TRƯỜNG ĐẠI HỌC ĐIỆN LỰC

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN**

**HỌC MÁY NÂNG CAO**

**ĐỀ TÀI:**

**TÌM HIỂU KỸ THUẬT ATTENTION**

**VÀ GIẢI CÁC BÀI TOÁN DỰ ĐOÁN**

|  |  |
| --- | --- |
|  |  |
| |  |  | | --- | --- | | **Sinh viên thực hiện** | **: TÔ XUÂN HIẾU**  **PHẠM TUẤN CƯỜNG**  **NGUYỄN NGỌC QUANG** | | **Giảng viên hướng dẫn** | **: PHẠM THỊ KIM DUNG** | | | **Ngành** | **: CÔNG NGHỆ THÔNG TIN** | | | **Chuyên ngành** | **: CÔNG NGHỆ PHẦN MỀM** | | | **Lớp** | **: D14LTCNPM1** | | | **Khóa** | **: 2019-2021** | | |  |
| ***Hà Nội, tháng 1 năm 2021*** |  |

**PHIẾU CHẤM ĐIỂM**

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **Chữ ký** | **Ghi chú** |
| **Tô Xuân Hiếu** |  |  |
| **Phạm Tuấn Cường** |  |  |
| **Nguyễn Ngọc Quang** |  |  |

Sinh viên thực hiện:

Giảng viên chấm:

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **Chữ ký** | **Ghi chú** |
|  |  |  |
|  |  |  |

**MỤC LỤC**

[LỜI MỞ ĐẦU 4](#_Toc61482770)

[CHƯƠNG I. KỸ THUẬT ATTENTION, CÀI ĐẶT CỦA TENSORFLOW VÀ ỨNG DỤNG 5](#_Toc61482771)

[1. GIỚI THIỆU VỀ KỸ THUẬT ATTENTION 5](#_Toc61482772)

[1.1 Cơ chế Attention 7](#_Toc61482773)

[1.2 BiRNN 10](#_Toc61482774)

[1.3 Thuật toán tìm kiếm chùm (Beam search) 12](#_Toc61482775)

[1.4 Mô hình đề xuất 14](#_Toc61482776)

[2. CÀI ĐẶT TENSORFLOW VÀ ỨNG DỤNG 1](#_Toc61482777)

[CHƯƠNG II: GIẢI BÀI TOÁN 7](#_Toc61482778)

[2. Dự đoán giá chứng khoán sử dụng LSTM 7](#_Toc61482779)

[2.1 Tìm hiểu về LSTM 7](#_Toc61482780)

[2.2 Bài toán dự đoán giá chứng khoán 9](#_Toc61482781)

[CHƯƠNG 3: KẾT LUẬN 12](#_Toc61482782)

[TÀI LIỆU THAM KHẢO 13](#_Toc61482783)

LỜI MỞ ĐẦU

Ngày nay sự phát triển mạnh mẽ của công nghệ thông tin và được ứng dụng rất nhiều trong lĩnh vực đời sống. Chúng ta không thể không nhắc đến Học Máy (Machine Learning) là một lĩnh vực của Trí tuệ nhân tạo (Artificial Intelligence - AI). Những công nghệ như là nhận diện khuôn mặt, giọng nói, vân tay,… đã và đang mang lại rất nhiều tiện ích cho con người.

Trong bài báo cáo này, chúng em sẽ tìm hiểu cơ bản về kỹ thuật Attention trong Deep Learning. Kỹ thuật này đã chứng minh được tính hiệu quả trong các nhiệm vụ dịch máy hay xử lý ngôn ngữ tự nhiên.

Để thực hiện và hoàn thành tốt bài báo cáo này, chúng em đã nhận được sự giúp đỡ và hướng dẫn rất tận tình của các thầy cô trường Đại Học Điện Lực. Em xin cảm ơn các thầy cô thuộc bộ môn chuyên ngành đã cung cấp cho chúng em các thông tin, kiến thức vô cùng quý báu và cần thiết trong suốt thời gian quá để chúng em có thể thực hiện và hoàn thành đề tài của mình. Đặc biệt chúng em xin chân thành cảm ơn cô Phạm Thị Kim Dung, người đã trực tiếp hướng dẫn chúng em trong thời gian thực hiện đề tài này.

Cuối cùng, xin chân thành cảm ơn các bạn trong ngành công nghệ thông tin đã ủng hộ, giúp đỡ, chia sẻ kiến thức, kinh nghiệm và tài liệu có được giúp chúng tôi trong quá trình nghiên cứu và thực hiện đề tài.

Do giới hạn về mặt thời gian và kiến thức cũng như kinh nghiệm thực tiễn nên đề tài không tránh khỏi những sai sót. Chúng em rất mong nhận được sự thông cảm của quý thầy cô và mong đón nhận những góp ý của thầy cô và các bạn.

**Em xin chân thành cảm ơn!**

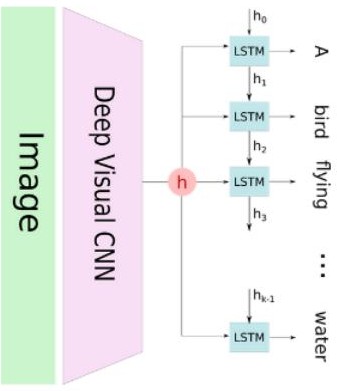
CHƯƠNG I. KỸ THUẬT ATTENTION, CÀI ĐẶT CỦA TENSORFLOW VÀ ỨNG DỤNG

1. GIỚI THIỆU VỀ KỸ THUẬT ATTENTION

Các quá trình xử lý mạng nơ-ron liên quan đến Attention đã được nghiên cứu nhiều trong lĩnh vực thần kinh học. Các nghiên cứu liên quan là hiện thực hóa Attention: rất nhiều loại động vật tập trung trong việc xác định thành phần cụ thể đầu vào để tính toán phản hồi phù hợp. Nguồn gốc có một lượng lớn ảnh hưởng đến khoa học thần kinh khi chúng ta phải lựa chọn những thông tin phù hợp nhất, hơn là việc sử dụng tất cả các thôn tin, chứa một lượng lớn các thông tin không phù hợp cho phản hồi nơ-ron. Ý tưởng tập trung vào các thành phần cụ thể của đầu vào được áp dụng trong các ứng dụng của học sâu như nhận dạng tiếng nói, dịch máy, lý giải và nhận dạng thị giác của đối tượng.

Bài toán mở đầu là: Sinh một tiêu đề cho ảnh.

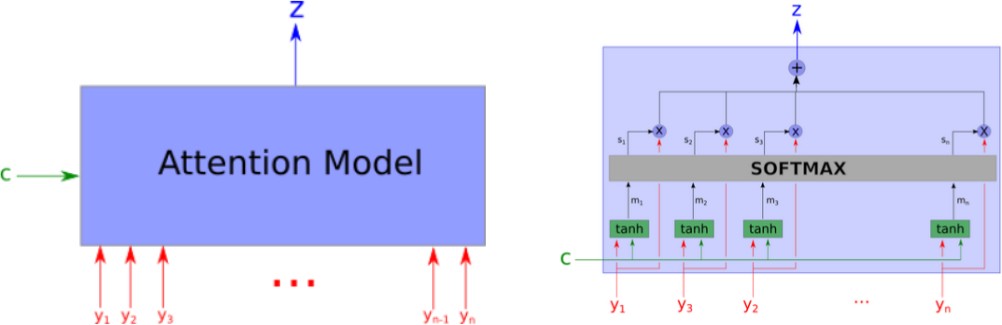
Một hệ thống cổ điển sinh tiêu đề có thể mã hóa hình ảnh, sử dụng một quá trình tiền xử lý CNN có thể đưa ra tầng ẩn h. Sau đó, nó có thể giải mã tầng ẩn bằng một mạng RNN, và sinh ra một đệ quy mỗi từ của tiêu đề.



Hình 1.1: Bài toán sinh tiêu đề

Vấn đề đặt ra với phương pháp này là khi mô hình cố gắng sinh ra một từ của tiêu đề, từ này thường chỉ mô tả duy nhất một phần của hình ảnh. Sử dụng toàn bộ biểu diễn của bức ảnh h để điều kiện hóa việc sinh mỗi từ sẽ không hiệu quả trong việc đưa ra một từ khác cho những thành phần khác của bức ảnh. Điều này lý giải cho lợi ích của kỹ thuật Attention.

Với kỹ thuật Attention, bức ảnh đầu tiên được chia thành n thành phần và chúng có thể tính toán với sự trình diễn CNN cho mỗi thành phần h1,...,hn. Khi RNN sinh ra một từ mới, kỹ thuật Attention tập trung vào những thành phần phù hợp của bức ảnh, vì thế quá trình giải mã chỉ sử dụng thành phần cụ thể của ảnh.



Hình 1.2: Sơ đồ mô hình Attention

Trước khi sử dụng Cơ chế Attention, các mô hình tóm tắt đều có cơ chế sử dụng Encoder-Decoder. Tại bước encoder, đầu vào của mạng RNN, LSTM, GRU là các vector được tạo ra từ mã hóa chuỗi từ với mô hình từ nhúng (word embedding). Pha decoder sử dụng một mạng RNN, LSTM hoặc GRU tương ứng để sinh ra một chuỗi từ mới dựa vào chuỗi đầu vào và các từ sinh ra phía trước. Trong mô hình tóm tắt văn bản tự động, thay vì tìm ra xác suất lớn nhất của mỗi từ sinh ra ở bước decoder, chúng ta tạo ra danh sách các từ ứng viên tại mỗi bước giải mã. Sau đó sử dụng giải thuật tìm kiếm chùm (Beam Search) để lựa chọn các từ ứng viên và kết nối danh sách các từ ứng viên đó lại thành một câu có điểm số cao nhất tạo ra một chuỗi tóm tắt.

1.1 Cơ chế Attention

1.1.1 Kiến trúc RNN Encoder-Decoder

Được đề xuất bởi Cho[12] và Sutskever[10] như là một kiến trúc hiện đại có thể học sự căn chỉnh và dịch ngay lập tức.

Trong Encoder-Decoder, một encoder đọc vào một câu - một chuỗi vector x = (x1,…,xTx) thành một vector c. Cách tiếp cận như sau:

ht = f(xt, ht-1)

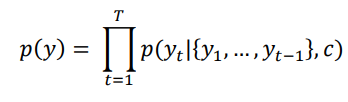
c = q({h1,…,hTx})

(1.1)

(1.2)

Trong đó ht là trạng thái ẩn tại thời điểm t, ht ∈ ℝn và c là vector được sinh ra từ một chuỗi các trạng thái ẩn. f và q là các hàm phi tuyến.

Pha decoder, được huấn luyện để dự đoán từ tiếp theo yt’ cho ngữ cảnh c và tất cả các từ dự đoán đằng trước {y1,…,yt’-1}. Hiểu theo cách khác decoder định nghĩa một xác xuất trên chuyển dịch y bằng việc phân tích xác suất liên kết thành thứ tự các điều kiện:



Trong đó y = (y1,…yTy).

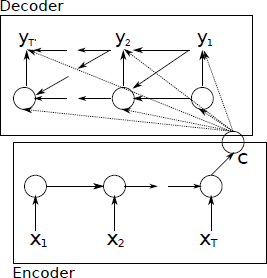
(1.3)

Với một mạng RNN, mỗi xác suất có điều kiện được mô hình bởi:



Trong đó g là hàm phi tuyến, yt là đầu ra và st là trạng thái ẩn của mạng RNN.

(1.4)



Hình 1.3: Minh họa kiến trúc của mạng Encoder-Decoder

1.1.2 Cơ chế Attention

Kiến trúc Encoder-Decoder có thể bị phá vỡ khi chuỗi đầu vào quá dài. Nguyên nhân là nếu ở mỗi bước nếu chỉ có một vector ngữ cảnh c giao tiếp giữa encoder và decoder, vector đó sẽ phải mã hóa cho toàn bộ chuỗi đầu vào, dẫn đến nó có thể bị tan biến khi nó xử lý chuỗi ký tự quá dài. Cơ chế Attention cho phép bộ giải mã tập trung vào một phần khác nhau từ đầu ra của encoder.

Định nghĩa mỗi xác suất có điều kiện như sau:



Trong đó:

Mỗi si là một trạng thái ẩn RNN tại thời điểm i, tính bằng công thức:

si = f(si-1, yi-1, ci)

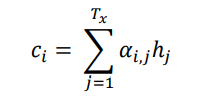
(1.5)

(3.6)

Điều này không giống với cách tiếp cận encoder-decoder, ở đây mỗi xác suất được điều kiện trên một ngữ cảnh riêng biệt ci cho mỗi từ mục tiêu yi.

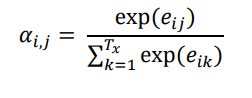
Vector ngữ cảnh ci phụ thuộc vào chuỗi trạng thái (h1,…,hTx) – để encoder ánh xạ câu đầu vào. Mỗi trạng thái hi chứa đựng thông tin của toàn bộ câu với một sự nhấn mạnh các thành phần xung quanh từ thứ i của câu đầu vào.

Ngữ cảnh c được tính toán như là trọng số tổng hợp của các trạng thái hi:



Trong đó: trọng số αi,j của mỗi trạng thái hj được tính như sau:

(1.7)

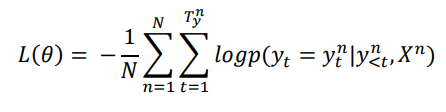


(1.8)

Với eij = a(si-1, hj) là hình thức căn lề tính điểm khả năng đầu vào xung quanh vị trí j và đầu ra tại vị trí i trùng nhau. Điểm số dựa trên trạng thái ẩn RNN si-1 và trạng thái gán nhãn hj của câu đầu vào.

Xác suất αij hay eij phản ánh độ quan trọng của trạng thái hj với trạng thái ẩn đằng trước si-1 để quyết định trạng thái tiếp theo si và đưa ra nhãn yi. Decoder quyết định thành phần của câu đầu vào để tập trung. Encoder toàn bộ thông tin câu thành một vector có độ dài cố định. Thông tin có thể trải dài thành chuỗi gán nhãn, có thể lựa chọn lấy lại bởi pha decoder tương ứng.

Toàn bộ mô hình được huấn luyện end-to-end bằng việc cực tiểu hóa xác suất có điều kiện:

 (1.9)

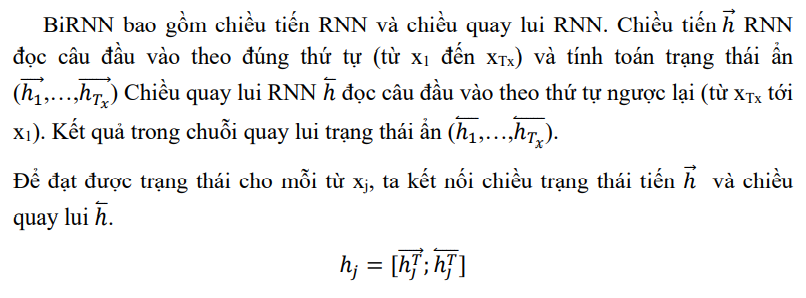
Trong đó: N là số lượng các cặp câu, Xn là câu đầu vào, ytn là nhãn đầu ra thứ t trong n cặp tương ứng.

1.2 BiRNN

Đối với rất nhiều nhiệm vụ gán nhãn chuỗi, việc truy cập vào thông tin tương lai rất có ích cho bối cảnh quá khứ. Ví dụ, khi phân loại một chữ viết tay, sẽ rất hữu ích khi biết chữ cái đến từ đằng sau cũng như chữ cái đến từ đằng trước nó. Tuy vậy, mạng RNN chuẩn xử lý chuỗi theo thứ tự thời gian, chúng bỏ qua tương lai của ngữ cảnh. Một giải pháp rõ ràng là thêm một cửa sổ trượt của ngữ cảnh tương lai vào mạng đầu vào. Tuy nhiên, nó làm tăng số lượng bộ trọng số đầu vào. Một cách tiếp cận khác là tạo sự trễ giữa các yếu tố đầu vào và mục tiêu, nhờ đó tạo cho mạng một số mốc thời gian của ngữ cảnh tương lai. Phương pháp này tuy vẫn duy trì được điểm mạnh của mạng RNN đối với sự biến dạng, nhưng nó vẫn yêu cầu phạm vi của ngữ cảnh phải xác định bằng tay. Hơn thế nữa nó đặt một gánh nặng không cần thiết lên mạng bằng cách buộc nó phải nhớ bản gốc đầu vào và bối cảnh trước đó của nó, trong suốt thời gian trễ. Trong các phương án trên, không có phương pháp nào loại bỏ sự không cân xứng giữa thông tin quá khứ và tương lai.

Mạng hai chiều RNN (BiRNN) được đưa ra như một giải pháp phù hợp. Ý tưởng cơ bản của BiRNN là trình bày mỗi chuỗi tiến và chuỗi lùi thành hai tầng ẩn hồi quy riêng biệt, cả hai đều được kết nối với nhau tới một tầng giống nhau. Cấu trúc này cung cấp cho mỗi tầng đầu ra với quá khứ hoàn chỉnh và bối cảnh tương lai cho mọi điểm trong chuỗi đầu vào, mà không phải di dời các đầu vào từ các mục tiêu phù hợp. BiRNN đã cải thiện kết quả trong các lĩnh vực khác nhau, chúng hoạt động tốt hơn RNN một chiều khi gán nhãn chuỗi.

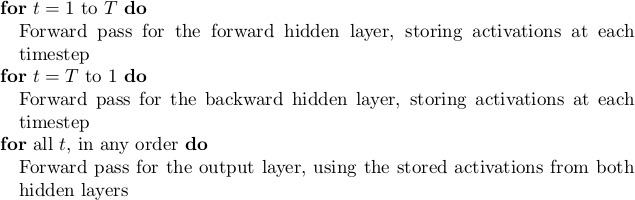
Thông thường RNN đọc câu đầu vào theo thứ tự bắt đầu của câu từ điểm đầu tiên x1 tới điểm cuối xTx. BiRNN được đề xuất để tổng hợp mỗi từ không chỉ đằng trước một từ mà còn từ đằng sau từ đó.



(1.10)

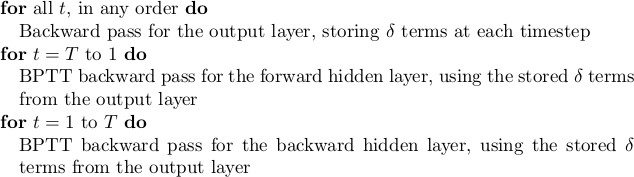
Trạng thái gán nhãn hj bao gồm thông tin tổng hợp của cả đằng trước và đằng sau từ đó. Phụ thuộc vào xu hướng RNN trình bày câu gần từ mà trạng thái ẩn hj sẽ tập trung xung quanh từ xj. Chuỗi trạng thái được sử dụng bởi decoder và model căn chỉnh để tính toán vector ngữ cảnh.

Pha tiến của tầng ẩn BiRNN giống như mạng RNN chuẩn, trừ việc chuỗi đầu ra được trình bày theo các hướng ngược nhau với hai lớp ẩn, tầng đầu ra không được cập nhật cho đến khi cả hai tầng ẩn đã được xử lý toàn bộ chuỗi đầu vào.

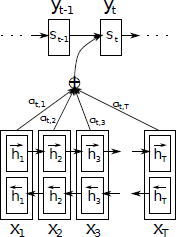


Hình 1.4: Pha tiến của mạng BiRNN

Tương tự quá trình quay lui như với một mạng RNN chuẩn trừ việc tất cả các tầng đầu ra δ được tính toán đầu tiên và sau đó quay trở lại hai tầng ẩn theo hướng ngược lại.



Hình 1.5: Pha quay lui của mạng BiRNN



Hình 1.6: Minh họa cơ chế Attention

1.3 Thuật toán tìm kiếm chùm (Beam search)

Trong mô hình tóm tắt, bộ giải mã được điều khiển bởi một câu đã được mã hóa để tạo ra câu mới. Tại mỗi bước lặp t, bộ giải mã cần đưa ra quyết định từ nào sinh ra từ thứ t trong câu. Vấn đề là chúng ta không biết chính xác chuỗi từ cần sinh ra để cực đại hóa xác suất có điều kiện tổng thể. Để giải quyết vấn đề này thuật tìm kiếm chùm sẽ được áp dụng. Thuật toán có độ rộng K sao cho tại mỗi bước đưa ra K đề xuất và tiếp tục giải mã với một trong số chúng.

Các mô hình phát triển giải quyết vấn đề sinh chuỗi thường hoạt động bằng sinh ra các phân phối xác suất thông qua từ điển các từ đầu ra. Chúng ta đối mặt với vấn đề này lúc làm việc với mạng nơ-ron truy hồi (RNN), khi mà văn bản được sinh ra như đầu ra. Ở tầng cuối cùng trong mạng nơ-ron có một mạng nơ-ron cho mỗi từ trong từ điển đầu ra và một hàm kích hoạt được sử dụng để đưa ra khả năng mỗi từ trong từ vựng là từ tiếp theo trong chuỗi.

Pha giải mã liên quan đến tìm kiếm thông qua tất cả các chuỗi đầu ra dựa trên khả năng của chúng. Kích thước tập từ vựng có thể tới hàng ngàn, hàng triệu từ. Vì thế vấn đề tìm kiếm là số mũ trong chiều dài cả chuỗi đầu ra và là vấn đề NP khó để hoàn tất tìm kiếm.

Thông thường, các phương pháp tìm kiếm thông minh được sử dụng để đưa ra chuỗi đầu ra được giải mã gần đúng cho sự dự đoán. Chuỗi ứng viên của các từ được ghi điểm dựa trên khả năng của chúng. Phương pháp phổ biến là tìm kiếm tham lam hoặc tìm kiếm chùm để định vị chuỗi ứng viên của văn bản.

Khác với các phương pháp thông minh, thuật toán tìm kiếm chùm mở rộng trên thuật toán tham lam và trả về danh sách phù hợp nhất các chuỗi đầu ra. thay vì tham lam chọn bước tiếp theo có khả năng nhất khi chuỗi được xây dựng, thuật toán tìm kiếm chùm mở rộng các khả năng có thể ở bước kế tiếp và giữa k trường hợp phù hợp nhất, trong đó k là tham số người dùng chỉ định và kiểm soát số lượng các chùm hoặc tìm kiếm song song thông qua chuỗi xác suất.

Thông thường độ rộng chùm là 1 tương ứng với thuật toán tìm kiếm tham lam và giá trị 5 hoặc 10 cho tiêu chuẩn chung của dịch máy. Độ rộng chùm kết quả lớn hơn dẫn tới hiệu suất tốt hơn của một mô hình vì các chuỗi ứng viên nhiều khả năng làm tăng khả năng kết hợp tốt hơn một chuỗi mục tiêu. Sự tăng hiệu suất này làm giảm tốc độ giải mã.

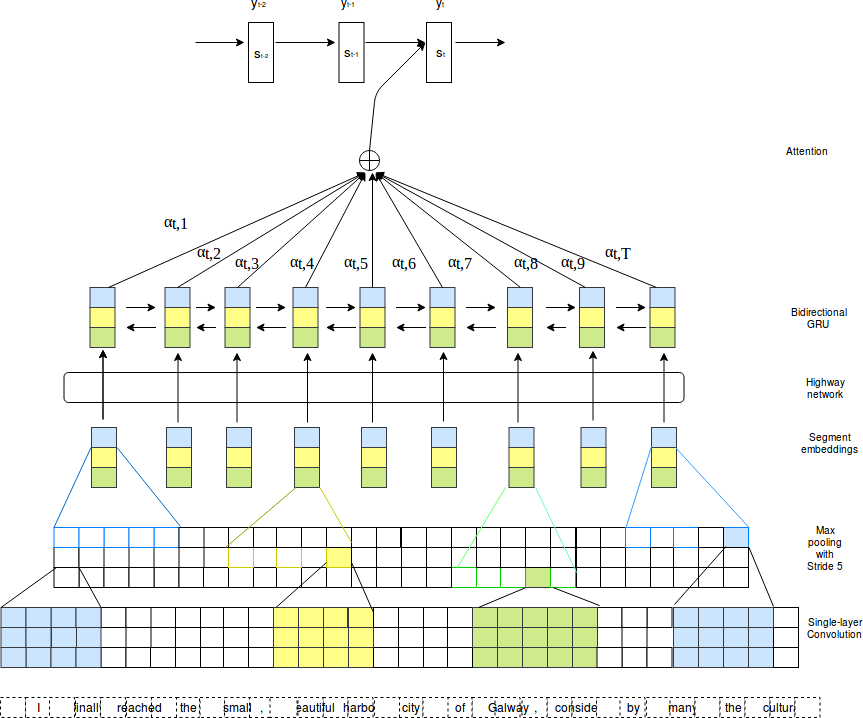
Cho (2014)[15] đã cài đặt một thuật toán tìm kiếm chùm tiêu chuẩn trong pha giải mã của dịch máy (Koehn, 2004) cho một hệ thống Encoder-Decoder trong GroundHog. Thuật toán chùm đã thành công trong việc giảm thiểu không gian tìm kiếm từ kích thước mũ sang kích thước đa thức.

Cho một pha encoder, một pha decoder và một đầu vào là x, chúng ta tìm kiếm chuỗi dịch tốt nhất ŷ = argmaxyp(y|x). Một nhóm các ngăn xếp được sử dụng để lưu lại các giả thuyết trong quá trình tìm kiếm. Kích thước chùm N được sử dụng để điều kiển không gian tìm kiếm bằng việc mở rộng đỉnh N giả thuyết trong ngăn xếp hiện tại. Với những cài đặt bên trên, phần dịch y được sinh ra từ bởi từ theo chiều từ trái sang phải. Ta định nghĩa một giả thuyết hoàn tất là câu chứa đầu ra EOS, trong đó EOS là từ đặc biệt chỉ ra kết thúc trong câu.

1.4 Mô hình đề xuất

Các mô hình học sâu áp dụng trong bài toán tóm tắt văn bản gồm: nhóm tác giả Rush [2] sử dụng mạng nơ-ron tích chập kết hợp với cơ chế attention. Sau đó, nhóm Chopra [3] sử dụng mạng nơ-ron tích chập và mạng RNN kết hợp với cơ chế attention. Nhóm Nallapati[19] sử dụng mô hình GRU và cơ chế attention đạt kết quả tốt hơn nhóm của Rush [2]. Hơn nữa mô hình của nhóm tác giả Nallapati[19] còn áp dụng được trên cả dữ liệu tóm tắt chứa nhiều câu văn. Điều mà nhóm tác giả Rush[2] và Chopra[3] chưa tiến hành thí nghiệm.

Do đó, tôi mở rộng nghiên cứu của nhóm tác giả Nallapati[19] bằng cách sử dụng mạng nơ-ron tích chập với mạng GRU kết hợp với cơ chế Attention. Câu đầu vào được đi qua các tầng Convolution rồi đến tầng mạng Highway. Đầu ra của tầng Hìghway sẽ là đầu vào của mạng GRU và đi vào cơ chế Attention.

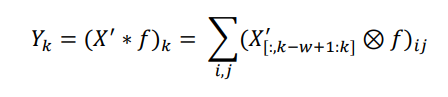


Hình 1.7: Mô hình đề xuất

Tầng nhúng (embedding): Giả sử ta có câu nguồn X = (x1, x2,... xTx) ∈ Rd x Tx. Trong đó: d là số chiều của một từ.

Tầng convolution:

Giả định ta có một hàm lọc f ∈ Rd x w với độ rộng là w, đầu tiên chúng ta áp dụng biên ở đầu và cuối của câu X. Do đó, biên của câu tạo thành X’ ∈ Rd x (Tx + w -1) là w- 1 từ. Ta áp dụng phép tích chập giữa X’ và f sao cho phần tử đầu ra thứ k được tính như sau:



Trong đó:

(3.11)

⊗ là phép nhân từng phần ma trận và phép toán \* là phép tích chập. X’[:,k-w+1:k] là một tập con của X’ chứa tất cả các hàng nhưng chỉ chứa w cột kề bên. Kiểu lựa chọn lề như vậy gọi là một nửa tích chập (half convolution). Điều này đảm bảo chiều dài của đầu ra là Y ϵ R1xTx.

Bên trên, ta minh họa trường hợp một bộ lọc tích chập cố định. Để trích chọn các mẫu thông tin với chiều dài khác nhau, ta đưa một tập các bộ lọc với chiều dài khác nhau. Cụ thể hơn, ta sử dụng một tập các bộ lọc F = {f1, , fm}. Trong đó, fi

= Rd x i x ni là một tập của các ni bộ lọc với độ rộng i. Mô hình của tôi sử dụng m=5, do đó có thể trích chọn được 5 gram chiều dài. Đầu ra của tất cả các hàm lọc được xếp chồng lại, đưa ra một sự biểu diễn đơn giản Y ϵ RNxTx, trong đó số chiều của mỗi cột được cho bởi tổng các bộ lọc N = ∑i=1m ni. Cuối cùng tầng kích hoạt được áp dụng theo từng phần tử của sự trình diễn.

Tầng max pooling:

Đầu ra của tầng convolution đầu tiên được phân thành các cụm với chiều dài là s, và tầng max pooling được áp dụng với mỗi cụm không giao nhau. Thủ tục lựa chọn các đặc trưng nổi bật nhất đưa ra một phân đoạn nhúng. Mỗi tầng nhúng là

một tóm tắt của một đoạn riêng biệt (hoặc chồng chéo) trong câu đầu vào. Điều này hoạt động như đơn vị ngôn ngữ bên trong từ tầng hiện tại đến tầng trên.

Sự rút ngắn sự biểu diễn nguồn theo s-fold: Y’ ϵ RNx(Tx/s). Theo kinh nghiệm, tôi sử dụng s=5.

Mạng highway (nhóm tác giả Srivastava 2015 [14]):

Mạng highway được áp dụng khi số tầng của mô hình học sâu tăng lên cùng với đó là sự tăng độ phức tạp tính toán. Mạng highway có thể sử dụng với hàng trăm tầng được huấn luyện trực tiếp cùng với phương pháp tối ưu SGD và các biến thể của hàm kích hoạt.

Chuỗi ma trận nhúng sau khi qua tầng max pooling của mạng nơ-ron tích chập được đưa đến mạng highway. Ý nghĩa tiềm ẩn là mạng highway chuyển đổi đầu ra của tầng max pooling thành các khoảng ngữ nghĩa, giúp các đặc trưng được học chính xác. Mạng này chuyển đổi đầu vào x với một cơ chế cổng để điều chỉnh thông tin theo luồng:

 (1.12)

Đẩu ra của tầng mạng highway được đưa tới mạng GRU hai chiều.

Cuối cùng, một tầng mạng hướng tiến tính toán điểm số attention của mỗi từ mục tiêu để sản sinh cho mỗi cụm thể hiện đầu vào.

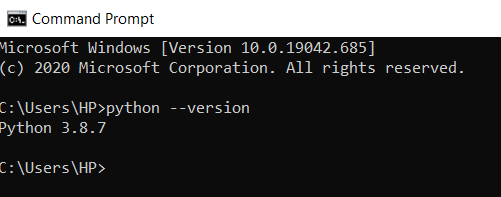
1. CÀI ĐẶT TENSORFLOW VÀ ỨNG DỤNG

Tensorflow là một thư viện mã nguồn mở do nhóm Google Drain nghiên cứu và phát triển. Sau đó được phát hành theo giấy phép mã nguồn mở Apache 2.0 vào ngày 9/11/2015. Tensorflow được sử dụng nhiều trong trí tuệ nhân tạo, machine learning,… Tensorflow 2.4 là phiên bản mới nhất ở thời điểm hiện tại. Vậy cài đặt như thế nào, chúng ta cùng tìm hiểu dưới đây nhé.

Tensorflow hỗ trợ cho cả PC và Mobile, đồng thời nó hỗ trợ một số ngôn ngữ lập trình như là Python, C/C++, Java, Go, Javascript,…

Và ở đây chúng ta sẽ cài đặt Tensorflow với ngôn ngữ lập trình là Python. Trước khi cài đặt thì chúng ta cần phải kiểm tra *version* Python hiện tại trên PC của mình bằng câu lệnh sau.





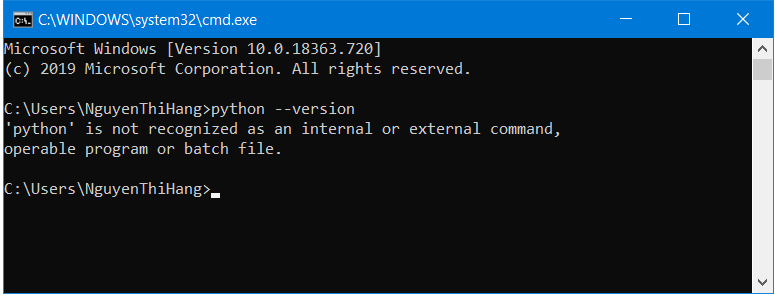
Theo như kết quả hiện thị trên màn hình thì PC của mình đang sử dụng Python 3 có version là 3.8.7.

Hiện tại thì **Tensorflow 2.4** chỉ hỗ trợ cho Python3 từ 3.5 → 3.8. Nếu trong PC của bạn đã có Python nhưng khác phiên bản thì sẽ không cài đặt được Tensorflow 2.4 và sẽ nhận được thông báo lỗi như sau:

ERROR: Could not find a version that satisfies the requirement tensorflow (from versions: none) ERROR: No matching distribution found for tensorflow

Bạn có thể fix lỗi đó bằng cách cài lại **Python 3.5.x** đến **3.8.x.**!

Nếu khi gõ lệnh python –version mà nhận được lỗi như hình bên dưới



Theo như lỗi thông báo ở trên thì PC có thể chưa có cài đặt Python hoặc biến môi trường của Python thiết lập chưa đúng.

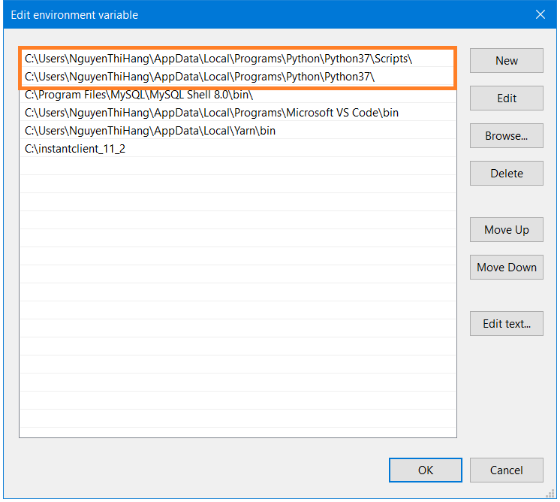
Trường hợp chưa cài đặt Python

Bạn vào trang chủ của [**Python**](https://www.python.org/downloads/)và chọn phiên bản[**Python**](https://www.python.org/downloads/)[**2x**](https://www.python.org/downloads/)hoặc [**3x**](https://www.python.org/downloads/)để tải về và cài đặt. Theo quan điểm cá nhân của mình thì nên xài [**Python**3x](https://www.python.org/downloads/) do nó tương thích nhiều với những **framework**mới. Còn nếu bạn vẫn muốn làm việc với những **framework**cũ thì [**Python 2x**](https://www.python.org/downloads/) sẽ là sự lựa chọn.



Sau khi cài đặt xong rồi thì bạn mở mới **command line** và gõ lệnh **python –version**lại nhé. Nếu vẫn bị lỗi như hình 2 thì bạn thử restart PC và kiểm tra lại. Nếu vẫn còn lỗi thì xem bước tiếp theo ở bên dưới nhé.

* **Trường hợp đã cài đặt nhưng vẫn bị lỗi**
* Lúc này bạn nhấp phải **My Computer** chọn **Properties**(hoặc vào **Control Panel / System and Security / System**).
* Nhìn về phía bên trái chọn **Advances System Settings**
* Một Popup mới hiện ra. Click vào button **Environment Variables ….**
* Trong group **User variables for {x}**. Bạn chú ý đến **Path**. Hãy Click chọn **Path**và Click vào button **Edit**.
* Bạn Click **New**và thêm giá trị cho nó chính là đường dẫn thư mục cài đặt của Python. Theo hình hình bến dưới thì mình đã cài Python 3x vào thư mục *C:UsersAppDataLocalProgramsPythonPython37*



Xong rồi, click vào button **Ok** để thiết lập của bạn mới có giá trị. Sau khi thiết lập xong thì bạn mở mới **command line** khác và kiểm tra lại việc thiết lập bằng lệnh python –version. Nếu vẫn còn lỗi thì hãy restart PC lại để refresh biến môi trường vừa mới thiết lập nhé.

Sau khi đã cài đặt được **Python** đúng phiên bản thì mình đã hoàn thành được **80%** việc cài đặt Tensorflow rồi đó. Công việc còn lại là cài đặt thêm một số thư viện liên quan và Tensorflow thôi. Chúng ta tiếp tục nào !!

Upgrade pip và virtualenv

Tiếp đến mình cần upgrade thư viện Pip và virtualenv trong python lên phiên bản mới nhất. Tensorflow cần **pip >=19.0** và **virtualenv**



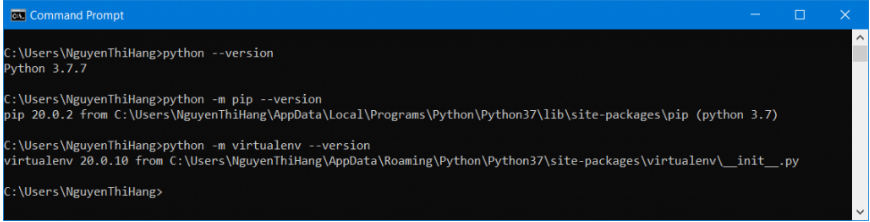
Ghi chú: Nếu bạn nhận được thông báo lỗi :

ERROR: Could not install packages due to an EnvironmentError: [WinError 5] Access is denied: ‘C:xxxxpip.exe’  
Consider using the --user option or check the permissions.

Thì bạn thêm **–user** vào câu lệnh bên trên hoặc vào thư mục bị **permissions**– rồi thay đổi quyền cho nó nhé.



Sau khi upgrade pip và virtualenv thành công, mình kiếm tra phiên bản của nó.



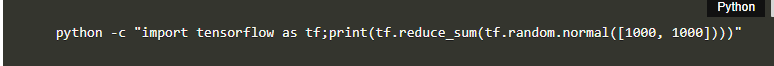
Cuối cùng thì mình sẽ cài đặt Tensorflow. Tensorflow có 3 phiên bản

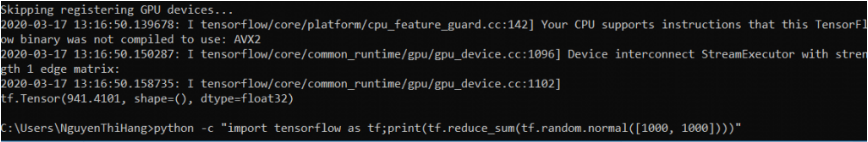
* tensorflow —Phiên bản mới nhất – hiện là 2x
* tf-nightly —Phiên bản đang phát triển.
* tensorflow==1.15 —Chỉ định phiên bản mà mình muốn cài đặt. Ở đây là phiên bản TensorFlow 1.15.

Ở đây mình cài đặt phiên bản mới nhất.



Sau khi cài đặt Tensorflow thành công. Mình cần kiểm tra lại xem nó có cài đặt đúng hay chưa? Từ command line. Mình chay đoạn script sau:

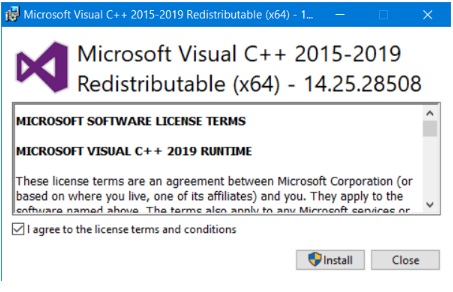




Nếu không đạt được kết quả như trên mà xuất hiện lỗi:

….ImportError: Could not find the DLL(s) ‘msvcp140\_1.dll’. TensorFlow requires that these DLLs be installed in a directory that is named in your %PATH% environment variable. You may install these DLLs by downloading “Microsoft C++ Redistributable for Visual Studio 2015, 2017 and 2019” for your platform from this URL: https://support.microsoft.com/help/2977003/the-latest-supported-visual-c-downloads.

Thì bạn cần cài đặt thêm thư viện [*Microsoft Visual C++ Redistributable for Visual Studio 2015, 2017, and 2019*](https://support.microsoft.com/en-us/help/2977003/the-latest-supported-visual-c-downloads) nữa là ok nhé.



Vậy là chúng đã cài đặt thành công **Tensorflow 2.4.0** trên **Window 10** với ngôn ngữ lập trình phổ biến là **Python**với phiên bản **3.8.7**.

CHƯƠNG II: GIẢI BÀI TOÁN

2. Dự đoán giá chứng khoán sử dụng LSTM

2.1 Tìm hiểu về LSTM

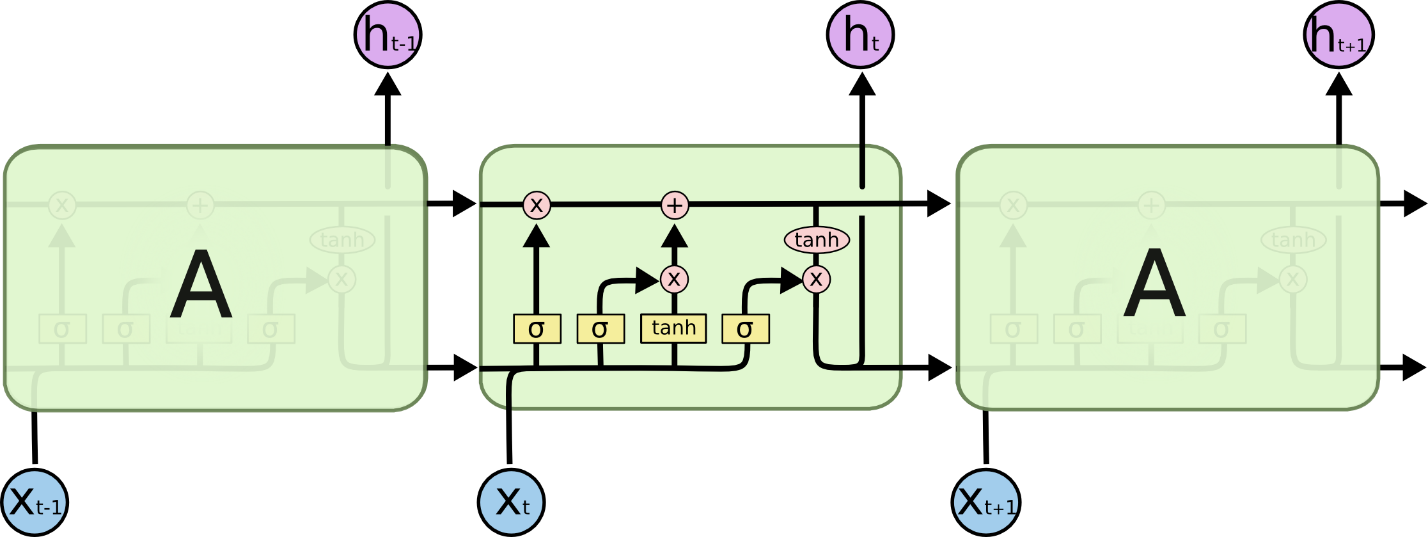
Trong bài báo cáo này, chúng ta sẽ xây dựng mô hình đơn giản để áp dụng vào tập dữ liệu giá chứng khoán. Mục tiêu là chúng ta sẽ dự đoán chỉ số S&P 500 sử dụng LSTM.

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)](http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf), 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.

Mọi mạng hồi quy đều có dạng là một chuỗi các mô-đun lặp đi lặp lại của mạng nơ-ron. Với mạng RNN chuẩn, các mô-dun này có cấu trúc rất đơn giản, thường là một tầng *tanh*.

LSTM cũng có kiến trúc dạng chuỗi như vậy, 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.



Ý tưởng cốt lõi của LSTM

Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

Trạng thái tế bào là một dạng giống như băng truyền. Nó chạy xuyên suốt tất cả các mắt xích (các nút mạng) và chỉ tương tác tuyến tính đôi chút. Vì vậy mà các thông tin có thể dễ dàng truyền đi thông suốt mà không sợ bị thay đổi.



LSTM có khả năng bỏ đi hoặc thêm vào các thông tin cần thiết cho trạng thái tế báo, chúng được điều chỉnh cẩn thận bởi các nhóm được gọi là cổng (gate).

Các cổng là nơi sàng lọc thông tin đi qua nó, chúng được kết hợp bởi một tầng mạng sigmoid và một phép nhân.



Tầng sigmoid sẽ cho đầu ra là một số trong khoản [0, 1][0,1], mô tả có bao nhiêu thông tin có thể được thông qua. Khi đầu ra là 00 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 11 thì có nghĩa là cho tất cả các thông tin đi qua nó.

Một LSTM gồm có 3 cổng như vậy để duy trì và điều hành trạng thái của tế bào.

2.2 Bài toán dự đoán giá chứng khoán

Chuẩn bị dữ liệu:

Nguồn dữ liệu: <https://github.com/AlexBlack2202/alexmodel/blob/master/GSPC.csv>

Gồm 17114 dòng, 7 cột

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Date | Open | High | Low | Close | Adj Close | Volume |
| 0 | 1950-11-09 | 19.790001 | 19.790001 | 19.790001 | 19.790001 | 19.790001 | 1760000 |
| 1 | 1950-11-10 | 19.940001 | 19.940001 | 19.940001 | 19.940001 | 19.940001 | 1640000 |
| 2 | 1950-11-13 | 20.010000 | 20.010000 | 20.010000 | 20.010000 | 20.010000 | 1630000 |
| 3 | 1950-11-14 | 19.860001 | 19.860001 | 19.860001 | 19.860001 | 19.860001 | 1780000 |
| 4 | 1950-11-15 | 19.820000 | 19.820000 | 19.820000 | 19.820000 | 19.820000 | 1620000 |

Cột đầu tiên là ngày, sau đó là giá mở cửa, giá giao dịch cao nhất, giá giao dịch thấp nhât, giá đóng cửa, giá đóng cửa đã điều chỉnh, khối lượng giao dịch.

Đầu tiên, khai báo các thư viện cần thiết:

import numpy as np

import pandas as pd

from subprocess import check\_output

from keras.layers.core import Dense, Activation,Dropout

from keras.layers.recurrent import LSTM

from keras.models import Sequential

from sklearn.cross\_validation import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

import matplotlib.pyplot as plt from numpy

Load dataset:

file\_name ='GSPC.csv' prices\_dataset = pd.read\_csv(file\_name, header=0)

Chia tập train và test:

Chia dữ liệu thành 2 phần với 80% là train và 20% còn lại là test

seq\_len = 20

seq\_prices\_dataset = generate\_data(prices\_dataset\_norm,seq\_len)

x\_train, y\_train, x\_test, y\_test = generate\_train\_test(seq\_prices\_dataset, 0.8) print('x\_train.shape = ',x\_train.shape)

print('y\_train.shape = ', y\_train.shape)

print('x\_test.shape = ', x\_test.shape)

print('y\_test.shape = ',y\_test.shape)

Kết quả:

x\_train.shape = (13675, 19, 4)

y\_train.shape = (13675, 4)

x\_test.shape = (3419, 19, 4)

y\_test.shape = (3419, 4)

Xây dựng mô hình sử dụng Keras

model = Sequential()

model.add(LSTM(

input\_dim=4,

output\_dim=50,

return\_sequences=True))

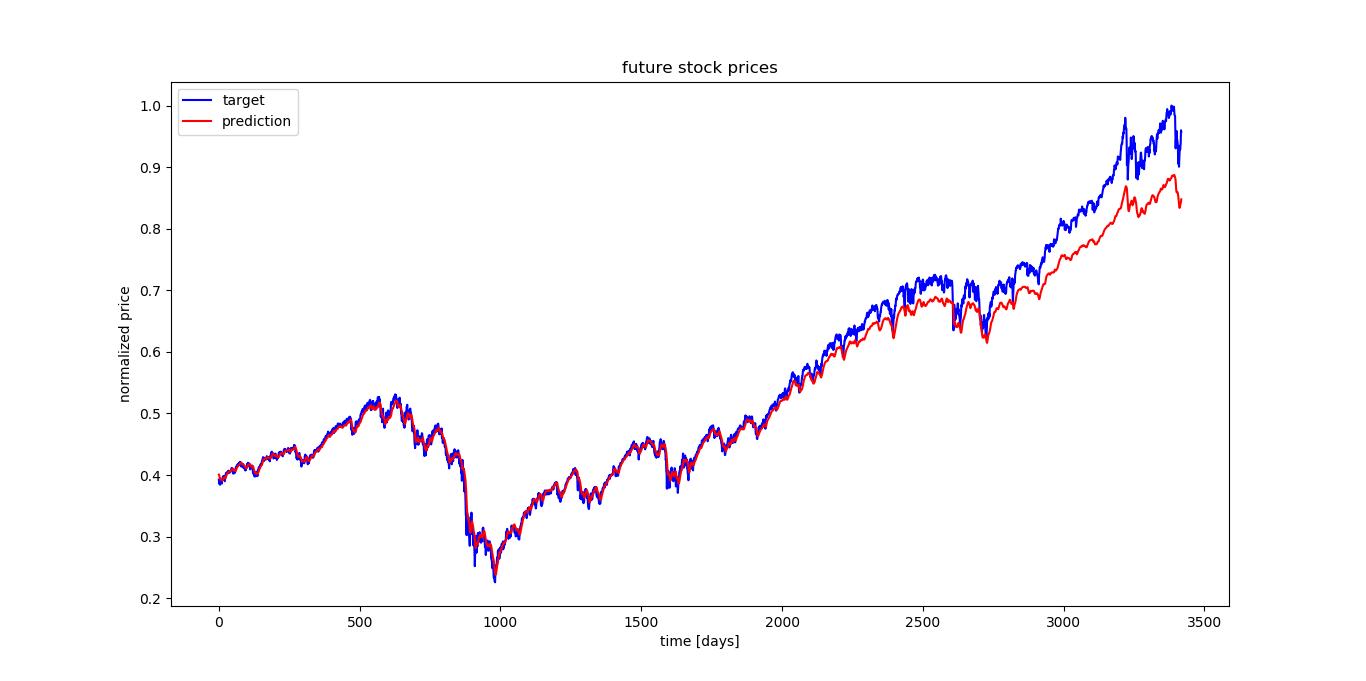
model.add(Dropout(0.2)) model.add(LSTM( 100, return\_sequences=False)) model.add(Dropout(0.2)) model.add(Dense( output\_dim=4))

model.add(Activation('linear')) model.compile(loss='mean\_squared\_error', optimizer='adam', metrics=['accuracy'])

checkpoint = ModelCheckpoint(filepath=sp500.h5', verbose=1, save\_best\_only=True)

hist = model.fit(x\_train, y\_train, epochs=300, batch\_size=128, verbose=1, callbacks=[checkpoint], validation\_split=0.2)

Kết quả:



Kết quả của mô hình trông khá tốt, về hình dạng thì khá tương đồng với kết quả. Chúng ta có thể cải tiến model bằng cách nâng số lượng layter/ hidden node.

CHƯƠNG 3: KẾT LUẬN

**-** Mạng LSTM

Với kỹ LSTM là một kỹ thuật rất mạnh mẽ nhưng khi áp dụng vào dataset này thì độ chính xác chưa được cao.

- Phương án giải quyết:

Cần áp dụng các thuật toán cũng như kỹ thuật học máy khác để xử lý bộ dữ liệu này. Tìm ra các thuật toán và kỹ thuật máy học đem lại độ chính sác cao nhất, tối ưu nhất.

TÀI LIỆU THAM KHẢO

[1] <https://123doc.net/document/5707719-tom-tat-van-ban-su-dung-cac-ky-thuat-trong-deep-learning.htm#>

[2] <https://www.phamduytung.com/blog/2018-11-10-stock-prediction_v1/>

[3] <https://wikipedia.org>