ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

CÁCH TIẾP CẬN HIỆN ĐẠI TRONG XỬ LÝ NGÔN NGỮ TỰ NHIÊN

Đề tài

Xây dựng mô hình phân loại cảm xúc (Sentimental Classification)

GVHD: PGS.TS Quản Thành Thơ
SV: Đoàn Thanh Nam - 1813130

TP. HỒ CHÍ MINH, THÁNG 11/2021

Mục lục

1 Giới thiệu bài toán **2**

2 Phân tích tập dữ liệu VLSP và tiền xử lý dữ liệu **2**
2.1 Dữ liệu thô . . . 2
2.2 Tiền xử lý dữ liệu . . . 2
2.3 Phân tích tập dữ liệu đã được tiền xử lý . . . 3
2.4 Định nghĩa các tham số và xây dựng mô hình Word2Vec . . . 4

3 Lựa chọn mô hình **4**
3.1 Mô hình CNN . . . 4
3.2 Mô hình DeepCNN - Reduce Dim . . . 7
3.3 Mô hình LSTM . . . 9
3.4 Mô hình BiLSTM - Reduce Dim . . . 11
3.5 Mô hình CNN kết hợp song song với LSTM . . . 12
3.6 Mô hình CNN tuần tự với LSTM . . . 14
3.7 Mô hình LSTM tuần tự với CNN . . . 15

4 Tổng hợp kết quả và nhận xét **17**

1 Giới thiệu bài toán

Bài toán phân tích cảm xúc thuộc dạng bài toán phân tích ngữ nghĩa văn bản. Vì vậy, ta cần phải xây dựng một mô hình để hiểu được ý nghĩa của câu văn, đoạn văn để quyết định xem câu văn đó hoặc đoạn văn đó mang màu sắc cảm xúc chủ đạo nào. Đây là một vấn đề không mới trong lĩnh vực xử lý ngôn ngữ tự nhiên, tuy nhiên, với tập dữ liệu tiếng Việt và dữ liệu được thu thập từ internet sẽ là một vấn đề không hề đơn giản để giải quyết.

2 Phân tích tập dữ liệu VLSP và tiền xử lý dữ liệu

2.1 Dữ liệu thô

Bộ dữ liệu bao gồm 2 tập huấn luyện (train set) và tập kiểm thử (test set). Dữ liệu trong mỗi tập bao gồm 2 trường:

- Data - là trường chứa các bình luận về một sản phẩm công nghệ với các cảm xúc của người dùng
- Class - là các nhãn đã được gán tương ứng với các trạng thái cảm xúc : -1 (tiêu cực) ; 0 (trung lập) ; 1 (tích cực).

Dữ liệu thô có nhiều từ không có trong từ điển như teencode và các kí hiệu đặc biệt không mang nhiều giá trị ngữ nghĩa. Có tổng cộng 5100 dòng ở tập huấn luyện và 1050 dòng ở tập kiểm thử.

2.2 Tiền xử lý dữ liệu

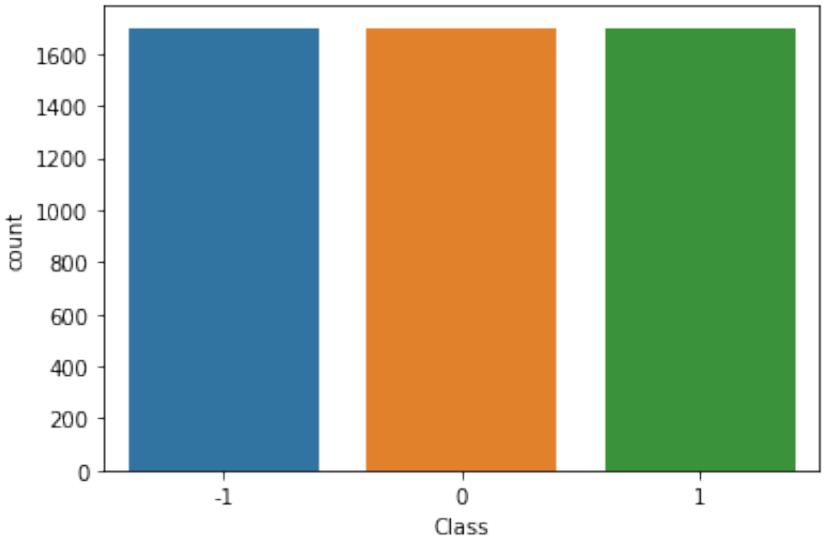
Tiền xử lý dữ liệu giúp cho việc học của mô hình nhanh và chính xác hơn. Công việc tiền xử lý dữ liệu ở văn bản gồm rất nhiều các tác vụ khác nhau. Ở đây, các thao tác chính của em trong việc tiền xử lý dữ liệu Tiếng Việt lần lượt là:

- Loại bỏ kí tự chữ số: các giá trị số thường không mang lại nhiều giá trị ngữ nghĩa trong các câu bình luận nên ta sẽ loại bỏ chúng đi để giúp cho từ điển sau khi xây dựng mang tính chính xác cao hơn
- Tách từ (Tokenizer): khác với tiếng Anh, tiếng Việt có nhiều từ phải đi liền kề nhau mới mang lại ý nghĩa. Do đó sử dụng 1 bộ tách từ có hỗ trợ tiếng Việt là cần thiết. Bộ tách từ em sử dụng là ViTokenizer
- Xóa các kí tự không cần thiết như: @,!,... v.v.
- Xóa các khoảng trắng thừa
- Đưa tất cả chứ in hoa về chữ thường

Sau khi thực hiện xong việc làm sạch dữ liệu và tách từ, ta xây dựng 1 bộ từ điển và đánh chỉ mục cho bộ từ điển bao gồm các từ tìm được trong dữ liệu đã được làm sạch ở trên. Ở đây bộ từ điển của em có 7893 từ và 01 token <OOV> đại diện cho những từ không tìm thấy trong từ điển.

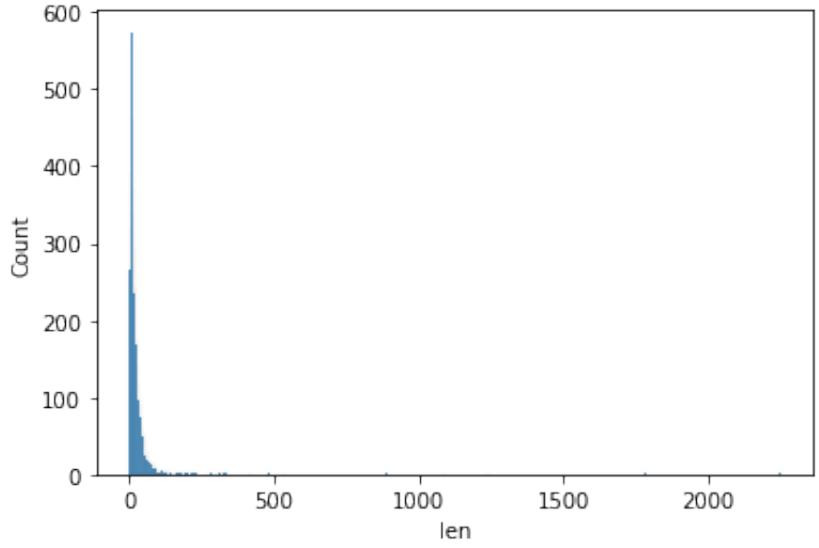
2.3 Phân tích tập dữ liệu đã được tiền xử lý

Đầu tiên nói về sự phân bổ của các lớp cảm xúc trên tập dữ liệu, ta thấy được số lượng phân bố rất cân bằng trên cả 3 trạng thái cảm xúc.



Hình 1: Phân bố dữ liệu

Tuy nhiên bên cạnh sự phân bố khá đồng đều của tập dữ liệu huấn luyện, độ dài các câu bình luận lại rất khác nhau sau khi tiền xử lý.



Hình 2: Độ dài bình luận

Ta thấy rằng các câu từ 500 từ trở xuống chiếm đa số, các câu có số từ lớn hơn chỉ là các

trường hợp đặc biệt. Do đó cần phải xem xét vấn đề chọn độ dài của câu trước khi đưa vào xử lý.

2.4 Định nghĩa các tham số và xây dựng mô hình Word2Vec

Từ những phân tích ở trên. em tiến hành chọn tham số $MAXLEN$ cho 1 câu bình luận lần lượt là 100 và 300. Lý do chọn 100 là vì các bình luận chứa ít hơn 100 từ chiếm 98% tập dữ liệu. Và 300 sẽ gần như lấy hết tất cả các từ trong mọi câu bình luận có ở tập dữ liệu. Việc chọn lựa $MAXLEN$ tốt sẽ giúp ta tiết kiệm được thời gian huấn luyện cũng như tăng độ chính xác của mô hình dự đoán.

Sau khi lựa chọn $MAXLEN$ cho tập dữ liệu, ta tiến hành xây dựng mô hình Word2Vec để biến các câu thành các vector mang nhiều ý nghĩa hơn. Ở đây em sử dụng lại các trọng số đã được train của mô hình Word2Vec CBOW đã được cung cấp.Đầu vào của mô hình Word2Vec hay còn gọi là tầng Embedding sẽ là 1 câu có số từ là $MAXLEN$. Đầu ra của tầng Embedding là một vector 2 chiều với định dạng ($MAXLEN$;400). Những từ không có trong mô hình Word2Vec đã cung cấp thì vector sẽ được gán giá trị ngẫu nhiên.

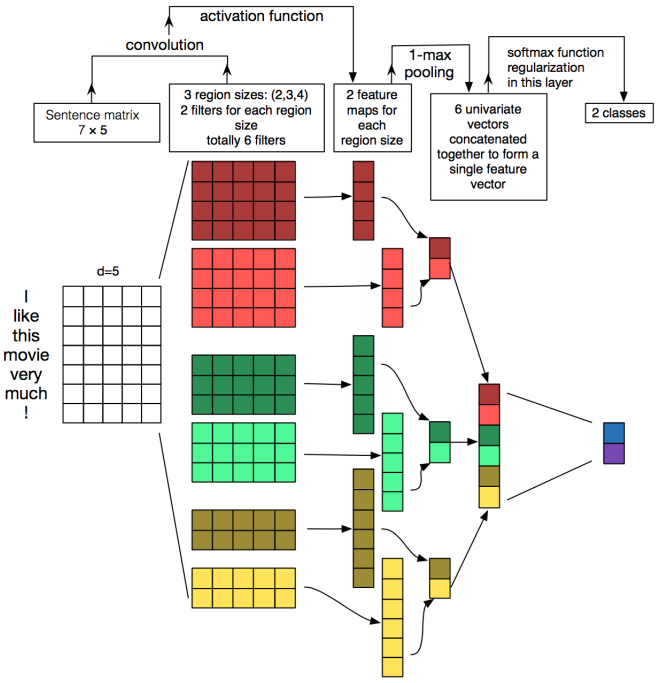
3 Lựa chọn mô hình

Để xây dựng một mô hình có độ chính xác tốt để phân loại tập dữ liệu trên, em đã thử nghiệm và chọn ra các mô hình tốt nhất trong suốt quá trình xây dựng và điều chỉnh tham số.

3.1 Mô hình CNN

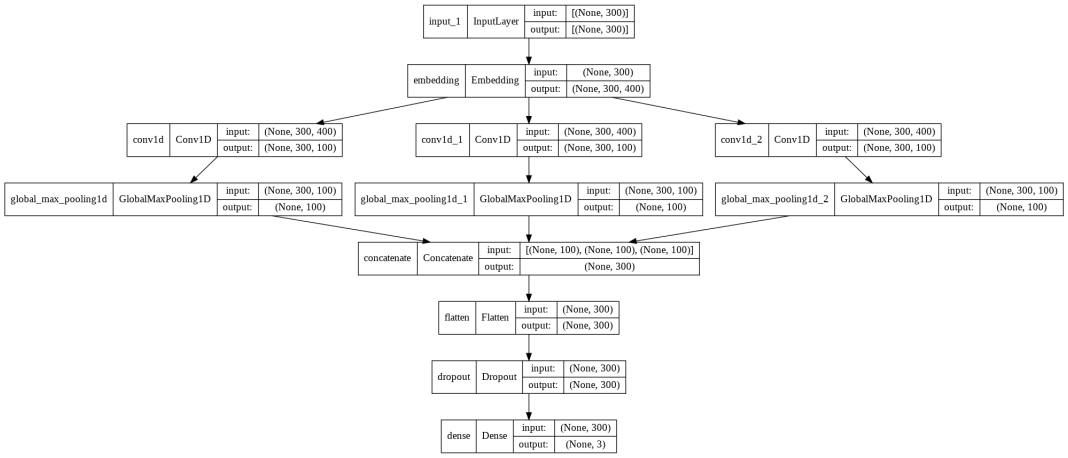
Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Thông thường CNNs được xử dụng nhiều trong các bài toán xử lý ảnh hay thị giác máy tính. Thế nhưng bên cạnh đó, việc áp dụng CNNs vào xử lý ngôn ngữ tự nhiên (NLP) cũng đem lại nhiều kết quả đáng chú ý.

CNNs phù hợp với việc trích xuất các đặc trưng xung quang ảnh, vậy để ứng dụng vào NLP ta áp dụng như thế nào? Thay vì input đầu vào là các điểm ảnh trong Computer Vision, đầu vào của phân tích ngôn ngữ tự nhiên là các mệnh đề, các văn bản được biểu diễn như một ma trận. Mỗi dòng của một ma trận tương ứng với một mã, đa phần đó là một từ, nhưng nó cũng có thể là một kí tự. Mối hàng chính là một vector đại diện cho một từ. Thông thường các vector word này được trình bày ở mức thấp như dạng word2vec hay Glove, nhưng nó cũng có thể là một vector với việc các từ sẽ được đánh chỉ số thuộc một bộ từ vựng. Và khi chung ta sử dụng nhiều chiều ánh xạ một câu thì nó sẽ tạo cho chúng ta một ma trận nhiều chiều, như thế nó đã được biến thành input image đầu vào.



Hình 3: CNN trong NLP

Cụ thể trong bài toán của chúng ta, dữ liệu đầu vào (1 câu bình luận) sau khi đi qua tầng Embedding ta được 1 vector 2 chiều có định dạng ($MAXLEN$;400). Sử dụng các cửa sổ trượt với kernel size lần lượt là (3;400), (4;400), (5;400) với số lượng filter là 100 mỗi cửa sổ trượt. Sau đó ứng với mỗi filter ở trên, ta cho qua tầng MaxPooling1D để lấy max của mỗi filter. Tiếp tục cho qua tầng concatenate để nối các vector lại với nhau. Cuối cùng vector đặc trưng được cho qua tầng Dropout để tránh overfitting và cho qua tầng Dense gồm 3 units để phân loại. Ở mô hình đầu tiên, em chọn $MAXLEN = 300$



Hình 4: Mô hình CNN - 300

Số lượng tham số

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================================
input_1 (InputLayer)         [(None, 300)]             0           []

embedding (Embedding)        (None, 300, 400)          3157600     ['input_1[0][0]']

conv1d (Conv1D)              (None, 300, 100)          120100      ['embedding[0][0]']

conv1d_1 (Conv1D)            (None, 300, 100)          160100      ['embedding[0][0]']

conv1d_2 (Conv1D)            (None, 300, 100)          200100      ['embedding[0][0]']

global_max_pooling1d (GlobalMa  (None, 100)            0           ['conv1d[0][0]']
xPooling1D)

global_max_pooling1d_1 (Global  (None, 100)            0           ['conv1d_1[0][0]']
MaxPooling1D)

global_max_pooling1d_2 (Global  (None, 100)            0           ['conv1d_2[0][0]']
MaxPooling1D)

concatenate (Concatenate)    (None, 300)               0           ['global_max_pooling1d[0][0]',
                                                                   'global_max_pooling1d_1[0][0]',
                                                                   'global_max_pooling1d_2[0][0]']

flatten (Flatten)            (None, 300)               0           ['concatenate[0][0]']

dropout (Dropout)            (None, 300)               0           ['flatten[0][0]']

dense (Dense)                (None, 3)                 903         ['dropout[0][0]']

==================================================================================================
Total params: 3,638,803
Trainable params: 3,638,803
Non-trainable params: 0
```

Hình 5: Tham số mô hình CNN - 300

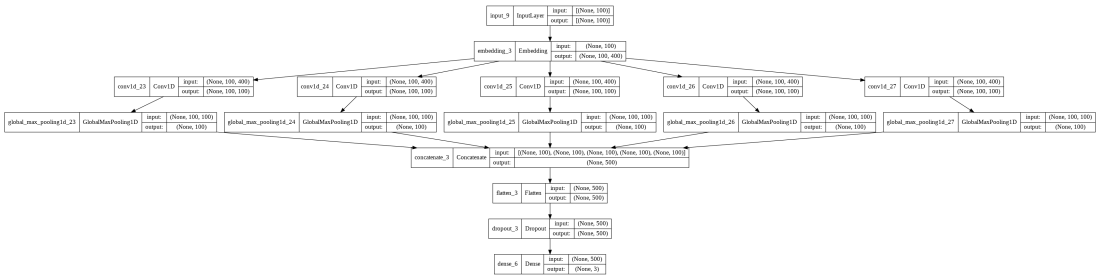
- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Tầng CNN với kernel size là 3, 100 filters sẽ có số lượng tham số là $100*(3*400+1) = 120100$

với 1 là số lượng bias

- Tầng CNN với kernel size là 4, 100 filters sẽ có số lượng tham số là $100*(4*400+1) = 160100$
- Tầng CNN với kernel size là 5, 100 filters sẽ có số lượng tham số là $100*(5*400+1) = 200100$
- Các tầng MaxPooling, Concat và Dropout không có tham số học tập
- Sau khi Concat ta được vector 300 chiều, qua tầng Fully Connected (Dense) với 3 units ta có số lượng tham số là $3 * (300 + 1) = 903$ với 1 là số lượng bias
- Tổng cộng ta có $3157600 + 120100 + 160100 + 200100 + 903 = 3638803$ tham số cần học

3.2 Mô hình DeepCNN - Reduce Dim

Tương tự như mô hình CNNs ở trên, ở đây em cải tiến mô hình bằng cách cho thêm 2 lớp CNN với kernel size là 6 và 7. Bên cạnh đó em giảm $MAXLEN$ xuống 100 để xem xét độ hiệu quả.



Hình 6: Mô hình CNN - 100

Số lượng tham số

Layer (type)	Output Shape	Param #	Connected to
input_9 (InputLayer)	[(None, 100)]	0	[]
embedding_3 (Embedding)	(None, 100, 400)	3157600	['input_9[0][0]']
conv1d_23 (Conv1D)	(None, 100, 100)	120100	['embedding_3[0][0]']
conv1d_24 (Conv1D)	(None, 100, 100)	160100	['embedding_3[0][0]']
conv1d_25 (Conv1D)	(None, 100, 100)	200100	['embedding_3[0][0]']
conv1d_26 (Conv1D)	(None, 100, 100)	240100	['embedding_3[0][0]']
conv1d_27 (Conv1D)	(None, 100, 100)	280100	['embedding_3[0][0]']
global_max_pooling1d_23 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_23[0][0]']
global_max_pooling1d_24 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_24[0][0]']
global_max_pooling1d_25 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_25[0][0]']
global_max_pooling1d_26 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_26[0][0]']
global_max_pooling1d_27 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_27[0][0]']
concatenate_3 (Concatenate)	(None, 500)	0	['global_max_pooling1d_23[0][0]', 'global_max_pooling1d_24[0][0]', 'global_max_pooling1d_25[0][0]', 'global_max_pooling1d_26[0][0]', 'global_max_pooling1d_27[0][0]']
flatten_3 (Flatten)	(None, 500)	0	['concatenate_3[0][0]']
dropout_3 (Dropout)	(None, 500)	0	['flatten_3[0][0]']
dense_6 (Dense)	(None, 3)	1503	['dropout_3[0][0]']

Total params: 4,159,603

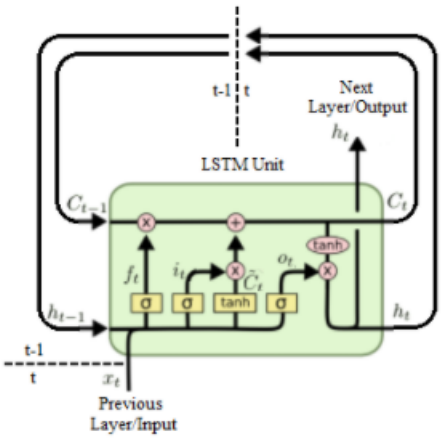
Hình 7: Tham số mô hình CNN - 100

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Tầng CNN với kernel size là 3, 100 filters sẽ có số lượng tham số là $100*(3*400+1) = 120100$ với 1 là số lượng bias
- Tầng CNN với kernel size là 4, 100 filters sẽ có số lượng tham số là $100*(4*400+1) = 160100$
- Tầng CNN với kernel size là 5, 100 filters sẽ có số lượng tham số là $100*(5*400+1) = 200100$
- Tầng CNN với kernel size là 6, 100 filters sẽ có số lượng tham số là $100*(6*400+1) = 240100$
- Tầng CNN với kernel size là 7, 100 filters sẽ có số lượng tham số là $100*(7*400+1) = 280100$
- Các tầng MaxPooling, Concat và Dropout không có tham số học tập
- Sau khi Concat ta được vector 300 chiều, qua tầng Fully Connected (Dense) với 3 units ta có số lượng tham số là $3 * (500 + 1) = 1503$ với 1 là số lượng bias
- Tổng cộng ta có $3157600 + 120100 + 160100 + 200100 + 240100 + 280100 + 1503 = 4159603$ tham số cần học

3.3 Mô hình LSTM

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 giới thiệu bởi Hochreiter Schmidhuber (1997), và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

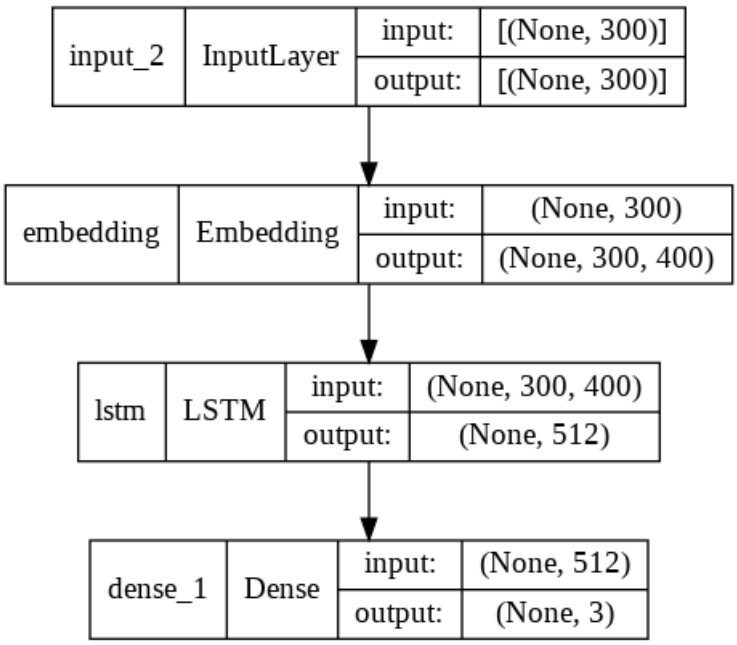
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Hình 8: Kiến trúc của LSTM

Ở đây đầu vào là một câu có độ dài $MAXLEN$ được chọn là 300. Dữ liệu đầu vào được đưa qua tầng Embedding để cho đầu ra là 1 vector có dạng ($MAXLEN$;400). Sau đó vector này được cho qua kiến trúc LSTM với số lượng tầng ẩn là 512, dropout rate là 0.5. Đầu ra cuối cùng của LSTM là một vector 512 chiều. Cho vector này qua Fully Connected (Dense) với units là 3 để phân loại.



Hình 9: Mô hình LSTM - 300

Số lượng tham số

Layer (type)	Output Shape	Param #
input_5 (InputLayer)	[(None, 50)]	0
embedding_1 (Embedding)	(None, 50, 400)	3157600
lstm_4 (LSTM)	(None, 512)	1869824
dense_4 (Dense)	(None, 3)	1539

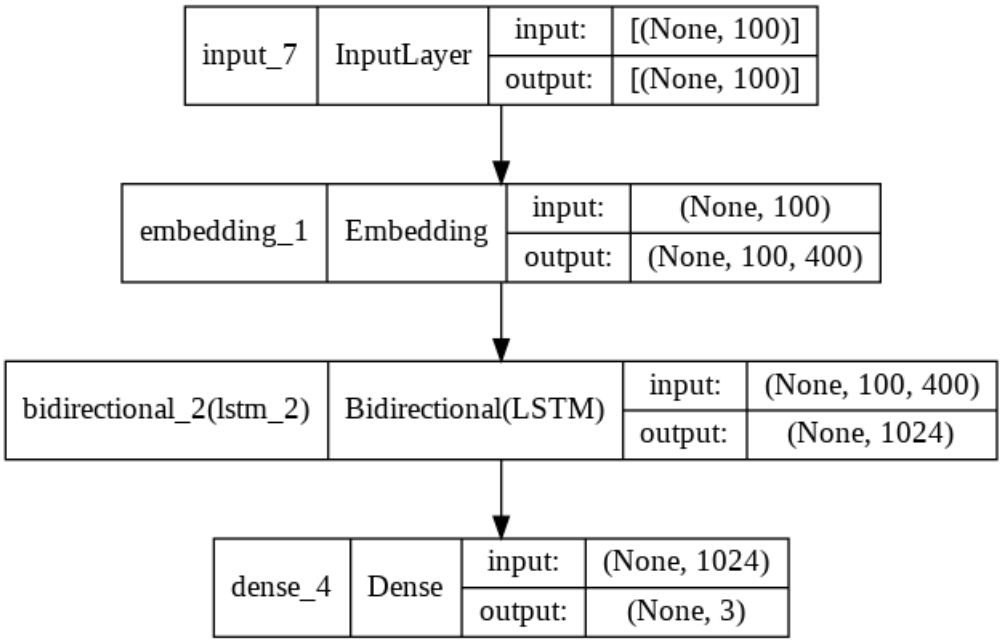
Total params: 5,028,963

Hình 10: Tham số mô hình LSTM - 300

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Ở hình trên ta thấy với mỗi cổng cần $400 * 512 + 512 * 512 + 1 * 512 = 467456$ tham số học tập
- Có tất cả 4 cổng với hệ số W cần học vậy có $4 * 467456 = 1869824$ tham số học tập
- Tầng FC có $3 * (512 + 1) = 1539$ tham số học tập
- Tổng cộng ta có $3157600 + 1869824 + 1539 = 5028963$ tham số cần học

3.4 Mô hình BiLSTM - Reduce Dim

Em thay đổi mô hình LSTM trên bằng cách cho LSTM học 2 chiều (Bidirectional) và áp dụng $MAXLEN = 100$



Hình 11: Mô hình BiLSTM - 100

Số lượng tham số

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, 100)]             0

embedding_1 (Embedding)      (None, 100, 400)          3157600

bidirectional_2 (Bidirectio  (None, 1024)              3739648
nal)

dense_4 (Dense)              (None, 3)                 3075

=================================================================
Total params: 6,900,323
```

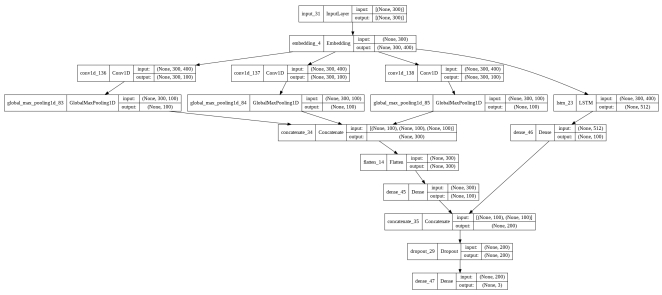
Hình 12: Tham số mô hình BiLSTM - 100

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Ở hình trên ta thấy với mỗi cổng cần $400 * 512 + 512 * 512 + 1 * 512 = 467456$ tham số học tập

- Có tất cả 4 cổng với hệ số W cần học vậy có $4 * 467456 = 1869824$ tham số học tập
- Do đây là mô hình Bidirectional nên tổng tham số học tập là $1869824 * 2 = 3739648$
- Tầng FC có $3 * (512 * 2 + 1) = 3075$ tham số học tập
- Tổng cộng ta có $3157600 + 3739648 + 3075 = 6900323$ tham số cần học

3.5 Mô hình CNN kết hợp song song với LSTM

Một cách làm khá thú vị là kết hợp giữa CNN và LSTM. CNN có khả năng học các vùng đặc trưng của câu văn trong khi LSTM lưu trữ được thông tin liên tiếp nhau của một câu. Do đó khả năng cao khi kết hợp lại ta sẽ được một mô hình học đầy đủ hơn các thông tin của câu.



Hình 13: Mô hình CNN song song LSTM - 300

Em chọn $MAXLEN = 300$ và sử dụng 3 cửa sổ với kernel size lần lượt là 3,4,5. Tất cả cửa sổ trượt sau khi concat được đi qua 1 tầng Fully Connected với số units là 100. Song song đó, sau khi qua tầng Embedding thì dữ liệu được cho đi qua tầng LSTM, cuối cùng là qua 1 tầng Fully Connected với số units là 100. Sau đó ta Concat cả 2 tầng FC 100 units lại và tiến hành cho qua Dropout. Cuối cùng là tầng Dense với units là 3 để phân loại.

Số lượng tham số

Layer (type)	Output Shape	Param #	Connected to
input_31 (InputLayer)	[(None, 300)]	0	[]
embedding_4 (Embedding)	(None, 300, 400)	3157600	['input_31[0][0]']
conv1d_136 (Conv1D)	(None, 300, 100)	120100	['embedding_4[6][0]']
conv1d_137 (Conv1D)	(None, 300, 100)	160100	['embedding_4[6][0]']
conv1d_138 (Conv1D)	(None, 300, 100)	200100	['embedding_4[6][0]']
global_max_pooling1d_83 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_136[0][0]']
global_max_pooling1d_84 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_137[0][0]']
global_max_pooling1d_85 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_138[0][0]']
concatenate_34 (Concatenate)	(None, 300)	0	['global_max_pooling1d_83[0][0]', 'global_max_pooling1d_84[0][0]', 'global_max_pooling1d_85[0][0]']
flatten_14 (Flatten)	(None, 300)	0	['concatenate_34[0][0]']
lstm_23 (LSTM)	(None, 512)	1869824	['embedding_4[6][0]']
dense_45 (Dense)	(None, 100)	30100	['flatten_14[0][0]']
dense_46 (Dense)	(None, 100)	51300	['lstm_23[0][0]']
concatenate_35 (Concatenate)	(None, 200)	0	['dense_45[0][0]', 'dense_46[0][0]']
dropout_29 (Dropout)	(None, 200)	0	['concatenate_35[0][0]']
dense_47 (Dense)	(None, 3)	603	['dropout_29[0][0]']

Total params: 5,589,727

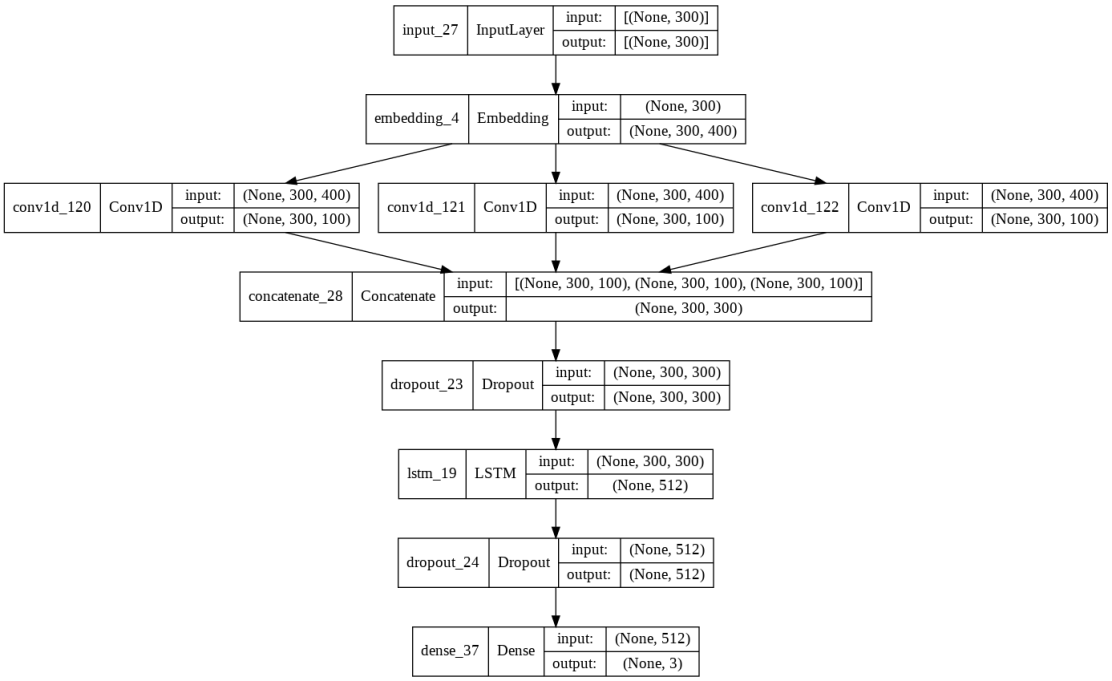
Hình 14: Tham số mô hình CNN song song LSTM - 300

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Tầng CNN với kernel size là 3, 100 filters sẽ có số lượng tham số là $100*(3*400+1) = 120100$ với 1 là số lượng bias
- Tầng CNN với kernel size là 4, 100 filters sẽ có số lượng tham số là $100*(4*400+1) = 160100$
- Tầng CNN với kernel size là 5, 100 filters sẽ có số lượng tham số là $100*(5*400+1) = 200100$
- Như đã phân tích ở trên, tầng LSTM cần 1869824 tham số học tập
- Tầng FC với CNN có $100 * (300 + 1) = 30100$ tham số học tập
- Tầng FC với CNN có $100 * (512 + 1) = 51300$ tham số học tập

- Tầng FC cuối cùng có $3 * (200 + 1) = 603$ tham số học tập
- Tổng cộng ta có $3157600 + 120100 + 160100 + 200100 + 30100 + 51300 + 603 + 1869824 = 5589727$ tham số cần học

3.6 Mô hình CNN tuần tự với LSTM

Bên cạnh cách kết hợp song song, ta có thể cho các tầng CNN học được các đặc trưng của từng khối từ lân cận rồi sau đó cho qua tầng LSTM để hiểu được sự liên kết của các khối từ với nhau



Hình 15: Mô hình CNN tuần tự LSTM - 300

Em chọn $MAXLEN = 300$ và sử dụng 3 cửa sổ với kernel size lần lượt là 3,4,5. Tất cả cửa sổ trượt sau khi concat được đi qua 1 tầng Dropout. Sau đó dữ liệu được cho đi qua tầng LSTM. Tiếp tục cho qua tầng Dropout. Cuối cùng là tầng Dense với units là 3 để phân loại.

Số lượng tham số

Layer (type)	Output Shape	Param #	Connected to
input_32 (InputLayer)	[(None, 300)]	0	[]
embedding_4 (Embedding)	(None, 300, 400)	3157600	['input_32[0][0]']
conv1d_139 (Conv1D)	(None, 300, 100)	120100	['embedding_4[7][0]']
conv1d_140 (Conv1D)	(None, 300, 100)	160100	['embedding_4[7][0]']
conv1d_141 (Conv1D)	(None, 300, 100)	200100	['embedding_4[7][0]']
concatenate_36 (Concatenate)	(None, 300, 300)	0	['conv1d_139[0][0]', 'conv1d_140[0][0]', 'conv1d_141[0][0]']
lstm_24 (LSTM)	(None, 512)	1665024	['concatenate_36[0][0]']
dropout_30 (Dropout)	(None, 512)	0	['lstm_24[0][0]']
dense_48 (Dense)	(None, 3)	1539	['dropout_30[0][0]']

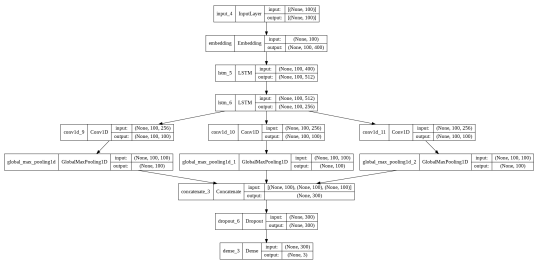
Total params: 5,304,463

Hình 16: Tham số mô hình CNN tuần tự LSTM - 300

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số lượng tham số ở tầng này là $400 * 7984 = 3157600$
- Tầng CNN với kernel size là 3, 100 filters sẽ có số lượng tham số là $100*(3*400+1) = 120100$ với 1 là số lượng bias
- Tầng CNN với kernel size là 4, 100 filters sẽ có số lượng tham số là $100*(4*400+1) = 160100$
- Tầng CNN với kernel size là 5, 100 filters sẽ có số lượng tham số là $100*(5*400+1) = 200100$
- Như đã phân tích ở trên, thay vì chiều của input là 400, ở đây ta chỉ có 300 chiều cho input, như vậy tầng LSTM cần $4 * (300 * 512 + 512 * 512 + 1 * 512) = 1665024$ tham số học tập
- Tầng FC cuối cùng có $3 * (512 + 1) = 1539$ tham số học tập
- Tổng cộng ta có $3157600 + 120100 + 160100 + 200100 + 1665024 + 1539 = 5304463$ tham số cần học

3.7 Mô hình LSTM tuần tự với CNN

Em cũng có thử ngược lại khi ta cho dữ liệu qua tầng LSTM trước rồi mới trích xuất đặc trưng bằng CNN.



Hình 17: Mô hình LSTM tuần tự CNN - 100

Em chọn $MAXLEN = 100$ và cho dữ liệu đi qua các 2 tầng LSTM liên tiếp nhau. Sau đó CNN được áp dụng với kernel size là 3,4,5 và 100 filters. Concat tất cả lại và cho đi qua Dropout. Cuối cùng là tầng Fully Connected với units là 3 để phân loại.

Số lượng tham số

Layer (type)	Output Shape	Param #	Connected to
input_4 (InputLayer)	[(None, 100)]	0	[]
embedding (Embedding)	(None, 100, 400)	3157600	['input_4[0][0]']
lstm_5 (LSTM)	(None, 100, 512)	1869824	['embedding[3][0]']
lstm_6 (LSTM)	(None, 100, 256)	787456	['lstm_5[0][0]']
conv1d_9 (Conv1D)	(None, 100, 100)	76900	['lstm_6[0][0]']
conv1d_10 (Conv1D)	(None, 100, 100)	102500	['lstm_6[0][0]']
conv1d_11 (Conv1D)	(None, 100, 100)	128100	['lstm_6[0][0]']
global_max_pooling1d (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_9[0][0]']
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_10[0][0]']
global_max_pooling1d_2 (GlobalMaxPooling1D)	(None, 100)	0	['conv1d_11[0][0]']
concatenate_3 (Concatenate)	(None, 300)	0	['global_max_pooling1d[0][0]', 'global_max_pooling1d_1[0][0]', 'global_max_pooling1d_2[0][0]']
dropout_6 (Dropout)	(None, 300)	0	['concatenate_3[0][0]']
dense_3 (Dense)	(None, 3)	903	['dropout_6[0][0]']

Total params: 6,123,283

Hình 18: Tham số mô hình LSTM tuần tự CNN - 100

- Ở tầng Embedding, mỗi từ sẽ là 1 vector 400 chiều. Số từ trong từ điển là 7894. Do đó số

lượng tham số ở tầng này là $400 * 7984 = 3157600$

- Tầng LSTM đầu tiên ta có $4 * (400 * 512 + 512 * 512 + 1 * 512) = 1869824$ tham số
- Tầng LSTM thứ hai ta có $4 * (512 * 256 + 256 * 256 + 1 * 256) = 787456$ tham số
- Tầng CNN với kernel size là 3, 100 filters sẽ có số lượng tham số là $100*(3*256+1) = 76900$ với 1 là số lượng bias
- Tầng CNN với kernel size là 4, 100 filters sẽ có số lượng tham số là $100*(4*256+1) = 102500$
- Tầng CNN với kernel size là 5, 100 filters sẽ có số lượng tham số là $100*(5*256+1) = 128100$
- Tầng FC cuối cùng có $3 * (300 + 1) = 903$ tham số học tập
- Tổng cộng ta có $3157600 + 1869824 + 787456 + 76900 + 102500 + 128100 + 903 = 6123283$ tham số cần học

Bên cạnh các mô hình đã nêu, em cũng đã thử giảm chiều toàn bô các mô hình để thử nghiệm. Kết quả sẽ được tổng hợp ở phần sau.

4 Tổng hợp kết quả và nhận xét

Sau đây là bảng tổng hợp kết quả sau khi chạy các mô hình đã diễn giải ở trên.

Mô hình	*Thời gian train 1 epoch*	*Độ chính xác*
CNN	**3s**	69.43%
Deep CNN – Reduce Dim	**3s**	**69.71%**
LSTM	39s	69.14%
BiLSTM – Reduce Dim	34s	68.57%
CNN song song LSTM	51s	67.43%
CNN tuần tự LSTM	51s	66.19%
CNN song song LSTM – Reduce Dim	20s	67.90%
CNN tuần tự LSTM – Reduce Dim	18s	64.95%
LSTM tuần tự CNN – Reduce Dim	34s	68.86%
Deep CNN – Reduce Dim – New CBOW	3s	69.52%
LSTM tuần tự CNN – Reduce Dim – New CBOW	33s	68.48%

Hình 19: Bảng tổng hợp kết quả

Bên cạnh các mô hình đã nêu, em đã train lại Word2Vec CBOW bằng bộ dữ liệu sau khi tiền xử lý. Chọn 2 mô hình tốt nhất khác loại để trên, ta thu được kết quả gần giống tương đối so với bộ Word2Vec CBOW đã cung cấp.

Có thể thấy rằng mô hình CNN mà em phát triển lại đang có kết quả cao nhất, tuy nhiên độ chính xác của các mô hình không quá chênh lệch. Bên cạnh đó, vấn đề về thời gian train cũng đã có thể giải quyết ít nhiều khi ta thực hiện giảm chiều đầu vào.

Về độ chính xác xấp xỉ 70%, em có một số nhận xét sau đây:

- Đây là một bộ dữ liệu khá ít, vì vậy việc huấn luyện khó đạt được độ chính xác cao. Cần phải cung cấp thêm dữ liệu cho các mô hình phức tạp như mô hình kết hợp CNN và LSTM.

- Dữ liệu bên cạnh đó cũng được gán nhãn khá nhập nhằng so với các từ được trích xuất, có một số câu mang tính chất trung lập nhưng giọng văn lại hoàn toàn tiêu cực, hay một số câu toàn những từ tích cực lại được gán nhãn tiêu cực vì câu đó mang hàm ý khá nhiều.
- Ta có thể thấy được rằng dù có một số mô hình khá phức tạp lại không thể đạt được hiệu suaasrt tốt hơn các mô hình đơn giản. Lý do thì theo chủ quan em nghĩ là do dữ liệu chưa đủ lớn để các mô hình này phát huy tác dụng.
- CNN vẫn thống trị bảng xếp hạng như inital code đã đạt được.

Tài liệu

[1] Slides bài giảng trong môn học NLP của thầy Quản Thành Thơ