ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

DEEP LEARNING TRONG KHOA HỌC DỮ LIỆU
LAB 5

MẠNG NEURAL HỒI QUY (RNN)

Họ và tên : Lưu Quang Tiến Hoàng
MSSV : 20521342
Lớp : DS201.N11

Mục tiêu

- Ôn tập kiến thức cơ bản về mô hình mạng neural hồi quy (RNN).
- Biết cách xây dựng mô hình mạng neural hồi quy cơ bản và sử dụng một số mô hình mạng neural hồi quy nổi tiếng.
- Áp dụng các mô hình trên vào bài toán phân tích cảm xúc.

1, BỘ DỮ LIỆU

Phân tích cảm xúc (Sentiment Analysis) là một kỹ thuật xác định và phân tích cảm xúc của người dùng dựa vào một đoạn văn hoặc một câu văn.

Bộ dữ liệu IMDB Movies Reviews được xây dựng nhằm phục vụ cho bài toán phân tích cảm xúc của người dùng đối với các bộ phim trên IMDB dựa vào các bình luận (review) của họ.

Có hai nhãn cảm xúc chính trong bộ dữ liệu:

- **Tích cực (positive)** - ký hiệu là 1.
- **Tiêu cực (negative)** - ký hiệu là 0.

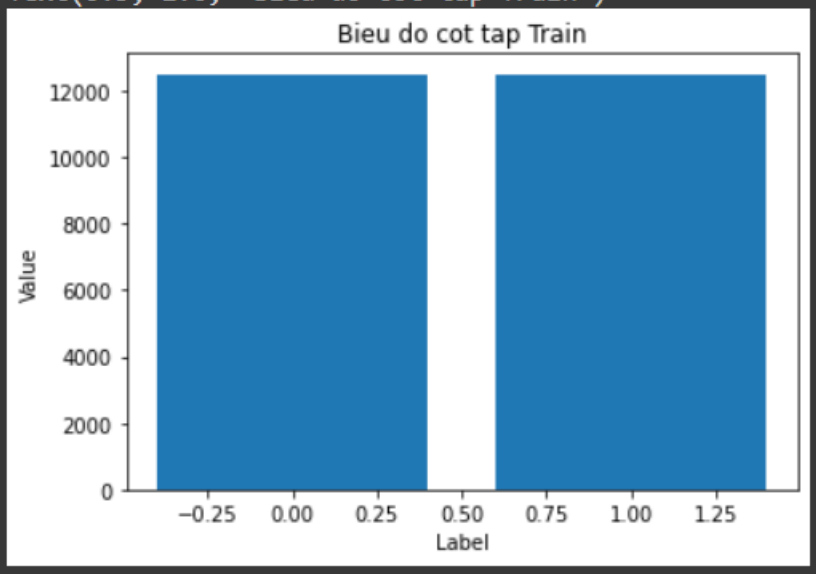
(?) Cho biết số lượng phần tử của tập train và tập test?

```
[80] len(training_targets)

     25000

[81] len(testing_targets)

     25000
```

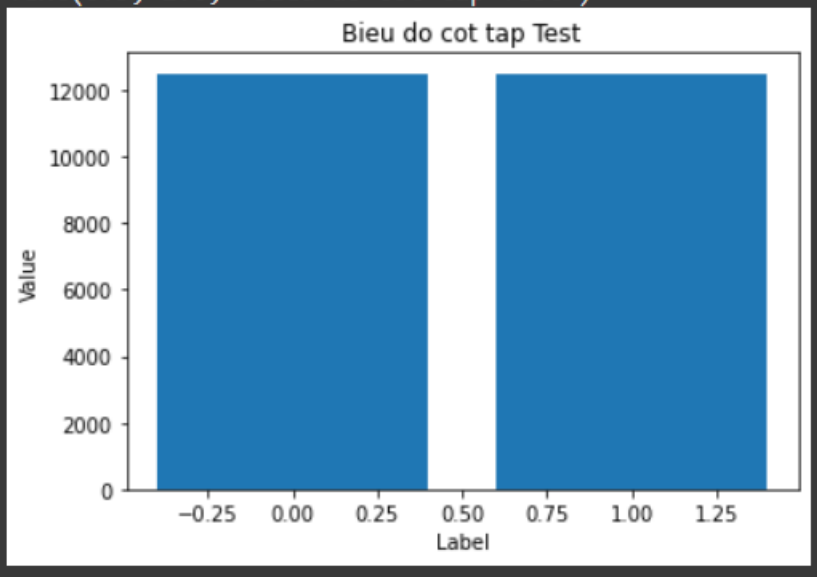
- Tập train: 25000
- Tập test: 25000

(?) Vẽ biểu đồ cột thể hiện số lượng nhãn của mỗi tập dữ liệu và biểu đồ tròn thể hiện tỉ lệ phân bố nhãn của mỗi tập dữ liệu?

```
[90] plt.bar(list_labels.keys(),list_labels.values())
     plt.xlabel('Label')
     plt.ylabel('Value')
     plt.title("Bieu do cot tap Train")

     Text(0.5, 1.0, 'Bieu do cot tap Train')
```



```
[92] plt.bar(list_labels1.keys(),list_labels1.values())
     plt.xlabel('Label')
     plt.ylabel('Value')
     plt.title("Bieu do cot tap Test")

     Text(0.5, 1.0, 'Bieu do cot tap Test')
```



- Mỗi tập có 2 nhãn 0 và 1
- Mỗi tập nhãn phân bố đồng đều tỉ lệ 1:1 đều bằng 12500 phần tử

(?) In ra 5 câuđầu tiên trong tập train và tập test kèm nhãn tương ứng?

```
for i in range(5):
  print(i,'-->',training_data[i],'-->',training_targets[i])
```

```
0 --> [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385,
1 --> [1, 194, 1153, 194, 8255, 78, 228, 5, 6, 1463, 4369, 5012, 134, 26, 4, 715, 8, 118, 1634, 14, 394, 20, 13, 119, 954, 189, 102, 5, 207, 110, 3103, 21, 14, 69, 188, 8, 30, 23, 7, 4
2 --> [1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 5974, 54, 61, 369, 13, 71, 149, 14, 22, 112, 4, 2401, 311, 12, 16, 3711, 33, 75, 43, 1829, 296, 4, 86, 320, 35, 534, 19, 263, 4821, 1
3 --> [1, 4, 2, 2, 33, 2804, 4, 2040, 432, 111, 153, 103, 4, 1494, 13, 70, 131, 67, 11, 61, 2, 744, 35, 3715, 761, 61, 5766, 452, 9214, 4, 985, 7, 2, 59, 166, 4, 105, 216, 1239, 41, 17
4 --> [1, 249, 1323, 7, 61, 113, 10, 10, 13, 1637, 14, 20, 56, 33, 2401, 18, 457, 88, 13, 2626, 1400, 45, 3171, 13, 70, 79, 49, 706, 919, 13, 16, 355, 340, 355, 1696, 96, 143, 4, 22, 3
```

```
[95] for i in range(5):
       print(i,'-->',testing_data[i],'-->',testing_targets[i])
```

```
0 --> [1, 591, 202, 14, 31, 6, 717, 10, 10, 2, 2, 5, 4, 360, 7, 4, 177, 5760, 394, 354, 4, 123, 9, 1035, 1035, 1035, 10, 10, 13, 92, 124, 89, 488, 7944, 100, 28, 1668, 14, 31, 23, 27,
1 --> [1, 14, 22, 3443, 6, 176, 7, 5063, 88, 12, 2679, 23, 1310, 5, 109, 943, 4, 114, 9, 55, 606, 5, 111, 7, 4, 139, 193, 273, 23, 4, 172, 270, 11, 7216, 2, 4, 8463, 2801, 109, 1603, 2
2 --> [1, 111, 748, 4368, 1133, 2, 2, 4, 87, 1551, 1262, 7, 31, 318, 9459, 7, 4, 498, 5076, 748, 63, 29, 5161, 220, 686, 2, 5, 17, 12, 575, 220, 2507, 17, 6, 185, 132, 2, 16, 53, 928,
3 --> [1, 13, 1228, 119, 14, 552, 7, 20, 190, 14, 58, 13, 258, 546, 1786, 8, 1968, 4, 268, 237, 13, 191, 81, 15, 13, 80, 43, 3824, 44, 12, 14, 16, 427, 3192, 4, 183, 15, 593, 19, 4, 35
4 --> [1, 40, 49, 85, 84, 1040, 146, 6, 783, 254, 4386, 337, 5, 13, 447, 14, 500, 10, 10, 14, 500, 517, 1076, 357, 21, 1684, 72, 45, 290, 12, 17, 515, 17, 25, 380, 129, 3305, 4, 2191,
```

(?) Cho biết 5 từ đầu tiên của tập từ vựng?

```
index
```

```
{'fawn': 34701,
 'tsukino': 52006,
 'nunnery': 52007,
 'sonja': 16816,
 'vani': 63951,
```

(?) Cho biết tập từ vựng của bộ dữ liệu có bao nhiêu từ?

```
[180] len(index)
```

```
88584
```

(?) Cho biết độ dài của câu văn vừa được decode?

```
[101] len(decoded)
```

```
1117
```

(?) Decode 5 câu tiếp theo trong tập train?

```
for i in range(1,6):
  reverse_index = dict([(value,key)for (key,value) in index.items()])
  decoded = ' '.join([reverse_index.get(i,"#") for i in training_data[i]])
  print(decoded)
```

```
the thought solid thought senator do making to is spot nomination assumed while he of jack in where picked as getting on was did hands fact characters to always life thrillers not as m
the as there in at by br of sure many br of proving no only women was than doesn't as you never of hat night that with ignored they bad out superman plays of how star so stories film c
the of and and they halfway of identity went plot actors watch of share was well these can this only and ten so failing feels only novak killer theo of bill br and would find of films
the sure themes br only acting i i was favourite as on she they hat but already most was scares minor if flash was well also good 8 older was with enjoy used enjoy phone too i'm of you
the effort still been that usually makes for of finished sucking ended and an because before if just though something know novel female i i slowly lot of above and with connect in of s
```

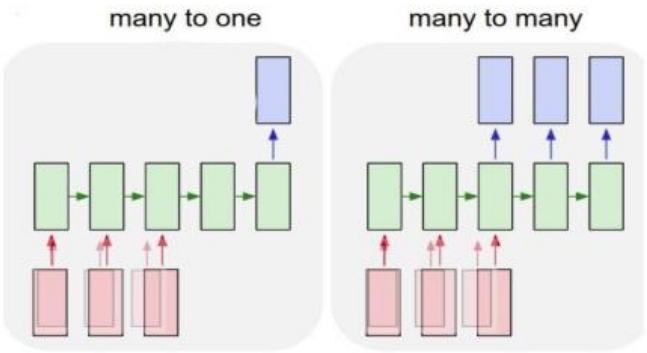
2. XÂY DỰNG MÔ HÌNH

SimpleRNN

Là layer gồm các units kết nối với nhau theo phương pháp Fully-connected.

Các tham số quan trọng bao gồm:

- **units:** số lượng units trong layer.
- **activation:** hàm kích hoạt.
- **dropout:** sử dụng dropount hay không.
- **return_sequences:** trả về output là một sequence (đối với Many-to-many) hoặc là một giá trị (Many-to-one).



- **recurrent_initializer**, **kernel_initializer** và **bias_initializer**: khởi tạo giá trị.
- **recurrent_regularizer**, **kernel_regularizer**: chuẩn hoá mạng neural.

Xây dựng mạng neural hồi quy đơn giản gồm lớp với 200 units (ứng với độ dài sequence)

SimpleRNN cần đầu vào có shape như sau: [batch, timesteps, feature].

Ý nghĩa như sau:

- **batch**: số lượng sample - điểm dữ liệu (đã chia theo kích thước mỗi batch, có thể xem lại batch and mini-batch training để hiểu rõ hơn).
- **timesteps**: Số lượng từ, ký tự trong câu, ở đây timesteps chính là độ dài chuỗi.
- **features**: số lượng features đưa vào mỗi steps trong sequence, features sẽ bằng với **chiều (dimension) của word embedding**.

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.

```
[115] from tensorflow.keras.optimizers import Adam
      from tensorflow.keras.losses import CategoricalCrossentropy
      optimizer = Adam(learning_rate=0.01)
```

- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

```
loss = CategoricalCrossentropy()
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn_2 (SimpleRNN)    (None, None, 200)         40400

 simple_rnn_3 (SimpleRNN)    (None, 200)               80200

 dense_5 (Dense)             (None, 2)                 402

=================================================================
Total params: 121,002
Trainable params: 121,002
Non-trainable params: 0
_________________________________________________________________
```

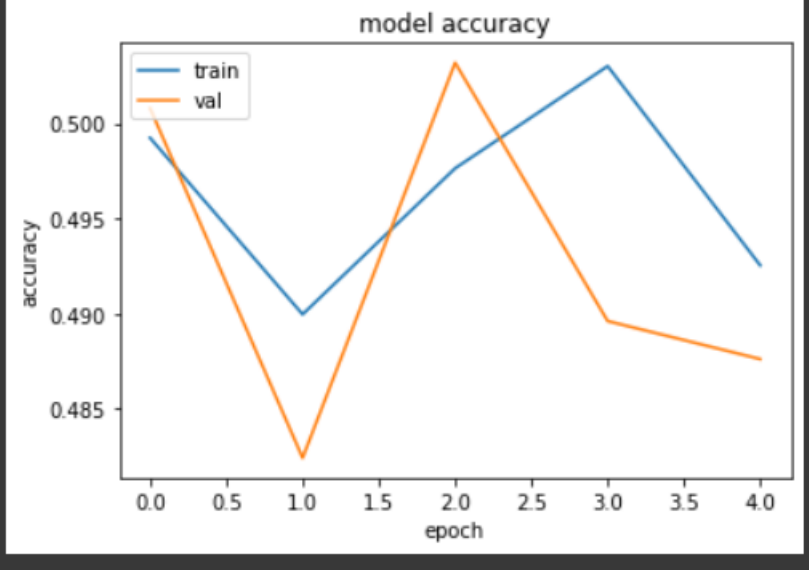
(?) Huấn luyện mô hình với các thông số sau:

- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

```
[116] history=model.fit(X_train_new, y_train, batch_size=128,epochs=5,validation_split=0.1)

      Epoch 1/5
      176/176 [==============================] - 77s 431ms/step - loss: 130.7133 - accuracy: 0.4992 - val_loss: 1.2459 - val_accuracy: 0.5008
      Epoch 2/5
      176/176 [==============================] - 77s 438ms/step - loss: 1.1032 - accuracy: 0.4900 - val_loss: 0.8317 - val_accuracy: 0.4824
      Epoch 3/5
      176/176 [==============================] - 75s 426ms/step - loss: 0.8490 - accuracy: 0.4976 - val_loss: 0.8564 - val_accuracy: 0.5032
      Epoch 4/5
      176/176 [==============================] - 74s 422ms/step - loss: 0.8021 - accuracy: 0.5030 - val_loss: 0.6940 - val_accuracy: 0.4896
      Epoch 5/5
      176/176 [==============================] - 76s 434ms/step - loss: 0.6934 - accuracy: 0.4925 - val_loss: 0.6934 - val_accuracy: 0.4876
```

(?) Vẽ đồ thị học với Accuracy và Loss?

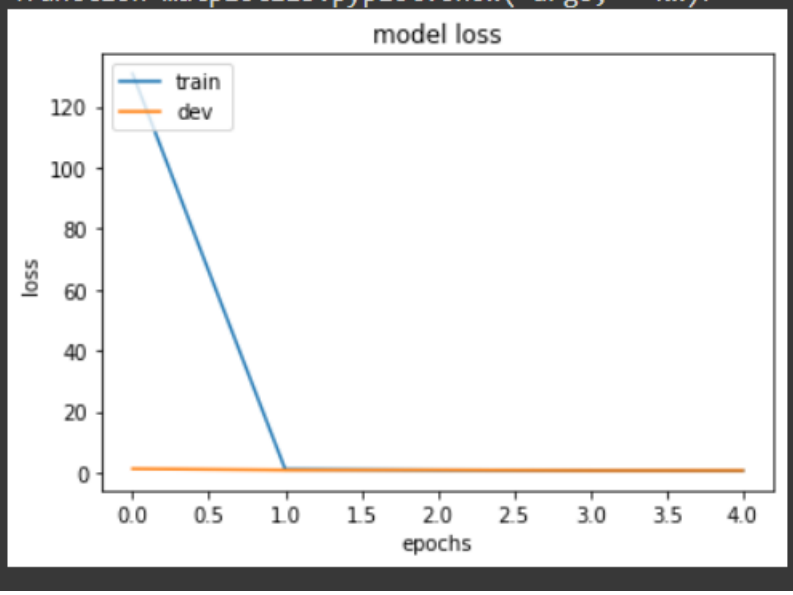
```
[117] plt.plot(history.history['accuracy'])
      plt.plot(history.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train','val'], loc = 'upper left')
      plt.show()
```



Đồ thị học với Accuracy

```
[118] plt.plot(history.history['loss'])
      plt.plot(history.history['val_loss'])
      plt.title('model loss')
      plt.ylabel('loss')
      plt.xlabel('epochs')
      plt.legend(['train','dev'],loc='upper left')
      plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



Đồ thị học với Loss

(?) Tính độ chính xác đánh giá của mô hình bằng độ đo Accuracy?

```
[119] import numpy as np
      from sklearn.metrics import accuracy_score
      y_pred = model.predict(X_test_new)
      y_pred_label = np.argmax(y_pred,axis=-1)
```

```
      782/782 [==============================] - 24s 31ms/step
```

```
[120] from sklearn.metrics import accuracy_score
      accuracy_score(y_test, y_pred_label)*100
```

```
      50.019999999999996
```

Mô hình có độ chính xác là 50.01%

3. LSTM, BILSTM VÀ EMBEDDING

Embedding

Kỹ thuật Embedding sẽ chuyển từ thành vector đặc (dense vector). Đây là một trong các kỹ thuật được dùng nhiều trong NLP nhằm biểu diễn ngữ nghĩa cho một từ.

Lớp Embedding trong thư viện Keras có các thông số quan trọng sau:

- **input_dim:** kích thước từ vựng. thường là giá trị: len(word_index) + 1.
- **output_dim:** chiều của vector (đôi khi còn được gọi là embedding_dim).
- **embeddings_initializer:** phương pháp khởi tạo giá trị của embedding.
- **embeddings_regularizer:** chuẩn hoá giá trị của embedding.

Lớp LSTM

Xây dựng một lớp LSTM cho mạng neural.

Các thông số quan trọng bao gồm:

- **units:** số lượng units trong layer..
- **activation:** hàm kích hoạt.
- **dropout:** sử dụng dropount hay không.
- **return_sequences:** trả về output là một sequence (đối với Many-to-many) hoặc là một giá trị (Many-to-one).
- **recurrent_initializer**, **kernel_initializer** và **bias_initializer**: khởi tạo giá trị.
- **recurrent_regularizer**, **kernel_regularizer**: chuẩn hoá mạng neural.

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.
- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

```
[125] optimizer = Adam(learning_rate=0.01)
      loss = CategoricalCrossentropy()
      model1.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])
```

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

```
] model1.summary()

Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, None, 128)         11338752

 lstm_4 (LSTM)               (None, 200)               263200

 dense_6 (Dense)             (None, 2)                 402

=================================================================
Total params: 11,602,354
Trainable params: 11,602,354
Non-trainable params: 0
_________________________________________________________________
```

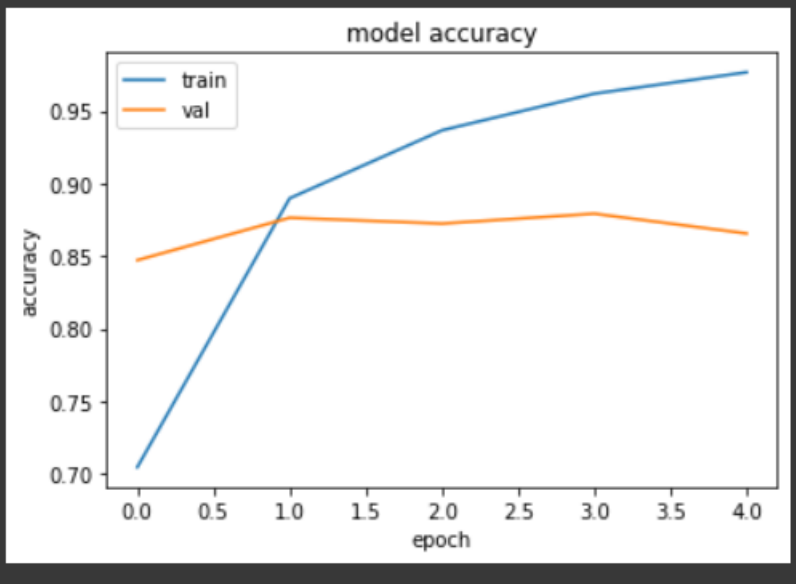
(?) Huấn luyện mô hình với các thông số sau:

- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

```
[126] history1=model1.fit(X_train_new, y_train, batch_size=128,epochs=5,validation_split=0.1)

Epoch 1/5
176/176 [==============================] - 7s 32ms/step - loss: 0.5753 - accuracy: 0.7048 - val_loss: 0.3754 - val_accuracy: 0.8472
Epoch 2/5
176/176 [==============================] - 5s 30ms/step - loss: 0.2653 - accuracy: 0.8898 - val_loss: 0.2983 - val_accuracy: 0.8764
Epoch 3/5
176/176 [==============================] - 5s 30ms/step - loss: 0.1628 - accuracy: 0.9366 - val_loss: 0.3234 - val_accuracy: 0.8724
Epoch 4/5
176/176 [==============================] - 5s 30ms/step - loss: 0.1047 - accuracy: 0.9618 - val_loss: 0.3778 - val_accuracy: 0.8792
Epoch 5/5
176/176 [==============================] - 5s 30ms/step - loss: 0.0662 - accuracy: 0.9765 - val_loss: 0.4417 - val_accuracy: 0.8656
```

(?) Vẽ đồ thị học với Accuracy và Loss?

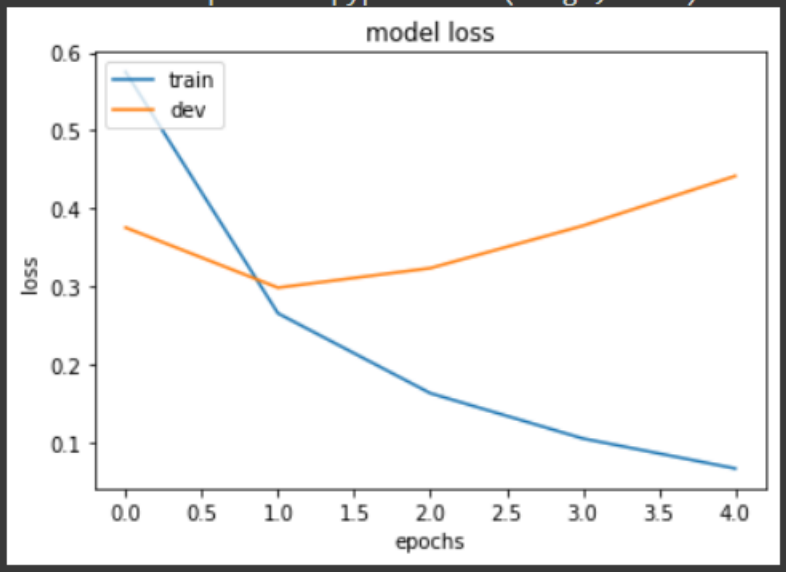
```
[127] plt.plot(history1.history['accuracy'])
      plt.plot(history1.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train','val'], loc = 'upper left')
      plt.show()
```



Đồ thị học với Accuracy

```
[128] plt.plot(history1.history['loss'])
      plt.plot(history1.history['val_loss'])
      plt.title('model loss')
      plt.ylabel('loss')
      plt.xlabel('epochs')
      plt.legend(['train','dev'],loc='upper left')
      plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



Đồ thị học với Loss

(?) Đánh giá độ chính xác mô hình bằng độ đo Accuracy?

```
[129] y_pred = model1.predict(X_test_new)
      y_pred = np.argmax(y_pred,axis = -1)
      accuracy_score(y_test,y_pred)*100
```

```
782/782 [==============================] - 5s 6ms/step
85.544
```

Mô hình có độ chính xác khá cao đạt 85.54%

Lớp Bidirectional

Sử dụng để xây dựng BiLSTM hoặc BiGRU.

Có hai thông số quan trọng gồm:

- layer: Một lớp (layer) của thư viện Keras, có thể là LSTM hoặc GRU.
- merge_mode: phương pháp dùng để merge giá trị forward và backward (xem thêm về LSTM và GRU để rõ hơn), mặc định là "concat".

(?) Compile mô hình với các thông số sau:

- Hàm tối ưu: Adam với learning_rate = 0.01.
- Hàm loss: Binary Cross-Entropy.
- Độ đo: Accuracy.

```
[131] optimizer = Adam(learning_rate = 0.01)
      loss = BinaryCrossentropy()
      model2.compile(optimizer = optimizer, loss = loss, metrics = ['accuracy'])
```

(?) Sử dụng hàm summary để xem cấu trúc mô hình đã xây dựng?

```
model2.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, None, 128)         11338752

 bidirectional_3 (Bidirectio  (None, 400)              526400
 nal)

 dense_7 (Dense)             (None, 2)                 802

=================================================================
Total params: 11,865,954
Trainable params: 11,865,954
Non-trainable params: 0
_________________________________________________________________
```

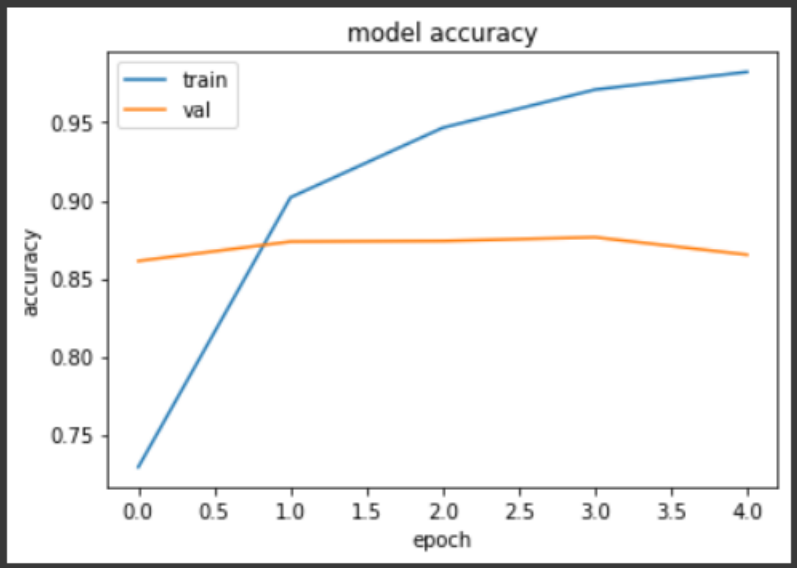
(?) Huấn luyện mô hình với các thông số sau:

- Batch size: 128.
- Số epochs: 5.
- Tỉ lệ tạo tập dev: 10%.

```
[133] history2 = model2.fit(X_train_new, y_train, batch_size = 128,epochs =5, validation_split = 0.1)

      Epoch 1/5
      176/176 [==============================] - 14s 59ms/step - loss: 0.5261 - accuracy: 0.7292 - val_loss: 0.3372 - val_accuracy: 0.8612
      Epoch 2/5
      176/176 [==============================] - 10s 55ms/step - loss: 0.2446 - accuracy: 0.9019 - val_loss: 0.2942 - val_accuracy: 0.8736
      Epoch 3/5
      176/176 [==============================] - 10s 55ms/step - loss: 0.1468 - accuracy: 0.9465 - val_loss: 0.3316 - val_accuracy: 0.8740
      Epoch 4/5
      176/176 [==============================] - 10s 55ms/step - loss: 0.0839 - accuracy: 0.9710 - val_loss: 0.3854 - val_accuracy: 0.8764
      Epoch 5/5
      176/176 [==============================] - 10s 55ms/step - loss: 0.0529 - accuracy: 0.9823 - val_loss: 0.5096 - val_accuracy: 0.8652
```

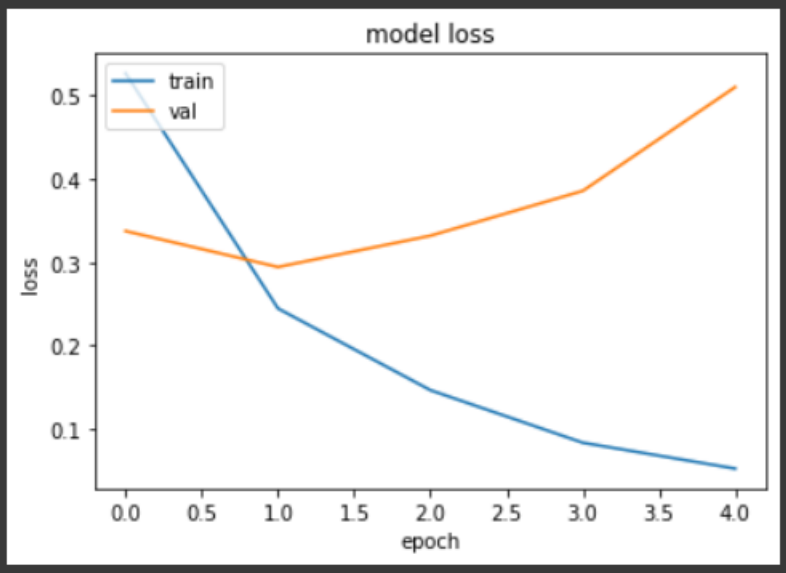
(?) Vẽ đồ thị học với Accuracy và Loss?

```
[134] plt.plot(history2.history['accuracy'])
      plt.plot(history2.history['val_accuracy'])
      plt.title('model accuracy')
      plt.ylabel('accuracy')
      plt.xlabel('epoch')
      plt.legend(['train','val'], loc = 'upper left')
      plt.show()
```



Đồ thị học với Accuracy

```
[135] plt.plot(history2.history['loss'])
      plt.plot(history2.history['val_loss'])
      plt.title('model loss')
      plt.ylabel('loss')
      plt.xlabel('epoch')
      plt.legend(['train','val'], loc = 'upper left')
      plt.show()
```



Đồ thị học với Loss

(?) Đánh giá độ chính xác mô hình bằng độ đo Accuracy?

```
y_pred = model2.predict(X_test_new)
y_pred = np.argmax(y_pred,axis = -1)
accuracy_score(y_test,y_pred)*100
```

```
782/782 [==============================] - 9s 10ms/step
85.668
```

Mô hình có độ chính xác khá cao đạt 85.66%