữ cảnh dài hạn của tiếng Việt mà CNN bỏ sót.

6. ÁP DỤNG CÁC MÔ HÌNH DEEP LEARNING VÀO BÀI TOÁN SENTIMENT ANALYSIS

Hình 3. Phân tích cảm xúc – Sentiment Analysis

6.1. Định nghĩa bài toán

6.1.1. Làm sạch dữ liệu

Phân tích cảm xúc (Sentiment Analysis), hay còn gọi là Khai phá ý kiến (Opinion Mining), là một trong những bài toán cốt lõi và thách thức nhất trong lĩnh vực Xử lý ngôn ngữ tự nhiên (NLP). Mục tiêu chính của bài toán này là sử dụng các thuật toán máy tính để tự động xác định, trích xuất và định lượng trạng thái cảm xúc, thái độ hoặc ý kiến chủ quan của người viết đối với một thực thể cụ thể (sản phẩm, dịch vụ, tổ chức, hoặc sự kiện).

Trong kỷ nguyên số, người dùng internet không chỉ là người tiếp nhận thông tin mà còn là người tạo ra nội dung (User-Generated Content). Hàng triệu bình luận, đánh giá (reviews) được sinh ra mỗi ngày trên các nền tảng thương mại điện tử (Shopee, Tiki, Lazada...) và mạng xã hội. Đây là nguồn dữ liệu vô giá nhưng lại ở dạng phi cấu trúc (unstructured text), khiến việc phân tích thủ công trở nên bất khả thi.

6.1.2. Bối cảnh và Mục tiêu của dự án

Bài toán Phân tích cảm xúc cấp độ câu (Sentence-level Sentiment Analysis) trên miền dữ liệu cụ thể là Sản phẩm công nghệ (Smartphone, Laptop, Phụ kiện điện tử...). Đây là một miền dữ liệu đặc thù với nhiều thuật ngữ chuyên ngành (ví dụ:

"ram", "chip", "đơ", "lag", "pin trâu"...) đòi hỏi mô hình phải có khả năng hiểu ngữ cảnh sâu sắc.

Việc đọc và phân loại thủ công hàng ngàn bình luận không chỉ tốn kém thời gian, nhân lực mà còn dễ bị ảnh hưởng bởi sự chủ quan của con người. Do đó, việc ứng dụng Deep Learning (Học sâu) là giải pháp tối ưu để tự động hóa quy trình này với độ chính xác cao và khả năng mở rộng lớn.

Mục tiêu cụ thể: Xây dựng một mô hình máy học có khả năng nhận đầu vào là một đoạn văn bản tiếng Việt (bình luận về sản phẩm) và tự động gán nhãn phân loại vào một trong 3 cảm xúc xác định.

6.1.3. Không gian nhãn

Bài toán được định nghĩa là một bài toán phân loại đa lớp (Multi-class Classification) với 3 nhãn đầu ra (C=3). Việc định nghĩa rõ ràng các nhãn là cực kỳ quan trọng để mô hình học được các đặc trưng đúng đắn:

Nhãn Tiêu cực (Negative - Label -1):

Nhãn Tiêu cực (Negative – Label -1) bao gồm những bình luận thể hiện sự không hài lòng, thất vọng, tức giận hoặc chê bai trực tiếp sản phẩm hay dịch vụ. Các câu thuộc nhóm này thường chứa những từ khóa mang sắc thái phủ định mạnh hoặc các tính từ mô tả chất lượng kém, giúp mô hình dễ dàng nhận diện đặc trưng của lớp tiêu cực.

Ví dụ thực tế:

- Máy dùng rất nóng, pin tụt nhanh như tụt quần." (Chê bai hiệu năng).
- "Giao hàng quá chậm, shop làm ăn tắc trách, sẽ không quay lại." (Phàn nàn dịch vụ).
- "Màn hình bị ám vàng, nhìn rất đau mắt." (Lỗi kỹ thuật).

Nhãn Trung tính (Neutral - Label 0):

Nhãn Trung tính bao gồm những bình luận không thể hiện rõ cảm xúc tích cực hay tiêu cực, khiến đây trở thành lớp khó phân loại nhất do ranh giới ngữ nghĩa thường rất mờ. Các câu thuộc nhóm này thường là những câu hỏi tìm kiếm thông tin, nhận xét mang tính mô tả khách quan về thông số kỹ thuật, hoặc các đánh giá pha trộn cả ưu và nhược điểm một cách cân bằng (mixed reviews), khiến mô hình khó

suy luận cảm xúc một cách dứt khoát.

Ví dụ thực tế:

- "Máy này có hỗ trợ sạc nhanh không shop?" (Câu hỏi - Query).
- "Cấu hình chip Snap 865, Ram 8GB." (Liệt kê thông số - Fact).
- "Máy đẹp nhưng giá hơi cao." (Ý kiến trái chiều cân bằng).

Nhãn Tích cực (Positive - Label 1):

Nhãn Tích cực bao gồm những bình luận thể hiện sự hài lòng, yêu thích, phấn khích hoặc khuyến khích người khác nên mua sản phẩm. Các câu thuộc nhóm này thường chứa những từ khóa mang sắc thái ca ngợi, các tính từ tích cực và đôi khi xuất hiện kèm theo các biểu tượng cảm xúc vui vẻ, giúp mô hình dễ dàng nhận diện cảm xúc tích cực một cách rõ ràng.

Ví dụ thực tế:

- "Điện thoại chụp ảnh siêu nét, đáng đồng tiền bát gạo." (Khen ngợi chất lượng).
- "Giao hàng thần tốc, đóng gói rất cẩn thận, 5 sao cho shop." (Hài lòng về dịch vụ).
- "Pin trâu lắm, dùng 2 ngày mới phải sạc." (Trải nghiệm tốt).

6.1.4. Chọn phương pháp Deep Learning

Các phương pháp máy học truyền thống (như SVM, Naive Bayes) dựa trên tần suất từ (Bag-of-Words) thường gặp hạn chế lớn khi xử lý tiếng Việt.

Một hạn chế quan trọng của các mô hình truyền thống như Bag-of-Words hoặc TF-IDF là mất mát thông tin về thứ tự từ. Khi văn bản được biểu diễn dưới dạng tập hợp từ không có cấu trúc, mô hình không thể phân biệt được sự khác biệt về ý nghĩa giữa các câu như “Rất tốt, không tệ” và “Rất tệ, không tốt”. Mặc dù các câu này chứa cùng một tập từ, cách sắp xếp khác nhau lại dẫn đến sắc thái cảm xúc hoàn toàn trái ngược. Tuy nhiên, với biểu diễn không xét đến thứ tự, mô hình sẽ coi chúng tương đương nhau, dẫn đến sai lệch nghiêm trọng trong phân loại cảm xúc.

Bên cạnh đó, các phương pháp truyền thống còn gặp nhiều khó khăn trong quá trình trích xuất đặc trưng. Việc thiết kế các đặc trưng thủ công (feature engineering) đòi hỏi rất nhiều công sức và chuyên môn, từ việc lựa chọn n-gram phù hợp, xử lý phủ định cho đến tách cụm từ cảm xúc. Không những tốn thời gian, phương pháp này còn dễ bỏ sót các đặc trưng ngữ nghĩa quan trọng, khiến mô hình

khó đạt hiệu quả cao trong các nhiệm vụ xử lý ngôn ngữ tự nhiên phức tạp như phân tích cảm xúc.

Các mô hình Deep Learning như CNN (Convolutional Neural Networks) và LSTM (Long Short-Term Memory) mang lại ưu thế vượt trội:

- CNN: Có khả năng trích xuất các đặc trưng cục bộ (local features) xuất sắc, giúp nhận diện nhanh các cụm từ khóa quan trọng (n-grams) mang tính cảm xúc (ví dụ: "quá đẹp", "tệ hại").
- LSTM: Có khả năng ghi nhớ thông tin dài hạn, giúp mô hình hiểu được ngữ cảnh toàn cục của câu, giải quyết tốt các cấu trúc câu phức tạp, câu phủ định hoặc châm biếm mà mô hình đơn giản thường đoán sai.

Việc kết hợp và so sánh các phương pháp này trên bộ dữ liệu VLSP sẽ cung cấp cái nhìn toàn diện về hiệu quả của Deep Learning trong xử lý ngôn ngữ tự nhiên tiếng Việt.

6.2. Thu thập và tiền xử lý dữ liệu

6.2.1. Tổng quan về bộ dữ liệu

Bộ dữ liệu bao gồm các ý kiến, bình luận (reviews) của người dùng về các sản phẩm công nghệ (điện thoại, máy tính, phụ kiện...) được thu thập từ các trang thương mại điện tử và mạng xã hội.

- Cấu trúc dữ liệu: Mỗi mẫu dữ liệu bao gồm 2 trường thông tin chính:
  - Content (Data): Nội dung văn bản bình luận của người dùng.
  - .Label (Class): Nhãn cảm xúc đã được gán thủ công bởi con người.
- Không gian nhãn: Bài toán được định nghĩa là phân loại đa lớp (Multi-class Classification) với 3 nhãn:
  - Tiêu cực (-1): Chê bai chất lượng, phàn nàn về giá cả, dịch vụ (Ví dụ: "Pin yếu quá, dùng nửa ngày đã hết").
  - Trung tính (0): Hỏi thông tin, nhận xét khách quan không mang sắc thái tình cảm rõ rệt (Ví dụ: "Máy này cấu hình ram bao nhiêu?").

- Tích cực (1): Khen ngợi, hài lòng, khuyên người khác mua (Ví dụ: "Chụp ảnh siêu nét, đáng đồng tiền bát gạo").

- Thống kê kích thước:
  - Tập huấn luyện (Training Set): ~5.100 mẫu.
  - Tập kiểm thử (Test Set): ~1.050 mẫu.

Bảng 1. Thống kê đặc trưng của tập dữ liệu VLSP

Đặc trưng (Feature)	**Giá trị (Value)**	**Mô tả (Description)**
Tổng số mẫu (Total Samples)	~6,150	Tổng số lượng bình luận trong cả tập Train và Test.
Kích thước tập Train	5,100 (83%)	Dữ liệu dùng để huấn luyện mô hình.
Kích thước tập Test	1,050 (17%)	Dữ liệu dùng để kiểm thử độc lập.
Chiều dài câu tối đa (Max Len)	180	Số lượng từ tối đa trong một câu (sau khi Padding).
Kích thước từ điển (Vocab Size)	12,000	Số lượng từ vựng phổ biến nhất được giữ lại.
Số chiều Embedding	400	Kích thước vector biểu diễn cho mỗi từ (Word2Vec).

Bảng 2. Phân bố nhãn và Mẫu dữ liệu minh họa

STT	**Nhãn (Class)**	**Ý nghĩa (Sentiment)**	**Ví dụ nội dung gốc (Raw Data Example)**	**Đặc điểm nhận dạng**
1	-1	Tiêu cực (Negative)	"máy xài chán quá, pin tuột như tụt quần, thất vọng ghê"	Chứa từ chê bai mạnh (chán, tụt, thất vọng), phàn nàn về tính năng.
2	0	Trung tính (Neutral)	"cho mình hỏi con này ram mấy gb vậy shop, có màu đen ko"	Câu hỏi thông tin, không thể hiện thái độ yêu/ghét.
3	1	Tích cực (Positive)	"hàng ngon bổ rẻ, chụp hình đẹp lung linh, giao	Chứa từ khen ngợi (ngon, rẻ, đẹp, nhanh), thể hiện sự hài

			hàng nhanh"	lòng.

Trong phân bố dữ liệu cảm xúc, các bình luận mang sắc thái Tiêu cực (-1) và Tích cực (1) thường chiếm tỷ lệ lớn hơn nhiều so với Trung tính (0). Điều này phản ánh xu hướng phổ biến của người dùng khi để lại đánh giá: họ thường bày tỏ cảm xúc mạnh (hài lòng hoặc không hài lòng) hơn là đưa ra nhận xét trung lập.

Ngược lại, số lượng bình luận Trung tính thường ít hơn đáng kể và thậm chí dễ bị lẫn lộn với hai còn lại do sắc thái biểu đạt không rõ ràng. Các câu trung tính đôi khi chứa các từ ngữ không mang cảm xúc rõ rệt, hoặc có cấu trúc khiến mô hình khó phân biệt đâu là ý kiến đánh giá trung lập, đâu là nhận xét nhẹ nhàng mang tính tích cực hoặc tiêu cực.

Hệ quả của sự mất cân bằng này là mô hình có xu hướng nghiêng về việc dự đoán các lớp chiếm đa số nếu không có biện pháp điều chỉnh phù hợp. Vì vậy, trong quá trình huấn luyện, kỹ thuật Class Weighting được áp dụng nhằm gán trọng số phạt cao hơn cho các mẫu thuộc lớp thiểu số. Điều này giúp mô hình học cân bằng hơn, tránh thiên lệch và cải thiện độ chính xác tổng thể, đặc biệt đối với dữ liệu Trung tính.

6.2.2. Thách thức từ dữ liệu thực tế

Dữ liệu văn bản tiếng Việt trên mạng xã hội có đặc thù là rất "nhiễu" (noisy data), gây khó khăn cho máy tính trong việc hiểu ngữ nghĩa:

- Teencode/Viết tắt: Người dùng thường xuyên sử dụng các từ không chuẩn mực (VD: "hok" thay vì "không", "dt" thay vì "điện thoại", "đc" thay vì "được").
- Sai chính tả và không dấu: Nhiều bình luận thiếu dấu câu hoặc gõ sai (VD: "hang chat luong" có thể hiểu là "hàng chất lượng" hoặc "hàng chật lượng").
- Ký tự lạ & Icon: Chứa nhiều emoji, ký tự đặc biệt (@, #, $%) không mang lại giá trị ngữ nghĩa cho mô hình CNN/LSTM.
- Cấu trúc ngữ pháp lỏng lẻo: Câu văn không tuân theo ngữ pháp chuẩn, thường là các câu cụt hoặc câu ghép phức tạp.

6.2.3. Quy trình tiền xử lý chi tiết

Để chuyển đổi dữ liệu thô thành dạng vector số học mà mạng nơ-ron có thể

xử lý, thiết lập một quy trình (pipeline) gồm 4 bước chính:

Bước 1: Vòng lặp sàng lọc (The Condensing Loop)

Trong bước làm sạch dữ liệu, các biểu thức chính quy (Regular Expressions – Regex) được sử dụng để lọc bỏ các thành phần gây nhiễu. Trước hết, các ký tự số và ký tự đặc biệt được loại bỏ hoặc thay thế bằng khoảng trắng, bởi những thông tin này – chẳng hạn như ngày tháng hay thông số kỹ thuật – thường không mang giá trị cảm xúc. Tiếp theo, toàn bộ văn bản được chuẩn hóa về chữ thường nhằm đồng nhất dữ liệu và giảm kích thước bộ từ điển, giúp mô hình xử lý hiệu quả hơn. Cuối cùng, các khoảng trắng thừa ở đầu, cuối hoặc giữa câu được loại bỏ để đảm bảo văn bản sạch và chuẩn trước khi đưa vào bước xử lý tiếp theo.

Bước 2: Tách từ tiếng Việt (Vietnamese Tokenization)

Khác với tiếng Anh, nơi từ được tách rõ ràng bằng dấu cách, tiếng Việt là ngôn ngữ đơn âm tiết nhưng lại cho phép hình thành từ đa âm tiết dưới dạng từ ghép. Nếu chỉ dựa vào dấu cách để tách từ, các cụm như "đồng hồ" sẽ bị tách thành "đồng" và "hồ", hoàn toàn sai lệch về mặt ngữ nghĩa. Để khắc phục vấn đề này, hệ thống sử dụng thư viện PyVi với thuật toán tách từ dựa trên thống kê, giúp nhận diện chính xác các từ ghép. Kết quả thu được là các cụm từ nhiều âm tiết được chuẩn hóa bằng cách nối bằng dấu gạch dưới, chẳng hạn "đồng_hồ", đảm bảo giữ nguyên nghĩa khi đưa vào mô hình xử lý ngôn ngữ.Sử dụng thư viện PyVi (Python Vietnamese Core) với thuật toán tách từ thống kê.

```python
def clean_text(text: str) -> str:
    # Drop digits/extra spaces and lowercase to reduce noise
    text = re.sub(r"\d+", " ", str(text))
    text = re.sub(r"\s+", " ", text).strip().lower()
    return text


def tokenize_vi(text: str) -> str:
    # Vietnamese word segmentation via pyvi (keeps compound words)
    return ViTokenizer.tokenize(text)
```

Bước 3: Xây dựng từ điển và Mã hóa chuỗi (Indexing & Vectorization)

Quá trình xây dựng từ điển (vocabulary) được thực hiện bằng cách quét toàn bộ tập huấn luyện và lựa chọn 12.000 từ xuất hiện thường xuyên nhất, tương ứng với giá trị MAX_VOCAB_SIZE. Những từ hiếm gặp hoặc ít xuất hiện sẽ bị loại bỏ nhằm

giảm nhiễu và hạn chế nguy cơ overfitting. Sau đó, mỗi từ được ánh xạ thành một chỉ số nguyên duy nhất; chẳng hạn, "điện_thoại" có thể được gán index 1, "tốt" là 2 và "pin" là 3. Cuối cùng, toàn bộ câu văn bản được chuyển đổi thành chuỗi các số nguyên tương ứng, tạo đầu vào chuẩn hoá cho mô hình học sâu.

Bước 4: Cắt/Thêm độ dài cố định (Padding/Truncating)

- Thiết lập độ dài chuẩn MAX_SEQUENCE_LENGTH = 180.
- Thực hiện:
  - Nếu câu dài hơn 180 từ: Cắt bỏ phần sau.
  - Nếu câu ngắn hơn 180 từ: Thêm các số 0 (Padding) vào phía trước cho đủ độ dài.

```python
def load_and_prepare(df: pd.DataFrame):
    # Pipeline: clean -> tokenize -> return tokens and one-hot labels
    cleaned = df['Data'].astype(str).apply(clean_text).apply(tokenize_vi)
    tokens = [text.split() for text in cleaned]
    labels = encode_labels(df['Class'].values)
    return tokens, labels


def pad(tokenizer: Tokenizer, sequences, maxlen: int):
    # Convert text to ids and pad to fixed length
    ids = tokenizer.texts_to_sequences(sequences)
    return pad_sequences(ids, maxlen=maxlen)
```

Bảng 3. Minh họa quy trình tiền xử lý dữ liệu (Preprocessing Pipeline)

Bước xử lý	**Trạng thái dữ liệu (Data State)**	**Ví dụ minh họa (Example)**
1. Đầu vào thô	Văn bản gốc (chứa nhiễu, viết hoa, số)	"ĐT này xài OK lắm, pin 4000mAh trâu bò!"
2. Làm sạch	Lowercase, bỏ số, bỏ ký tự lạ	"đt này xài ok lắm pin trâu bò"
3. Tách từ (Tokenization)	Gộp từ ghép tiếng Việt (pyvi)	"đt này xài ok lắm pin trâu_bò"
4. Mã hóa (Indexing)	Chuyển từ sang số nguyên (Index)	[45, 12, 33, 89, 5, 204]
5. Padding	Cố định độ dài (Max len = 180)	[0, 0, ..., 45, 12, 33, 89, 5, 204]

6.2.4. Nhúng từ (Word Embedding) - Pre-trained Word2Vec

Thay vì để mô hình tự học ý nghĩa của từng từ từ đầu (Random Initialization), phương pháp này sử dụng tri thức chuyển giao (Transfer Learning) để tận dụng những mô hình đã được huấn luyện trước. Cách tiếp cận này giúp mô hình không phải bắt đầu từ con số 0, mà kế thừa ngay lập tức những hiểu biết ngữ nghĩa đã có sẵn từ các tập dữ liệu lớn.

Trong nghiên cứu này, mô hình Word2Vec với kiến trúc CBOW được sử dụng làm nền tảng tiền huấn luyện. Mô hình này đã được huấn luyện trước trên một lượng dữ liệu tiếng Việt rất lớn, bao gồm Wikipedia, các nguồn báo chí và nhiều kho ngữ liệu khác. Nhờ vậy, các biểu diễn từ thu được mang tính khái quát cao và phản ánh tốt các quan hệ ngữ nghĩa trong tiếng Việt.

Cơ chế hoạt động của Word2Vec cho phép mỗi từ (dưới dạng một chỉ số — Index) được ánh xạ sang một vector trong không gian 400 chiều (EMBEDDING_DIM = 400). Các vector này không chỉ là những con số rời rạc, mà là sự mã hóa thông tin ngữ nghĩa và ngữ cảnh mà mô hình đã học được từ tập dữ liệu gốc.

Lợi ích quan trọng nhất là các từ có ý nghĩa tương đồng sẽ nằm gần nhau hơn trong không gian vector. Ví dụ, vector biểu diễn cho từ "đẹp" sẽ gần với "xinh", trong khi từ "tệ" sẽ nằm gần "kém". Điều này giúp mô hình học nhanh hơn, giảm thời gian hội tụ và đặc biệt là tăng khả năng hiểu ngữ nghĩa sâu của văn bản, từ đó cải thiện hiệu quả trong các tác vụ xử lý ngôn ngữ tự nhiên.Triển khai các mô hình và Tối ưu hóa

Dựa trên đặc thù của dữ liệu văn bản tiếng Việt và mục tiêu phân loại cảm xúc, đã thiết kế và thực nghiệm 3 kiến trúc Deep Learning khác nhau: TextCNN, Bi-LSTM và mô hình lai CNN-BiLSTM. Quá trình này không chỉ dừng lại ở việc áp dụng các mô hình cơ bản (Baseline) mà còn bao gồm các chiến lược tối ưu hóa nâng cao (phiên bản v2) để cải thiện hiệu suất.

6.3. Triển khai các mô hình và Tối ưu hóa

6.3.1. Mô hình 1: TextCNN (Convolutional Neural Network for Text)

Kiến trúc chi tiết: Đầu vào là ma trận câu có kích thước (180×400) (độ dài câu × số chiều embedding).

- Convolutional Layer (Lớp tích chập): Thay vì dùng kernel vuông (2D) như trong xử lý ảnh, sử dụng các bộ lọc 1 chiều (1D-Filters) trượt dọc theo chiều dài của câu. sử dụng đồng thời 3 kích thước bộ lọc khác nhau:
    - Kernel size = 3 (Trigrams): Bắt các cụm 3 từ (VD: "dùng rất thích", "pin hơi yếu").
    - Kernel size = 4 (4-grams): Bắt các cụm từ dài hơn (VD: "màn hình sắc nét").
    - Kernel size = 5 (5-grams): Bắt các cấu trúc câu phức tạp hơn.

Trong giai đoạn rút trích đặc trưng, mô hình sử dụng cơ chế Global Max Pooling để chọn lọc thông tin quan trọng nhất. Đối với mỗi bản đồ đặc trưng được tạo ra từ các bộ lọc tích chập, lớp này chỉ giữ lại giá trị lớn nhất. Cách làm này tương đương với việc lựa chọn "tín hiệu cảm xúc mạnh nhất" xuất hiện trong câu, đồng thời loại bỏ các yếu tố nhiễu như stop-words hay những từ không mang nhiều ý nghĩa cảm xúc. Nhờ đó, mô hình có khả năng tập trung hiệu quả vào những đặc trưng quan trọng nhất phục vụ cho quá trình phân loại.

Sau khi các đặc trưng quan trọng được trích xuất, chúng được ghép nối (concatenate) và chuyển đến lớp Fully Connected, trước khi đi qua hàm kích hoạt Softmax để tạo ra phân lớp cuối cùng. Đây là bước tổng hợp và ra quyết định, nơi mô hình biến toàn bộ biểu diễn đặc trưng thành xác suất tương ứng với các nhãn cảm xúc.

Một trong những ưu điểm nổi bật của mô hình CNN cho xử lý văn bản là tốc độ huấn luyện rất nhanh. Các phép tính tích chập được thực hiện song song trên nhiều bộ lọc, giúp rút ngắn đáng kể thời gian huấn luyện mà vẫn đảm bảo hiệu quả trích xuất đặc trưng. Ngoài ra, mô hình đặc biệt nhạy bén trong việc phát hiện các từ khóa mang tính quyết định, chẳng hạn như "tệ", "ngon", "xuất sắc", bất kể vị trí của chúng ở đâu trong câu. Điều này khiến CNN trở thành một trong những lựa chọn hiệu quả nhất khi bài toán cần phát hiện tín hiệu cảm xúc rõ ràng và trực tiếp.

```python
def build_textcnn():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    convs = []
    for k in [3, 4, 5]:
        # Kernels 3-5 capture different n-grams
        c = layers.Conv1D(128, k, activation="relu", padding="valid")(x)
        p = layers.GlobalMaxPooling1D()(c)
        convs.append(p)
    x = layers.Concatenate()(convs)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="textcnn")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

6.3.2. Mô hình 2: Bi-LSTM (Bidirectional Long Short-Term Memory)

Cơ chế hoạt động: LSTM Cells: Sử dụng các cổng (Input, Output, Forget Gate) để quyết định giữ lại hay quên thông tin, giúp giải quyết vấn đề biến mất đạo hàm (vanishing gradient) khi câu quá dài.

- Cơ chế hai chiều (Bidirectional): Mô hình bao gồm hai lớp LSTM chạy song song:
  - Lớp xuôi (Forward): Đọc câu từ trái sang phải (từ đầu đến cuối) để hiểu ngữ cảnh quá khứ.
  - Lớp ngược (Backward): Đọc câu từ phải sang trái (từ cuối lên đầu) để hiểu ngữ cảnh tương lai.

Trong kiến trúc mô hình, lớp SpatialDropout1D được sử dụng nhằm giảm thiểu hiện tượng quá khớp (Overfitting). Khác với dropout thông thường chỉ loại bỏ các nút riêng lẻ, SpatialDropout1D áp dụng dropout lên toàn bộ chiều đặc trưng (feature dimension). Điều này buộc mô hình phải học các biểu diễn mang tính khái quát hơn, tránh việc phụ thuộc vào một vài từ cụ thể và giúp cải thiện khả năng tổng quát hóa.

Bên cạnh đó, mô hình Bi-LSTM mang lại nhiều ưu thế nổi bật. Một trong những ưu điểm quan trọng là khả năng hiểu ngữ cảnh toàn diện của câu nhờ cơ chế đọc hai chiều — từ trái sang phải và từ phải sang trái. Chẳng hạn, trong câu "Máy này cấu hình thì mạnh nhưng pin lại quá yếu", Bi-LSTM có thể nhận ra rằng về sau ("pin yếu") mang trọng số cảm xúc mạnh hơn về trước, từ đó đưa ra đánh giá chính xác hơn.

Mô hình này cũng xử lý hiệu quả các cấu trúc phủ định và sắc thái châm biếm

vốn rất phổ biến trong tiếng Việt. Ví dụ, câu "Không phải là không đẹp" mang ý nghĩa tích cực, và Bi-LSTM có khả năng nắm bắt được sự đảo nghĩa này nhờ cơ chế kết hợp thông tin từ cả hai hướng đọc của chuỗi.

Cuối cùng, Bi-LSTM thường được xem là một trong những kiến trúc đạt hiệu quả cao nhất trong các mô hình không sử dụng Transformer. Nhờ khả năng ghi nhớ thông tin dài hạn và hiểu bối cảnh tốt, đây là mô hình được kỳ vọng đem lại độ chính xác vượt trội trong bài toán phân tích cảm xúc tiếng Việt.

```
def build_bilstm():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    x = layers.SpatialDropout1D(0.2)(x)  # dropout across the embedding sequence
    x = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(x)
    x = layers.GlobalMaxPooling1D()(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="bilstm")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

6.3.3. Mô hình 3: CNN + BiLSTM (Hybrid Model / C-RNN)

Kiến trúc tầng:

- Input → Embedding.
- CNN Layer: Sử dụng Conv1D để trích xuất các đặc trưng n-grams quan trọng, giảm chiều dài của chuỗi dữ liệu nhưng giữ lại thông tin cốt lõi.
- Bi-LSTM Layer: Nhận đầu vào là các đặc trưng đã được CNN "cô đặc", tiếp tục học mối quan hệ thời gian giữa các đặc trưng này.
- Dense Layer: Phân loại.

Mục đích dùng để giảm tải khối lượng tính toán cho lớp LSTM (do chuỗi đầu vào đã được CNN làm ngắn lại) trong khi vẫn giữ được thông tin ngữ cảnh. Tuy nhiên, mô hình này phức tạp và khó huấn luyện (khó hội tụ) hơn so với hai mô hình đơn lẻ.

```python
def build_cnn_bilstm():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    x = layers.Conv1D(128, 5, activation="relu", padding="same")(x)
    x = layers.MaxPooling1D(2)(x)
    x = layers.Bidirectional(layers.LSTM(96, return_sequences=True))(x)  # combine conv features with longer context
    x = layers.GlobalMaxPooling1D()(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="cnn_bilstm")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

Bảng 4. Các siêu tham số huấn luyện

Tham số (Parameter)	**Giá trị thiết lập (Value)**	**Giải thích (Explanation)**
Embedding Dimension	400	Kích thước vector biểu diễn từ (dựa trên Word2Vec).
Max Sequence Length	180	Độ dài tối đa của một câu bình luận.
Batch Size	256	Số lượng mẫu dữ liệu được đưa vào mạng trong một lần học.
Epochs	6 - 8	Số vòng lặp huấn luyện tối đa.
Optimizer	Adam	Thuật toán tối ưu hóa với Learning rate khởi tạo = 1e-3.
Loss Function	Categorical Cross-entropy	Hàm mất mát cho bài toán phân loại đa lớp.

6.3.4. Các kỹ thuật Tối ưu hóa nâng cao

Bước 1: Vòng lặp sàng lọc (The Condensing Loop)

Cơ chế hai chiều (Bidirectional) trong mô hình sử dụng hai lớp LSTM chạy song song theo hướng xuôi và ngược, giúp nắm bắt ngữ cảnh đầy đủ hơn. Thay vì khởi tạo trọng số ngẫu nhiên, mô hình được tích hợp trọng số Word2Vec (CBOW) đã được huấn luyện trên hàng triệu văn bản tiếng Việt từ báo chí và Wikipedia. Nhờ đó, mô hình sở hữu kiến thức ngôn ngữ nền tảng ngay từ đầu; chẳng hạn, nó đã biết rằng vector của từ “tốt” nằm gần “tuyệt_vời” trong không gian biểu diễn, giúp quá trình học nhanh hơn và hội

tụ ổn định hơn.

Bước 2: Fine-tuning Embedding Layer (trainable=True):

Mặc dù sử dụng Pre-trained, nhưng từ vựng trong miền công nghệ có ý nghĩa rất đặc thù/ Cho phép cập nhật lại lớp Embedding trong quá trình huấn luyện để các vector từ thích nghi (adapt) với ngữ cảnh của tập dữ liệu VLSP.

Bước 3: Xử lý mất cân bằng dữ liệu (Class Weighting):

Trong tập dữ liệu thực tế, phân bố nhãn bị lệch đáng kể khi các bình luận Tiêu cực và Tích cực chiếm ưu thế, trong khi lớp Trung tính xuất hiện rất ít. Sự mất cân bằng này khiến mô hình dễ có xu hướng nghiêng về các lớp đa số. Để khắc phục, chúng tôi tính toán và áp dụng trọng số lớp (class weights) trực tiếp vào hàm mất mát. Cơ chế này đảm bảo rằng khi mô hình dự đoán sai một mẫu thuộc lớp Trung tính – lớp thiểu số – mức phạt sẽ cao hơn so với dự đoán sai ở lớp Tiêu cực. Nhờ đó, mô hình buộc phải chú trọng hơn vào việc học đặc trưng của các lớp hiếm, cải thiện khả năng phân loại tổng thể.

```
# Use class weights to reduce imbalance across labels
CLASS_WEIGHTS = dict(
    enumerate(compute_class_weight(class_weight="balanced", classes=np.unique(label_ids), y=label_ids))
)
```

Bước 4: Tối ưu hóa tốc độ học (ReduceLROnPlateau):

Sử dụng cơ chế Learning Rate Decay Nếu sau một số epochs mà độ lỗi (Validation Loss) không giảm, hệ thống sẽ tự động giảm tốc độ học (Learning Rate) xuống (ví dụ: chia đôi). Điều này giúp mô hình di chuyển chậm lại để tìm được điểm cực tiểu toàn cục (Global Minima) chính xác hơn ở giai đoạn cuối của quá trình huấn luyện.

```
def run_experiment_v2(name, builder, builder_kwargs=None, use_class_weights=True):
    model = builder(**(builder_kwargs or {}))
    cbs = [
        callbacks.EarlyStopping(monitor="val_loss", patience=2, restore_best_weights=True),
        callbacks.ReduceLROnPlateau(monitor="val_loss", factor=0.5, patience=1, min_lr=1e-5),
    ]
    cw = CLASS_WEIGHTS if use_class_weights else None
    history = model.fit(
        x_train,
        y_train,
        validation_data=(x_val, y_val),
        epochs=EPOCHS + 2,
        batch_size=BATCH_SIZE,
        callbacks=cbs,
        class_weight=cw,
        verbose=2,
    )
```

7. CHIẾN LƯỢC TRỌNG SỐ LỚP VÀ TINH CHỈNH MÔ HÌNH

Kết quả phân lớp trên tập dữ liệu gốc có thể bị ảnh hưởng tiêu cực bởi sự mất cân bằng dữ liệu. Trong kỹ thuật huấn luyện mạng nơ-ron truyền thống, vai trò của mỗi mẫu dữ liệu đóng góp vào hàm lỗi (Loss function) được coi là ngang bằng nhau. Điều này bộc lộ hạn chế lớn đối với tập dữ liệu VLSP:

- Các nhãn chiếm đa số (như Tiêu cực -1) xuất hiện quá nhiều, khiến mô hình có xu hướng "học vẹt" và thiên vị dự đoán nhãn này để đạt độ chính xác ảo cao.
- Nhãn thiểu số (như Trung tính 0) xuất hiện ít, khiến mô hình không học đủ đặc trưng và thường dự đoán sai.

Giải pháp: Áp dụng kỹ thuật Cost-Sensitive Learning (Học dựa trên chi phí) thông qua việc gán trọng số cho từng lớp (Class Weighting) và Tinh chỉnh Embedding. Nguyên tắc cốt lõi là: "Lớp càng ít mẫu thì trọng số phạt càng cao, bắt buộc mô hình phải 'học kỹ' các mẫu hiếm này"

Để hiện thực hóa ý tưởng trên, sử dụng các kỹ thuật sau trong quá trình huấn luyện (phiên bản v2 trong code):

Cách 1: Cân bằng trọng số lớp (Class Weighting)

Đây là phương pháp thay đổi hàm mất mát để trừng phạt nặng hơn các sai số trên lớp thiểu số. Trọng số W_j cho lớp j được tính nghịch đảo với tần suất xuất hiện của nó.

Công thức tổng quát được sử dụng (từ thư viện sklearn trong code):

$$W_j = \frac{N}{C \times N_j}$$

Trong đó:

- N: Tổng số mẫu trong tập huấn luyện.
- C: Tổng số lớp (3 lớp: -1, 0, 1).
- N_j: Số lượng mẫu của lớp j.

Nhận xét: Khi áp dụng công thức này vào hàm mất mát Categorical Crossentropy, nếu mô hình đoán sai một mẫu thuộc nhãn hiếm (Trung tính), giá trị lỗi (Loss) sẽ bị nhân lên gấp nhiều lần so với đoán sai nhãn phổ biến. Điều này giúp mô hình đạt được chỉ số Macro F1-Score cao hơn (~0.68), cân bằng hiệu suất giữa các lớp.

Cách 2: Tinh chỉnh trọng số nhúng (Embedding Fine-tuning)

Trong các mô hình cơ bản (Baseline), lớp Embedding thường được "đóng băng" (Frozen) để giữ nguyên các giá trị từ mô hình Word2Vec gốc. Tuy nhiên, điều này có hạn chế là các từ vựng chuyên ngành công nghệ (ví dụ: "trâu" trong "pin trâu") có thể không được biểu diễn chính xác trong bộ Word2Vec phổ thông.

Giải pháp: Chuyển tham số trainable=True.

Khi đó, các vector từ v_i không còn là hằng số mà trở thành các tham số cần học:

$$v_i^{(t+1)} = v_i^{(t)} - \eta \cdot \frac{\partial L}{\partial v_i}$$

Quá trình lan truyền ngược (Backpropagation) sẽ tinh chỉnh vị trí của các từ trong không gian vector sao cho phù hợp nhất với nhiệm vụ phân tích cảm xúc cụ thể của tập dữ liệu này.

Cách 3: Điều chỉnh tốc độ học (Learning Rate Scheduling)

Để mô hình hội tụ tốt hơn tại các vùng cực tiểu toàn cục, không dùng một tốc độ học cố định. Thay vào đó, áp dụng cơ chế ReduceLROnPlateau.

Cơ chế hoạt động như hàm bậc thang:

- Theo dõi độ lỗi trên tập kiểm định (Validation Loss).
- Nếu sau một số chu kỳ (patience) mà lỗi không giảm, thực hiện giảm tốc độ học

$$LR_{new} = LR_{old} \times factor$$

Nhận xét: Phương pháp này giúp mô hình "bước chậm lại" khi đến gần đích, tránh việc bước quá đà (overshooting) và giúp đạt độ chính xác cao nhất có thể (lên

tới ~67.9% cho BiLSTM v2).

8. ĐÁNH GIÁ HIỆU QUẢ THỰC NGHIỆM VÀ SO SÁNH

Để đảm bảo tính khách quan và khoa học, quá trình đánh giá được thực hiện trên tập kiểm thử (Test Set) độc lập gồm 1.050 mẫu dữ liệu, hoàn toàn không tham gia vào quá trình huấn luyện. chia quá trình thực nghiệm thành hai giai đoạn: (1) Chạy các mô hình cơ sở (Baseline) và (2) Áp dụng các kỹ thuật tối ưu hóa (Improved v2). Các chỉ số đánh giá được sử dụng:

- Accuracy (Độ chính xác): Tỷ lệ dự đoán đúng trên tổng số mẫu.
- Precision (Độ chuẩn xác): Đo lường mức độ tin cậy của mô hình khi đưa ra dự đoán một nhãn cụ thể.
- Recall (Độ phủ / Độ nhạy): Đo lường khả năng bao quát của mô hình đối với một nhãn cụ thể.
- F1-Score (Trung bình điều hòa): Điều hòa giữa Precision và Recall, giúp cân bằng giữa độ chính xác và độ phủ.
- Confusion Matrix (Ma trận nhầm lẫn): Biểu diễn chi tiết sự tương quan giữa nhãn thực tế và nhãn dự đoán.
- Macro F1-Score: Trung bình điều hòa giữa Precision và Recall, tính bình quân cho các lớp. Chỉ số này đặc biệt quan trọng đối với tập dữ liệu VLSP do có sự mất cân bằng giữa các nhãn (Tiêu cực > Tích cực > Trung tính).

8.1. Giai đoạn 1: Kết quả các mô hình cơ sở (Baseline Models)

Bảng 5. Chi tiết diễn biến độ chính xác (Accuracy) và Hàm mất mát (Loss) qua các Epochs (Giai đoạn Baseline)

Epoch	Metric (Chỉ số)	TextCNN (Mô hình 1)	Bi-LSTM (Mô hình 2)	CNN + Bi-LSTM (Mô hình 3)
1	Train Acc Train Loss	0.4133 3.1625	0.4377 1.1032	0.3961 1.1095

	Val Acc	0.5176	0.5314	0.5431
	Val Loss	1.3604	0.9626	0.9898
2	Train Acc	0.5845	0.5547	0.5375
	Train Loss	1.4143	0.9361	0.9457
	Val Acc	0.5490	0.5706	0.5294
	Val Loss	1.2505	0.9024	0.9238
3	Train Acc	0.6784	0.6031	0.6227
	Train Loss	0.8846	0.8713	0.8476
	Val Acc	0.6000	0.6098	0.5686
	Val Loss	1.0354	0.8712	0.8779
4	Train Acc	0.7196	0.6438	0.6776
	Train Loss	0.7017	0.8215	0.7584
	Val Acc	0.5941	0.6196	0.6039
	Val Loss	0.9443	0.8459	0.8450
5	Train Acc	0.8070	0.6725	0.7183
	Train Loss	0.5026	0.7672	0.6820
	Val Acc	0.6059	0.6196	0.6373
	Val Loss	0.9385	0.8299	0.8186
6	Train Acc	0.8514	0.6780	0.7664
	Train Loss	0.3955	0.7496	0.5819
	Val Acc	0.6333	0.6373	0.6373
	Val Loss	0.9184	0.8598	0.8281

```
Epoch 1/6
18/18 - 15s - 853ms/step - accuracy: 0.4133 - loss: 3.1625 - val_accuracy: 0.5176 - val_loss: 1.3604
Epoch 2/6
18/18 - 7s - 401ms/step - accuracy: 0.5845 - loss: 1.4143 - val_accuracy: 0.5490 - val_loss: 1.2505
Epoch 3/6
18/18 - 1s - 38ms/step - accuracy: 0.6784 - loss: 0.8846 - val_accuracy: 0.6000 - val_loss: 1.0354
Epoch 4/6
18/18 - 1s - 38ms/step - accuracy: 0.7196 - loss: 0.7017 - val_accuracy: 0.5941 - val_loss: 0.9443
Epoch 5/6
18/18 - 1s - 38ms/step - accuracy: 0.8070 - loss: 0.5026 - val_accuracy: 0.6059 - val_loss: 0.9385
Epoch 6/6
18/18 - 1s - 69ms/step - accuracy: 0.8514 - loss: 0.3955 - val_accuracy: 0.6333 - val_loss: 0.9184
Epoch 1/6
18/18 - 7s - 407ms/step - accuracy: 0.4377 - loss: 1.1032 - val_accuracy: 0.5314 - val_loss: 0.9626
Epoch 2/6
18/18 - 1s - 75ms/step - accuracy: 0.5547 - loss: 0.9361 - val_accuracy: 0.5706 - val_loss: 0.9024
Epoch 3/6
18/18 - 1s - 81ms/step - accuracy: 0.6031 - loss: 0.8713 - val_accuracy: 0.6098 - val_loss: 0.8712
Epoch 4/6
18/18 - 1s - 75ms/step - accuracy: 0.6438 - loss: 0.8215 - val_accuracy: 0.6196 - val_loss: 0.8459
Epoch 5/6
18/18 - 1s - 73ms/step - accuracy: 0.6725 - loss: 0.7672 - val_accuracy: 0.6196 - val_loss: 0.8299
Epoch 6/6
18/18 - 1s - 73ms/step - accuracy: 0.6780 - loss: 0.7496 - val_accuracy: 0.6373 - val_loss: 0.8598
Epoch 1/6
...
Epoch 5/6
18/18 - 1s - 45ms/step - accuracy: 0.7183 - loss: 0.6820 - val_accuracy: 0.6373 - val_loss: 0.8186
Epoch 6/6
18/18 - 1s - 43ms/step - accuracy: 0.7664 - loss: 0.5819 - val_accuracy: 0.6373 - val_loss: 0.8281
```

Hình 4. Diễn biến độ chính xác qua các Epochs

Kết quả trong khối log của mô hình TextCNN, ta quan sát thấy tốc độ hội tụ diễn ra rất nhanh: chỉ đến Epoch 6, độ chính xác trên tập Train đã tăng mạnh lên 85.14%. Tuy nhiên, mô hình thể hiện dấu hiệu Overfitting nghiêm trọng khi Train Accuracy đạt tới 0.85 trong khi Validation Accuracy chỉ dừng ở mức 0.63, tạo ra khoảng cách gần 22%. Nguyên nhân nằm ở bản chất của CNN: mô hình có khả năng học nhanh các đặc trưng cục bộ như từ khóa, nhưng lại thiếu khả năng nắm bắt ngữ cảnh tổng thể, dẫn đến việc mô hình "học vẹt" và tổng quát hóa kém trên dữ liệu mới.

Ở mô hình Bi-LSTM, tốc độ hội tụ chậm hơn nhưng ổn định hơn. Đến Epoch 6, Train Accuracy chỉ đạt 67.8%, song ưu điểm nổi bật là độ chênh lệch giữa Train và Validation rất thấp (khoảng 4%). Điều này cho thấy mô hình thực sự học được quy luật ngôn ngữ theo trình tự và ngữ cảnh, thay vì ghi nhớ máy móc từng đặc trưng riêng lẻ. Đây cũng là mô hình có tính ổn định cao nhất trong ba mô hình

được so sánh.

Đối với mô hình kết hợp CNN + Bi-LSTM, hành vi tổng thể nằm giữa hai mô hình trên. Train Accuracy đạt 76.6% và Validation Accuracy đạt 63.7%, phản ánh sự dung hòa giữa khả năng học nhanh của CNN và tính tuần tự của Bi-LSTM. Tuy vậy, mô hình vẫn xuất hiện Overfitting nhẹ với mức chênh khoảng 13%, cho thấy khả năng tổng quát hóa chưa tối ưu hoàn toàn.

Sau khi huấn luyện xong, các mô hình được đánh giá trên tập Test độc lập. Kết quả tổng hợp như sau:

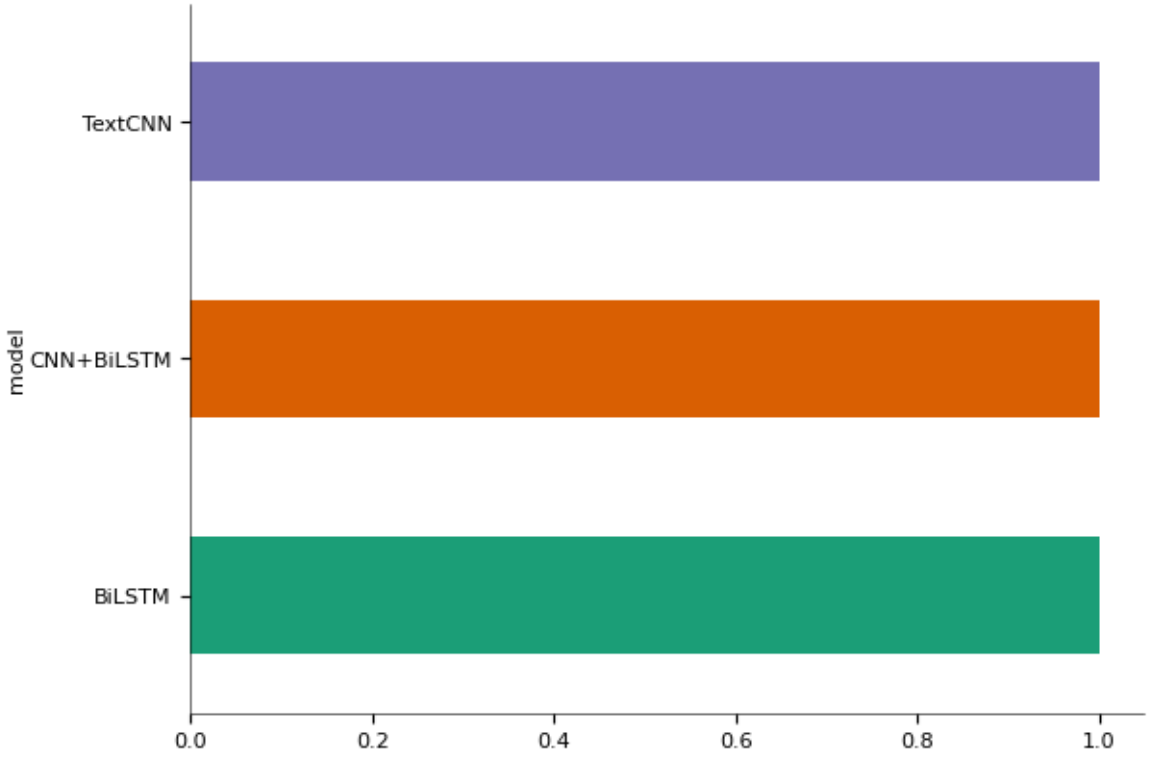
Bảng 6. Bảng Kết quả thực nghiệm giai đoạn Baseline

Hạng	**Mô hình (Model)**	**Best Val Acc**	**Test Acc**	**Số lượng tham số (Params)**	**Đánh giá hiệu quả**
1	CNN + Bi-LSTM	0.6372	0.6542	3,591,907	Hiệu quả nhất (Baseline). Đạt độ chính xác cao nhất với số lượng tham số ít nhất (nhẹ nhất).
2	Bi-LSTM	0.6372	0.6476	3,704,867	Ổn định, ít bị overfitting nhưng độ chính xác thấp hơn mô hình lai một chút.
3	TextCNN	0.6333	0.6285	3,778,339	Kém nhất trên tập Test. Mô hình nặng nhất nhưng hiệu quả thấp nhất do bị Overfitting.

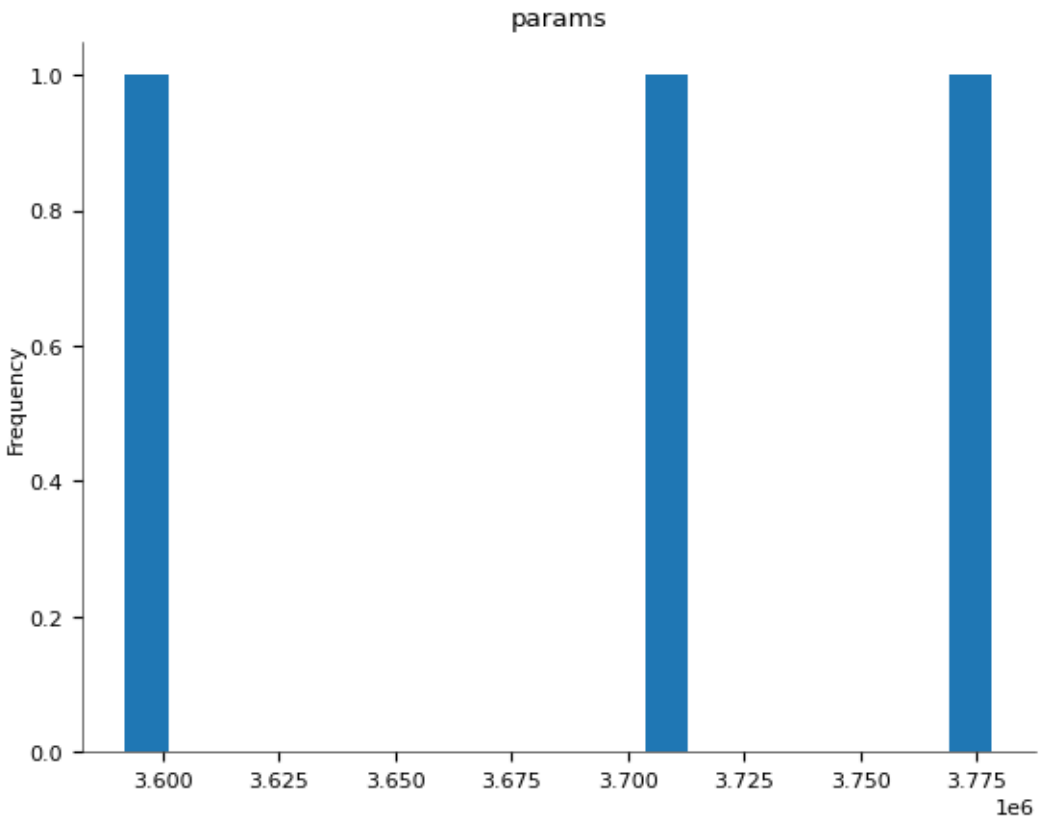
	model	best_val_acc	test_acc	params
0	TextCNN	0.633333	0.628571	3778339
1	BiLSTM	0.637255	0.647619	3704867
2	CNN+BiLSTM	0.637255	0.654286	3591907

Hình 5. Kết quả thực nghiệm giai đoạn Baseline

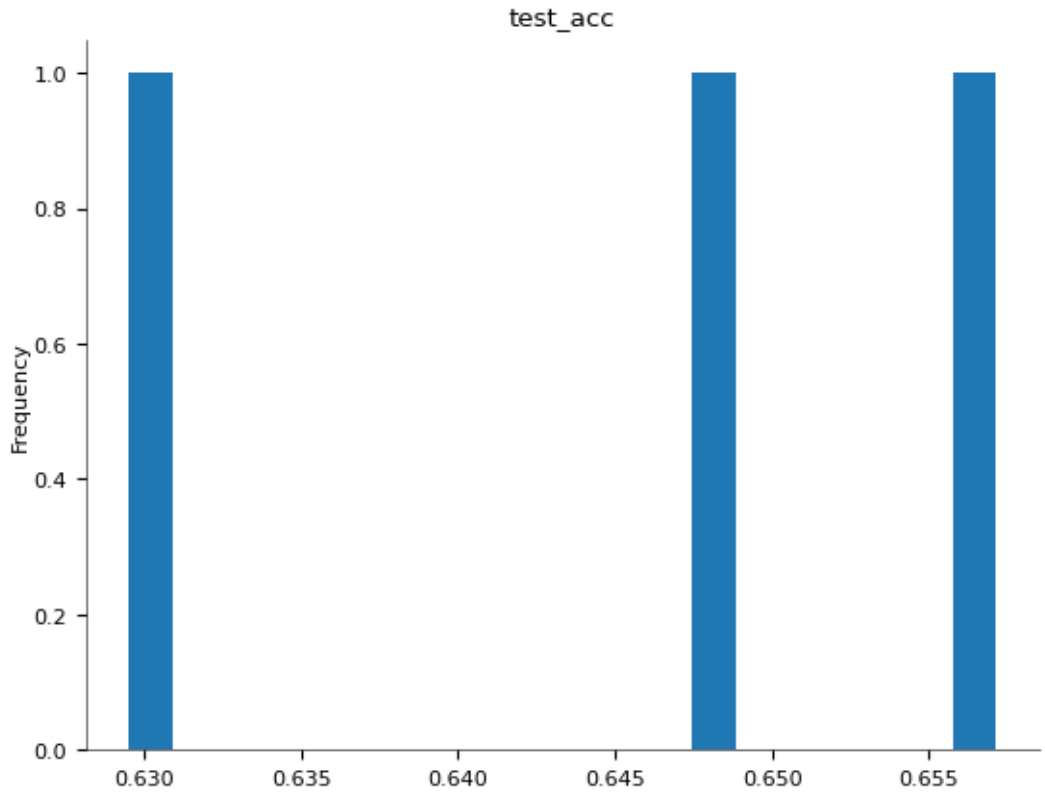
Ở giai đoạn này, mô hình lai CNN + Bi-LSTM đạt hiệu suất tốt nhất với độ chính xác xấp xỉ 65.4%. Kết quả này là hợp lý về mặt kỹ thuật, bởi kiến trúc lai cho phép lớp CNN thực hiện bước tiền xử lý quan trọng: loại bỏ nhiễu và trích xuất các đặc trưng n-gram cục bộ, trước khi các đặc trưng này được đưa vào Bi-LSTM để học quan hệ phụ thuộc theo chuỗi và ngữ cảnh dài hạn. Trong khi đó, mô hình TextCNN cho kết quả thấp nhất do hạn chế cố hữu của CNN trong việc mô hình hóa ngữ cảnh — một yếu tố đặc biệt quan trọng trong tiếng Việt, nơi các cấu trúc phủ định hoặc đa tầng nghĩa (ví dụ: “không phải là không đẹp”) đòi hỏi khả năng hiểu chuỗi và quan hệ phụ thuộc mà CNN khó có thể nắm bắt chính xác.



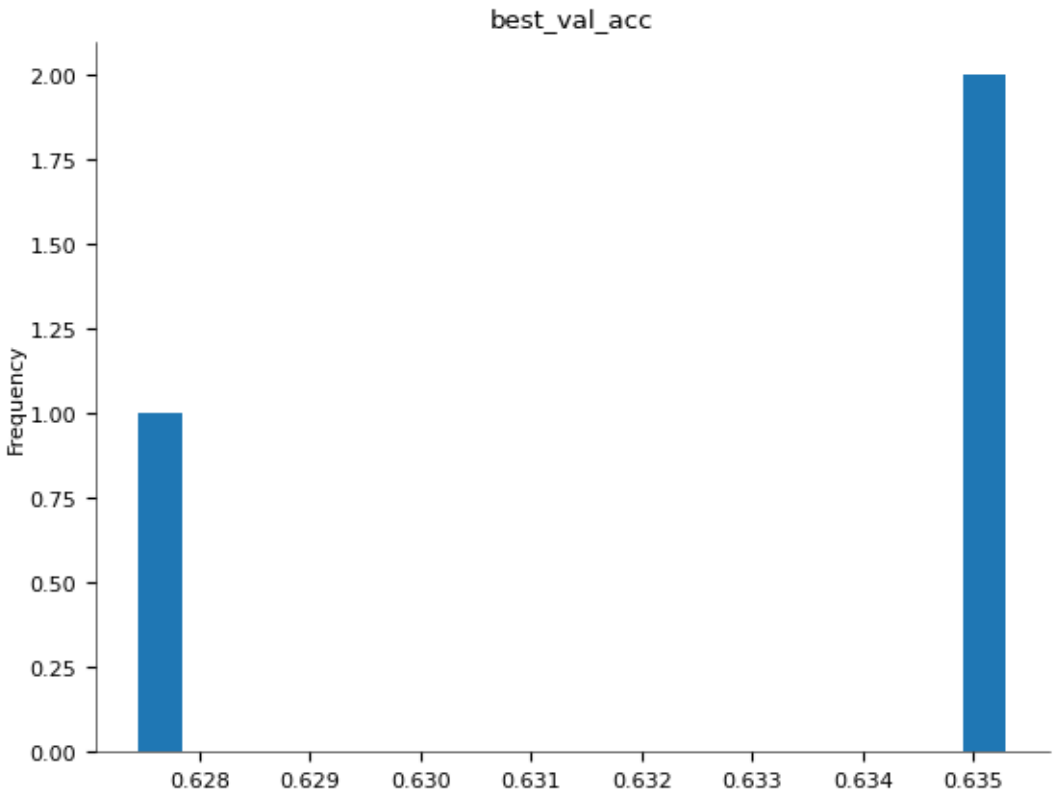
Hình 6. Các Categorical Distributions



Hình 7. Params Distributions



Hình 8. Test_acc Distributions



Hình 9. Best_val_acc Distributions

8.2. Giai đoạn 2: Kết quả sau khi tối ưu hóa (Improved v2)

Nhận thấy kết quả Baseline chưa thực sự thuyết phục, đã triển khai bộ chiến lược cải tiến (phiên bản v2) nhằm nâng cao hiệu suất mô hình:

1. Fine-tuning Embedding: Cho phép cập nhật trọng số Word2Vec (trainable=True) để thích nghi với từ vựng chuyên ngành công nghệ.
2. Class Weighting: Cân bằng sự chênh lệch mẫu giữa các lớp.
3. Learning Rate Schedule: Tự động giảm tốc độ học khi Loss không giảm.

Bảng 7. Chi tiết diễn biến độ chính xác (Accuracy) và Hàm mất mát (Loss) qua các Epochs (Giai đoạn Tối ưu hóa - v2)

Epoch	Metric (Chỉ số)	TextCNN v2 (Mô hình 1)	Bi-LSTM v2 (Mô hình 2)	CNN + Bi-LSTM v2 (Mô hình 3)
1	Train Acc	0.4170	0.4316	0.3771
	Train Loss	3.0792	1.1144	1.1553
	Val Acc	0.5118	0.5118	0.4765
	Val Loss	1.4702	0.9511	1.0410

2	Train Acc	0.6011	0.5429	0.5111
	Train Loss	1.3233	0.9443	0.9845
	Val Acc	0.5784	0.6137	0.5333
	Val Loss	1.0150	0.8813	0.9574
3	Train Acc	0.6797	0.6142	0.6004
	Train Loss	0.9046	0.8544	0.8688
	Val Acc	0.6235	0.6235	0.5686
	Val Loss	0.9260	0.8437	0.8894
4	Train Acc	0.7634	0.6484	0.6612
	Train Loss	0.6088	0.7948	0.7729
	Val Acc	0.6216	0.6216	0.6157
	Val Loss	0.9281	0.8221	0.8551
5	Train Acc	0.8089	0.6728	0.7183
	Train Loss	0.4853	0.7559	0.6896
	Val Acc	0.6196	0.6314	0.6529
	Val Loss	0.9205	0.8095	0.8083
6	Train Acc	0.8388	0.6967	0.7863
	Train Loss	0.4208	0.7128	0.5689
	Val Acc	0.6294	0.6275	0.6608 (Đỉnh)
	Val Loss	0.9092	0.8115	0.8045
7	Train Acc	0.8617	0.7153	0.8410
	Train Loss	0.3675	0.6780	0.4431
	Val Acc	0.6255	0.6549 (Đỉnh)	0.6529
	Val Loss	0.8961	0.7793	0.8747
8	Train Acc	0.8824	0.7327	0.8869
	Train Loss	0.3126	0.6451	0.3382
	Val Acc	0.6235	0.6529	0.6255
	Val Loss	0.9094	0.7862	0.9301

```
Epoch 1/8
18/18 - 24s - 1s/step - accuracy: 0.4170 - loss: 3.0793 - val_accuracy: 0.5118 - val_loss: 1.4707 - learning_rate: 1.0000e-03
Epoch 2/8
18/18 - 1s - 74ms/step - accuracy: 0.6017 - loss: 1.3229 - val_accuracy: 0.5784 - val_loss: 1.0150 - learning_rate: 1.0000e-03
Epoch 3/8
18/18 - 1s - 74ms/step - accuracy: 0.6813 - loss: 0.9045 - val_accuracy: 0.6275 - val_loss: 0.9266 - learning_rate: 1.0000e-03
Epoch 4/8
18/18 - 1s - 73ms/step - accuracy: 0.7667 - loss: 0.6081 - val_accuracy: 0.6216 - val_loss: 0.9284 - learning_rate: 1.0000e-03
Epoch 5/8
18/18 - 1s - 74ms/step - accuracy: 0.8089 - loss: 0.4856 - val_accuracy: 0.6196 - val_loss: 0.9205 - learning_rate: 5.0000e-04
Epoch 6/8
18/18 - 1s - 74ms/step - accuracy: 0.8390 - loss: 0.4203 - val_accuracy: 0.6275 - val_loss: 0.9087 - learning_rate: 5.0000e-04
Epoch 7/8
18/18 - 1s - 76ms/step - accuracy: 0.8617 - loss: 0.3668 - val_accuracy: 0.6255 - val_loss: 0.8997 - learning_rate: 5.0000e-04
Epoch 8/8
18/18 - 1s - 79ms/step - accuracy: 0.8856 - loss: 0.3115 - val_accuracy: 0.6255 - val_loss: 0.9080 - learning_rate: 5.0000e-04
Epoch 1/8
18/18 - 5s - 277ms/step - accuracy: 0.4314 - loss: 1.1144 - val_accuracy: 0.5137 - val_loss: 0.9510 - learning_rate: 1.0000e-03
Epoch 2/8
18/18 - 2s - 136ms/step - accuracy: 0.5436 - loss: 0.9444 - val_accuracy: 0.6118 - val_loss: 0.8807 - learning_rate: 1.0000e-03
Epoch 3/8
18/18 - 2s - 134ms/step - accuracy: 0.6148 - loss: 0.8544 - val_accuracy: 0.6176 - val_loss: 0.8440 - learning_rate: 1.0000e-03
Epoch 4/8
18/18 - 2s - 133ms/step - accuracy: 0.6477 - loss: 0.7950 - val_accuracy: 0.6275 - val_loss: 0.8231 - learning_rate: 1.0000e-03
Epoch 5/8
...
Epoch 7/8
18/18 - 1s - 74ms/step - accuracy: 0.8244 - loss: 0.4663 - val_accuracy: 0.6608 - val_loss: 0.8283 - learning_rate: 8.0000e-04
Epoch 8/8
18/18 - 1s - 74ms/step - accuracy: 0.8885 - loss: 0.3392 - val_accuracy: 0.6294 - val_loss: 0.8584 - learning_rate: 4.0000e-04
```

Hình 10. Diễn biến độ chính xác qua các Epochs (Giai đoạn Tối ưu hóa - v2)

Bảng 8. Bảng Kết quả thực nghiệm giai đoạn Tối ưu hóa

Hạng	Mô hình (Model)	Best Val Acc	Test Acc	Số lượng tham số (Params)	Đánh giá hiệu quả
1	Bi-LSTM v2	0.6569	0.6790	4,074,403	Hiệu quả nhất. Đạt độ chính xác và Macro F1 (0.68) cao nhất trên tập Test. Mô hình học ổn định nhất, hiểu tốt ngữ cảnh tiếng Việt.
2	CNN + Bi-LSTM v2	0.6608	0.6771	3,894,403	Tiềm năng cao. Đạt độ chính xác Validation cao nhất trong quá trình huấn luyện. Kết quả Test rất sát sao với Bi-LSTM nhưng nhẹ hơn một chút.

3	TextCNN v2	0.6275	0.6667	3,932,323	Cải thiện đáng kể. Tăng gần 4% so với Baseline nhờ Fine-tuning. Tuy nhiên vẫn thấp nhất trong v2 và có xu hướng Overfitting nặng.
4	CNN + Bi-LSTM	0.6372	0.6542	3,591,907	Giữ nguyên từ Baseline
5	Bi-LSTM	0.6372	0.6476	3,704,867	Giữ nguyên từ Baseline
6	TextCNN	0.6333	0.6285	3,778,339	Giữ nguyên từ Baseline

	model	best_val_acc	test_acc	params	test_macro_f1
0	TextCNN	0.627451	0.626667	3778339	NaN
1	BiLSTM	0.649020	0.647619	3704867	NaN
2	CNN+BiLSTM	0.643137	0.655238	3591907	NaN
3	TextCNN v2	0.627451	0.666667	3932323	0.667421
4	BiLSTM v2	0.656863	0.679048	4074403	0.680003
5	CNN+BiLSTM v2	0.660784	0.677143	3894403	0.675205

Hình 11. Kết quả thực nghiệm (Giai đoạn Tối ưu hóa - v2)

Sau khi áp dụng các kỹ thuật tối ưu hóa, bao gồm fine-tuning embedding (cho phép cập nhật trọng số Word2Vec), class weighting nhằm cân bằng phân bố mẫu giữa các lớp, và learning rate scheduler để tự động giảm tốc độ học khi hàm mất mát có dấu hiệu chững lại, đã thu được bộ kết quả huấn luyện qua 8 Epochs với nhiều điểm đáng chú ý.

Đối với TextCNN v2, mô hình đạt độ chính xác rất cao trên tập huấn luyện (88.24%) nhưng lại chỉ đạt mức trung bình trên tập kiểm định (~62.9%). Mô hình hội tụ rất nhanh, thể hiện qua việc Train Loss giảm mạnh từ 3.07 xuống còn 0.31 chỉ trong vài Epoch đầu. Tuy nhiên, khoảng cách lớn giữa Train và Validation Accuracy—lên đến gần 26%—cho thấy dấu hiệu Overfitting đáng kể. Điều này phản ánh việc TextCNN chủ yếu học thuộc các từ khóa bề mặt thay vì nắm bắt ngữ nghĩa chung. Mặc dù cơ chế giảm Learning Rate ở Epoch 5 giúp ổn định Loss, hiệu suất trên tập kiểm định vẫn không được cải thiện thêm.

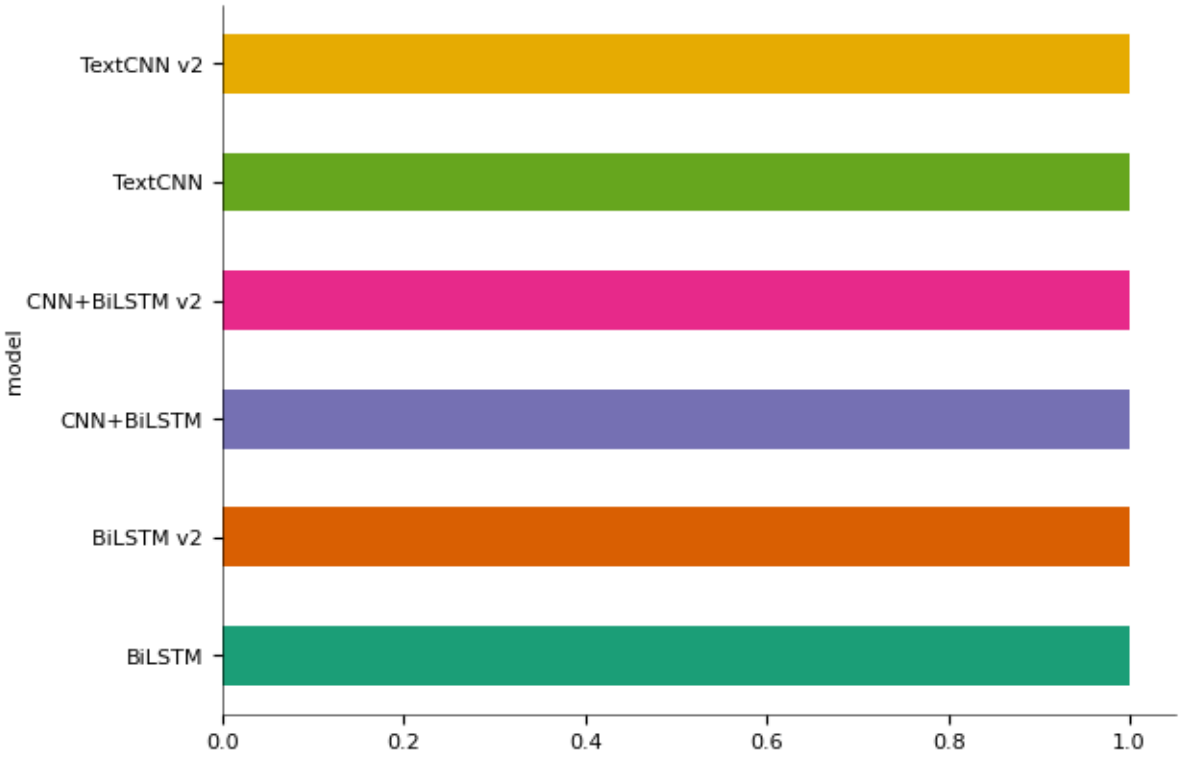
Trong khi đó, Bi-LSTM v2 đạt độ chính xác trên tập Train ở mức 73.27% và đạt điểm cao nhất trên tập Validation tại Epoch 7 với 65.49%. Tốc độ học của mô hình này chậm hơn CNN, nhưng đường cong Loss và Accuracy lại rất ổn định. Khoảng cách giữa Train và Validation chỉ khoảng 8%, thể hiện khả năng tổng quát hóa vượt trội. Điều này cho thấy Bi-LSTM thực sự học được các quy luật ngôn ngữ và ngữ cảnh trong tiếng Việt, phù hợp với đặc thù của bài toán phân tích cảm xúc.

Đối với mô hình lai CNN + Bi-LSTM v2, mô hình đạt điểm Validation cao nhất ở Epoch 6 (66.08%), nhưng hiệu suất này không duy trì được và giảm xuống 62.55% vào Epoch cuối. Mặc dù độ chính xác Train lên đến 88.69%, Validation Loss lại tăng trở lại ở các Epoch sau, cho thấy mô hình bắt đầu bị Overfitting. Điều này phản ánh tính chất "mạnh nhưng không ổn định" của mô hình lai: kết hợp được ưu điểm của cả CNN và Bi-LSTM nhưng cũng dễ bị quá khớp khi độ phức tạp mô hình tăng cao.

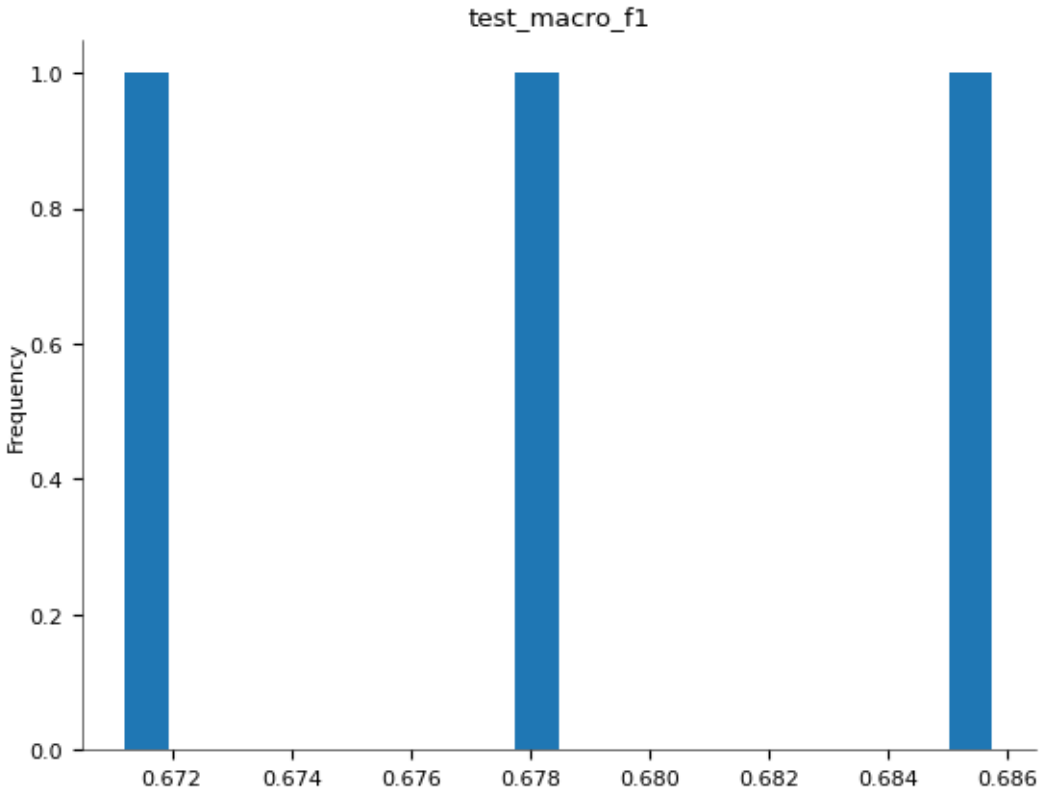
So sánh tổng hợp cho thấy TextCNN v2 và CNN+BiLSTM v2 có độ chính xác Train rất cao nhưng lại thiếu ổn định, với khoảng cách Train–Validation lần lượt khoảng 25% và 26%. Trong khi đó, Bi-LSTM v2 đạt độ ổn định tốt nhất và có mức chênh lệch nhỏ nhất, phản ánh rủi ro Overfitting thấp hơn đáng kể. Xét về hiệu suất đỉnh, CNN+BiLSTM có Validation Accuracy cao nhất (66.08%), nhưng Bi-LSTM đạt độ ổn định và khả năng tổng quát hóa tốt hơn.

Từ các biểu đồ log và bảng thống kê, nhận thấy learning rate scheduler đã phát huy hiệu quả quan trọng. Khi Validation Loss bắt đầu chững lại ở các Epoch 5, 6 và 8, cơ chế ReduceLROnPlateau đã tự động giảm tốc độ học (từ 1e-3 xuống 5e-4 và 4e-4), giúp mô hình hội tụ ổn định hơn và tránh việc rơi vào cực trị cục bộ. Khi cân nhắc bài toán tối ưu, nếu ưu tiên tốc độ huấn luyện và khả năng bắt đặc trưng n-gram nhanh, TextCNN hoặc mô hình lai CNN+BiLSTM có thể là lựa chọn phù hợp nhưng cần đi kèm các biện pháp Regularization mạnh hơn. Ngược lại, nếu mục tiêu là độ tin cậy và hiệu suất ổn định trên dữ liệu thực tế, Bi-LSTM chứng tỏ là mô hình hiệu quả nhất.

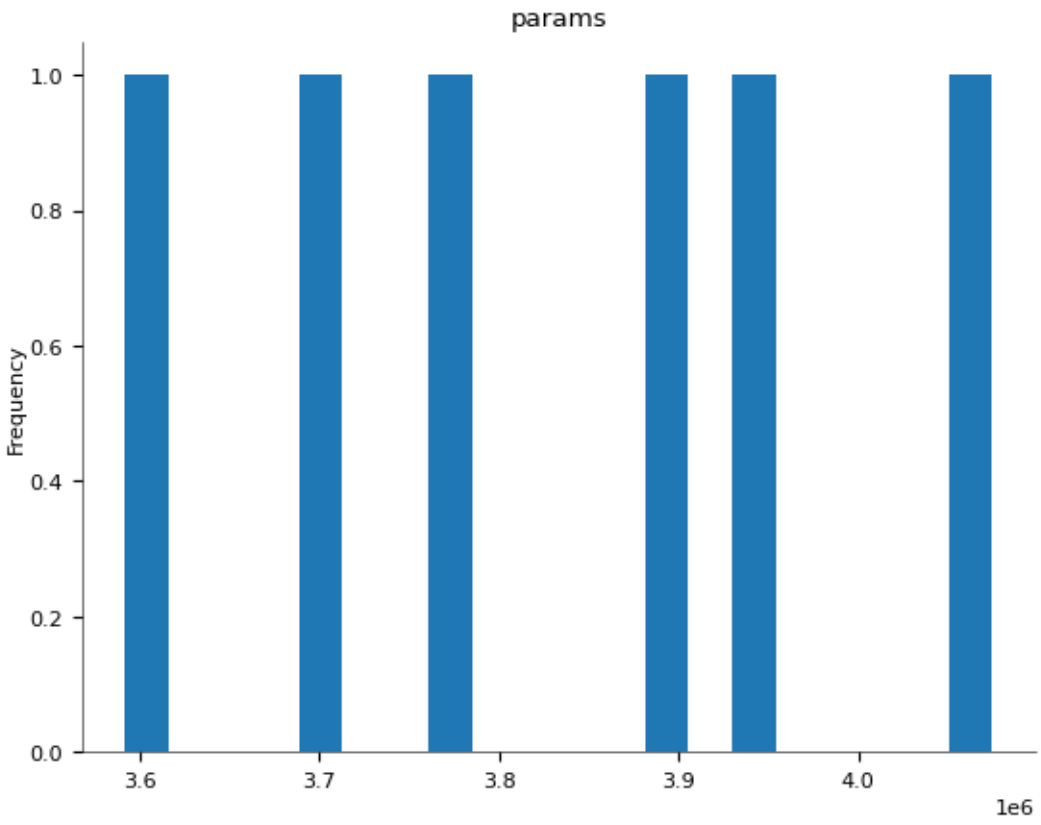
Chọn Bi-LSTM v2 làm mô hình đề xuất chính thức cho bài toán, nhờ khả năng cân bằng hài hòa giữa độ chính xác và mức độ tổng quát hóa trên dữ liệu tiếng Việt, đồng thời hạn chế tốt hiện tượng Overfitting.



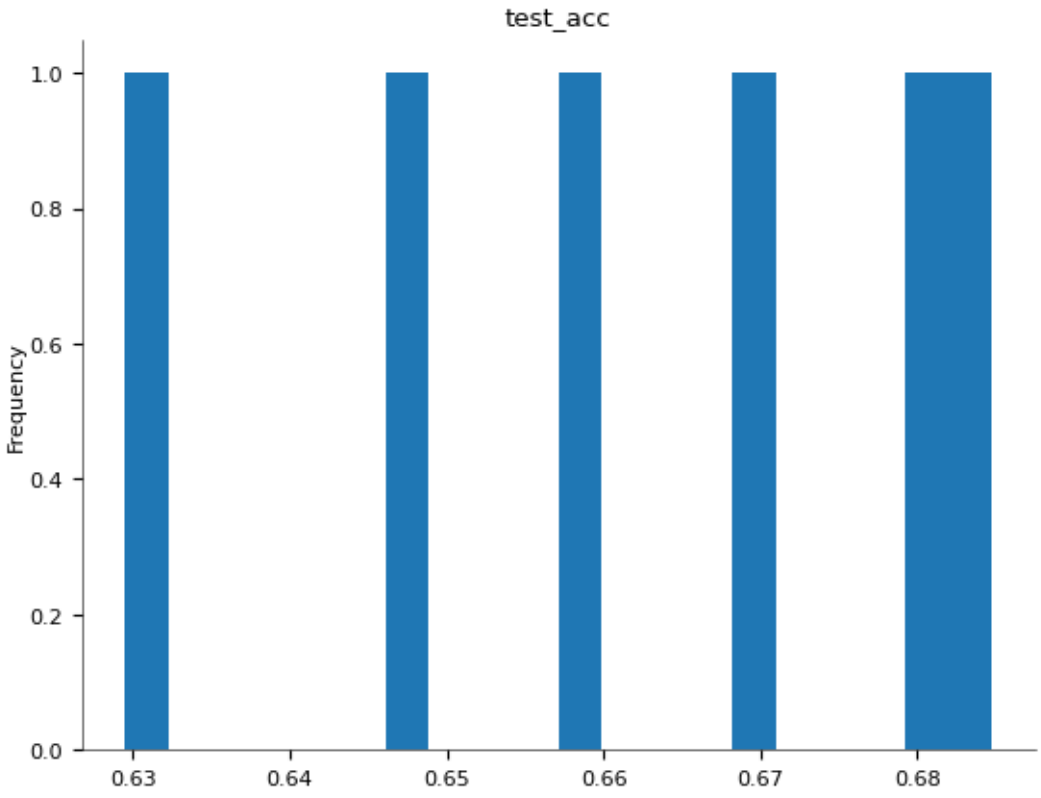
Hình 12. Các Categorical Distributions giai đoạn tối ưu



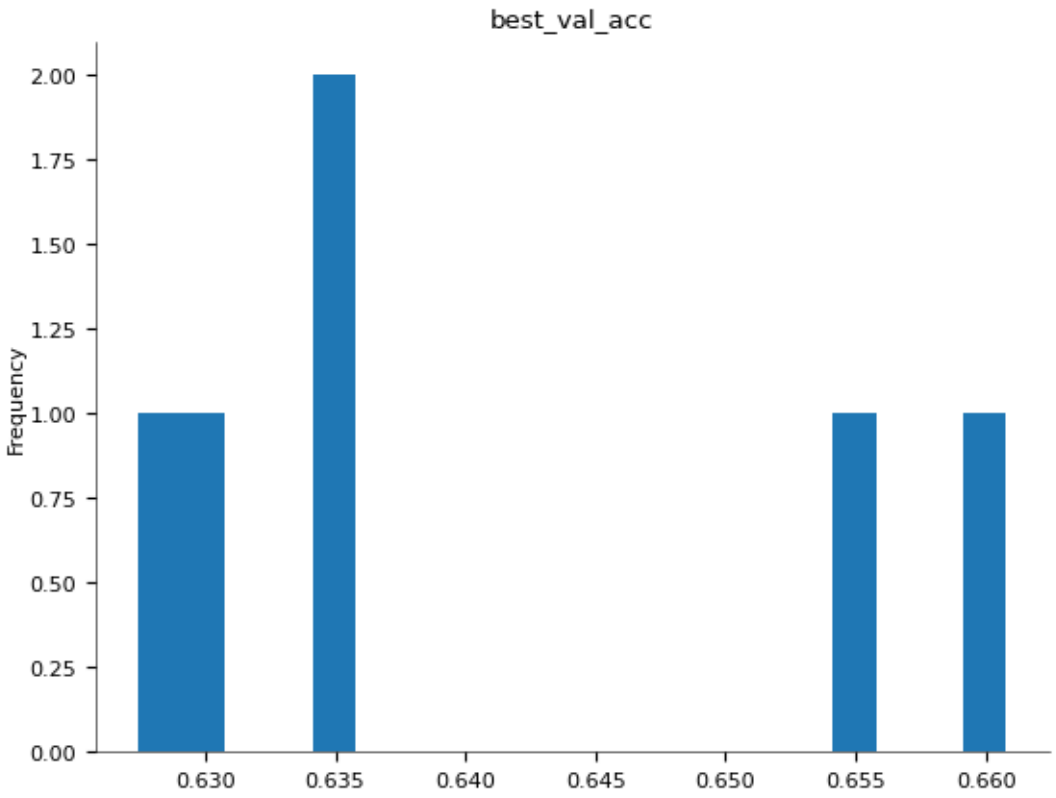
Hình 13. Test_macro_f1 Distributions



Hình 14. Params Distributions



Hình 15. Test_acc Distributions



Hình 16. Best_vall_acc Distributions

9. KẾT LUẬN

Trong dự án này, tiến hành nghiên cứu và giải quyết bài toán phân tích cảm xúc (Sentiment Analysis) trên tập dữ liệu tiếng Việt VLSP, tập trung vào các bình luận liên quan đến sản phẩm công nghệ. Bài toán đặt ra nhiều thách thức do đặc thù của ngôn ngữ tiếng Việt, bao gồm văn bản chứa nhiều nhiễu như teencode, lỗi chính tả, cấu trúc câu không tuân thủ ngữ pháp chuẩn và đặc biệt là sự mất cân bằng nghiêm trọng giữa các nhãn, trong đó bình luận mang sắc thái Tiêu cực chiếm đa số. Để giải quyết những vấn đề này, đã thiết kế một quy trình xử lý ngôn ngữ tự nhiên hoàn chỉnh, từ tiền xử lý (sử dụng pyvi để tách từ), biểu diễn từ bằng Word2Vec, đến việc triển khai và so sánh ba mô hình học sâu gồm TextCNN, Bi-LSTM và mô hình lai CNN–BiLSTM.

Qua hai giai đoạn thực nghiệm gồm Baseline và phiên bản Tối ưu hóa v2, đã rút ra một số kết luận quan trọng. Trước hết, xét về hiệu suất mô hình, Bi-LSTM là kiến trúc đạt kết quả cao nhất sau tối ưu hóa, với Accuracy khoảng 67.9% và Macro F1 đạt 0.68 trên tập kiểm thử. Điều này cho thấy đối với tiếng Việt, khả năng nắm bắt ngữ cảnh hai chiều và phụ thuộc dài hạn đóng vai trò quyết định, vượt trội hơn phương pháp trích xuất đặc trưng cục bộ như CNN. Mô hình này cũng thể hiện mức độ ổn định cao nhất, với chênh lệch Train–Validation chỉ khoảng 8%.

Ngược lại, TextCNN mặc dù có tốc độ huấn luyện nhanh và đạt Accuracy trên tập Train lên đến 88%, nhưng lại bị Overfitting nghiêm trọng, chỉ đạt khoảng 62.9% trên tập kiểm định. Mô hình này có xu hướng "học vẹt" các mẫu n-gram phổ biến thay vì hiểu ngữ nghĩa tổng quát. Mô hình lai CNN + Bi-LSTM đạt kết quả tốt nhất ở giai đoạn Baseline (~65.4%) nhờ kết hợp ưu

thế của cả hai kiến trúc, nhưng trong giai đoạn tối ưu hóa, do độ phức tạp tăng lên, mô hình dễ bị bão hòa và khó hội tụ so với Bi-LSTM thuần.

Về mặt chiến lược tối ưu hóa, phiên bản v2 mang lại cải thiện rõ rệt, giúp tăng độ chính xác trung bình từ 3–4% so với Baseline. Trong đó, fine-tuning embedding đóng vai trò quan trọng khi giúp các vector từ Word2Vec thích nghi tốt hơn với ngôn ngữ công nghệ (ví dụ: "mượt", "trâu", "lag"). Kỹ thuật class weighting cũng góp phần nâng cao Macro F1 bằng cách giảm thiên vị đối với lớp Tiêu cực và tăng mức phạt cho việc dự đoán sai lớp Trung tính. Ngoài ra, learning rate scheduler giúp mô hình đạt được điểm cực tiểu tốt hơn ở các Epoch cuối, giảm hiện tượng dao động của Loss.

Dựa trên sự cân bằng giữa độ chính xác, độ ổn định và khả năng tổng quát hóa, kết luận rằng mô hình Bi-LSTM v2, kết hợp với fine-tuning embedding và class weighting, là lựa chọn tối ưu nhất cho bài toán phân tích cảm xúc tiếng Việt trên tập dữ liệu VLSP. Mô hình không đạt mức Accuracy Train quá cao như TextCNN hay mô hình lai, nhưng lại cho hiệu suất ổn định và phù hợp hơn khi triển khai thực tế trên dữ liệu mới.

10. MÃ NGUỒN

10.1. Giải thuật TextCNN, Bi-LSTM và mô hình lai CNN–BiLSTM được triển khai trên môi trường Google Colab.

Link mã nguồn: Tham khảo tại đây

Một số thư viện cần thiết được dùng trong bài gồm:

- Thư viện tensorflow (và keras): Framework nòng cốt để xây dựng kiến trúc mạng (Layers), thiết lập huấn luyện (Optimizers, Callbacks) và đánh giá các mô hình Deep Learning (TextCNN, BiLSTM).

- Thư viện pyvi (Python Vietnamese Core): Thư viện NLP chuyên dụng cho tiếng Việt, đóng vai trò tách từ (Tokenizer) để gộp các từ ghép chính xác.

- Thư viện gensim: Dùng để tải và truy xuất các vector từ trong mô hình ngôn ngữ Word2Vec đã huấn luyện sẵn (vi-model-CBOW.bin).

- Thư viện pandas: Dùng để đọc và xử lý dữ liệu dạng bảng (DataFrame) từ các file CSV/TSV đầu vào.

- Thư viện numpy: Dùng để xử lý mảng đa chiều và các phép tính số học trên ma trận.

- Thư viện sklearn: Hỗ trợ chia tập dữ liệu (Train/Val split) và tính toán các độ đo Cài

đặt các thư viện cần thiết đánh giá như F1-Score, Class Weights.

- Thư viện re (Regex): Dùng biểu thức chính quy để làm sạch văn bản (xóa số, ký tự lạ) trong khâu tiền xử lý.

- Thư viện hệ thống (os, pathlib, random): Dùng để quản lý đường dẫn file trên Google Drive và cố định hạt giống ngẫu nhiên (Seeding) để đảm bảo tính tái lập của kết quả.

```
!pip install -q pyvi gensim scikit-learn tensorflow
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.5/8.5 MB 70.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 69.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 48.6 MB/s eta 0:00:00
```

```
import os

import random
import re
from pathlib import Path

import numpy as np
import pandas as pd
import tensorflow as tf
from gensim.models.keyedvectors import KeyedVectors
from pyvi import ViTokenizer
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, models, callbacks, optimizers
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
```

10.2. Tải bộ dữ liệu

Thực hiện bước này nhằm tải bộ dữ liệu VLSP Sentiment Dataset (bộ dữ liệu chuẩn cho bài toán phân tích cảm xúc tiếng Việt) từ thư mục lưu trữ (Google Drive) vào môi trường làm việc. Đây là bước khởi đầu quan trọng để cung cấp dữ liệu văn bản đầu vào cho quá trình tiền xử lý và huấn luyện mô hình.

Kết nối và định nghĩa đường dẫn dữ liệu: Đầu tiên, hệ thống thực hiện kết nối với Google Drive thông qua lệnh drive.mount('/content/drive') nhằm truy cập kho dữ liệu lưu trữ trực tuyến. Tiếp theo, thư viện pathlib được sử dụng để định nghĩa biến BASE_DIR, trỏ đến thư mục gốc của dự án. Từ đó, các đường dẫn con TRAIN_PATH và TEST_PATH được thiết lập để liên kết trực tiếp đến hai tệp dữ

liệu vlsp_sentiment_train.csv và vlsp_sentiment_test.csv. Cách tổ chức này giúp mã nguồn trở nên gọn gàng, linh hoạt và dễ dàng điều chỉnh khi cấu trúc thư mục thay đổi.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Định nghĩa tên cột (Column definition):File dữ liệu gốc được lưu trữ dưới dạng bảng nhưng cần được gán nhãn cột rõ ràng để thuận tiện cho việc truy xuất. Danh sách tên cột được định nghĩa lại bao gồm 2 thuộc tính chính:

- Class: Nhãn cảm xúc của bình luận (Biến mục tiêu - Target). Nhãn này nhận 3 giá trị số nguyên:
  - -1: Cảm xúc Tiêu cực (Negative).
  - 0: Cảm xúc Trung tính (Neutral).
  - 1: Cảm xúc Tích cực (Positive).
- Data: Nội dung văn bản bình luận của người dùng về các sản phẩm công nghệ (Biến đặc trưng - Feature).

```
# Duong dan dat theo cau truc tren Google Drive/Colab
BASE_DIR = Path('/content/drive/MyDrive/NLP_Lab/')
TRAIN_PATH = BASE_DIR / 'Lab5/vlsp_sentiment_train.csv'
TEST_PATH = BASE_DIR / 'Lab5/vlsp_sentiment_test.csv'
W2V_PATH = BASE_DIR / 'Lab5/vi-model-CBOW.bin'
```

Đọc dữ liệu vào DataFrame (pd.read_csv):

- Hàm pd.read_csv của thư viện Pandas được sử dụng để đọc dữ liệu từ các đường dẫn TRAIN_PATH và TEST_PATH.
- Tham số sep='\t': Đây là thông số quan trọng nhất. Dữ liệu VLSP trong file PDF thực chất là định dạng TSV (Tab-Separated Values - phân tách bằng dấu Tab) dù đuôi file là .csv. Việc khai báo sep='\t' giúp Pandas tách cột chính xác, tránh việc nội dung bình luận bị cắt lỗi.

- Gán tên cột: Sau khi đọc, các cột được gán lại tên thông qua lệnh train_df.columns = ['Class', 'Data'] để chuẩn hóa cấu trúc dữ liệu trước khi đưa vào quy trình tiền xử lý.

```
# Doc du lieu TSV tu bo du lieu VLSP
train_df = pd.read_csv(TRAIN_PATH, sep='\t')
train_df.columns = ['Class', 'Data']
test_df = pd.read_csv(TEST_PATH, sep='\t')
test_df.columns = ['Class', 'Data']
```

10.3. Thiết Lập Cấu Hình và Siêu Tham Số

10.3.1. Cố định tính ngẫu nhiên

Trong các mô hình Deep Learning, có nhiều thành phần mang tính ngẫu nhiên ảnh hưởng trực tiếp đến kết quả huấn luyện, chẳng hạn như việc khởi tạo trọng số ban đầu của các lớp mạng (Dense, LSTM, Conv1D), quá trình chia tách dữ liệu thành tập huấn luyện và kiểm thử, cũng như thao tác xáo trộn dữ liệu trong suốt quá trình training. Nếu không cố định các yếu tố này, mỗi lần chạy lại mô hình sẽ cho ra kết quả khác nhau, khiến độ chính xác thay đổi theo hướng khó kiểm soát. Điều này gây khó khăn trong việc đánh giá liệu sự cải thiện hiệu suất có thực sự đến từ mô hình hay chỉ đơn thuần là do yếu tố may mắn.

- random.seed(42): Cố định ngẫu nhiên cho các thư viện chuẩn của Python.
- np.random.seed(42): Cố định ngẫu nhiên cho thư viện NumPy (quan trọng cho việc chia tách dữ liệu).
- tf.random.set_seed(42): Cố định việc khởi tạo trọng số trong TensorFlow/Keras.

```
# Co dinh random seed de lap lai ket qua
tf.random.set_seed(42)
np.random.seed(42)
random.seed(42)
```

10.3.2. Thiết lập Siêu tham số

Tham số EMBEDDING_DIM = 400 xác định số chiều của vector biểu diễn từ, đảm bảo tương thích với mô hình Word2Vec tiền huấn luyện (vi-model-CBOW.bin). Kích thước 400 giúp mô hình học được nhiều sắc thái ngữ nghĩa hơn của tiếng Việt.

Tham số MAX_VOCAB_SIZE = 12000 giới hạn số lượng từ vựng để loại bỏ các từ hiếm, từ sai chính tả và giảm chi phí bộ nhớ. Những từ ngoài phạm vi này được thay thế bằng token <UNK>.

Tham số MAX_SEQUENCE_LENGTH = 180 quy định độ dài cố định của các chuỗi đầu vào. Các câu ngắn sẽ được padding, trong khi câu dài hơn sẽ bị cắt bớt. Con số này được chọn dựa trên phân bố độ dài trung bình của bình luận.

Tham số VAL_SIZE = 0.1 tách 10% dữ liệu huấn luyện làm tập kiểm định nhằm đánh giá mô hình theo từng epoch và phát hiện sớm hiện tượng overfitting.

Tham số BATCH_SIZE = 256 xác định số lượng mẫu xử lý trong mỗi lần cập nhật trọng số, giúp tăng tốc độ huấn luyện và làm cho hàm mất mát ổn định hơn.

Cuối cùng, EPOCHS = 6 quy định số vòng lặp huấn luyện. Con số nhỏ giúp hạn chế overfitting, đặc biệt với mô hình TextCNN; đồng thời cơ chế EarlyStopping cho phép dừng sớm nếu mô hình không còn cải thiện.

```
# Sieu tham so chung cho ba mo hinh
EMBEDDING_DIM = 400
MAX_VOCAB_SIZE = 12000
MAX_SEQUENCE_LENGTH = 180
VAL_SIZE = 0.1
BATCH_SIZE = 256
EPOCHS = 6
```

10.4. Tiền Xử Lý Dữ Liệu

10.4.1. Làm sạch và Chuẩn hóa văn bản

- re.sub(r"\d+", " ", str(text)): Trong phân tích cảm xúc, các con số (ví dụ: "giá 5 triệu", "ngày 20/10") thường không mang sắc thái tích cực hay tiêu cực. Việc giữ lại chúng làm tăng kích thước từ điển vô ích. Dòng code này dùng Regex để thay thế toàn bộ chuỗi số bằng khoảng trắng.
- .strip().lower():
    - lower(): Chuyển toàn bộ về chữ thường. Máy tính phân biệt "Đẹp" và "đẹp" là 2 từ khác nhau. Việc chuyển về lower giúp đồng nhất dữ liệu.

- re.sub(r"\s+", " ", ...): Loại bỏ các khoảng trắng thừa (ví dụ do lỗi đánh máy "rất tốt"), chỉ giữ lại 1 dấu cách duy nhất giữa các từ để tránh làm nhiễu bộ tách từ.

```python
def clean_text(text: str) -> str:
    # Loai bo chu so/ki tu thua va dua ve lowercase de giam do nhieu
    text = re.sub(r"\d+", " ", str(text))
    text = re.sub(r"\s+", " ", text).strip().lower()
    return text
```

10.4.2. Tách từ tiếng Việt

- Tiếng Anh dùng dấu cách để phân tách từ (VD: "I love you"). Nhưng tiếng Việt lại có từ ghép (VD: "đồng hồ", "học sinh"). Nếu chỉ tách bằng dấu cách, máy sẽ hiểu "đồng hồ" là hai từ "đồng" (kim loại) và "hồ" (nước), làm sai lệch hoàn toàn ngữ nghĩa.
- Thư viện pyvi sẽ tự động nhận diện từ ghép và nối chúng lại bằng dấu gạch dưới _.
  - Input: "chiếc điện thoại này rất tốt"
  - Output: "chiếc điện_thoại này rất tốt"
  - Điều này giúp mô hình Word2Vec học được vector của cả cụm từ "điện_thoại" thay vì các từ đơn lẻ.

```python
def tokenize_vi(text: str) -> str:
    # Tach cum tu tieng Viet bang pyvi (giu cac tu ghep)
    return ViTokenizer.tokenize(text)
```

10.4.3. Mã hóa nhãn

- Input: Nhãn gốc từ file CSV là các số nguyên: -1 (Tiêu cực), 0 (Trung tính), 1 (Tích cực).
- Mapping: Mạng nơ-ron thường yêu cầu nhãn bắt đầu từ 0. Code thực hiện ánh xạ:
  - -1 → 0
  - 0 → 1
  - 1 → 2
- to_categorical: Chuyển đổi sang dạng One-hot Encoding. Đây là định dạng bắt buộc cho hàm mất mát categorical_crossentropy.

- Nhãn 0 (Tiêu cực) → [1, 0, 0]
- Nhãn 1 (Trung tính) → [0, 1, 0]
- Nhãn 2 (Tích cực) → [0, 0, 1]

```
def encode_labels(labels: np.ndarray) -> np.ndarray:
    # Chuyen nhan {-1, 0, 1} ve one-hot 3 lop
    label_map = {-1: 0, 0: 1, 1: 2}
    encoded = np.array([label_map[int(l)] for l in labels])
    return to_categorical(encoded, num_classes=3)
```

10.4.4. Quy trình xử lý tổng hợp

```
def load_and_prepare(df: pd.DataFrame):
    # Quy trinh: lam sach -> tokenize -> tach tokens va nhan one-hot
    cleaned = df['Data'].astype(str).apply(clean_text).apply(tokenize_vi)
    tokens = [text.split() for text in cleaned]
    labels = encode_labels(df['Class'].values)
    return tokens, labels


def pad(tokenizer: Tokenizer, sequences, maxlen: int):
    # Chuyen tu sang id va padding ve chieu dai co dinh
    ids = tokenizer.texts_to_sequences(sequences)
    return pad_sequences(ids, maxlen=maxlen)


def build_embedding_matrix(word_index, max_vocab=MAX_VOCAB_SIZE):
    # Nap vector tu vi-model-CBOW.bin; neu thieu file thi tao ngau nhien
    vocab_size = min(len(word_index) + 1, max_vocab)
    matrix = np.zeros((vocab_size, EMBEDDING_DIM), dtype=np.float32)
    try:
        w2v = KeyedVectors.load_word2vec_format(str(W2V_PATH), binary=True)
    except FileNotFoundError:
        print(f"[!] Khong tim thay file embedding tai {W2V_PATH}")
        return matrix, vocab_size

    for word, idx in word_index.items():
        if idx >= max_vocab:
            continue
        if word in w2v:
            matrix[idx] = w2v[word]
        else:
            matrix[idx] = np.random.normal(0, np.sqrt(0.25), EMBEDDING_DIM)
    del w2v
    return matrix, vocab_size
```

Làm sạch và Chuẩn hóa bằng cách sử dụng Hàm load_and_prepare gọi các hàm con để loại bỏ ký tự nhiễu, chuyển văn bản về chữ thường và tách từ tiếng Việt (sử dụng pyvi). Nhãn cảm xúc cũng được chuyển sang dạng One-hot vector tại đây.

Mã hóa và Padding bằng Hàm pad sử dụng Tokenizer để biến đổi mỗi từ thành một con số (Index) duy nhất. Sau đó, các câu được chuẩn hóa về cùng độ dài 180 từ (pad_sequences): câu ngắn được thêm số 0, câu dài bị cắt bớt.

Khởi tạo Embedding dùng Hàm build_embedding_matrix xây dựng ma trận trọng số cho lớp đầu vào. Thay vì học từ đầu, mô hình kế thừa tri thức từ bộ Word2Vec tiếng Việt (400 chiều) đã huấn luyện sẵn. Các từ không có trong từ điển sẽ được khởi tạo ngẫu nhiên.

10.5. Chuyển đổi dữ liệu

```
# Doc du lieu TSV tu bo du lieu VLSP
train_df = pd.read_csv(TRAIN_PATH, sep='\t')
train_df.columns = ['Class', 'Data']
test_df = pd.read_csv(TEST_PATH, sep='\t')
test_df.columns = ['Class', 'Data']

# Tien xu ly text va tach tokens/labels
train_tokens, y_full = load_and_prepare(train_df)
test_tokens, y_test = load_and_prepare(test_df)

# Fit tokenizer tren toan bo train de giu vocab on dinh
tokenizer = Tokenizer(num_words=MAX_VOCAB_SIZE, lower=False, oov_token="<unk>")
tokenizer.fit_on_texts(train_tokens)

# Convert sang padding va tach tap train/val co stratify
x_full = pad(tokenizer, train_tokens, MAX_SEQUENCE_LENGTH)
x_test = pad(tokenizer, test_tokens, MAX_SEQUENCE_LENGTH)
label_ids = np.argmax(y_full, axis=1)

x_train, x_val, y_train, y_val = train_test_split(
    x_full, y_full, test_size=VAL_SIZE, random_state=42, stratify=label_ids
)

# Tao embedding matrix tu tu dien co san
embedding_matrix, VOCAB_SIZE = build_embedding_matrix(tokenizer.word_index, MAX_VOCAB_SIZE)
print(f"Train: {x_train.shape}, Val: {x_val.shape}, Test: {x_test.shape}")
```

Nạp và làm sạch dữ liệu:Dữ liệu được đọc từ file .csv nhưng thực chất có cấu trúc TSV với ký tự tab (\t) làm dấu phân tách. Văn bản sau đó được đưa qua pipeline load_and_prepare để loại bỏ nhiễu, chuẩn hóa và tách từ tiếng Việt (ví dụ: "điện thoại" → "điện_thoại"). Đồng thời, các nhãn cảm xúc cũng được chuyển đổi sang dạng one-hot vector để phù hợp với mô hình học sâu.

Sử dụng Tokenizer để ánh xạ từng từ thành chỉ số nguyên. Tokenizer chỉ được huấn luyện trên tập Train, vì vậy những từ chưa xuất hiện trong tập này nhưng lại xuất hiện ở tập Test sẽ được thay thế bằng token <unk>. Các câu được chuẩn hóa về độ dài cố định 180 từ thông qua pad_sequences, trong đó câu ngắn được thêm số 0 và câu dài sẽ bị cắt bớt.

Tập dữ liệu huấn luyện được chia thành hai phần: 90% dùng để Train và 10% dành cho Validation. Kỹ thuật stratify được áp dụng nhằm duy trì tỷ lệ phân bố nhãn đồng nhất giữa các tập, giúp đánh giá mô hình một cách chính xác hơn trong bối cảnh dữ liệu mất cân bằng.

Tạo ra ma trận embedding từ mô hình Word2Vec đã được huấn luyện sẵn. Ma trận này đóng vai trò như lớp đầu vào, giúp mô hình học sâu hiểu được ngữ nghĩa tiếng Việt ngay từ giai đoạn đầu, thay vì học từ đầu hoàn toàn.

```
Train: (4590, 180), Val: (510, 180), Test: (1050, 180)
```

Hình 17. Kết quả in ra log output

10.6. Áp dụng các mô hình Deep Learning vào bài toán

10.6.1. Mô hình TextCNN

- Lớp Input & Embedding:
  - Đầu vào là chuỗi số nguyên có độ dài cố định (MAX_SEQUENCE_LENGTH).
  - Lớp Embedding sử dụng trọng số từ Word2Vec (weights=[embedding_matrix]) và được đóng băng (trainable=False) trong giai đoạn này để giữ nguyên tri thức từ vựng gốc.
- Lớp Tích chập (Convolution Layers):
  - Sử dụng chiến lược Multi-kernel: Thay vì dùng một kích thước bộ lọc cố định, mô hình sử dụng 3 bộ lọc với kích thước kernel lần lượt là 3, 4, 5.
  - Kernel size = 3 sẽ quét qua 3 từ liên tiếp (trigrams), tương tự với 4 và 5. Điều này giúp mô hình bắt được các cụm từ (n-grams) mang tính cảm xúc ở nhiều độ dài khác nhau (ví dụ: "rất tốt" - 2 từ, "không thích lắm" - 3 từ).
- Lớp Global Max Pooling:
  - Với mỗi bản đồ đặc trưng tạo ra từ lớp tích chập, chỉ lấy giá trị lớn nhất.

- Chỉ giữ lại tín hiệu cảm xúc mạnh nhất trong câu và loại bỏ các thông tin nhiễu, giúp giảm chiều dữ liệu đáng kể.

- Concatenate & Dense:
  - Các đặc trưng từ 3 nhánh kernel được gộp lại (Concatenate), đi qua lớp Dropout(0.5) để chống học vẹt (Overfitting), cuối cùng qua lớp Dense với hàm kích hoạt Softmax để phân loại ra 3 nhãn.

```
def build_textcnn():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    convs = []
    for k in [3, 4, 5]:
        # Kernel 3-5 de bat bien the n-gram
        c = layers.Conv1D(128, k, activation="relu", padding="valid")(x)
        p = layers.GlobalMaxPooling1D()(c)
        convs.append(p)
    x = layers.Concatenate()(convs)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="textcnn")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

10.6.2. Mô hình Bi-LSTM

- SpatialDropout1D(0.2):
  - Khác với Dropout thường (tắt ngẫu nhiên các nơ-ron), SpatialDropout tắt toàn bộ một chiều đặc trưng (feature channel).
  - Trong NLP, các từ liền kề thường có mối tương quan mạnh. SpatialDropout giúp mô hình học được các đặc trưng độc lập hơn, tăng khả năng tổng quát hóa.
- Bidirectional LSTM:
  - Sử dụng lớp LSTM hai chiều: Một lớp đọc từ đầu câu đến cuối câu, một lớp đọc từ cuối về đầu.
  - Giúp mô hình nắm bắt được ngữ cảnh toàn diện. Ví dụ, trong câu *"Phim không hay lắm"*, từ *"không"* (ở đầu) sẽ tác động phủ định lên từ *"hay"* (ở sau). LSTM hai chiều xử lý tốt các cấu trúc này.
- GlobalMaxPooling1D: Trích xuất đặc trưng nổi bật nhất từ chuỗi trạng thái ẩn (hidden states) của LSTM trước khi đưa vào phân loại.

```
def build_bilstm():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    x = layers.SpatialDropout1D(0.2)(x)  # dropout tren toan bo embedding sequence
    x = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(x)
    x = layers.GlobalMaxPooling1D()(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="bilstm")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

10.6.3. Mô hình kết hợp CNN + BiLSTM

- Tầng CNN (Conv1D + MaxPooling):
  - Đầu tiên, dữ liệu đi qua lớp Conv1D (kernel size = 5) để trích xuất các đặc trưng cục bộ.
  - Sau đó qua lớp MaxPooling1D(2) để giảm kích thước chuỗi đi một nửa.
  - Đóng vai trò như một bộ lọc nhiễu và nén dữ liệu, giúp giảm tải khối lượng tính toán cho lớp LSTM phía sau.
- Tầng Bi-LSTM:
  - Nhận đầu vào là chuỗi đặc trưng đã được CNN "cô đặc". Lớp này chịu trách nhiệm học mối quan hệ thời gian/ngữ cảnh giữa các đặc trưng đó.
- Mô hình này thường nhẹ hơn Bi-LSTM thuần (do chiều dài chuỗi giảm) nhưng vẫn giữ được khả năng hiểu ngữ cảnh.

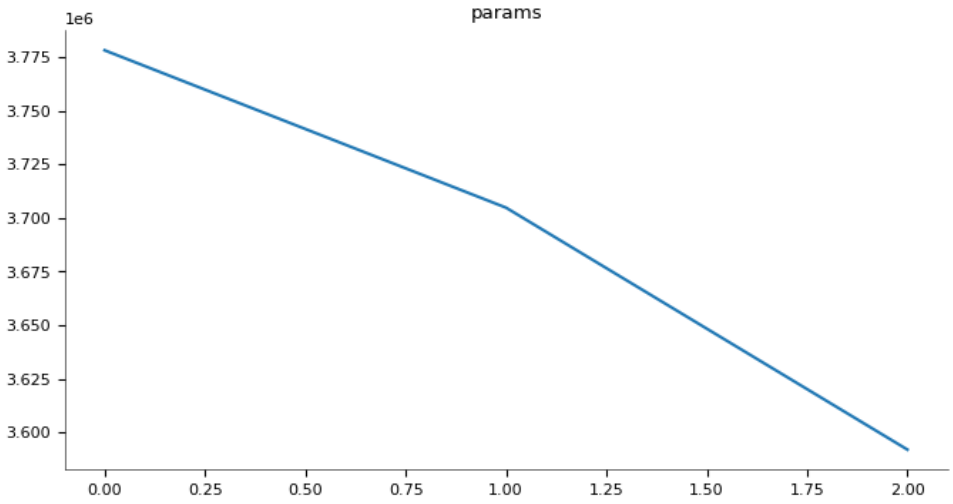
```
def build_cnn_bilstm():
    inputs = layers.Input(shape=(MAX_SEQUENCE_LENGTH,), name="text")
    x = layers.Embedding(VOCAB_SIZE, EMBEDDING_DIM, weights=[embedding_matrix], trainable=False)(inputs)
    x = layers.Conv1D(128, 5, activation="relu", padding="same")(x)
    x = layers.MaxPooling1D(2)(x)
    x = layers.Bidirectional(layers.LSTM(96, return_sequences=True))(x)  # ket hop conv + ngu canh dai
    x = layers.GlobalMaxPooling1D()(x)
    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(3, activation="softmax")(x)
    model = models.Model(inputs, outputs, name="cnn_bilstm")
    model.compile(optimizer=optimizers.Adam(1e-3), loss="categorical_crossentropy", metrics=["accuracy"])
    return model
```

10.6.4. Quy trình Huấn luyện để thực thi kết quả

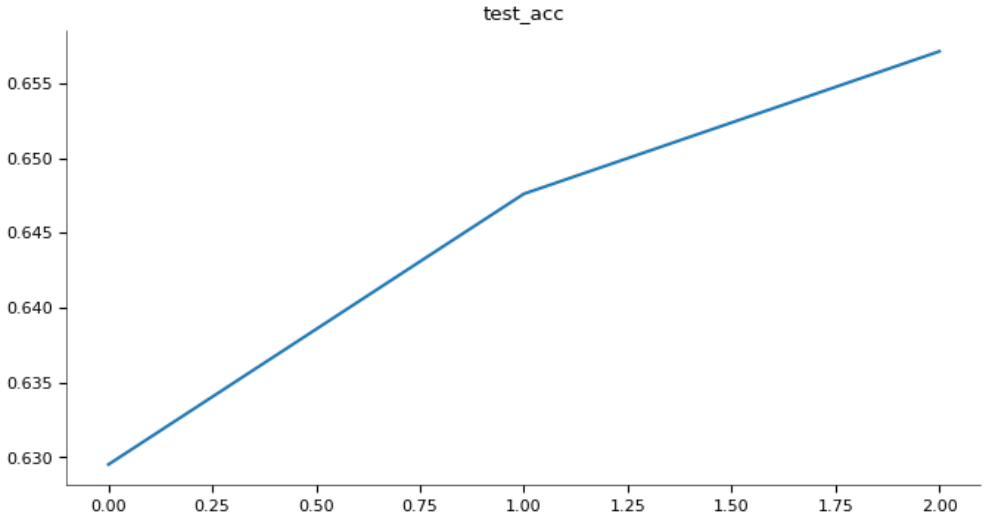
```
def run_experiment(name, builder):
    model = builder()
    es = callbacks.EarlyStopping(monitor="val_loss", patience=2, restore_best_weights=True)  # dung som khi val_loss tang
    history = model.fit(
        x_train,
        y_train,
        validation_data=(x_val, y_val),
        epochs=EPOCHS,
        batch_size=BATCH_SIZE,
        callbacks=[es],
        verbose=2,
    )
    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)
    return {
        "model": name,
        "best_val_acc": float(max(history.history["val_accuracy"])),
        "test_acc": float(test_acc),
        "params": model.count_params(),
    }


results = []
# Bo comment cac dong duoi neu muon train lan luot 3 mo hinh
results.append(run_experiment("TextCNN", build_textcnn))
results.append(run_experiment("BiLSTM", build_bilstm))
results.append(run_experiment("CNN+BiLSTM", build_cnn_bilstm))

if results:
    display(pd.DataFrame(results))
else:
    print("Chua chay huan luyen (dang o dry-run)")
```

- Cơ chế Dừng sớm (Early Stopping):
  - Code: callbacks.EarlyStopping(monitor="val_loss", patience=2, restore_best_weights=True)
  - Trong quá trình huấn luyện, nếu sau 2 epoch liên tiếp mà độ lỗi trên tập kiểm định (val_loss) không giảm, quá trình sẽ dừng lại ngay lập tức.
    1. Chống Overfitting: Ngăn mô hình học thuộc lòng dữ liệu huấn luyện khi đã đạt đỉnh hiệu suất.
    2. Tiết kiệm thời gian: Không chạy hết số epoch nếu không cần thiết.
    3. Restore Best Weights: Đảm bảo mô hình cuối cùng được lưu lại là phiên bản tốt nhất (có val_loss thấp nhất) chứ không phải là phiên bản ở epoch cuối cùng (thường đã bị overfit).
- Thiết lập Compile:
  - Optimizer: Adam với learning rate 1e-3 (chuẩn mực cho các bài toán NLP).
  - Loss function: categorical_crossentropy (bắt buộc cho bài toán phân loại đa lớp One-hot).
  - Metric: accuracy (độ chính xác).

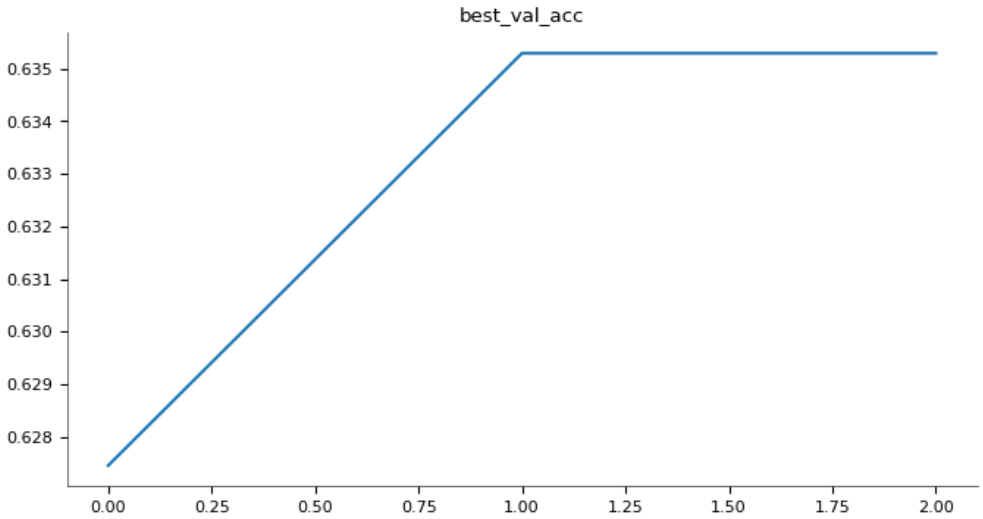
	model	best_val_acc	test_acc	params
0	TextCNN	0.633333	0.628571	3778339
1	BiLSTM	0.637255	0.647619	3704867
2	CNN+BiLSTM	0.637255	0.654286	3591907



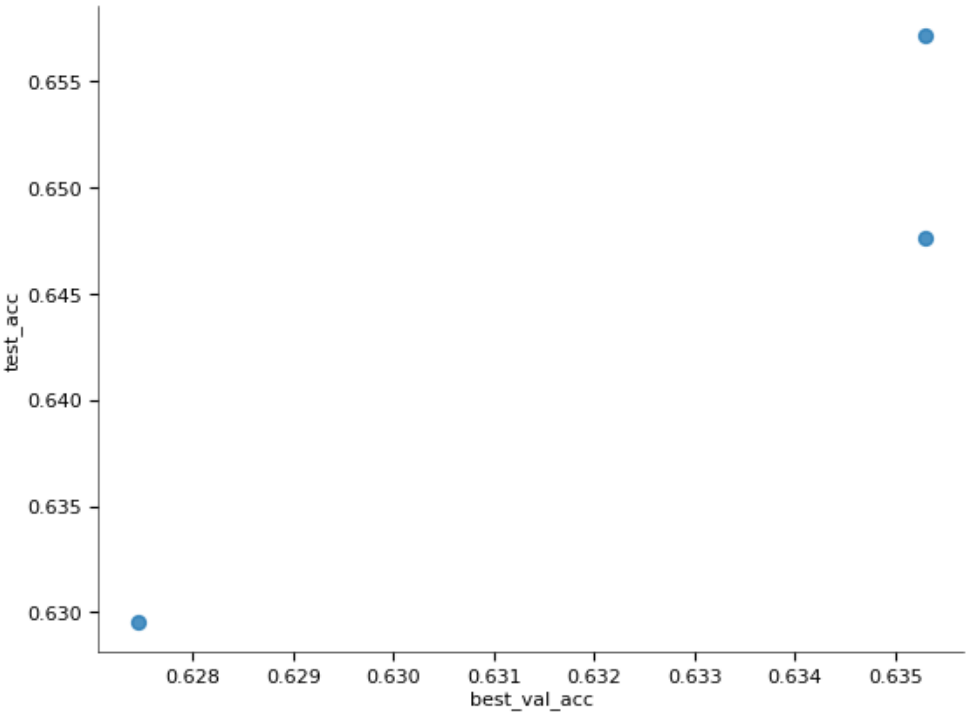
Hình 18. Đồ thị đường Params



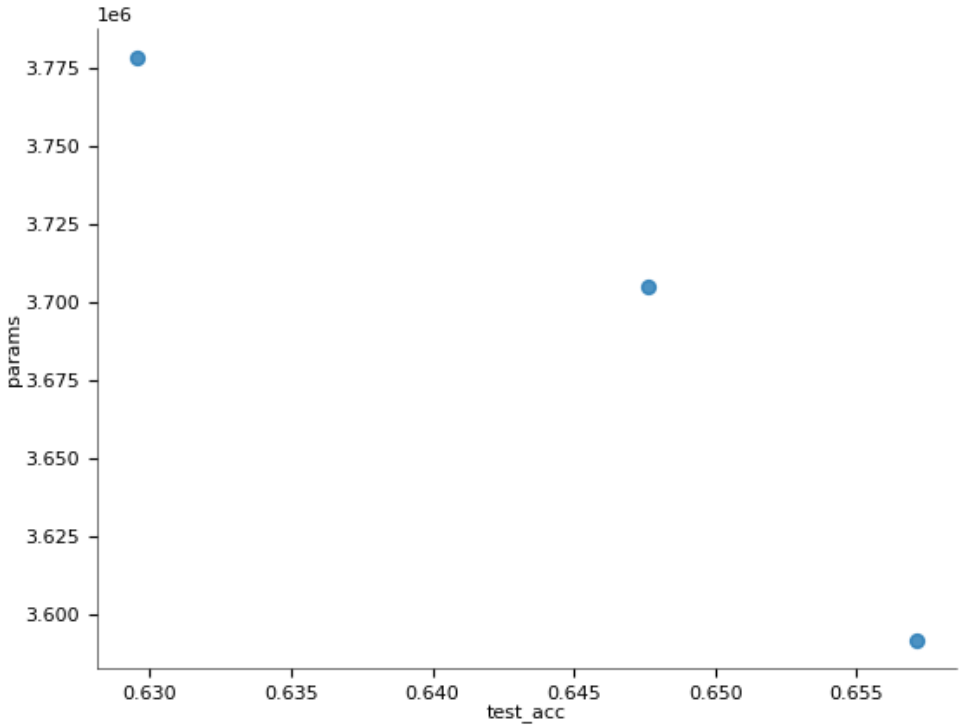
Hình 19. Đồ thị đường test_acc



Hình 20. Đồ thị đường best_val_acc

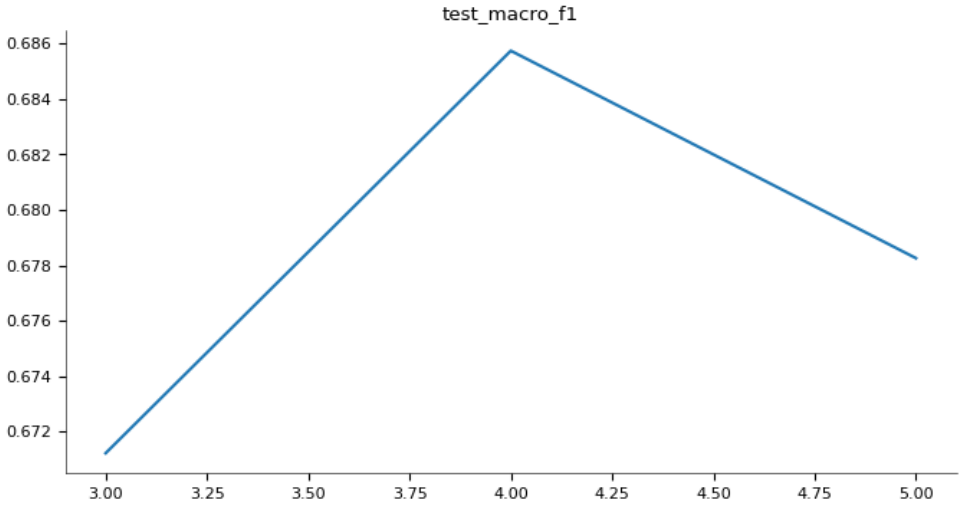


Hình 21. Đồ thị test_acc

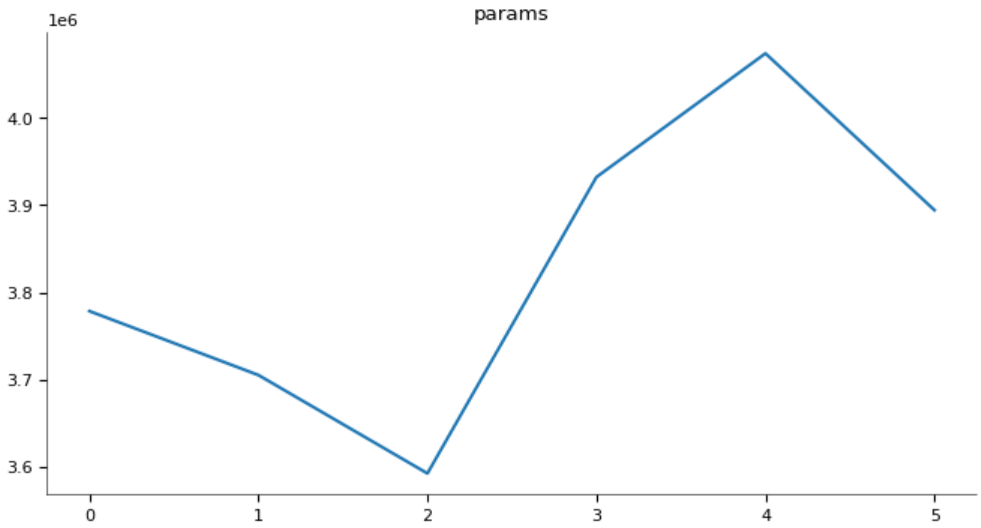


Hình 22. Đồ thị params

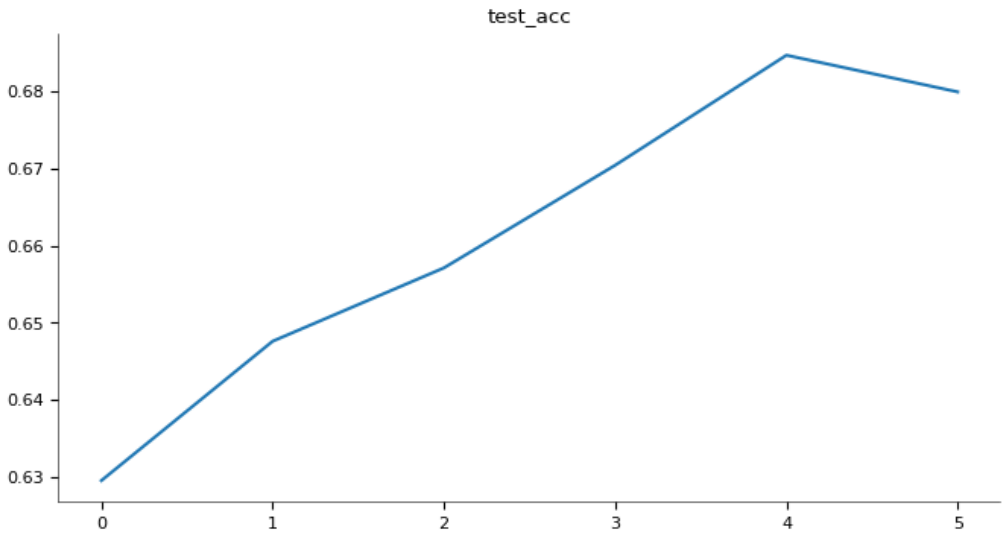
10.6.5. Mô hình chạy kết quả tối ưu



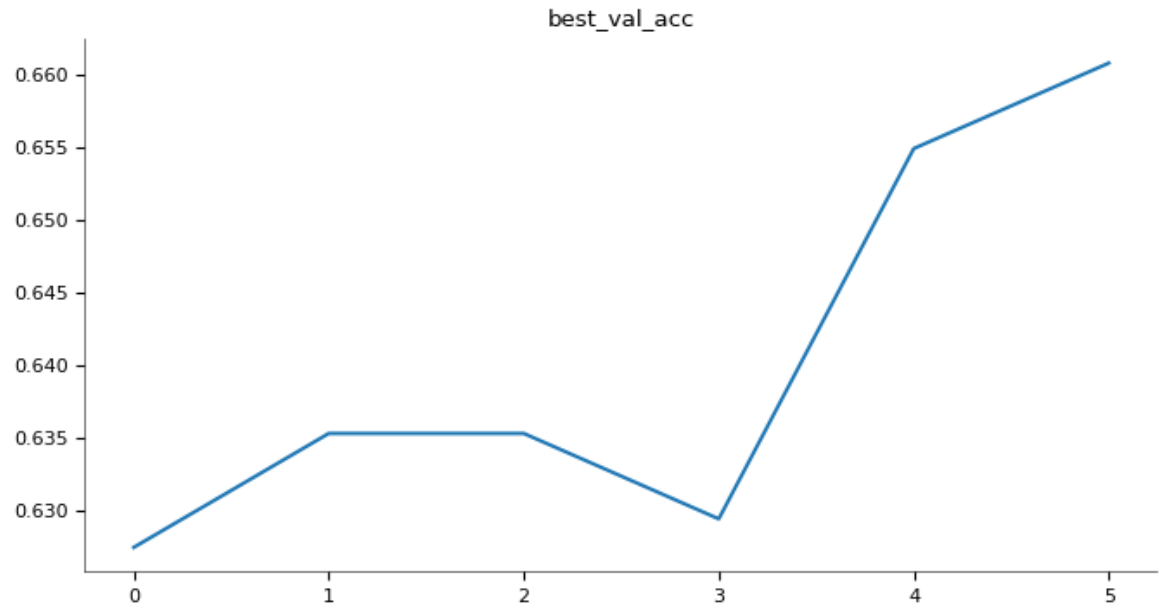
Hình 23. Đồ thị test_macro_f1 tối ưu



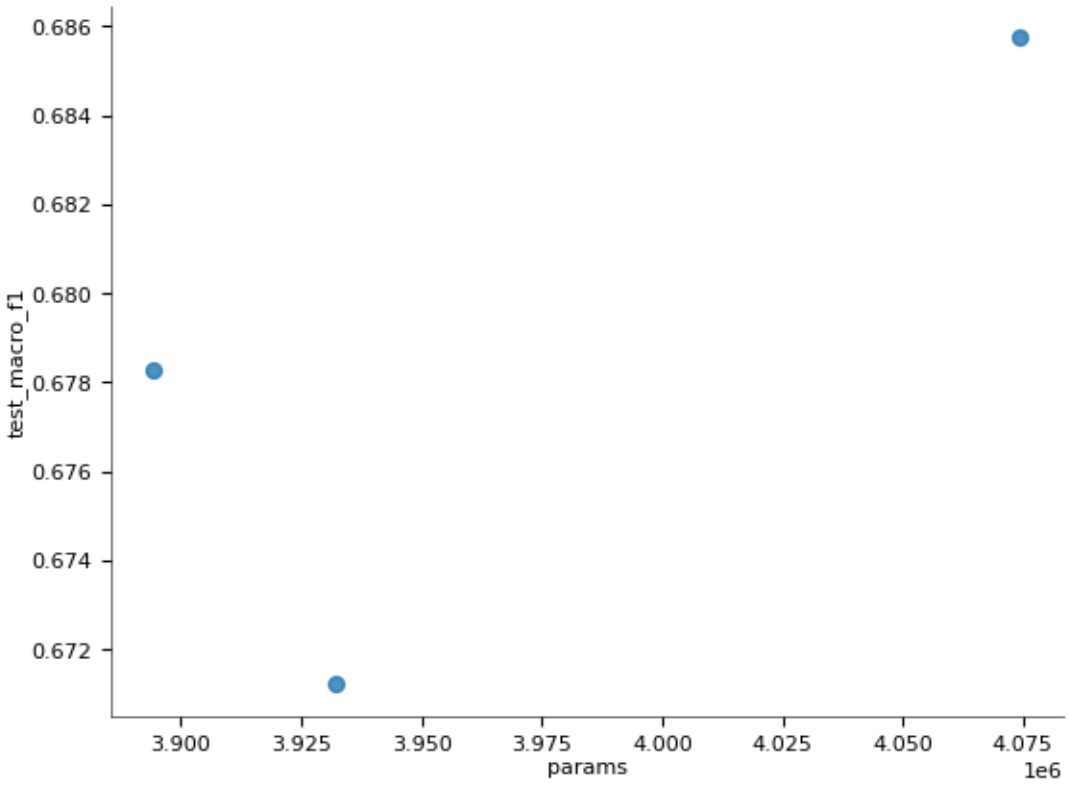
Hình 24. Đồ thị params tối ưu



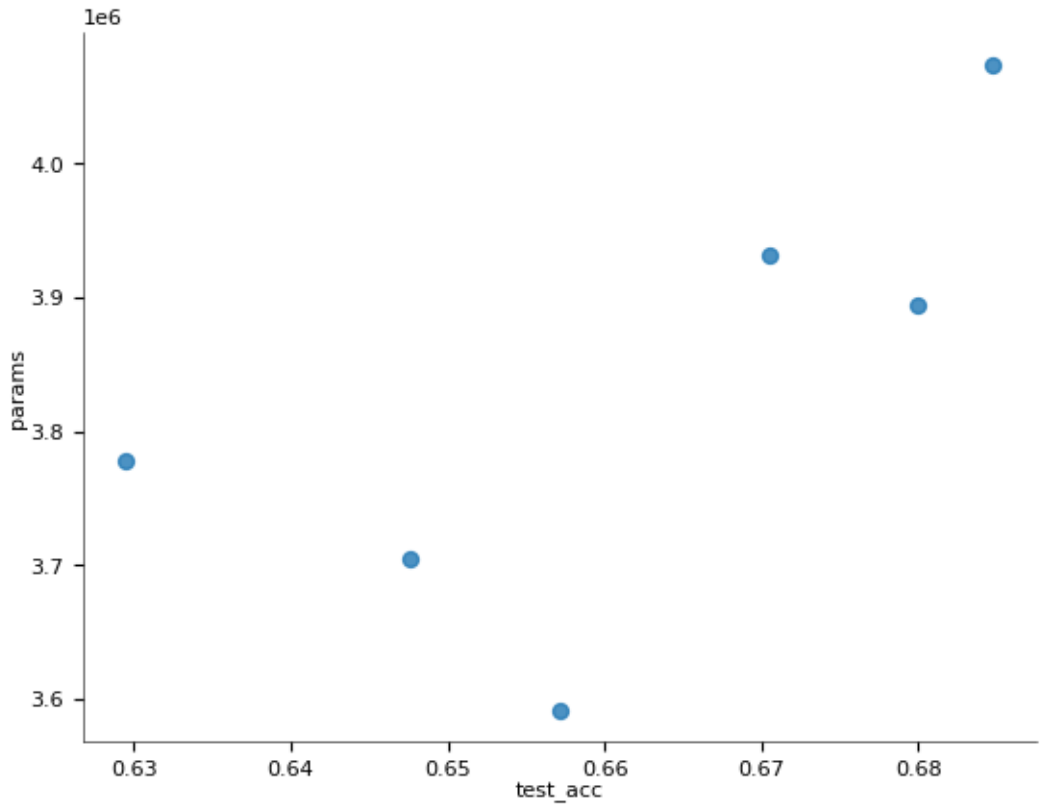
Hình 25. Đồ thị test_acc tối ưu



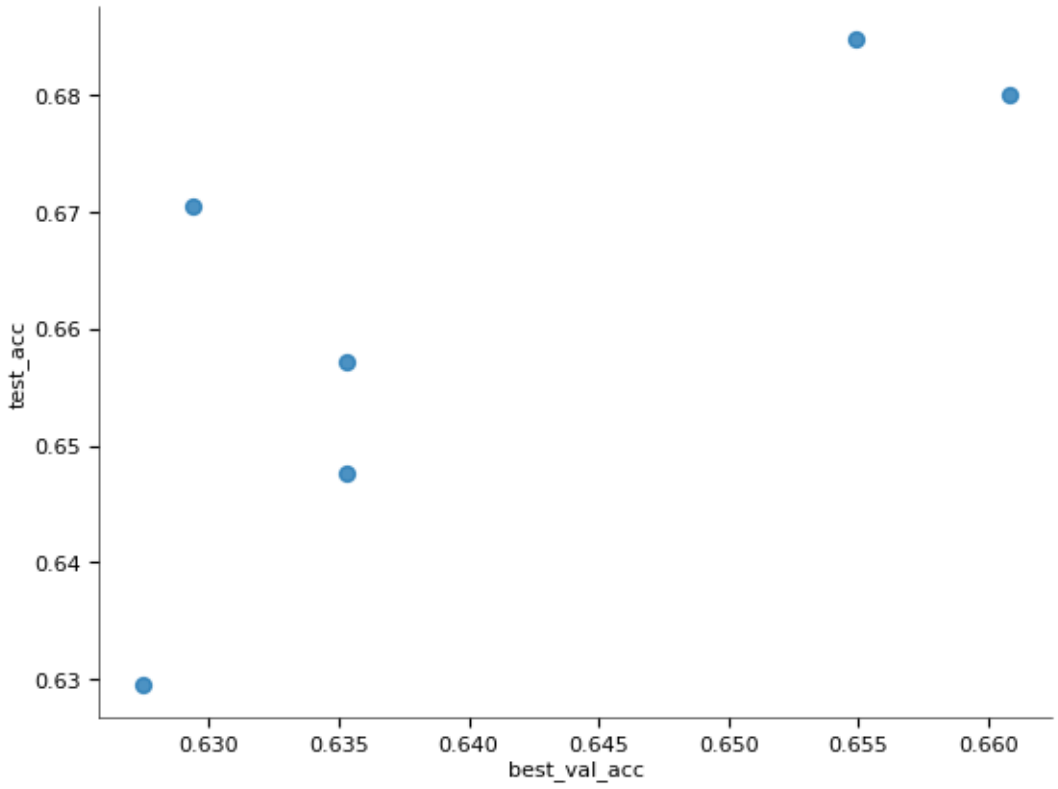
Hình 26. Đồ thị best_val_acc tối ưu



Hình 27. Đồ thị test_macro_f1 tối ưu dạng Scatter Plot



Hình 28. Đồ thị test_acc tối ưu dạng Scatter Plot



Hình 29. Đồ thị best_val_acc tối ưu dạng Scatter Plot

11. TÀI LIỆU THAM KHẢO

[1] PGS.TS Quản Thành Thơ, Chương 3 – CNN and RNN for Sentiment Analysis, Cách tiếp cận hiện đại trong xử lý ngôn ngữ tự nhiên , Khoa Khoa Học và Kỹ Thuật Máy Tính, Đại Học Bách Khoa TPHCM.

[2] PGS.TS Quản Thành Thơ, lab 5 – CNN+W2V_Sentiment Analysis, Cách tiếp cận hiện đại trong xử lý ngôn ngữ tự nhiên , Khoa Khoa Học và Kỹ Thuật Máy Tính, Đại Học Bách Khoa TPHCM.

[3] PGS.TS Quản Thành Thơ, lab 6 – W2V+[LSTM, CRNN] Sentiment Analysis, Cách tiếp cận hiện đại trong xử lý ngôn ngữ tự nhiên , Khoa Khoa Học và Kỹ Thuật Máy Tính, Đại Học Bách Khoa TPHCM.