Phân loại đánh giá thực phẩm bằng kỹ thuật phân tích cảm xúc

Độ chính xác cao nhất: 0.91809

Tóm tắt nội dung. **Phân loại văn bản tự động được coi là một phần quan trọng trong việc quản lý và xử lý tài liệu dưới dạng số, ngày càng gia tăng. Trong khi có một số nghiên cứu đề cập đến vấn đề phân loại tài liệu tiếng Anh thì có rất ít nghiên cứu đề cập đến vấn đề phân loại tài liệu tiếng Việt. Trong bản báo cáo này, nhóm đề xuất một số mô hình học máy để phân loại văn bản tiếng Việt. Kết quả thực nghiệm được thử nghiệm trên bộ dữ liệu trong cuộc thi, cùng với dữ liệu thu thập được từ trang web Foody.**

Keywords: Sentiment Analysis, SVM, Linear Regression, LSTM, Attention, BERT, GRU, Hierarchical attention Networks.

GIỚI THIỆU

Với mục tiêu là đánh giá cảm xúc được thể hiện trong văn bản, bài toán Sentiment Analysis thường được sử dụng để phân tích phản hồi của khách hàng, phản hồi khảo sát và đánh giá sản phẩm.

Bài toán Sentiment Analysis được sử dụng trong đề tài với mục tiêu đánh giá các phản hồi của khách hàng trong ứng dụng Foody.

A. Bộ dữ liệu

Mỗi mẫu dữ liệu trong tập "full_train.csv" bao gồm 4 thuộc tính chính như sau:

RevId: Review ID.

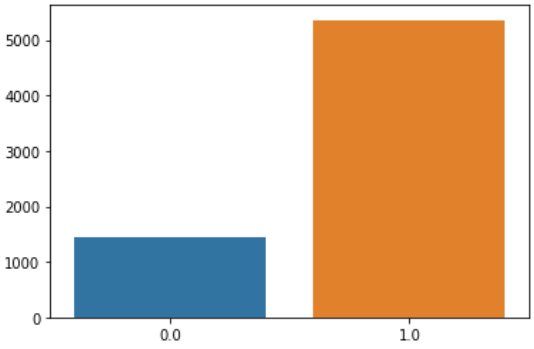
UserId: User ID.

Comment: User's comment on the review.

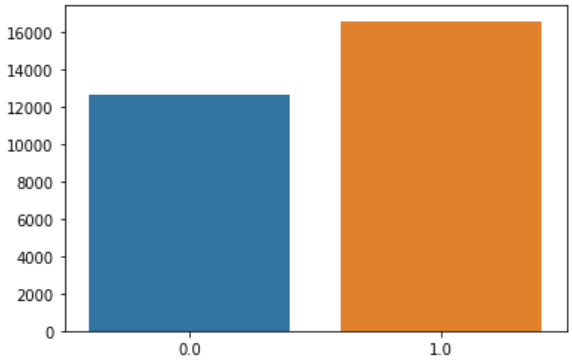
image_urls: The URL address contains images related to the review.

Với Rating là nhãn thể hiện cho đánh giá về phản hồi của người dùng và sẽ được loại bỏ trong tập "test.csv".

Tuy nhiên, theo thống kê thì lượng dữ liệu đang bị chênh lệch khá lớn giữa hai lớp:



Do đó, để tăng độ chính xác cho mô hình thì ta đã thêm dữ liệu cho mô hình từ tập dữ liệu của ứng dụng foody, sau khi xử lý file thì ta sử dụng dữ liệu từ file data1.csv



B. Yêu cầu bài toán

Yêu cầu của bài toán là xác định một đánh giá của người dùng là tích cực (positive) hay tiêu cực (negative) từ các thuộc tính bên trên như: Comment, image_urls. Nhóm thực hiện đánh giá bằng thuộc tính Comment, như vậy bài toán có thể đưa về việc phân loại văn bản dưới 2 nhãn 0 và 1, với 0 là nhãn dành cho phản hồi tích cực (positive) và 1 dành cho phản hồi tiêu cực (negative).

C. Giải pháp đề xuất

Với việc được quy về dưới dạng bài toán phân loại văn bản với các nhãn 0 và 1 thì những thuật toán phân loại trong học máy quen thuộc sẽ được áp dụng vào bài toán là SVM, Logistic Regression, LSTM with Attention và BERT.. dựa trên phần dữ liệu 'Comment' trong tập dữ liệu.

II. PHƯƠNG PHÁP

A. Preprocessing (Tiền xử lý dữ liệu)

Việc huấn luyện và kiểm thử dữ liệu đều được tiến hành trên file "full_train.csv". Với việc có khá nhiều nhiễu trong tập dữ liệu, nên để có thể đạt được kết quả tốt trong việc áp dụng các thuật toán học máy đã nêu ở trên thì dữ liệu cần được xử lý một cách tỉ mỉ.

Đầu tiên, vì dữ liệu (bình luận của người dùng) ở dạng văn nói nên người dùng thường không quan tâm đến chữ hoa và chữ thường khi gõ, vậy nên chúng ta đưa hết các chữ cái về dạng chữ thường.

Tiếp theo, vì dữ liệu ở dạng văn nói nên các dấu câu (punctuations) và các ký tự nhiễu thường không có ý nghĩa, nên có thể loại bỏ hết những dấu và và các ký tự nhiễu này.

Ngoài ra, một số người dùng thường có xu hướng kéo dài các ký tự ở cuối nên cũng cần loại bỏ những ký tự kéo dài này, ví dụ như: Áo đẹp quáaaaaaaa ⇒ Áo đẹp quá. Các thẻ html, các đường dẫn URL hay các ký tự xuống dòng "/n" cũng cần được loại bỏ.

B. Cơ sở lý thuyết

1) Logistic Regression

a) Định nghĩa:

Hồi quy logistic là một mô hình phân tích thống kê thường được sử dụng để phân loại và phân tích dự đoán. Hồi

quy logistic ước tính xác suất xảy ra các kết quả, chẳng hạn như đã bỏ phiếu hoặc không bỏ phiếu, dựa trên tập dữ liệu nhất định gồm các biến độc lập. Vì kết quả là một xác suất nên kết quả đầu ra của mô hình này có giới hạn từ 0 đến 1. Đây cũng là lý do mà nhóm hướng tới mô hình học máy này cho bài toán phân loại Positive và Negative bên trên.

b) Xây dựng mô hình:

Tập dữ liệu của mô hình là $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ với các giá trị $x_i \in R^d$ là một điểm dữ liệu gồm các thuộc tính cần phân tích dự đoán, $y_i \in \{0, 1\}$ là giá trị đầu ra tương ứng với điểm dữ liệu x_i. Ngoài ra các tham số mô hình $\theta = (w, w_0)$ cũng được chọn từ đó xác định hàm kích hoạt của mô hình:

$$f(x) = w^T x + w_0$$

Hàm kích hoạt được sử dụng là hàm sigmoid:

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

Đầu ra dự đoán của mô hình là:

$$\sigma(f(x)) = \sigma(w^T x + w_0)$$

Mô hình hồi quy Logistic thực hiện xác định các giá trị class y_i từ các điểm dữ liệu x_i cũng như các tham số mô hình bằng việc so sánh giá trị đầu ra dự đoán của mô hình $\sigma(f(x))$ với 0.5. Nếu lớn hơn thì ta kết luận điểm dữ liệu thuộc class 1, ngược lại thì nó thuộc class 0.

2) SVM (Support Vector Machine)

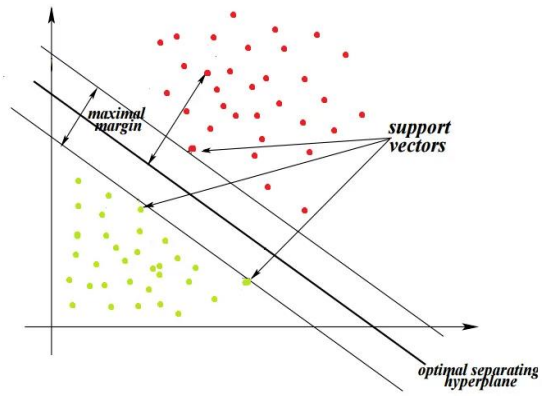
a) Định nghĩa:

SVM (Support Vector Machine) hay được gọi là máy vector hỗ trợ là một thuật toán giám sát, nó được sử dụng cho cả việc phân loại hoặc đệ quy. Với yêu cầu phân loại cảm xúc thành Negative và Positive thì đây là bài toán Classifier nên phương án nhóm sử dụng là SVM. SVM dạng chuẩn nhận dữ liệu vào và phân loại chúng vào hai lớp khác nhau.

Mục tiêu của SVM là tìm ra một siêu phẳng trong không gian N chiều (ứng với N đặc trưng) chia dữ liệu thành hai phần tương ứng với lớp của chúng. Gọi những điểm dữ liệu gần với siêu phẳng là các vector hỗ trợ, các vector hỗ trợ này nằm trên một mặt biên song song với mặt phẳng phân cách và khoảng cách giữa chúng với mặt phân cách được gọi là lề (margin).

Bài toán tối ưu SVM là đi tìm một siêu phẳng sao cho lề là lớn nhất.

Tuy nhiên nếu chỉ dừng lại ở trên SVM sẽ chỉ dừng lại ở việc phân loại được các bộ dữ liệu khả tuyến. Với những bộ dữ liệu không khả tuyến, SVM đưa ra giải pháp chuyển không gian hiện tại lên chiều không gian lớn hơn giúp tăng khả năng phân loại, để chuyển được từ không gian này sang không gian khác thì cần dùng kernel (nhân). Trong SVM có rất nhiều kernel với bài toán phân loại cảm xúc thì nhóm nhận thấy Linear Kernel rất phù hợp cho mục đích phân loại này



b) Xây dựng mô hình:

Các cặp dữ liệu của training set là $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ với vector $x_i \in R^d$ thể hiện đầu vào của một điểm dữ liệu và y_i là nhãn của điểm dữ liệu đó có $y_i \in \{-1,1\}$, d là số chiều của dữ liệu và N là số điểm dữ liệu.

Gọi siêu phẳng phân cách phân chia hai điểm dữ liệu là:

$$b + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N = b + \mathbf{w}^\intercal \mathbf{x} = 0$$

b là hệ số tự do, **w** là các vector hệ số, **x** là các vector quan sát đầu vào

Như vậy, khoảng cách từ một điểm bất kỳ $Z_n(x_n, y_n)$ tới đường biên là siêu phẳng H có phương trình $b + \mathbf{w}^\intercal \mathbf{x} = 0$ sẽ là:

$$d(Z_n, H) = \frac{|b + \mathbf{w}^\intercal \mathbf{x}_n|}{||\mathbf{w}||_2} = \frac{y_n(b + \mathbf{w}^\intercal \mathbf{x}_n)}{||\mathbf{w}||_2}$$

Từ công thức khoảng cách sẽ đi tìm **w** và b đường biên có lề lớn nhất.

$$(\mathbf{w}, b) = \arg\max_{\mathbf{w},b} \left\{ \min_n \frac{y_n(\mathbf{w}^T \mathbf{x}_n + b)}{||\mathbf{w}||_2} \right\} = \arg\max_{\mathbf{w},b} \left\{ \frac{1}{||\mathbf{w}||_2} \min_n y_n(\mathbf{w}^T \mathbf{x}_n + b) \right\}$$

Sau khi xác định được **w** và b hay mặt phân cách $w^T x + b = 0$ rồi thì sẽ thực hiện xác định class cho một điểm dữ mới thông qua công thức:

$$class(x) = sgn(w^T x + b)$$

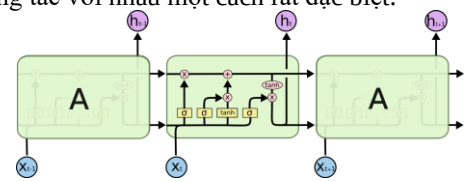
Trong đó hàm sgn là hàm xác định dấu, nhận giá trị là 1 nếu đối số là không âm và -1 nếu ngược lại..

3) LSTM

a) Định nghĩa:

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa.

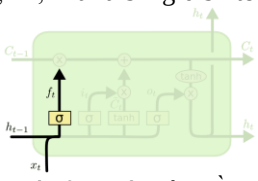
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kỳ can thiệp nào. LSTM cũng có kiến trúc dạng chuỗi như RNN, nhưng các mô-đun trong nó có cấu trúc khác với mạng RNN chuẩn. Thay vì chỉ có một tầng mạng nơ-ron, chúng có tới 4 tầng tương tác với nhau một cách rất đặc biệt.



Hình : Mạng LSTM.

b) Xây dựng mô hình:

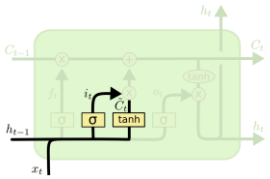
Bước 1: LSTM quyết định thông tin nào cần được loại bỏ từ trạng thái tế bào. Nó sẽ lấy đầu vào là h_(t-1) và x_t rồi đưa ra kết quả trong khoảng [0,1] cho mỗi số trong trạng thái tế bào C_(t-1). Đầu ra bằng 1 thể hiện việc dữ toàn bộ thông tin, đầu ra bằng 0 bỏ toàn bộ thông tin.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

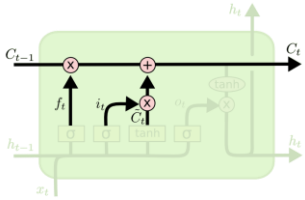
Bước 2: Bước này gồm 2 phần.

- Phần 1: Sử dụng tầng sigmoid để quyết định giá trị nào sẽ cập nhật
- Phần 2: tầng tanh tạo ra một vector cho giá trị mới Č_t để thêm vào cho trạng thái. Kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái tế bào cũ C_(t-1) thành trạng thái mới C_t.



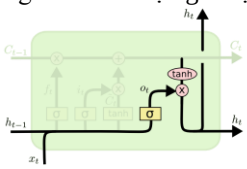
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Bước 3: Nhân trạng thái cũ với f_t để loại bỏ những thông tin dư thừa đã quyết định trước đó. Sau đó cộng thêm $i_t * \tilde{C}_t$.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Bước 4: Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa trạng thái tế bào qua một hàm tanh để đưa giá trị của nó về khoảng [-1, 1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra mong muốn.\



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

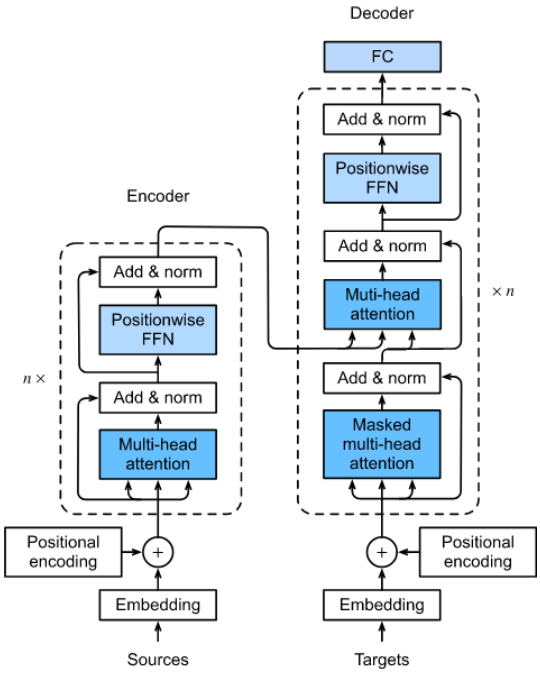
4) Transformer

a) Định nghĩa:

Để kết hợp các ưu điểm của CNN và RNN, Attention is All you Need đã thiết kế một kiến trúc mới bằng cách sử dụng cơ chế tập trung. Kiến trúc này gọi là Transformer, song song hóa bằng cách học chuỗi hồi tiếp với cơ chế tập trung, đồng thời mã hóa vị trí của từng phần tử trong chuỗi. Mô hình này mô tả đầu ra y_t bằng các đầu vào $x_{1:t}$ và đầu ra trước đó $y_{1:t-1}$, như vậy nếu có dữ liệu $x_{1:T}$, $y_{1:T}$thì có T dữ liệu để huấn luyện mô hình. Kết quả là ta có một mô hình tương thích với thời gian huấn luyện ngắn hơn đáng kể.

b) Xây dựng mô hình:

Dưới đây là cấu trúc của mô hình Transformer



Trong đó các khối chính bao gồm:

- **Embedding**

- Sử dụng một lớp Embedding để chuyển token đầu vào thành x_t

- Sử dụng một lớp Embedding để chuyển token đầu ra (đã dịch phải) thành y_{t-1}

- **Mã hóa vị trí (Positional Encoding)**

Mạng Transformer mã hoá thông tin vị trí bằng cách tạo ra các vector có cùng số chiều với embedding như sau:

$$\mathrm{PE}_{pos,2i} = \sin\left(pos/10000^{2i/d}\right)$$
$$\mathrm{PE}_{pos,2i+1} = \cos\left(pos/10000^{2i/d}\right)$$

embedding có d chiều ($i = 1,2,\ldots,d$) mỗi vị trí pos → vector d chiều trong đó các vị trí chẵn được mã hoá bằng hàm $sing$, các vị trí lẻ được mã hoá bằng hàm cos. Vector này được cộng với vector embedding để tạo ra đầu vào của các khối **Mã hóa** (Encoder) và **Giải mã** (Decoder).

- **Khối mã hoá (Encoder)**

Khối mã hoá gồm 2 mô đun: mô đun Chú ý (Attention) và một mạng MLP. Các mô đun này sử dụng kết nối phần dư (thêm đầu vào của một khối vào đầu ra của nó) giúp mạng có thể chồng được nhiều layers và sau đó chuẩn hoá (normalization). Các khối mã hoá có thể xếp chồng lên nhau (**stacking**) để mô tả các mối quan hệ phức tạp giữa các token đầu vào.

- **Khối giải mã (Decoder)**

Khối giải mã gồm 3 mô đun: 1 mô đun Chú ý (Attention) chạy trên các đầu ra phía trước ($y_{1:t-1}$), 1 mô đun Chú ý (Attention) chạy trên cả đầu ra của Khối Encoder

và 1 mạng MLP. Các mô đun này cũng sử dụng kết nối phần dư và chuẩn hoá

Điểm đặc biệt của Transformer là cơ chế chú ý (Attention mechanism). Cơ chế này cho phép mạng Transformer có bộ nhớ rất dài, hơn LSTM rất nhiều. Mạng Transformer có thể chú ý hoặc tập trung vào tất cả các token đầu vào (cả LSTM và RNN đều chỉ có thể truy xuất đến phần tử ngay trước đó). Quá trình này được thực hiện như sau:

1. Đầu vào: X là các embedding (đã cộng PE) của các token
2. Tính các ma trận Q (query), K (key), V (value) có số hàng bằng số token, các hệ số W_Q, W_K, W_V là hệ số mà mô hình cần huấn luyện

$$Q = XW_Q; K = XW_K; V = XW_V$$

3. Tính ma trận điểm số S là tương quan giữa Q và K (sự chú ý của từng vị trí tới các vị trí khác).

$$S = QK^T$$

4. Chia điểm số cho $\sqrt{d_k}$ (số đặc trưng của Q và K) để giảm độ lớn của điểm số sau khi nhân ma trận.
5. Sử dụng điểm số (qua softmax) làm trọng số để tổng hợp các cột trong V (trọng số của từng vị trí trong câu ảnh hưởng đến từng vị trí).

$$S(\frac{QK^T}{\sqrt{d_k}}) = S(S)$$

6. Đầu ra: các mã hoá mới có tính đến trọng số của từng vị trí

$$Attention(Q, K, V) = S(\frac{QK^T}{\sqrt{d_k}})V$$

Tuy nhiên, để có thể hiểu được vai trò của một từ trong một câu ta cần hiểu được sự liên quan giữa từ đó và các thành phần còn lại trong câu. Vì vậy, thay vì chỉ có một cơ chế attention như ở trên hay còn gọi là 1 "head" thì mô hình sẽ có nhiều "heads", mỗi head sẽ tập trung vào sự liên quan giữa từ và các thành phần còn lại. Đó chính là cấu trúc chú ý nhiều đầu (multi-head attention) trong Transformers

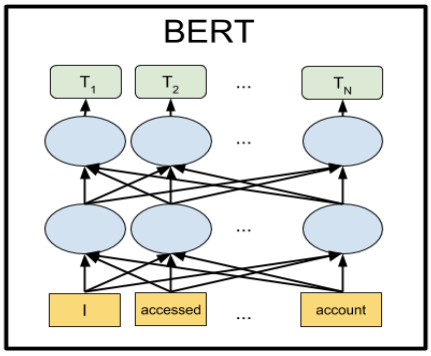
5) BERT

a) Định nghĩa:

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình ngôn ngữ (Language Model) được tạo ra bởi Google AI. Trước khi BERT ra đời thì các tác vụ như: phân loại cảm xúc văn bản (tốt hay xấu, tích cực hay tiêu cực), sinh văn bản, dịch máy,…. đều sử dụng kiến trúc RNN. Kiến trúc này có nhiều nhược điểm như train chậm, mất quan hệ giữa các từ xa nhau… BERT tốt hơn so với RNN bởi nó có nhúng thêm ngữ cảnh (Context) vào trong các vector embedding (ngữ cảnh là thứ vô cùng quan trọng trong ngôn ngữ, với ngữ cảnh khác nhau thì các từ trong câu được hiểu theo nghĩa hoàn toàn khác nhau).

b) Xây dựng mô hình:

BERT chia Transformer làm 2 phần, chỉ giữ phần Encoder bên trái và bỏ phần DeCoder bên phải (giữ phần input văn bản đầu vào và đầu ra là các encoder output)



BERT được train đồng thời 2 task là Masked LM (dự đoán từ thiếu trong câu) và Next Sentence Prediction (NSP - dự đoán câu tiếp theo câu hiện tại). Hai task được trên đồng thời và loss tổng sẽ kết hợp loss của 2 task và Model Bert sẽ cố gắng tối ưu minimize loss tổng.

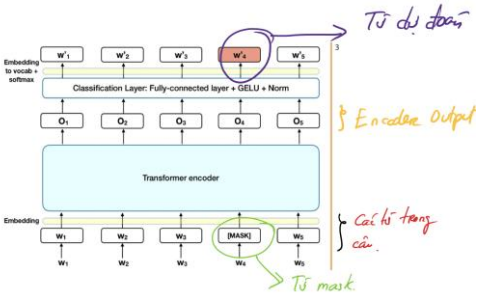
1. **Masked LM**

Với task này, khi train sẽ che đi 15% số từ trong câu và đưa vào model. Mục đích train để model predict ra các từ bị che dựa vào các từ đã có trước đó. Cụ thể:

Thêm 1 lớp classification lên trên lớp encoder output.

Đưa các vector trong encoder output về vector bằng với vocab size, sau đó softmax để chọn ra các từ tương ứng ở mỗi vị trí trong câu.

Loss sẽ được tính tại vị trí masked và bỏ qua các vị trí khác (để đánh giá xem model dự đoán từ mask đúng/sai ntn mà, các từ khác đâu có liên quan).



2. **Next Sentence Prediction (NSP)**

Model sẽ được feed cho một cặp câu và nhiệm vụ của nó là output ra giá trị 1 nếu câu thứ hai đúng là câu đi sau câu thứ nhất và 0 nếu không phải. Trong quá trình train, ta chọn 50% mẫu là Positive (output là 1) và 50% còn lại là Negative được ghép linh tinh (output là 0). Cụ thể:

+ Bước 1: Ghép 2 câu vào nhau và thêm 1 số token đặc biệt để phân tách các câu. Token [CLS] thêm vào đầu cầu Thứ nhất, token [SEP] thêm vào cuối mỗi câu. Ví dụ ghép 2 câu "Hôm nay em đi học" và "Học ở trường rất hay" thì sẽ thành [CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP]

Bước 2. Mỗi token trong câu sẽ được cộng thêm một vector gọi là Sentence Embedding, thực ra là đánh dấu xem từ đó thuộc câu Thứ nhất hay câu thứ 2 thôi. Ví dụ nếu thuộc câu Thứ nhất thì cộng thêm 1 vector toàn số "0" có kích thước bằng Word Embedding, và nếu thuộc câu thứ 2 thì cộng thêm một vector toàn số "1".

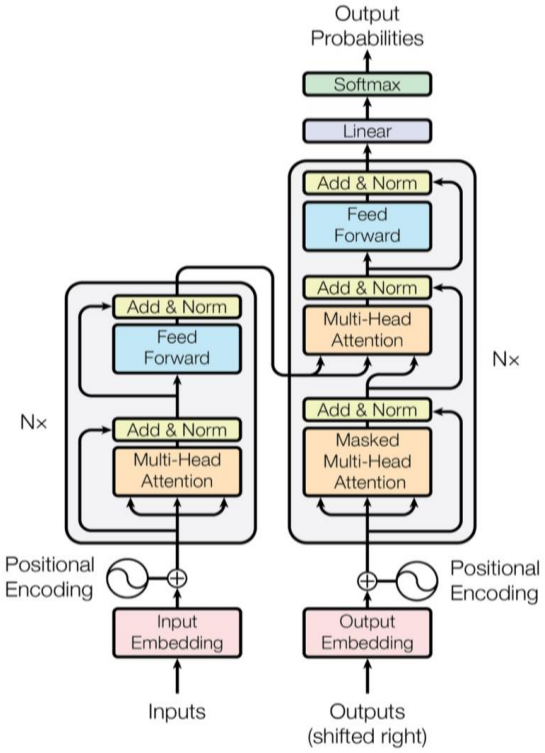
Bước 3. Sau đó các từ trong câu đã ghép sẽ được thêm vector Positional Encoding vào để đánh dấu vị trí từng từ trong câu đã ghép (bạn nào chưa biết thì xem lại bài về Transformer nhé).

Bước 4. Đưa chuỗi sau bước 3 vào mạng.

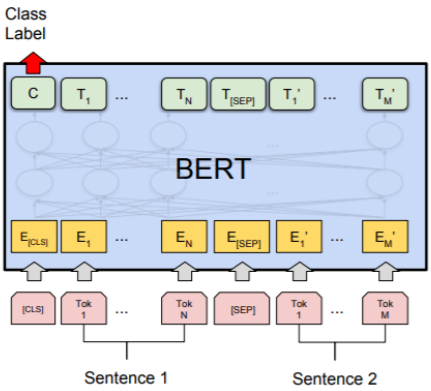
Bước 5. Lấy encoder output tại vị trí token [CLS] được transform sang một vector có 2 phần tử [c1 c2].

Bước 6. Tính softmax trên vector đó và output ra probability của 2 class: Đi sau và Không đi sau. Để thể hiện câu thứ hai là đi sau câu thứ nhất hay không, ta lấy argmax

Các bước tạo input:



Đây là cách lấy output đầu ra:



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

C. Pretrain (Tiền huấn luyện dữ liệu)

- *Vector hoá bằng thư viện TfidfTransformer*

Để áp dụng phân loại văn bản, định dạng của văn bản phải được chuyển đổi thành định dạng có cấu trúc vì lý do đơn giản là xử lý các con số sẽ dễ dàng hơn nhiều so với văn bản thuần. Điều này chủ yếu đạt được bằng cách chiếu nội dung văn bản vào mô hình không gian vector (Vector Space Model), nơi dữ liệu văn bản được chuyển đổi thành các vectơ số. Trong lĩnh vực phân loại văn bản, các tài liệu thường được xử lý như một túi từ (BoW), nghĩa là mỗi từ độc lập với các từ khác có trong tài liệu. Chúng được kiểm tra mà không quan tâm đến ngữ pháp cũng như trật tự từ. Trong một mô hình như vậy, tần số thuật ngữ (sự xuất hiện của mỗi từ) được sử dụng như một đặc điểm để huấn luyện bộ phân loại. Tuy nhiên, việc sử dụng thuật ngữ tần suất hàm ý rằng tất cả các thuật ngữ đều được coi là quan trọng như nhau. Như tên gọi của nó, tần suất của thuật ngữ chỉ đơn giản là đặt trọng số cho từng thuật ngữ dựa trên tần suất xuất hiện của chúng và không tính đến khả năng phân biệt đối xử của các thuật ngữ. Để giải quyết vấn đề này và xem xét các từ xuất hiện quá thường xuyên, mỗi từ được cung cấp một trọng số tf-idf được định nghĩa như sau:

$$tfidf_{t,d} = tf_{t,d} * idf_t$$

Trong đó:

$tf_{t,d} = \frac{Số\ lần\ từ\ t\ xuất\ hiện\ trong\ văn\ bản\ d}{Tổng\ số\ từ\ trong\ văn\ bản\ d}$

dùng để ước lượng tần suất xuất hiện của từ trong văn bản.

$idf_{t,d} = ln\left(\frac{Tổng\ số\ văn\ bản\ trong\ tập\ mẫu\ d}{Số\ văn\ bản\ có\ chứa\ từ\ t}\right)$

dùng để ước lượng mức độ quan trọng của từ đó như thế nào

- *PhoBert*

Với mục đích tạo đòn bẩy cho sự phát triển của xử lý ngôn ngữ tiếng Việt, PhoBERT đã ra đời. Mô hình này có cùng kiến trúc với BERT và sử dụng phương pháp tiền huấn luyện của RoBERTa.

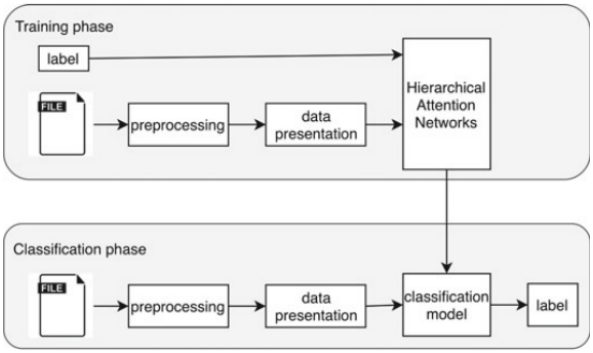
Word segmentation as input:

Tác giả của PhoBERT đã sử dụng RDRSegmenter của VnCoreNLP để phân đoạn từ cho văn bản trước khi đào tạo. Tuy nhiên, tác giả không có đưa ra kết quả của PhoBERT mà không sử dụng kỹ thuật phân đoạn từ (word segmentation), do đó ta không biết rõ hiệu quả của việc sử dụng kỹ thuật này. Do đó, ta sẽ so sánh việc đào tạo mô hình có hoặc không phân đoạn từ.

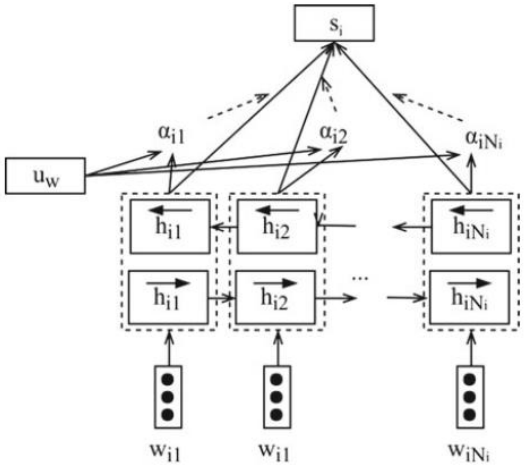
Training data:

Mô hình được đào tạo trên khoảng 20GB văn bản được thu thập từ kho ngữ liệu Wikipedia tiếng Việt và kho tin tức tiếng Việt. Kết quả thử nghiệm của bài báo PhoBERT cho thấy mô hình được đào tạo trước đạt được kết quả tiên tiến nhất trên một số tác vụ xuôi dòng, bao gồm gắn thẻ Phần lời nói, Phân tích cú pháp phụ thuộc, Nhận dạng thực thể được đặt tên và suy luận ngôn ngữ tự nhiên

- **Hierarchical Attention Networks (HAN)**



HAN là kỹ thuật dùng để phân loại tài liệu. Mạng này sẽ biểu diễn các vector bằng cách sử dụng hai bộ mã hoá dựa trên trình tự của các từ trong câu và thứ tự các câu, sau đó xếp chồng một lớp softmax để phân loại. Điểm nổi bật của model này là khả năng biểu diễn các từ và câu quan trọng bởi vì áp dụng cơ chế "chú ý phân cấp".



Hình : Kỹ thuật HAN: các lớp "sentence-level"

Gated Recurrent Unit (GRU): được sử dụng trong mô hình này để giải quyết vấn đề "vanish gradient" trong RNN. GRU sử dụng cổng cập nhật và cổng đặt lại để kiểm soát thông tin truyền đến đầu ra. Trong đó, cổng cập nhật z_t định lượng thông tin cũ và mới được cập nhật sang trạng thái mới h_t. Trong khi đó, cổng đặt lại r_t được sử dụng để xác định lượng thông tin trong quá khứ cần quên. Tại bước thời gian t, z_t và r_t được tính như sau:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

Trong đó, xt là vectơ của phần tử thứ t. W(z)/W(r) và U(z) /U(r) tương ứng là các trọng số cho x_t và trạng thái trước đó h_{t-1} trong các cổng cập nhật/đặt lại. Nội dung bộ nhớ hiện tại sử dụng cổng đặt lại để lưu trữ thông tin liên quan từ quá khứ:

$$h'_t = \tanh(Wx_t + r_t \odot (Uh_{t-1}))$$

Bộ nhớ cuối cùng tại bước thời gian hiện tại t bây giờ được tính toán từ thông tin mới được thêm vào và thông tin quá khứ như sau:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$

Sequence encoder using bidirectional GRU (bộ mã hóa hai chiều GRU): Cho một chuỗi S = $\{s_1, s_2, \dots, s_N\}$, trong đó s_i, i ∈ [1, N], được biểu diễn dưới dạng một vectơ có giá trị thực, chúng tôi sử dụng GRU hai chiều để mã hóa thông tin theo ngữ cảnh của chuỗi . Để làm điều đó, GRU hai chiều chứa GRU tiến $\vec{f}$ và GRU lùi $\overleftarrow{f}$ để đọc chuỗi từ trái sang phải và hướng ngược lại:

$$\overrightarrow{h}_t = \overrightarrow{\text{GRU}}(x_t)$$

$$\overleftarrow{h}_t = \overleftarrow{\text{GRU}}(x_t)$$

Trạng thái tại bước thời gian t được xác định bằng cách nối các trạng thái ẩn tiến và lùi, $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. ht chứa thông tin của cả dãy xoay quanh x_t.

Attention mechanism (cơ chế chú ý): GRU đọc một chuỗi và nén tất cả thông tin vào một vectơ duy nhất. Một số yếu tố có thể dẫn đến mất thông tin vì chúng có những đóng góp khác nhau vào ý nghĩa của chuỗi. Áp dụng cơ chế chú ý giải quyết một phần vấn đề này. Nó cho phép xem qua trình tự ban đầu và tập trung vào các yếu tố thông tin. Để thực hiện điều này, một vectơ ngữ cảnh ct được cắm giữa GRU và vectơ được mã hóa để tính toán phân bố xác suất cho từng phần tử. Công thức toán học được hiển thị dưới đây:

$$\alpha_{ts} = \frac{\exp(\text{score}(h_t, \bar{h}_s))}{\sum_{s'=1}^{N} \text{score}(h_t, \bar{h}'_s)}$$

$$c_t = \sum_{s=1} \alpha_{ts} \bar{h}'_s$$

$$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t])$$

GRU hierarchical attention (chú ý phân cấp GRU): Để xây dựng bộ phân loại, trước tiên chúng tôi áp dụng GRU hai chiều, chú ý ở cả cấp độ câu và cấp độ tài liệu để mã hóa tài liệu thành một vectơ có độ dài cố định. Sau đó, một lớp được kết nối đầy đủ và một softmax được xếp chồng lên nhau để phân loại.

D. Kết quả thí nghiệm

1) Kết quả với các mô hình học máy

Method	Acc	F1		
		1	0	weighted avg
LinearSVC	0.914	0.947	0.775	0.861
LogisticRegression	0.912	0.947	0.748	0.906
SVC	0.904	0.942	0.712	0.895
Bernoulli	0.846	0.896	0.702	0.799
LTSM	0.807	-	-	-
GRU	0.764	-	-	-

Các cài đặt trong những model trên đều sử dụng kỹ thuật trích xuất thông tin CountVector và TfidfVectorizer (max_features = 5000, stop_words = ‘english’) của thư viện Sklearn với pipeline gồm 3 bước:

Bước 1: Sử dụng trích xuất đặc trưng CountVectorizer với các thông số: ngram_range=(1, 4), max_df=0.5, min_df=5.

Bước 2: TfidfTransformer với thông số:norm='l2', sublinear_tf = True, use_idf = False

Bước 3: Sử dụng các model đơn giản của bảng trên

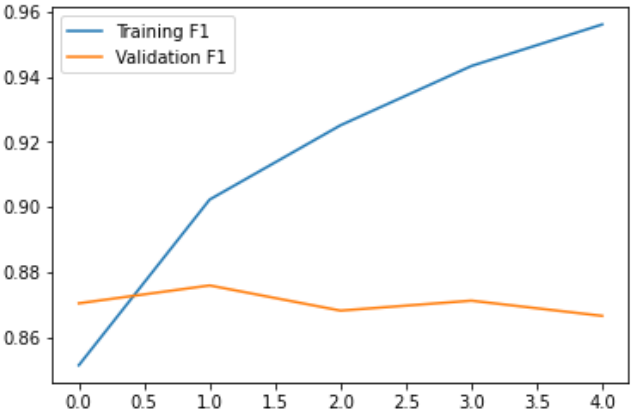
LinearSVC: fit_intercept = True, multi_class='ovr', C=1

LogisticRegression: C= 1, max_iter = 2000, n_jobs = -1

SVC: C= 0.2175, class_weight = None, verbose = True5

BernoulliNB()

2) Thử nghiệm với PhoBert

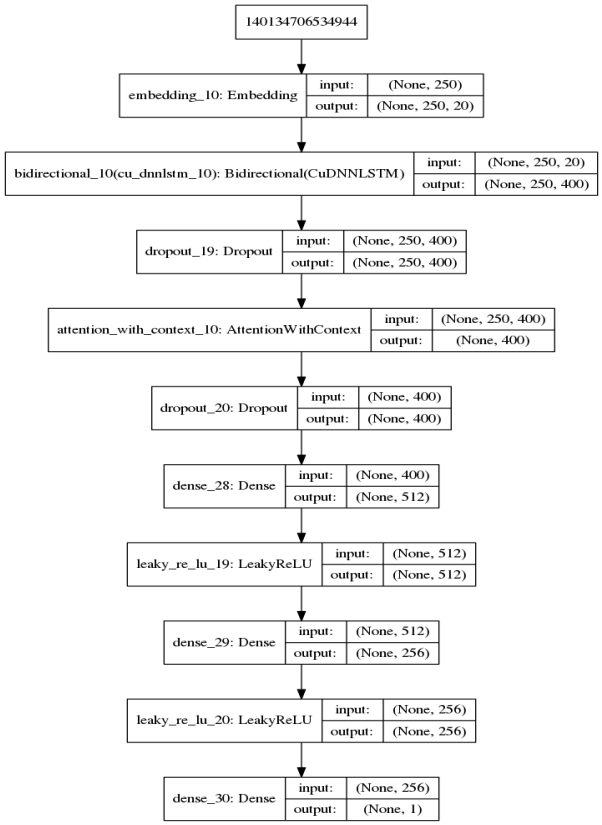


Kết quả trên tập valid: Acc = 0.889, tập test là 0.86

3. Model LSTM và mô hình Attention phân cấp

Sau quá trình chỉnh sửa các thông số trong mô hình, ta được mô hình có kết quả cao nhất trên bộ test như hình dưới.

Mô hình này đưa lại kết quả tốt nhất trên tập valid là 0.8914 và tập test là 0.91809, do đó đây là mô hình tốt nhất của nhóm.



III. KẾT LUẬN

Trong dự án này, nhóm đã thử nghiệm một số mô hình học máy cũng như kiến trúc học sâu về phân tích cảm xúc trong tiếng Việt. Độ chính xác trên tập thử nghiệm đều trên 80%, kết quả cho thấy việc sử dụng các mô hình học sâu sẽ cải thiện độ chính xác cho mô hình. Ngoài ra, ta nhận thấy cơ chế Attention thực sự hiệu quả vượt bậc so với các mô hình khác. Trong tương lai, hy vọng sẽ đạt được kết quả tương đương hoặc tốt hơn trên các tiêu chuẩn phân tích tâm lý khác của người Việt. Tuy kỹ lập trình cũng như áp dụng các mô hình học máy còn rất kém, nhưng trong quá trình hoàn thành bài cuối kỳ, ta đã học được rất nhiều kinh nghiệm trong việc xử lý dữ liệu, chọn các thông số sao cho phù hợp với mô hình đã cho và biết thêm một số mô hình học sâu vô cùng hiệu quả ở thời điểm hiện tại.

TÀI LIỆU THAM KHẢO

[1] [BERT Series] Chương 1. BERT là cái chi chi? - Mì AI (miai.vn)

[2] Khoa học dữ liệu (phamdinhkhanh.github.io)

[3] https://streetcodevn.com/blog/dataset

[4] Attention Is All You Need

[5] https://github.com/VinAIResearch/PhoBERT

[6] https://github.com/vncorenlp/VnCoreNLP

[7] Thư viện làm sạch văn bản https://pypi.org/project/clean-text

[8] Paper Hierarchical Attention Networks for Document Classification https://www.cs.cmu.edu/hovy/papers/16HLT-hierarchical-attention-networks.pdf