Giảng viên hướng dẫn: Lê Anh Cường

Sinh viên (Họ và tên – MSSV):  
1/ Nguyễn Minh Đăng Khoa – 51800882

2/ Nguyễn Hữu Lợi – 51702134

**DỰ ÁN CUỐI KỲ: BÀI 2**

MÔN: NHẬP MÔN XỬ LÝ NGÔN NGỮ TỰ NHIÊN

**I/ Mô tả bài toán**

- Xây dựng chương trình hỏi đáp tự động (question answering) cho tiếng Việt.

- Mô tả yêu cầu:

+ Nhập vào câu hỏi Q.

+ Hệ thống đưa ra câu trả lời A.

- Chương trình hỏi đáp tự động (Question Answering) là nhiệm vụ tự động trả lời câu hỏi dưới dạng ngôn ngữ tự nhiên. Để tìm câu trả lời cho một câu hỏi, một chương trình máy tính sử dụng một một tập các văn bản ngôn ngữ tự nhiên. Với câu hỏi đầu vào, chương trình đưa ra câu trả lời đơn giản với việc so khớp các từ khóa. Xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) được sử dụng để nhận ra cấu trúc cú pháp và ngữ nghĩa của câu hỏi và các tài liệu thích hợp chứa đựng các câu trả lời cho câu hỏi sẽ được trả lại.

**II/ Giải pháp**

Chúng ta sẻ sử dụng LSTM để giải quyết bài toán trên.

**2.1 LSTM là gì?**

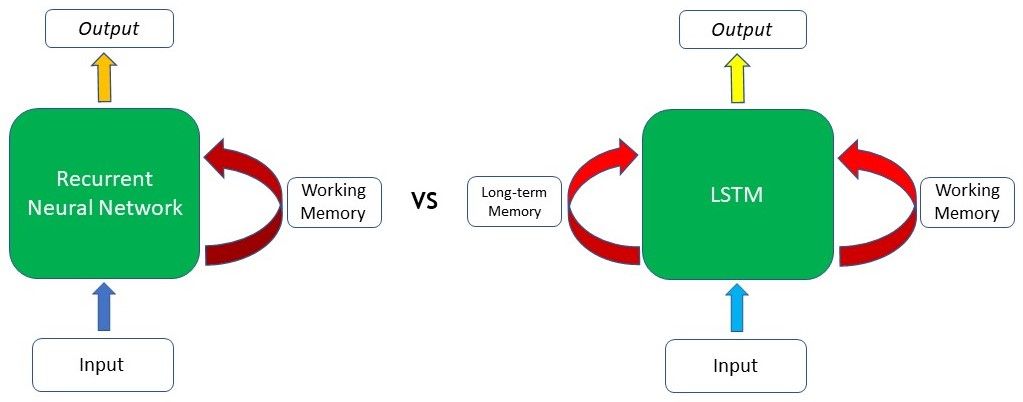
**-** LSTM là một biến thể được cải tiến từ RNN, RNN (Recurrent neural networks) là mạng nơ-ron hồi quy khác với các mạng nơ ron truyền thẳng chỉ output phụ thuộc vào input thì ở RNN output không phụ thuộc vào input mà còn phụ thuộc vào output trước đó.

- Nhờ vậy mà RNN được ứng dụng nhiều trong thực tế như: Nhận diện ngôn ngữ, Nhận diện ảnh, Dịch,…

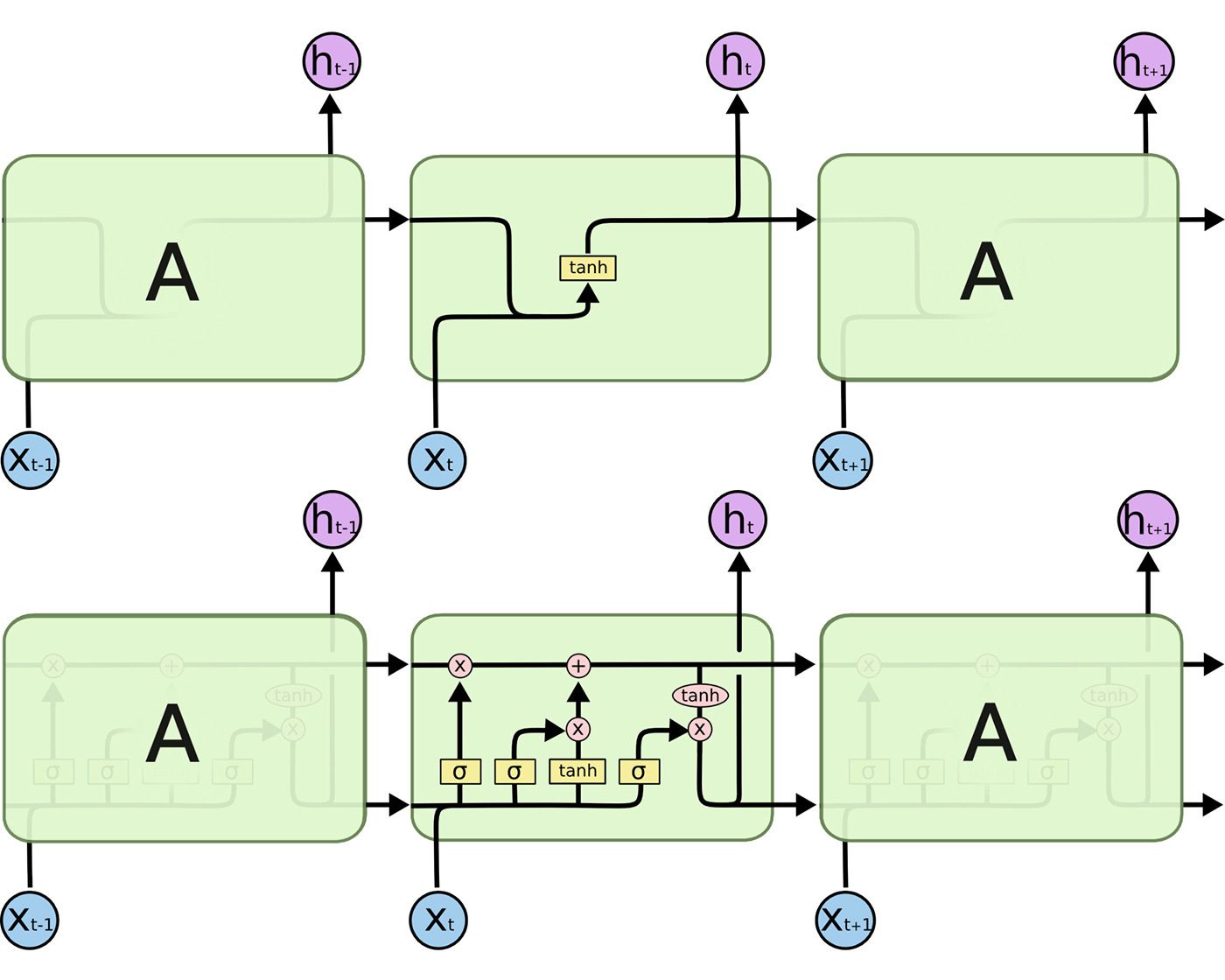
- Đặc điểm cơ bản mà một RNN cần có là thông tin hiện tại phải được đóng góp từ các thông tin trước đó của mạng từ đó RNN có thể học được các thông tin từ trước và sử dụng cho sau này. Tuy nhiên đối với các mạng có lớp sâu hơn thì RNN 2 vấn đề là Vanishing/ Exploding gradients. Vấn đề này phát sinh trong quá trình lan truyền ngược khi traning.

- Trong khi LSTM nhờ cơ chế Gating của nó là thứ tạo ra sự khác biệt. Tính năng này giải quyết vấn đề bộ nhớ ngắn hạn của RNN.

**2.2 Khác biệt giữa khiến trúc RNN và LSTM**

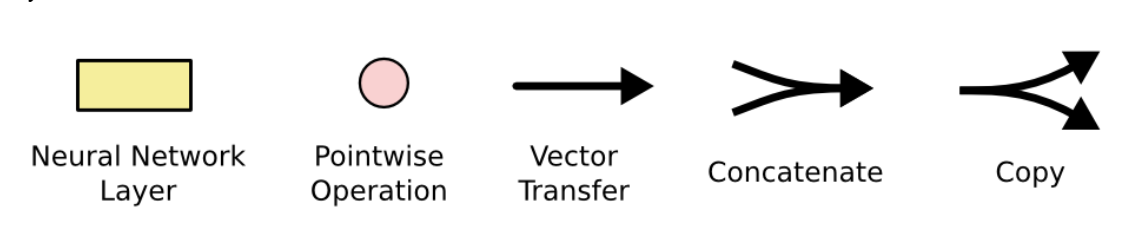


- Như hình thì ta thấy, sự khác biệt chủ yêu là ở LTSM khi lưu trữ “Long-term Memory”.



Từ hình trên ta thấy trong từng cell của RNN khác biệt với cell của LSTM. Cụ thể là RNN chỉ tính tanh của hidden state trước đó với input để cho ra output và lưu trữ hidden mới. Còn LSTM có nhiều bước tính hơn và có một bộ nhớ chạy xuyên suốt trong quá trình tính toán giữa các cell để lưu các thông tin được chọn.

**2.3 Các kí hiệu trong kiến trúc LSTM**

****

- Neural Net work Layer: layer được học trong từng mạng nơ ron.

- Poinwise: biểu diễn các phép tính như cộng hoặc nhân vector.

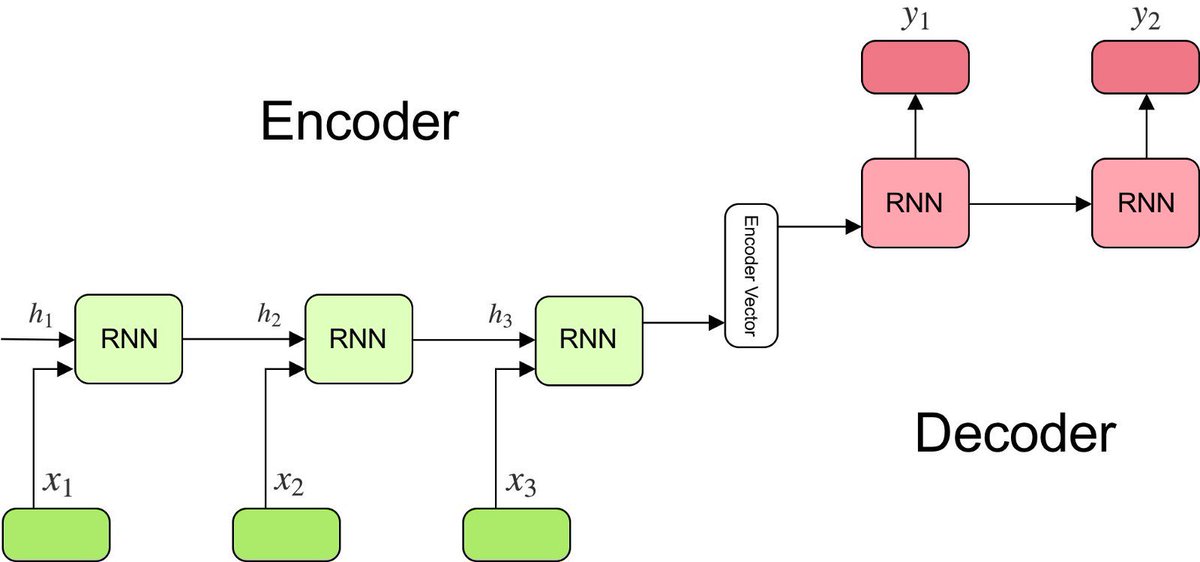
- Vector Transfer: biểu diễn mạng đầu vào của 1 nút tới 1 nút khác.

- Concatenate: thì nội dung các vector sẻ được kết hợp.

- Copy: thì nội dung sẻ được sao chép và chuyển tới các nói khác nhau.

**III/ Thuật toán**

Ý tưởng thuật toán là từ tập dữ liệu, ta áp dụng model LSTM để train dữ liệu. Cho một chuỗi đầu vào, hệ thống sẽ tạo một chuỗi đầu ra. Kết hợp với LSTM model cùng với việc sử dụng encoder (bộ mã hoá) và decoder (bộ giải mã). Với sự kết hợp đó đã đưa ra những kết quả tốt và ứng dụng vào các việc như: dịch ngôn ngữ, tóm tắt văn bản, dự báo chuỗi thời gian, hỏi đáp v..v..



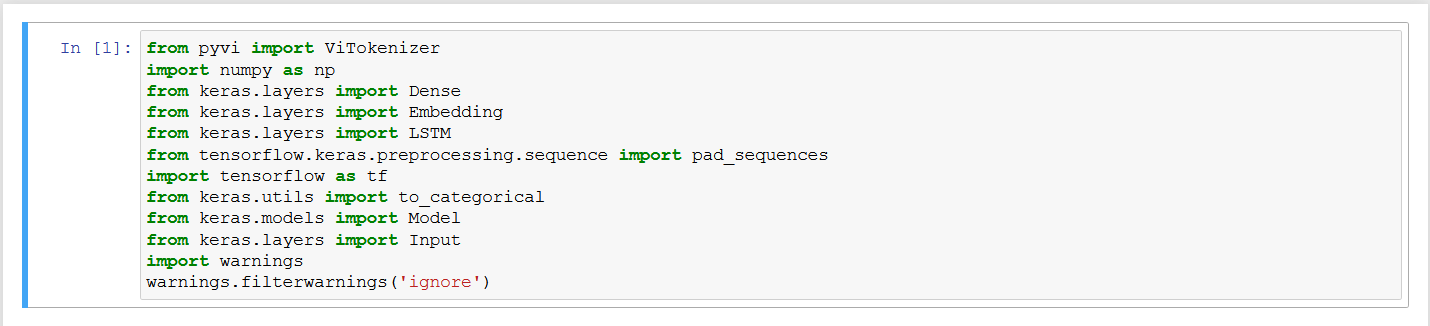
- Một lớp RNN (hoặc ngăn xếp của lớp đó) hoạt động như "bộ mã hóa": nó xử lý chuỗi đầu vào và trả về trạng thái bên trong của chính nó. Lưu ý rằng loại bỏ các đầu ra của bộ mã hóa RNN, chỉ khôi phục trạng thái. Trạng thái này sẽ đóng vai trò là "ngữ cảnh" hay "điều hòa" của bộ giải mã trong bước tiếp theo.

- Một lớp RNN khác (hoặc ngăn xếp của nó) hoạt động như "bộ giải mã": nó được huấn luyện để dự đoán các ký tự tiếp theo của chuỗi mục tiêu, cho các ký tự trước đó của chuỗi mục tiêu. Cụ thể, nó được đào tạo để biến các chuỗi mục tiêu thành các chuỗi giống nhau nhưng bù lại bằng một timestep trong tương lai, một quá trình đào tạo được gọi là " teacher forcing" trong bối cảnh này. Quan trọng hơn, bộ mã hóa sử dụng như trạng thái ban đầu các vectơ trạng thái từ bộ mã hóa, đó là cách bộ giải mã thu được thông tin về những gì nó được cho là tạo ra.

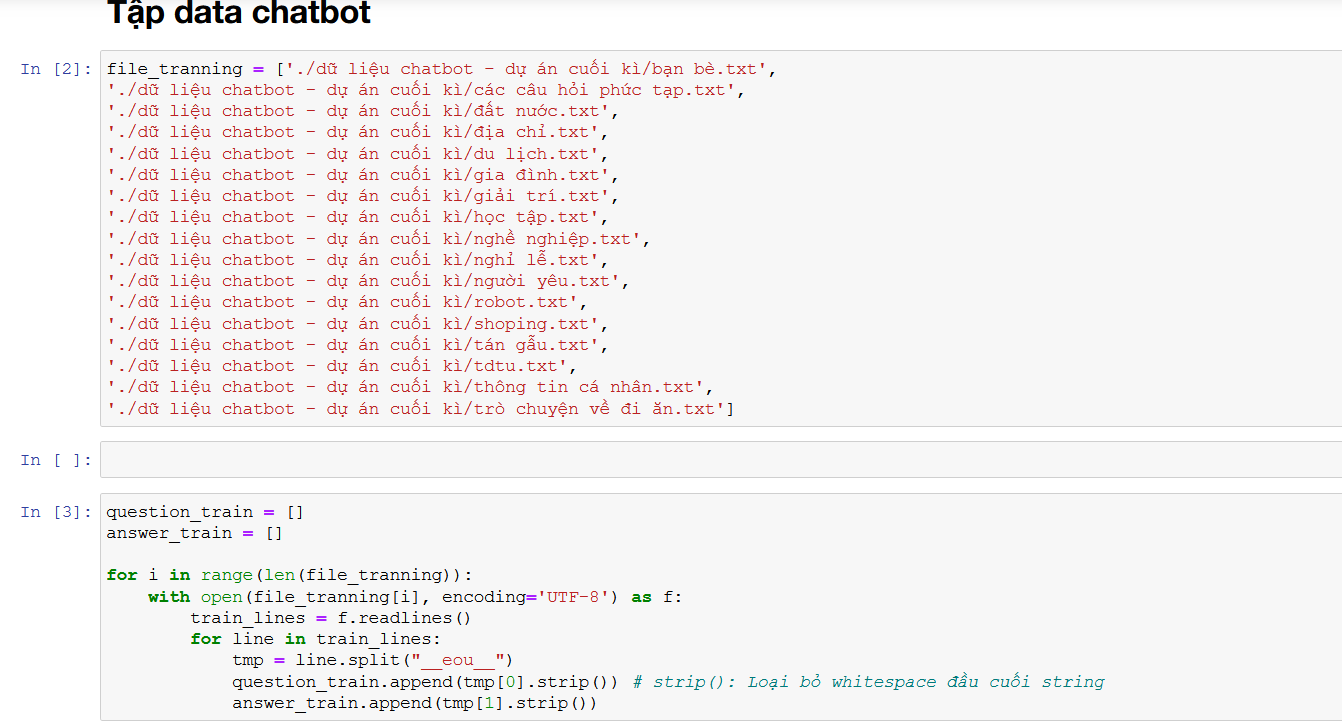
**IV/ Giải thích, kết quả**

**4.1 Import lib, Get data**

- Đầu tiên là import thư viện cần dùng.

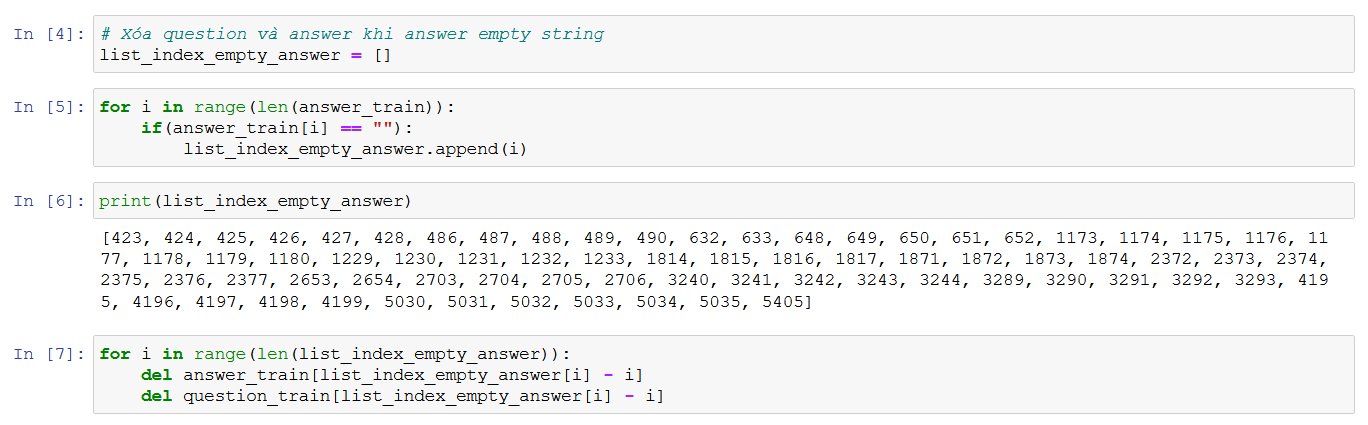


**-** Tiếp theo là đọc tất cả file txt trong “dữ liệu chatbot - dự án cuối kì” và chia thành 2 tập question, answer.

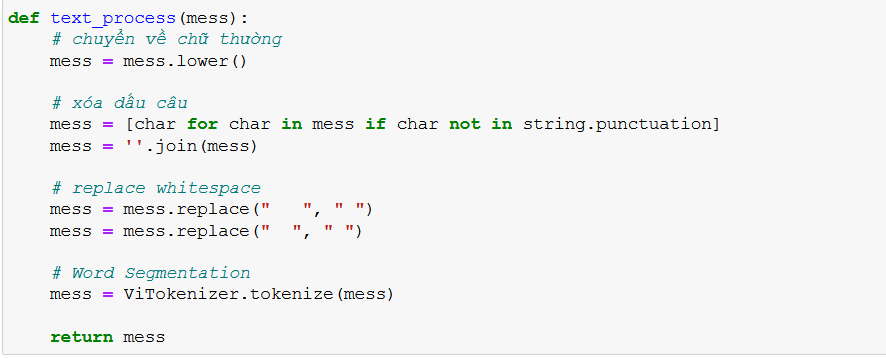


**4.2 Preprocessing**

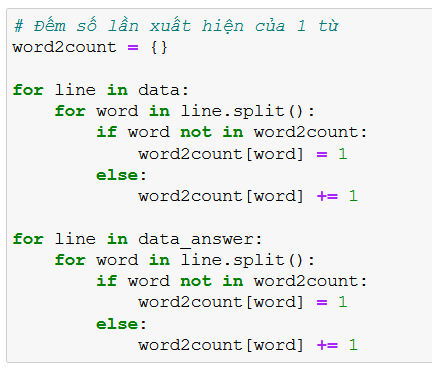
- Loại bỏ những câu ở 2 tập question, answer mà câu hỏi không có câu trả lời để tránh làm nhiễu tập data.



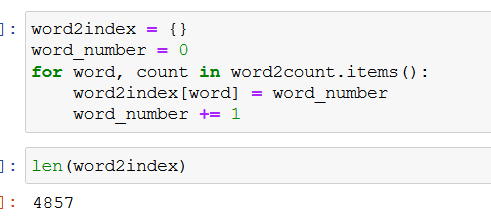
- Text\_process: chuyển tất cả về chữ thường, loại bỏ dấu câu, khoảng trắng thừa và word segmentation cho câu.



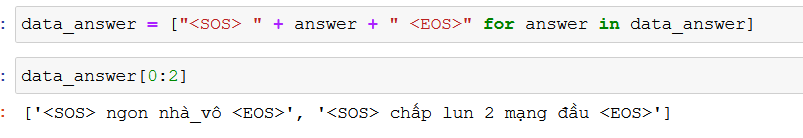
- Tạo một dict word2count để tính tần suất xuất hiện của các từ trong tập answer, question.



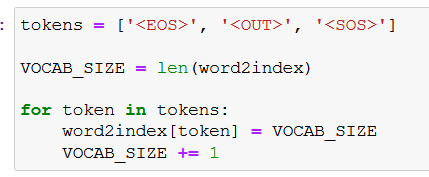
- Tạo dict word2index: một từ đại diện cho 1 số. Mục đích là convert dữ liệu đầu vào từ string thành một vector có các phần tử là index của từ đó trong word2index. index2word thì ngược lại với word2index.



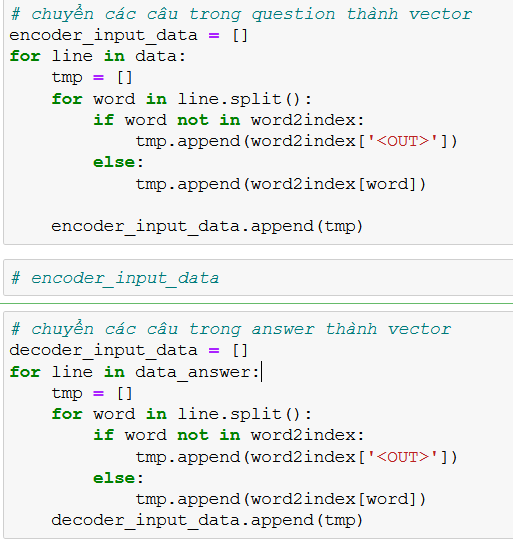
- Gắn 2 thẻ <SOS> (Start of sequence) và <EOS> (End of sequence) vào các câu trong answer



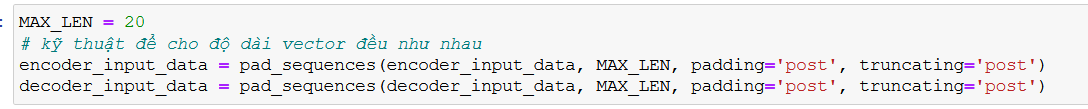
- Sau đó thêm 3 tag <SOS>, <EOS>, <OUT> vào word2index. (<OUT> đại diện cho các từ không có trong word2index). VOCAB\_SIZE là len cùa word2index.



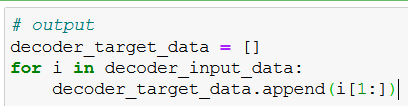
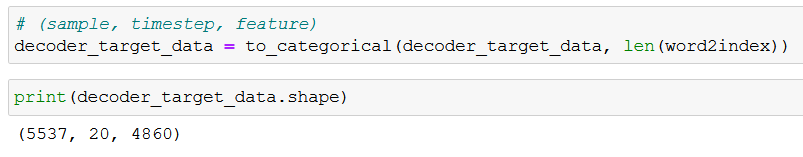
- Chuyển các câu trong question, answer thành vector đưa vào list encoder\_input\_data, decoder\_input\_data.



- Sau đó dùng pad\_sequences để cho độ dài các vector trong question và answer bằng nhau với độ dài là MAX\_LEN = 20.



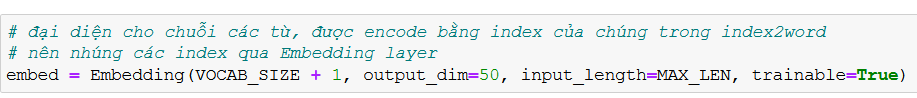
- decoder\_target\_data là output của model train, giống decoder\_input\_data nhưng được bỏ tag <SOS> . Sau đó, pad\_sequences lại decoder\_target\_data lại thành chiều dài MAX\_LEN. Tiếp theo sẽ one-hot encoder để decoder\_target\_data thành vector không gian ba chiều (samples, timestep, feature).

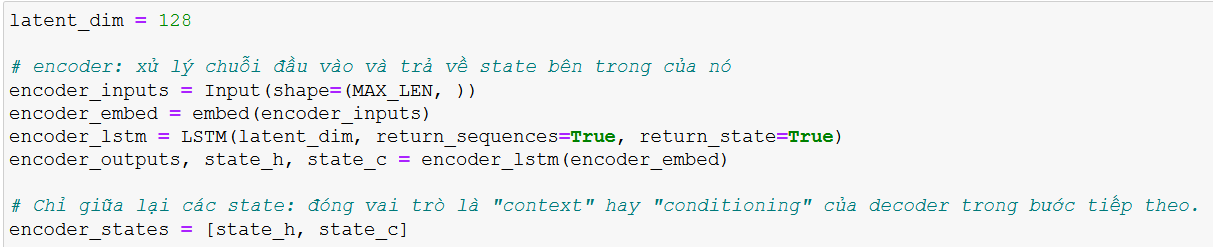
**4.3 LSTM model**

Latten\_dim = 128 là số chiều ẩn trong LSTM.

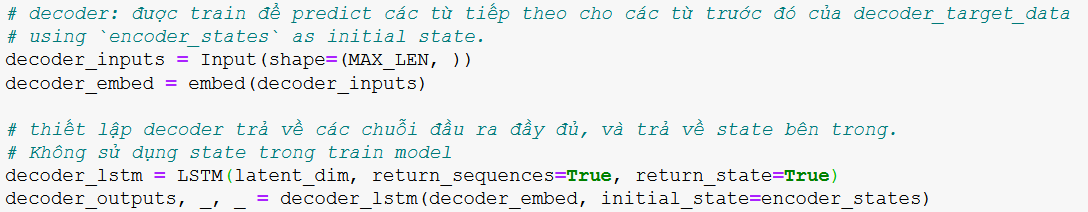
Số chiều output cho layer Embedding là 50.



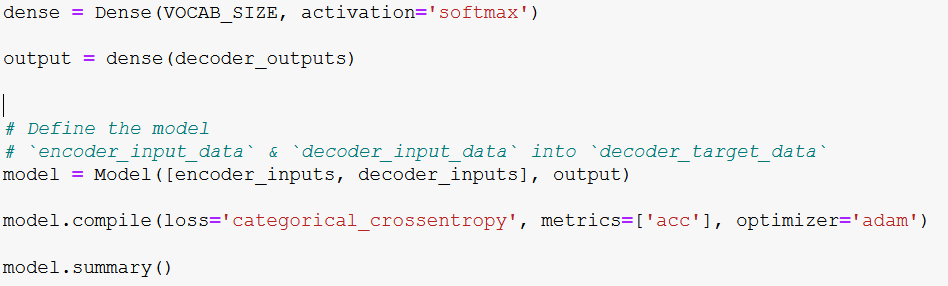
Định nghĩa lớp encoder: xử lý chuỗi đầu vào và trả về state bên trong của nó, trước tiên là embedding cho encoder input để đưa vào lớp LSTM. Đối với lớp LSTM encoder, chỉ giữa lại các state: đóng vai trò là "context" hay "conditioning" của decoder trong bước tiếp theo.

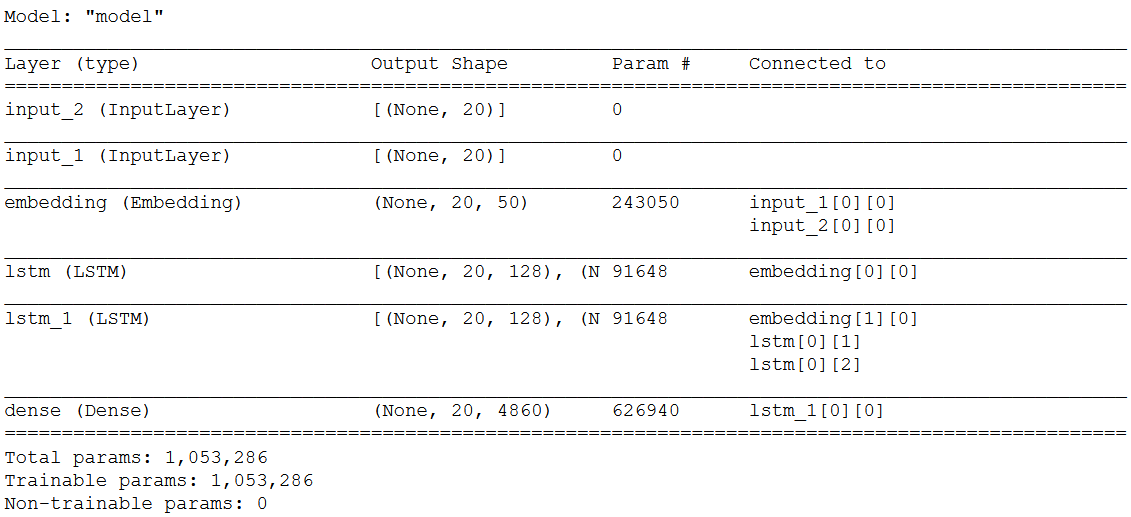


Định nghĩa lớp decoder: được train để predict các từ tiếp theo cho các từ trước đó của decoder\_target\_data, tiếp tục là embedding decoder input để đưa vào LSTM decoder, sử dụng encoder\_states của encoder như một state ban đầu của decoder. Decoder trả về các chuỗi đầu ra đầy đủ và trả về state bên trong. Lưu ý không sử dụng state đó trong train model.

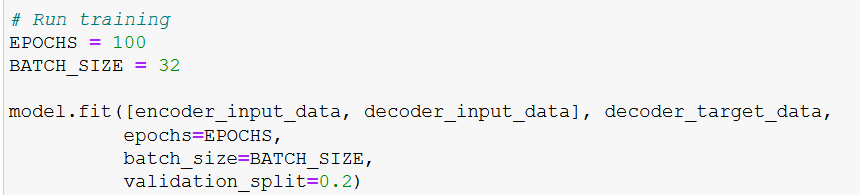


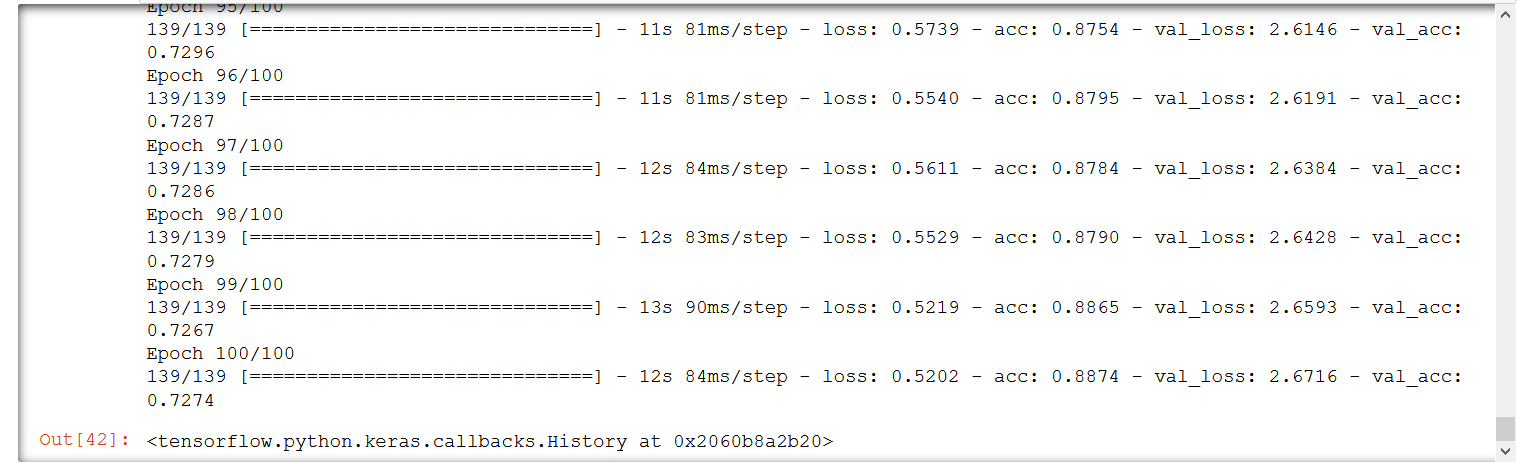
Lớp Dense với số chiều output là VOCAB\_SIZE và hàm kích hoạt là “softmax”. Sau đó định nghĩa cho model là từ “encoder\_input\_data” và “decoder\_input\_data” thành “decoder\_target\_data”.



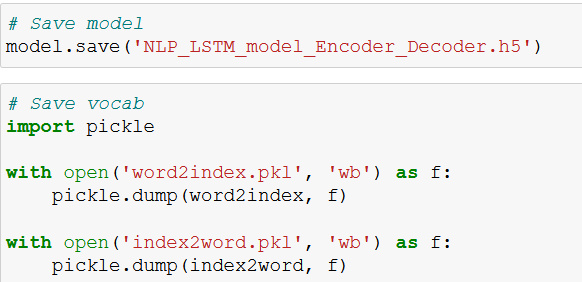


Cuối cùng là train model với EPOCH = 100, BATCH\_SIZE = 32.





Save model và index2word, word2index.



**4.4 Inference setup**

- Để decode một câu:

+ Encode câu đầu vào và truy xuất trạng thái ban đầu của encoder

+ Chạy một bước của decoder với trạng thái ban đầu và token "SOS" làm target. Ouput sẽ là từ tiếp theo.

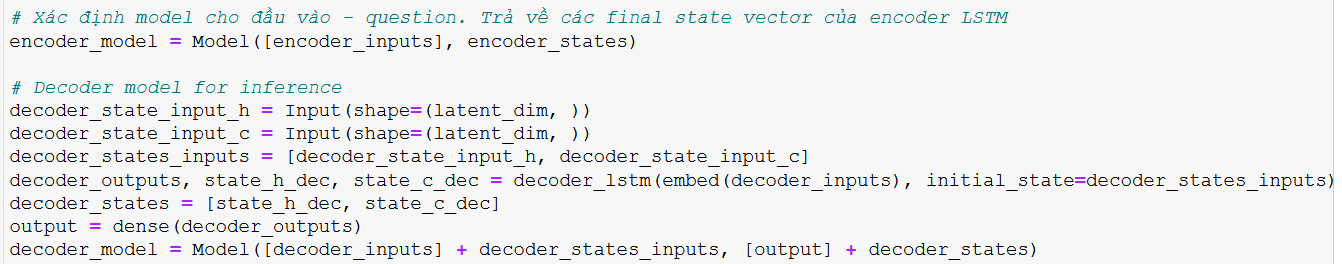
+ Nối từ được dự đoán và lặp lại.

- Sau khi save model và index2word, word2index. Chúng ta sẽ viết lại text\_process, load file word2index, index2word và load model lấy ra những thứ cần thiết.

- Định nghĩa encoder\_model: Xác định model cho đầu vào - question. Trả về các final state vector của encoder LSTM. Ở đây chúng ta cần load được cái encoder\_states của model đó.

- Đối với decoder\_model cần load lớp embbeding và dùng để embbeding decoder\_inputs.

- Giá trị trả về là decoder\_output và decoder\_state chính là từ tiếp theo được dự đoán.



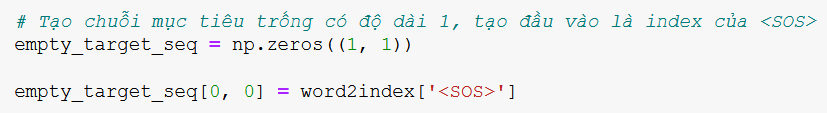
- Khi nhận vào câu hỏi, chúng ta phải preprocessing theo đúng dạng để đưa vào model dự đoán (dùng text\_process cho câu input, chuyển câu thành dạng vector và pad\_sequnces với độ dài MAX\_LEN). Nếu câu hỏi nhập vào là “stop” thì dừng chương trình, còn bỏ trống thì yêu cầu đăt lại câu hỏi khác.



- Chuyển câu hỏi đã xử lý cho encoder LSTM để nhận final encoder states của the encoder LSTM.



- Tạo chuỗi trống có độ dài 1, tạo đầu vào là index của <SOS>.



- Ta khởi tạo một vòng lập while, dùng decoder\_model với input là index của từ trước đó và state của encoder thì sẽ lấy được prediction, prediction\_h, prediction\_c. Từ prediction thì chúng ta tìm được index có khả năng xuất hiện cao nhất từ prediction trả về. Sau khi có index thì tìm được từ thông qua word2index. Tiếp đó là word sẽ nhận từ tìm được, và nếu word dự đoán không phải là <EOS> thì nối vào câu answer. Sau đó, khởi tạo lại empty\_target\_seq và đặt token\_word\_index thành đầu ra của current time step, nó sẽ chuyển làm đầu vào của next time step. Và update states trả về từ decoder\_model và tiếp tục dự đoán từ tiếp theo. Điều kiện dừng là khi word = <EOS> hoặc chiều dài câu lớn hơn MAX\_LEN. Sau khi dừng xuất ra answer cho câu question\_input.

