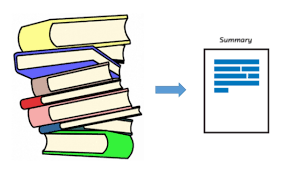


Đề Tài

Text Summarization sử dụng phương pháp Abstractive method



Giáo Viên Hướng Dẫn: Nguyễn Hồng Bửu Long

Sinh Viên Thực Hiện:

Mai Huy 43.01.104.065

Nguyển Trung Hiếu 43.01.104.052

Nguyễn Thành Đạt 43.01.104.020

Lớp : Khai thác nội dung văn bản

**BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC SƯ PHẠM HỒ CHÍ MINH**

Nội dụng

[I. Giới thiệu đề tài: 3](#_Toc58278308)

[II. Hướng tiếp cận: 4](#_Toc58278309)

[1. Abtractive summarization: 4](#_Toc58278310)

[2. Long Short-Term Memory (LSTM): 5](#_Toc58278311)

[3. Mô hình seq2seq: 9](#_Toc58278312)

[3.1) Training phrase: (giai đoạn huấn luyện mô hình) 9](#_Toc58278313)

[3.2) Inference phrase (giai đoạn dự đoán) 11](#_Toc58278314)

[4. Attention mechanism: 12](#_Toc58278315)

[III. Đánh giá kết quả trên ngữ liệu / Demo: 12](#_Toc58278316)

[1) Thu thập Dataset (Amazon fined food reviews): 12](#_Toc58278317)

[2) Tiền xử lý dữ liệu (Data preprocessing) 13](#_Toc58278318)

[3) Tokenization: 15](#_Toc58278319)

[4) Xây dựng mô hình: 17](#_Toc58278320)

[5) Đánh giá mô hình (evaluation): 18](#_Toc58278321)

[6) Tiến hành dự đoán 19](#_Toc58278322)

[IV. Hướng phát triển: 21](#_Toc58278323)

[V. Hướng dẫn sử dụng các file python đã được xây dựng cấu trúc thông qua môi trường Command Prompt: 21](#_Toc58278324)

[File 1: File Processing\_data.py 21](#_Toc58278325)

[File 2: File split\_data\_tokenization.py 21](#_Toc58278326)

[File 3: File visualize.py 21](#_Toc58278327)

[File 4: Train\_model.py 22](#_Toc58278328)

[File 5: predict\_model.py 23](#_Toc58278329)

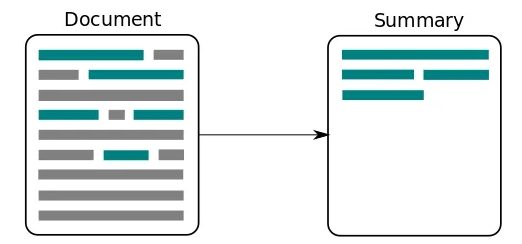
[VI. Tài liệu tham khảo: 24](#_Toc58278330)

1. **Giới thiệu đề tài:**

Trong một thời đại mà mỗi ngày, mỗi giờ , mỗi phút đều có một lượng thông tin khổng lồ được sinh ra, nhưng giới hạn về thời gian, về khả năng đọc và tiếp thu của con người là có hạn, việc hiểu và nắm bắt thật nhiều thông tin một cách nhanh chóng không phải là vấn đề đơn giản với bất kỳ ai.  
Đã bao giờ bạn tìm kiếm các kiến thức trên internet, hay đọc một cuốn sách mà nội dung của nó dài "lê thê", khiến cho bạn cảm thấy một chút khó khăn để có thể nắm bắt được nó chưa?

Đứng trước xu hướng con người ngày càng mất nhiều thời gian đọc email, báo điện tử và mạng xã hội, các thuật toán sử dụng machine learning để tự động tóm tắt các văn bản dài một cách gãy gọn và chính xác ngày càng trở nên cần thiết và có vai trò to lớn đối trong bất kỳ lĩnh vực nào.

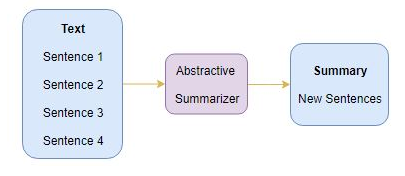
Tự động tóm tắt sẽ là một trong những công nghệ quan trọng có thể giúp con người giảm thiểu thời gian đọc email và thông tin, kiến thức mới để dành thời gian cho các công việc khác, mà vẫn có thể nắm bắt được gãy gọn những nội dung của nó.



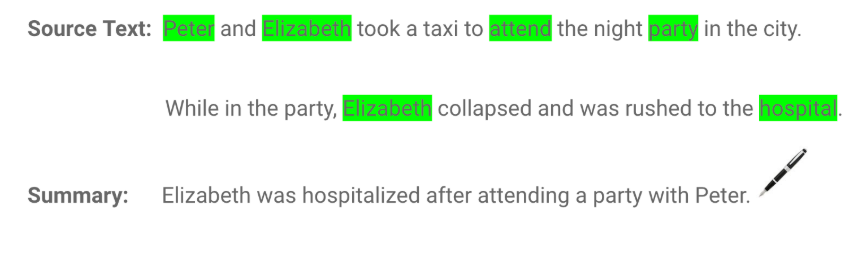
Tóm tắt văn bản là quá trình rút trích những thông tin quan trọng nhất từ một hoặc nhiều nguồn để tạo ra phiên bản cô đọng, ngắn gọn phục vụ cho một hoặc nhiều người dùng cụ thể, hay một hoặc nhiều nhiệm vụ cụ thể.

1. **Hướng tiếp cận:**
2. **Abtractive summarization:**

Tóm tắt trừu tượng là chúng ta sẽ tóm tắt thông tin từ văn bản lúc đầu thành một câu hoàn toàn mới. Phương pháp này hoàn toàn trái ngược với phương pháp trích xuất mà ta chỉ tóm tắt thông tin dựa trên các câu và từ có sẵn.



**Ví dụ:**

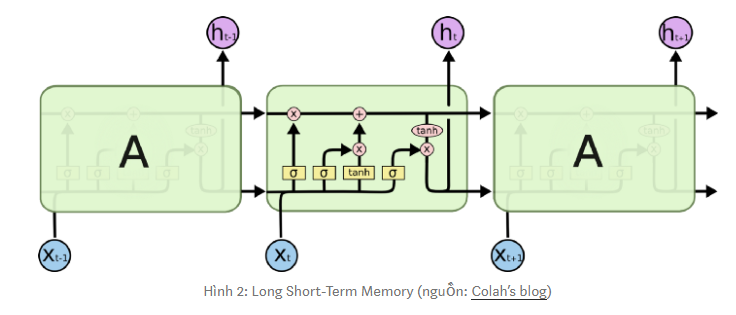
****

Ứng dụng tóm tắt trừu tượng quan sát toàn bộ các từ trong văn bản và nhớ các keyword- các từ mà ứng dụng đánh giá là quan trọng và nắm các ý chính trong đoạn văn bản, như trong ví dụ trên, các từ mà ứng dụng đánh giá là quan trọng bao gồm các danh từ và động từ như: Peter, Elizabeth, attend, party, hospital.

Từ các keyword này ứng dụng mới đoán được ý chính của văn bản và cho ra bản tóm tắt cuối cùng.

1. **Long Short-Term Memory (LSTM):**

**Long Short-Term Memory (LSTM),** một biến thể nổi tiếng của RNN, được đề xuất như là một giải pháp cho vấn đề vừa được nêu trên. Điểm chính trong kiến trúc mạng của LSTM chính là các **memory cell**với các cổng cho phép lưu trữ hoặc truy xuất thông tin. Các cổng này cho phép ghi đè (input gate), loại bỏ dư thừa (forget gate) và truy xuất (output gate) các thông tin được lưu trữ bên trong các memory cell.

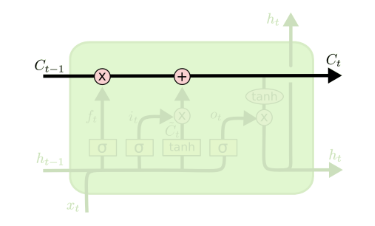




Từng Bước LSTM chạy qua:

Chìa khóa của LSTM là trạng thái ô, đường ngang chạy qua đỉnh sơ đồ.

Trạng thái tế bào giống như một băng chuyền. Nó chạy thẳng xuống toàn bộ chuỗi, chỉ với một số tương tác tuyến tính nhỏ. Rất dễ dàng để thông tin chỉ chảy dọc theo nó



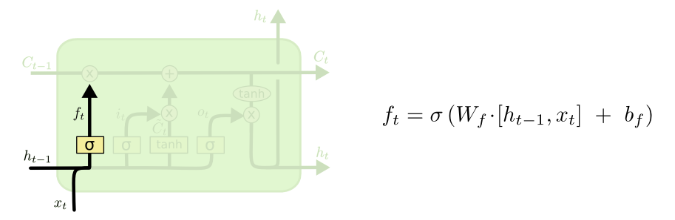
LSTM có khả năng loại bỏ hoặc thêm thông tin vào trạng thái tế bào, được điều chỉnh cẩn thận bởi các cấu trúc gọi là cổng vào

Gates là một cách để tùy ý để thông tin thông qua. Chúng được cấu tạo từ một lớp lưới thần kinh sigmoid và một phép toán nhân theo điểm.



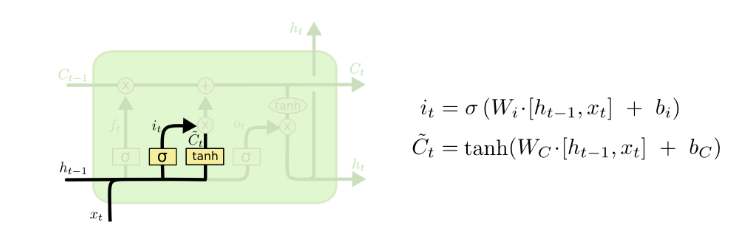
Lớp sigmoid xuất ra các số từ 0 đến 1, mô tả mức độ của mỗi thành phần sẽ được cho qua. Giá trị bằng 0 có nghĩa là không để bất cứ thứ gì qua, trong khi giá trị của 1 nghĩa là có thể cho phép mọi thứ thông qua!

Một LSTM có ba trong số các cổng này, để bảo vệ và kiểm soát trạng thái tế bào.



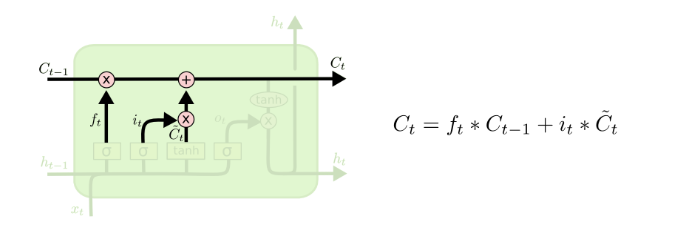
Bước tiếp là quyết định thông tin mới nào chúng em sẽ lưu trữ trong trạng thái tế bào. Điều này có hai phần. Đầu tiên, một lớp sigmoid được gọi là lớp cổng đầu vào của Google, quyết định giá trị nào chúng ta sẽ cập nhập. Tiếp theo, một lớp tanh tạo ra một vectơ các giá trị ứng cử viên mới , có thể được thêm vào. Trong bước tiếp theo, chúng em sẽ kết hợp cả hai để tạo ra một bản cập nhật cho trạng thái.

Trong ví dụ về mô hình ngôn ngữ của chúng em, chúng em muốn thêm trạng thái của chủ thể mới vào trạng thái tế bào, để thay thế mô hình cũ mà nó đang quên



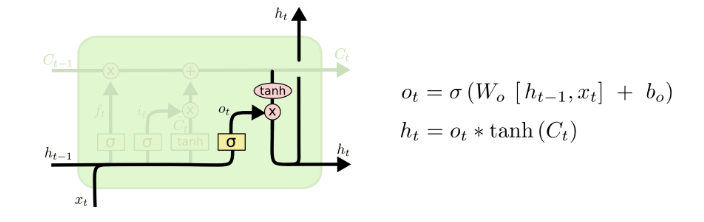
Chúng em nhân lên trạng thái cũ bằng ft, quên đi những điều chúng em quyết định quên trước đó. Sau đó, chúng em thêm  Đây là giá trị ứng cử viên mới, được chia tỷ lệ theo mức độ chúng em quyết định cập nhật từng giá trị trạng thái.

Trong trường hợp của mô hình ngôn ngữ, đây là nơi chúng em thực sự bỏ thông tin về giới tính của đối tượng cũ và thêm thông tin mới, như chúng em đã quyết định trong các bước trước.



Cuối cùng, chúng ta cần quyết định những gì chúng ta sẽ đầu ra. Đầu ra này sẽ dựa trên trạng thái ô của chúng em, nhưng sẽ là phiên bản được lọc. Đầu tiên, chúng em chạy một lớp sigmoid quyết định phần nào của trạng thái tế bào chúng ta sẽ xuất ra. Sau đó, chúng em đưa trạng thái tế bào thông qua tanh (để đẩy các giá trị ở giữa -1 và 1) và nhân nó với đầu ra của cổng sigmoid, do đó chúng em chỉ xuất ra các phần chúng em quyết định.

Đối với ví dụ về mô hình ngôn ngữ, vì nó chỉ nhìn thấy một chủ đề, nó có thể muốn xuất thông tin liên quan đến một động từ, trong trường hợp đó là những gì sắp diễn ra. Ví dụ, nó có thể xuất ra cho dù chủ ngữ là số ít hay số nhiều, để chúng ta biết dạng động từ nên được kết hợp thành gì nếu đó là những gì tiếp theo.

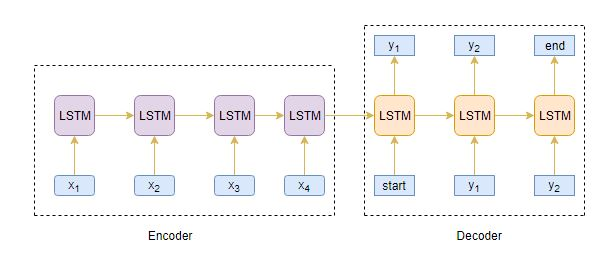


Việc nhận dạng chính xác tên riêng trong một đoạn văn bản phụ thuộc không chỉ vào các thông tin phía trước của từ đang xét mà còn cả các thông tin phía sau. Tuy nhiên, một kiến trúc LSTM truyền thống với một lớp duy nhất chỉ có thể dự đoán nhãn của từ hiện tại dựa trên thông tin có được từ các từ nằm trước đó.

1. **Mô hình seq2seq:**

Đây là mô hình giúp giải quyết những vấn đề liên quan đến thông tin tuần tự, có một số ứng dụng rất phổ biến như : sentiment classification, machine translation, named entity recognition.

Mô hình bao gồm 2 thành phần chính là encoder và decoder:



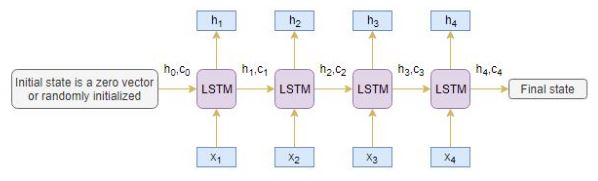
Kiến trúc Encoder-Decoder chủ yếu được sử dụng để giải quyết các vấn đề liên quan đến chuỗi (Seq2Seq) trong đó các chuỗi đầu vào và đầu ra có độ dài khác nhau.

Encoder được sử dụng dùng trích xuất các thông tin và ngữ cảnh của văn bản và encoder dùng dự đoán từng từ trong văn bản sau khi đã biết từ đứng trước nó.

### 3.1) Training phrase: (giai đoạn huấn luyện mô hình)

**Encoder**

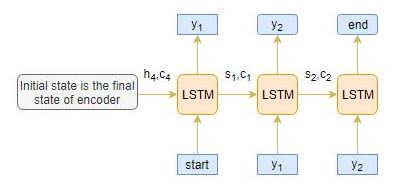
Lấy mô hình LSTM làm ví dụ, sử dụng encoder dùng để đọc toàn bộ chuỗi kí tự đầu vào, tại mỗi bước timestep, từng từ được đưa vào bộ encoder. Encoder giúp mô hình hiểu được rõ ý nghĩa và ngữ cảnh từng từ của câu input.



H0 và c0 được khởi tạo bằngc các vector random, sau mỗi timestep, các h(hidden state) và c (cell state) của mô hình LSTM được thay đổi. h và c cuối cùng (ở mô hình trên là h4 và c4 ) được dùng để làm giá trị khởi tạo đầu vào cho decoder .

**Decoder**

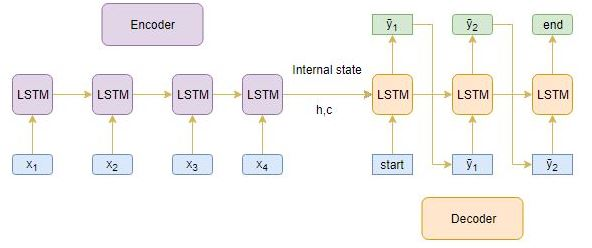
Decoder cũng là một mạng LSTM được huấn luyện để dự đoán từng từ trong câu output khi đã được biết trước từ trước đó trong mỗi timestep.



Token start và end là các token đặc biệt được thêm vào các chuỗi target y trước khi đưa vào decoder để mô hình training biết được đầu câu và cuối câu của target(câu summary trong data được huấn luyện).

### 3.2) Inference phrase (giai đoạn dự đoán)

Sau giai đoạn training, mô hình được thực nghiệm trên một câu mới. Nên ta cần encode và decode lại văn bản mới này.



Tương tự như training phrase, khi thực hiện dự đoán, văn bản mới được đưa vào encoder model để rút trích đặc trưng và ngữ cảnh, sau đó được đưa vào decoder model để thực hiện dự đoán.

Từng bước thực hiện của inference phrase:

1. Văn bản được đưa vào encoder model để rút trích đặc trưng, các hidden state(h) và cell state(c) trong mỗi timestep đều thay đổi, h và c ở timestep cuối cùng cũng chính là giá trị cho decoder.
2. Đưa token start như là input vào decoder model
3. Chạy mô hình decoder với mỗi timestep sẽ ứng với h và c khác nhau.
4. Decoder model sẽ dự đoán xác xuất của các từ sẽ xuất hiện tiếp theo sau token start và sẽ chọn từ có xác suất xuất hiện cao nhất.
5. Đưa từ được dự đoán trước đó vào timestep kế tiếp và sẽ update h và c theo timestep trước đó như là giá trị đầu vào.
6. Lặp lại bước 3->5 cho tới khi tiến tới token end hoặc vượt quá độ dài chuỗi summary cho phép .

## Attention mechanism:

Do kiến trúc encoder và decoder có một số điểm yếu khi giải quyết những câu quá dài do:

* Decoder cần xem xét toàn bộ các từ trong câu input dài để đưa ra dự đoán
* Khó cho encoder để có thể mã hoá những câu input dài thành một vector số nguyên.

Nên ta có cơ chế attention ( attention mechanism) dùng để khắc phục vấn đề này.

Ý tưởng chính của attention mechanism là thay vì nhìn và xem xét toàn bộ các từ trong câu input, ta có thể chỉ cần nhìn vào một phần quan trọng nhất trong câu input

Ví dụ

* **Source sequence:** “Which sport do you like the most?
* **Target sequence:** “I love cricket

Từ đầu ‘I’ trong câu output target sequence có mối liên hệ với từ ‘you’ ở vị trí thứ 4 trong câu input. Tương tự vậy, từ thứ 2 ‘love’ trong output cũng có mối liên hệ với từ ‘like’ trong câu input.

1. **Đánh giá kết quả trên ngữ liệu / Demo:**

Dưới đây là từng bước thực hiện đồ án text summerization:

1. **Thu thập Dataset (Amazon fined food reviews):**

Link: <https://www.kaggle.com/snap/amazon-fine-food-reviews>



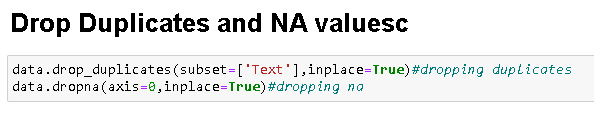
* Dataset gồm 500000 bài review về thức ở amazon đã được tóm tắt lại
* Thời gian từ 10-1999 đến 10-2012
* Mục tiêu : sử dụng dataset này để tạo ra một mô hình có thể tóm tắt lại các bài review về đồ ăn cho người tiêu dùng đọc dễ hiểu và nhanh chóng

1. **Tiền xử lý dữ liệu (Data preprocessing)**

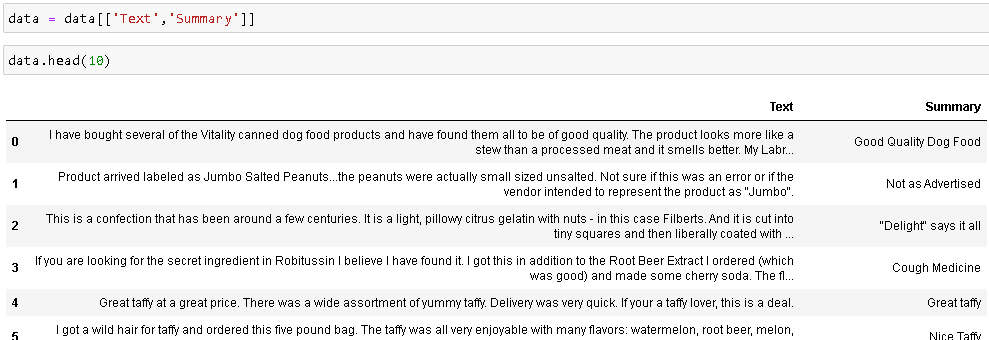
Sau khi đã thu thập xong tập dữ liệu (dataset), bước tiếp theo tụi em thực hiện là sẽ xử lý lại dữ liệu ban đầu (raw) này.

Các bước tiền xử lý bao gồm:

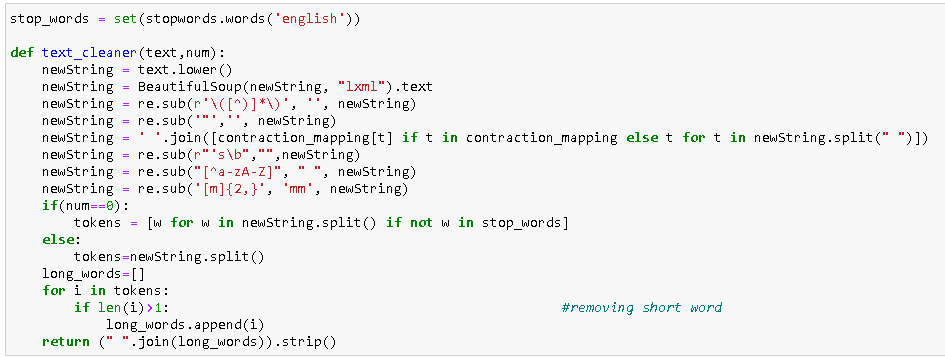
* 1. Bỏ đi các giá trị null



* 1. Lọc các cột không cần chỉ giữ lại 2 cột chính là text( văn bản ban đầu) và summary( văn bản sau khi tóm tắt)

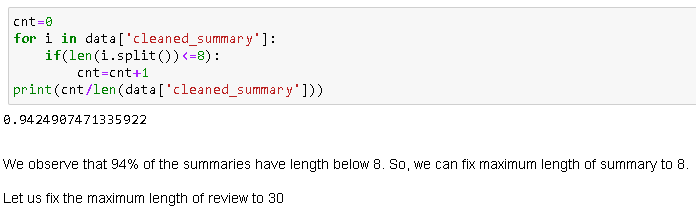


* 1. Tiến hành xử lý, làm gọn văn bản:



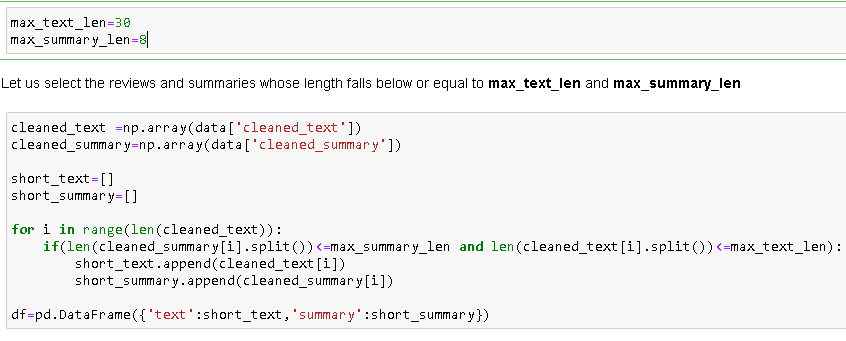
Quy trình này bao gồm các task :

* Chuyển chữ hoa thành chữ thường
* Bỏ đi html tags
* Ghi đầy đủ các từ, do một từ vẫn còn ghi tắt gây khó khăn trong quá trình training, này bước này ta sẽ chuyển các từ ghi tắt thành dạng đầy đủ
* Bỏ đi ‘s
* Bỏ các văn bên trong dấu ngoặc đơn ()
* Xoá chấm câu và kí tự đặc biệt
* Xoá từ dừng (stopwords)
* Xoá từ ngắn (short words)
  1. Thiết lập độ dài từ dài nhất cho văn bản summary và văn bản text ban đầu



Do ta thấy gần 95% tập dữ liệu có độ dài tập summary < 8 nên ta cũng sẽ giữ lại những sample có tập summary < 8 để rút ngọn tập dữ liệu và tránh những outlier không đáng.

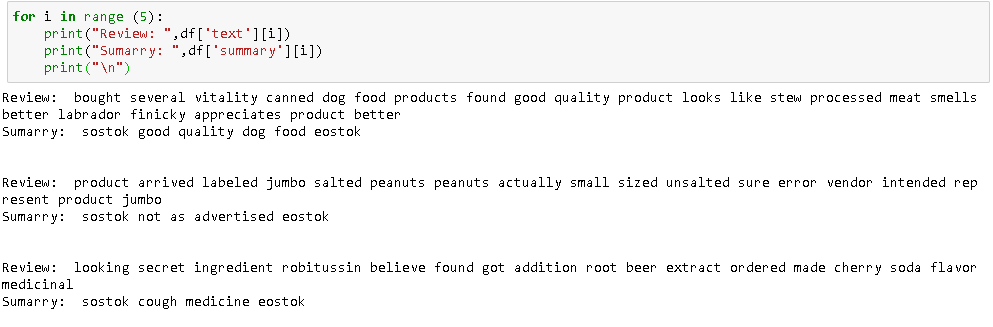
Ta cũng thiết lập độ dài tập text lớn nhất là 30 để làm nhẹ dataset



* 1. Thêm 2 token start và end vào tập summary



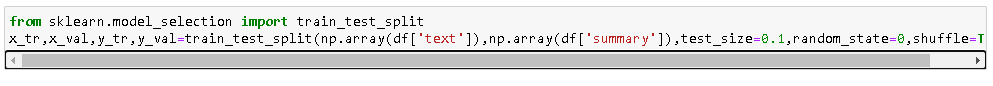
Ở đây ta thêm vào sostok tương ứng với start và eostok tương ứng end ở đầu và cuối các văn bản trong tập summary.



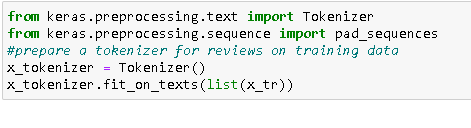
Văn bản ta có được như sau.

1. **Tokenization:**

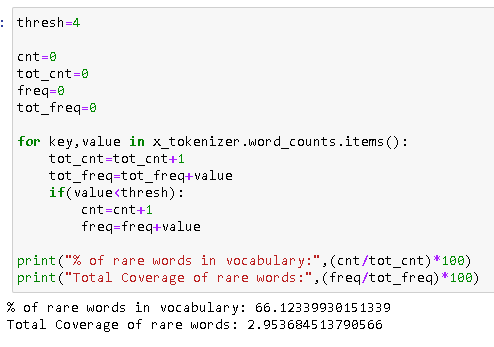
Ta chia tập train và tập validation theo tỉ lệ 9:1

****

Ta tiến hành xây dựng vocabulary chứa từng từ khác nhau (unique) trong các văn bản và chuyển các chuỗi từ trong văn bản thành các chuỗi số nguyên trước khi thực hiện quá trình training.



Ta sẽ loại bỏ đi các từ hiếm xuất hiện trong các sample.

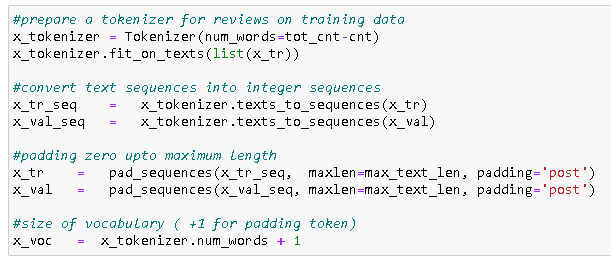


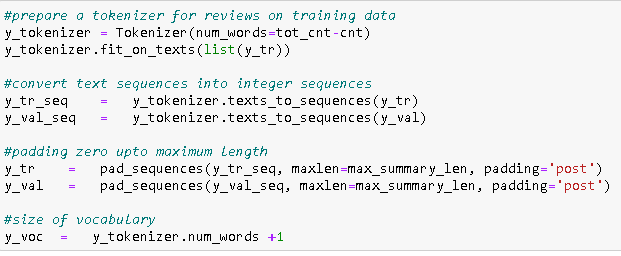
Ở đây ta đặt ngưỡng threshhold là tức là những từ nào xuất hiện dưới 4 sample thì sẽ được xem là từ hiếm, ta thấy tỉ lệ từ hiếm trong gói vocabulary là 66% nhưng tỉ lệ xuất hiện các từ hiếm này trong toàn bộ văn bản chỉ là dưới 3%, nên các từ hiếm này thực sự dư thừa và không cần thiết đặt vào vocabulary nên ta sẽ loại bỏ đi.

Tot\_cnt : kích thước vocabulary

Cnt : số lượng từ hiếm xuất hiện trong dưới 4 sample

Tot\_cnt –cnt cho ta được các từ phổ biến nhất



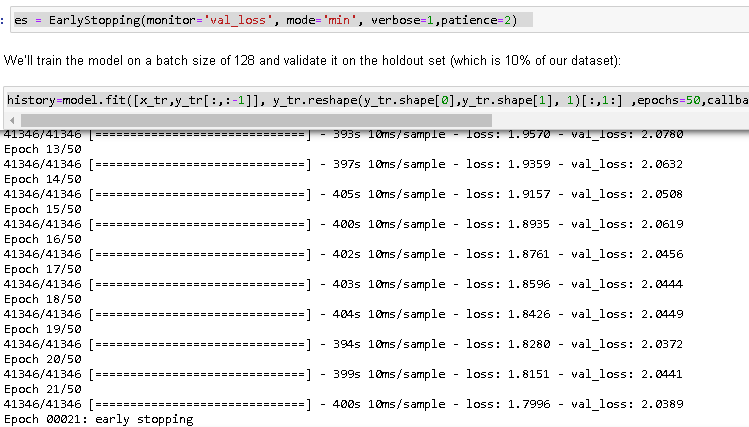


1. **Xây dựng mô hình:**

Trong giai đoạn training, ta sẽ xây dựng các mô hình encoder kết hợp embedding layer dùng để rút trích đặc trưng và đưa vào decoder để mô hình học các quy luật và ngữ cảnh của các từ theo thứ tự ( Ví dụ như từ này đứng trước thì từ sau tỉ lệ cao sẽ là từ nào).

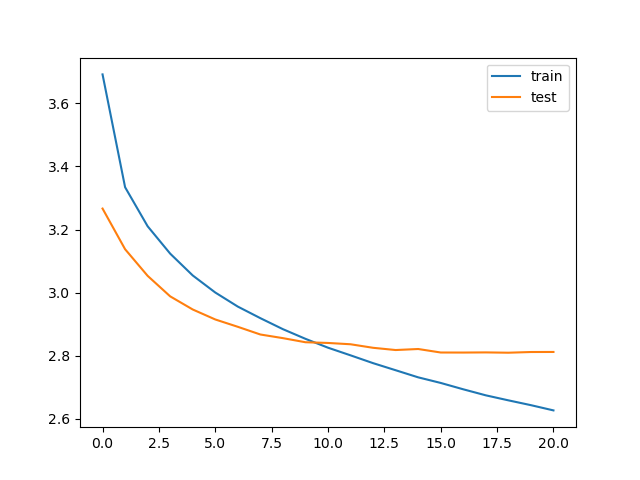
Ta cũng sẽ thêm một attention layer dùng để giúp mô hình encoder chỉ tập trung vào các từ keyword quan trọng chứ không tập trung vào toàn bộ các từ vì như vậy sẽ gặp nhiều khó khăn trong quá trình rút trích đặc trưng và chuyển đổi thành các vector số.

****

****

Ta tiến hành training và đặt điều kiện dừng sớm là khi giá trị loss của validation tăng mạnh.

1. **Đánh giá mô hình (evaluation):**

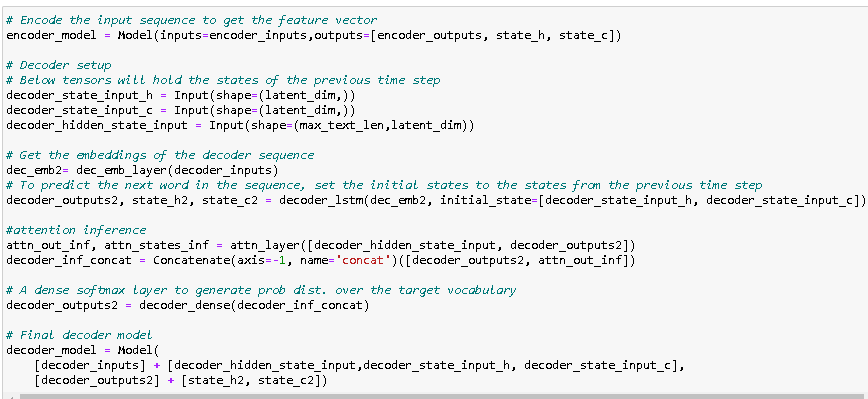


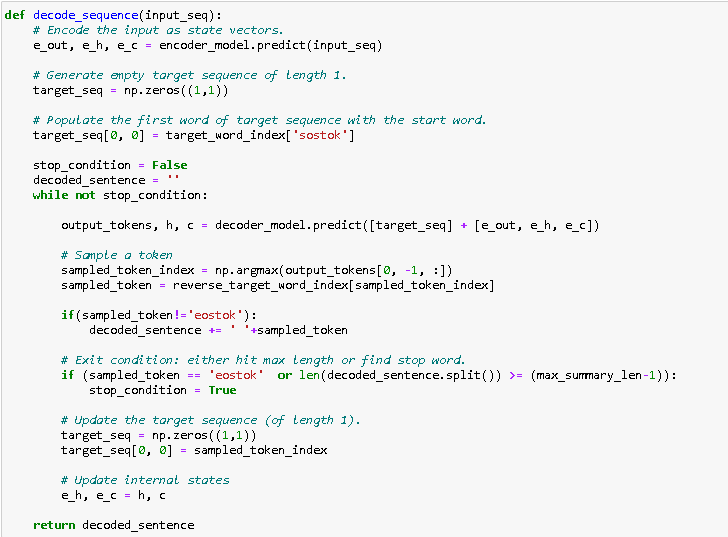
Em đã thử tuning các parameter : latent dim ( latent space của encoder model), embedding dim ( kích thước embedding layer) và rút ra kết luận rằng

Latent dim= 300 và embedding dim =100 đưa ra giá trị loss mô hình tốt nhất.

1. **Tiến hành dự đoán**

Đây cũng là giai đoạn inference (inference phrase) như trong lý thuyết ở trên



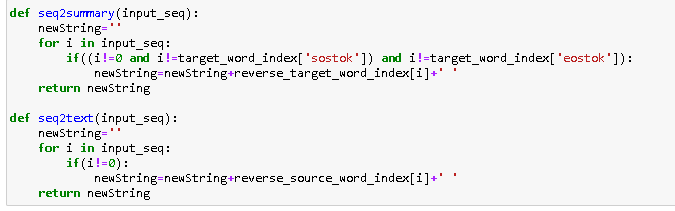


Ở đây là dùng thuật toán tham lam (argmax) để xác định từ có xác suất xuất hiện kế tiếp cao nhất.

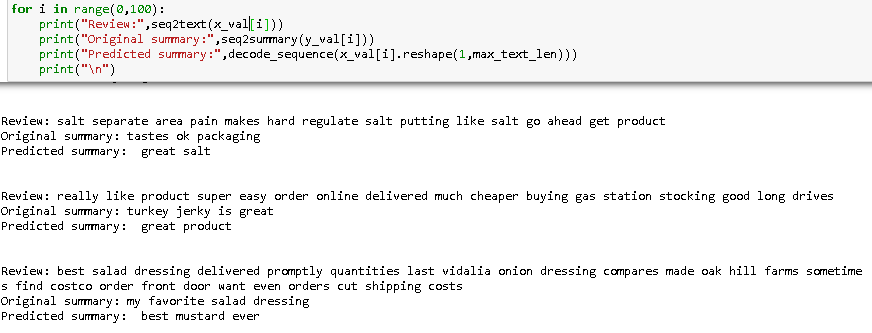
Ta sẽ thực hiện từng bước inference phrase để dự đoán như trên:

1. Văn bản được đưa vào encoder model để rút trích đặc trưng, các hidden state(h) và cell state(c) trong mỗi timestep đều thay đổi, h và c ở timestep cuối cùng cũng chính là giá trị cho decoder.
2. Đưa token start như là input vào decoder model
3. Chạy mô hình decoder với mỗi timestep sẽ ứng với h và c khác nhau.
4. Decoder model sẽ dự đoán xác xuất của các từ sẽ xuất hiện tiếp theo sau token start và sẽ chọn từ có xác suất xuất hiện cao nhất.
5. Đưa từ được dự đoán trước đó vào timestep kế tiếp và sẽ update h và c theo timestep trước đó như là giá trị đầu vào.
6. Lặp lại bước 3->5 cho tới khi tiến tới token end hoặc vượt quá độ dài chuỗi summary cho phép .

Do giá trị trả về của inference phrase vẫn là các vector số nên ta sẽ xây dựng 2 hàm để thành chuỗi string



Ta thực nghiệm trên 100 samples của tập validation



1. **Hướng phát triển:**

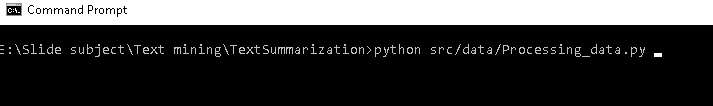
* Sử dụng **mô hình BiLSTM**
* Sử dụng **beam search strategy** trong quá trình decode để dự đoán từ tiếp theo thay vì thuật toán tham lam (argmax)
* Sử dụng **Bleu Score** để đánh giá mô hình

1. **Hướng dẫn sử dụng các file python đã được xây dựng cấu trúc thông qua môi trường Command Prompt:**

## File 1: File Processing\_data.py

Đây là file phục vụ quá trình tiền xử lý dữ liệu raw ở trong thư mục data:

* Để sử dụng cần chú ý đặt đúng đường dẫn trong command prompt đến đúng vị trí thư mục /TextSummarization
* Mở command prompt và dùng lệnh python src/data/Processing\_data.py để sử dụng
* File này sẽ xử lý dữ liệu và lưu dữ liệu mới ở đường dẫn TextSummarization/data/processed/data\_processed.csv



## File 2: File split\_data\_tokenization.py

Đây là file dùng để phục vụ quá trình chia data và tokenization, đây chỉ file chứa hàm tokenization để các file tham chiếu tới để sử dụng nên ở file này ta không cần gọi file trong command prompt.

## File 3: File visualize.py

Đây là file dùng để hiện đồ thị hiển thị distribution của văn bản text và văn bản summary. Nếu bạn muốn xem thử EDA của mô hình thì sử dụng file này

Cách sử dụng trong command prompt:

python src/visualization/visualize.py

## File 4: Train\_model.py

Đây là file dùng để training mô hình sử dụng kiến trúc seq2seq bao gồm các mô hình encoder và decoder.

Cách dùng:

* Để sử dụng cần chú ý đặt đúng đường dẫn trong command prompt đến đúng vị trí thư mục /TextSummarization
* Vào file train\_model.py sửa đường dẫn của src sau tương ứng máy của bạn



* Dùng lệnh python src/models/train\_model.py mặc định là sẽ thiết lập mô hình latent\_dim, embedding\_dim, epochs mặc định lần lượt là 300, 100, 50, nên nếu muốn thay đổi các thông số này ta dùng lệnh:

python src/models/train\_model.py –l {thông số latent\_dim} –e {thông số embedding\_dim} –ep {số epochs}

Ví dụ:

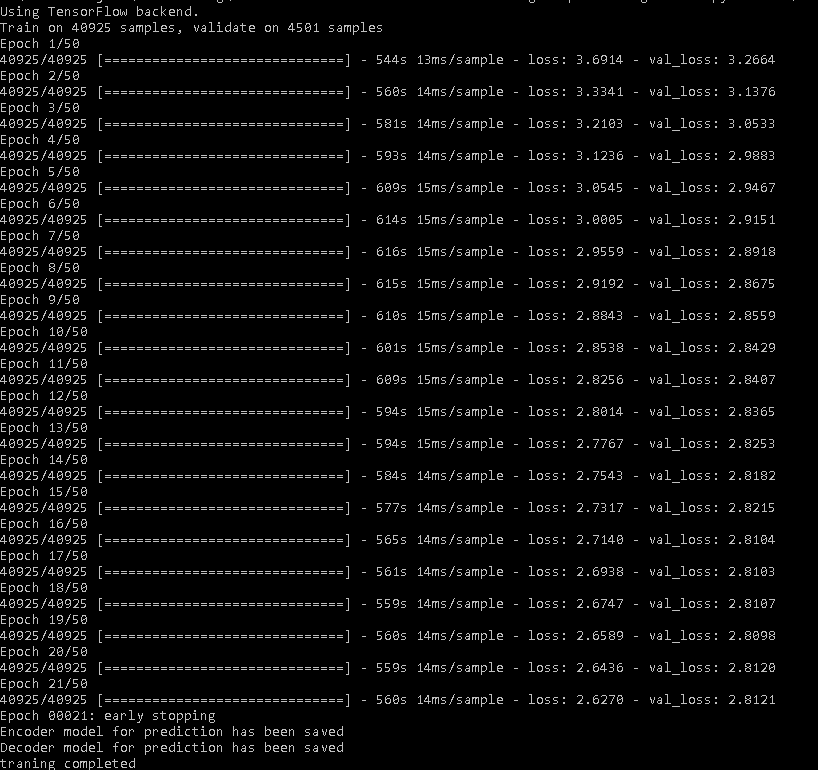


* Sau khi mô hình được training xong ( sau khoảng 3 giờ đồng hồ) các mô hình encoder và decoder sử dụng để dự đoán trong giai đoạn inference (inference phrase) sẽ được lưu lại trong file .h5

Các file đó gồm :

encoder\_model\_inference.h5 được lưu tại TextSummarization/models

decoder\_model\_inference.h5 được lưu tại TextSummarization/models

****

## File 5: predict\_model.py

Đây là file dùng để dự đoán sau khi đã training

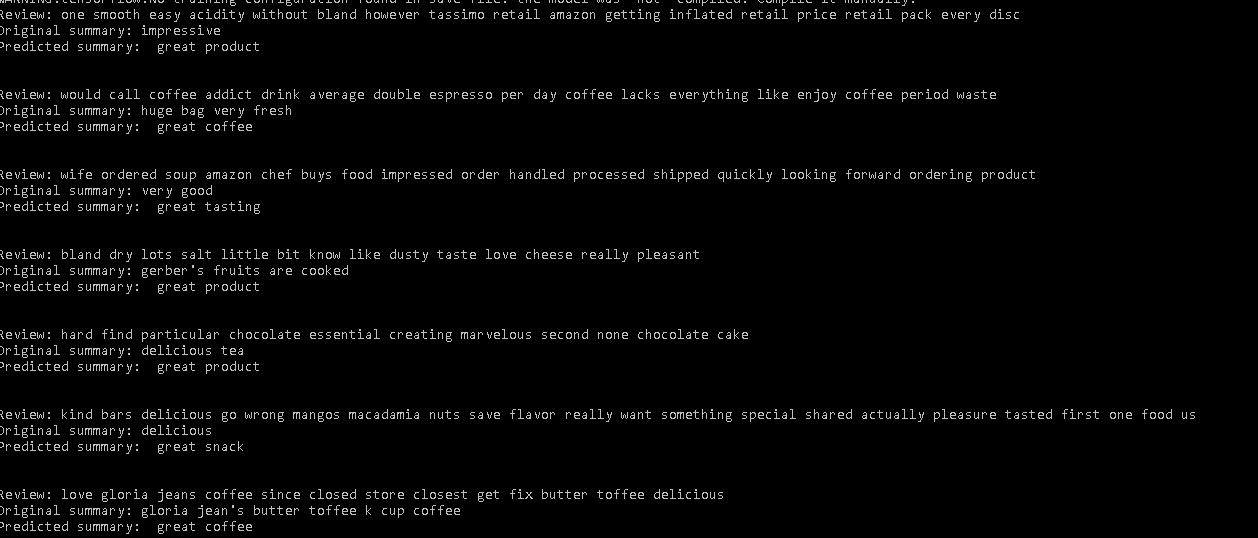
Cách dùng:

* Để sử dụng cần chú ý đặt đúng đường dẫn trong command prompt đến đúng vị trí thư mục /TextSummarization
* Vào file train\_model.py sửa đường dẫn của src sau tương ứng máy của bạn



* Dùng lệnh python src/models/predict\_model.py





Kết quả sẽ hiển thị 10 sample được dự đoán trong tập validation

1. **Tài liệu tham khảo:**

* <https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/>
* <https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/>
* <https://medium.com/jatana/unsupervised-text-summarization-using-sentence-embeddings-adb15ce83db1>
* <https://medium.com/sciforce/towards-automatic-text-summarization-extractive-methods-e8439cd54715>