Đại học quốc gia Thành phố Hồ Chí Minh

Trường Đại học Công nghệ thông tin

Khoa công nghệ phần mềm

Xây dựng chatbot sử dụng Machine Learning

Giảng viên hướng dẫn Huỳnh Tuấn Anh

Sinh viên thực hiện Phạm Hùng Vỹ - 15521037
Lê Nhựt Vinh - 15521016

13/06/2019

Version: 1.0

Lời cảm ơn

Nhóm chúng em hoàn thành được tốt đồ án môn học này, không thể không nói đến công lao của thầy Huỳnh Tuấn Anh. Chúng em xin chân thành cảm ơn thầy đã tận tâm hướng dẫn chúng em. Bên cạnh đó, nhóm cũng xin gửi lời cảm ơn chân thành đến các anh chị khóa trên, các bạn trong và ngoài lớp đã sẵn lòng chia sẻ tài liệu cũng như kinh nghiệm từng trải của bản thân để nhóm chúng em học tập và tránh mắc những sai lầm, tiết kiệm được thời gian trong quá trình thực hiện đồ án.

Tuy nhiên, do kiến thức và khả năng của chúng em còn nhiều hạn chế, do đó không tránh khỏi những thiếu sót, yếu kém. Chúng em rất mong nhận được những ý kiến đóng góp quý báu của thầy cô và các bạn học cùng lớp để đồ án được hoàn thiện hơn và rút ra được kinh nghiệm. Sau cùng, chúng em xin kính chúc quý thầy cô ở Khoa Công nghệ Phần mềm, đặc biệt là thầy Huỳnh Tuấn Anhthật dồi dào sức khỏe để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau. Nhóm xin chân thành cảm ơn!

Mục lục

1 Tổng quan

1.1 Giới thiệu

Công nghệ truyền thông đang tiến bộ rất nhanh. Và ngày nay, hầu hết mọi người thích kết nối thông qua tin nhắn văn bản hoặc ứng dụng nhắn tin. Để làm cho cuộc sống thuận tiện hơn, các tổ chức và ngành công nghiệp đang thúc đẩy truyền thông bằng cách phát triển và đầu tư vào chatbot.

Chatbots là các dịch vụ được lập trình trên máy tính có thể tương tác như con người thông qua giao diện trò chuyện, cả về mặt văn bản và thính giác. Còn được gọi là talkbots, smartbots, bot, chatterbots hoặc tác nhân tương tác, các chương trình này nhằm giao tiếp với người dùng và hành xử như thể một con người thực sự đứng sau cuộc trò chuyện.

Hiện tại, các bot này được tìm thấy trong các giải pháp trò chuyện hoặc nhắn tin lớn như Facebook Messenger, Kik, Slack, WeChat, Line, LiveChat và Telegram. Với sự phổ biến của công nghệ, ngay cả những người hầu như không chú ý đến công nghệ hiện đại có lẽ đã tương tác với các bot này nhiều lần rồi.

Hai loại Chatbots

Chatbots không phải là mới. Tuy nhiên, việc sử dụng các bot đã thu hút các ngành công nghiệp trong vài năm qua. Được thành lập lần đầu tiên vào những năm 1960, chatbot đã đi một chặng đường dài từ sự phát triển ban đầu của nó. Có hai loại chatbot. Loại chatbot phổ biến nhất là dựa trên quy tắc và loại tiên tiến hơn được cung cấp trí tuệ nhân tạo. Các chatbot trí tuệ nhân tạo (AI) sử dụng các hệ thống xử lý ngôn ngữ tự nhiên. Trong hệ thống này, các máy tính được lập trình để đọc, xử lý và phân tích số lượng lớn dữ liệu ngôn ngữ tự nhiên. Các công nghệ trí tuệ nhân tạo cũng bao gồm các thuật toán học sâu và máy học. Các bot AI học hỏi từ các cuộc trò chuyện và tương tác họ có với mọi người, mở rộng cơ sở dữ liệu của họ. Mặt khác, các bot dựa trên quy tắc được tạo thành từ các hệ thống đơn giản và do đó, có các phản hồi hạn chế. Hệ thống quét và xác định từ khóa hình thành đầu vào của người dùng và trả lời bằng lệnh tương ứng. Không giống như các chatbot dựa trên AI, các chatbot dựa trên quy tắc không còn phản hồi khi chúng gặp các lệnh lạ và các từ không được nhận dạng.

Sáng tạo công nghệ

Việc tạo ra các chatbot tương tự như mô hình phát triển các ứng dụng và trang web di động và

ban đầu bắt đầu với thiết kế. Thiết kế này mô tả sự tương tác của bot và người dùng. Mẫu này cũng bao gồm việc xây dựng bot liên quan đến phân tích đầu vào bằng cách sử dụng một công cụ xử lý ngôn ngữ tự nhiên. Sau các giai đoạn ban đầu, phân tích và bảo trì các bot sau đó được thực hiện. Việc phát triển Chatbot có thể được thực hiện trên các nền tảng được cung cấp bởi các nhà cung cấp Dịch vụ Nền tảng. Trong số này có IBM Watson, SnatchBot và Oracle Cloud Platform. Các nghiên cứu gần đây dường như cho thấy mọi người dành nhiều thời gian sử dụng các ứng dụng nhắn tin hơn phương tiện truyền thông xã hội. Do đó, các ứng dụng nhắn tin hiện cung cấp nhiều nền tảng hơn cho các công ty và doanh nghiệp để tiếp cận phần lớn người tiêu dùng. Hiệu quả của chatbot, đặc biệt là các ứng dụng sử dụng AI, lôi kéo và khuyến khích các công ty đầu tư vào các loại dịch vụ này. Sử dụng trong các ngành công nghiệp khác nhau Chatbots có một loạt các ứng dụng trong các lĩnh vực khác nhau như kinh doanh, giáo dục, thông tin và giải trí. Được sử dụng lần đầu tiên trong các trò chơi tương tác trực tuyến và nhắn tin tức thời, các bot này hiện đang được phân loại dựa trên việc sử dụng chúng trong giao tiếp, phân tích, thiết kế, du lịch, thể thao, mua sắm, cá nhân, thực phẩm và sức khỏe. Trong lĩnh vực kinh doanh, nhiều công ty đã sử dụng chatbot để cải thiện dịch vụ của họ và tăng doanh số bán hàng của họ. Từ việc xử lý các đơn đặt hàng trực tuyến đến tiếp thị và dịch vụ khách hàng và hỗ trợ, các bot hỗ trợ và phục vụ nhu cầu của người tiêu dùng bất cứ lúc nào trong ngày, có hoặc không có đại lý trực tiếp. Các hãng hàng không và các công ty thương mại điện tử cũng đã sử dụng chatbot trên trang web của họ để cung cấp thông tin và trả lời các câu hỏi cho khách hàng và khách hàng của họ. Một số công ty cũng đã đầu tư vào chatbot cho các vấn đề nội bộ, chẳng hạn như trong nguồn nhân lực. Các bot có thể hỗ trợ xử lý và bảo mật tài liệu. Các ngành công nghiệp ngân hàng cũng đang đầu tư vào chatbot thay cho các đại lý trung tâm cuộc gọi để phục vụ khách hàng của họ. Hơn nữa, các tập đoàn đồ chơi đang sử dụng đồ chơi dựa trên chatbot. Những thứ này cho phép trẻ tương tác tốt hơn với đồ chơi, cho phép chúng học tốt hơn. Nói chung, việc sử dụng chatbot trong các ngành tạo ra doanh thu lớn hơn trong khi tiết kiệm thời gian và tiền bạc. Họ cũng hướng dẫn người tiêu dùng tìm kiếm những gì họ quan tâm, mang lại cho họ trải nghiệm tốt hơn và độc đáo hơn.

Sự xuất hiện của trợ lý ảo

Chatbots cũng hoạt động như trợ lý ảo. Một số trong số các bot này là những người nổi tiếng mà bạn có thể đã nghe nói về Amazon Alexa, Google Assistant và Siri của Apple. Những trợ lý ảo này có thể thực hiện cả các nhiệm vụ đơn giản và cơ bản, cho phép người tiêu dùng tập trung vào những thứ quan trọng hơn. Những bot trợ lý ảo này giúp mọi người theo nhiều cách. Họ giữ cho người dùng của họ thông báo với các tin tức hiện tại và cập nhật thời tiết. Họ nhắc nhở họ về lịch trình của họ và hỗ trợ họ trong cửa hàng tạp hóa và tài chính của họ. Quan trọng hơn, bot cũng hoạt động như một người bạn và người bạn tâm tình. Ở châu Á, có một quốc gia có bot mà hàng triệu người nói chuyện. Vì vậy, thật an toàn khi nói rằng có vô số khả năng với bot.

Thành phần con người Sự phát triển của công nghệ và sự phát triển của chatbot chắc chắn đã thay đổi và cải thiện cách mọi người giao tiếp. Nó đóng vai trò là cầu nối giữa doanh nghiệp và

người tiêu dùng, giúp họ hoàn thành nhiệm vụ và đạt được mục tiêu của mình bất kể họ có thể ở đâu.

Chatbots, vào cuối ngày, được lập trình với các mã và lệnh mà mọi người tạo ra. Công nghệ này không hoàn hảo vì nó vẫn đang phát triển và do đó, dễ bị lỗi. Ngoài ra, việc sử dụng độc hại các chatbot trong quảng cáo và spam đã được báo cáo để lôi kéo mọi người và thu thập thông tin cá nhân. Bots và các công nghệ trí tuệ nhân tạo khác vẫn được con người theo dõi và duy trì. Khi giao dịch với bot, người tiêu dùng nên cẩn thận và bảo vệ an ninh và quyền riêng tư của họ mọi lúc.

Sự phát triển của công nghệ đã dẫn đến sự phát triển của trí tuệ nhân tạo và chatbot, mở ra cơ hội đáng kể cho các doanh nghiệp và công ty đồng thời mang lại sự tiện lợi cho người tiêu dùng. Tuy nhiên, như dự kiến, có những người cho rằng nó giới hạn các tương tác xã hội thực tế của con người.

Vâng, bất chấp sự tiện lợi mà nó mang lại, tương tác với bot và các công nghệ tương tự khác không thay thế (hoặc gây hại) cho sự tương tác của con người. Rốt cuộc, sự thành công của công nghệ phụ thuộc vào chính những người sử dụng nó và những người đứng đằng sau nó. Sử dụng các tiện ích nhắn tin chatbot có thể gây hại tới khả năng nhận thức và tương tác của con người dẫn đến kỹ năng giao tiếp không được trau dồi, con người trở nên thô lỗ và hung hăng, Mọi người đang trở nên lười biếng với thông tin và không quen kiểm tra mọi thứ hoặc không bao giờ dành thời gian suy nghĩ.

1.2 Bài toán

Mô hình hóa cuộc trò chuyện là một nhiệm vụ quan trọng để hiểu ngôn ngữ tự nhiên và hiểu biết về máy móc. Mặc dù các cách tiếp cận trước đây, chúng thường bị giới hạn trong chức năng hẹp cụ thể (ví dụ: đặt vé máy bay) và yêu cầu các quy tắc thủ công. Vấn đề đặt ra của bài toán là đưa dự đoán câu tiếp theo dựa trên câu trước đó hoặc một câu trong đoạn hội thoại cho các trường hợp mở.

2 Dữ liệu

2.1 Dữ liệu

Dữ liệu được dùng là *Cornell Movie-Dialogs Corpus <https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html>* là một bộ dữ liệu đa dạng chứa các đoạn hội thoại của nhân vật:

- 220.579 hội thoại giữa 10.292 cặp nhân vật trong phim.
- 9.035 nhân vật từ 617 phim.
- 304,713 câu thoại

Bộ dữ liệu này rất lớn và đa dạng, và có sự khác biệt lớn về hình thức ngôn ngữ, khoảng thời gian, cảm xúc.

File: movie_lines.txt
File này thông tin các câu hội thoại của từng nhân vật. Một số dòng dữ liệu mẫu

b'L1045 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ They do not!\n'
b'L1044 +++\$+++ u2 +++\$+++ m0 +++\$+++ CAMERON +++\$+++ They do to!\n'
b'L985 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ I hope so.\n'
b'L984 +++\$+++ u2 +++\$+++ m0 +++\$+++ CAMERON +++\$+++ She okay?\n'
b"L925 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Let's go.\n"
b'L924 +++\$+++ u2 +++\$+++ m0 +++\$+++ CAMERON +++\$+++ Wow\n'
b"L872 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Okay—you're gonna need to learn how to lie.\n"
b'L871 +++\$+++ u2 +++\$+++ m0 +++\$+++ CAMERON +++\$+++ No\n'
b'L870 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Iḿ kidding. You know how sometimes you just become this "persona"? And you donť know how to quit?\n'
b'L869 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Like my fear of wearing pastels?\n'

Các trường dữ liệu được phân cách bởi ký hiệu "+++\$+++ ". Các trường lần lượt là:

- Mã dòng
- Mã nhân vật.
- Mã phim
- Tên nhân vật
- Lời thoại

Ví dụ

L1045 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ They do not!\n'

Có nghĩa tại phim có mã 'm0' nhân vật 'u0' có tên là 'BIANCA' nói lời thoại "They do not!"

File: movie_conversations.txt
File này thông tin các câu nào tạo thành một đoạn hội thoại. Một số dòng dữ liệu mẫu

u0 +++\$+++ u2 +++\$+++ m0 +++\$+++ ['L870', 'L871', 'L872']
u0 +++\$+++ u2 +++\$+++ m0 +++\$+++ ['L924', 'L925']
u0 +++\$+++ u2 +++\$+++ m0 +++\$+++ ['L984', 'L985']

Các trường dữ liệu được phân cách bởi ký hiệu "+++\$+++ ". Các trường lần lượt là:

- Mã nhân vật đầu tiên của cuộc hội thoại.
- Mã nhân vật thứ hai của cuộc hội thoại.
- Mã phim mà cuộc hội thoại diễn ra
- Mã lời thoại trong file *movie_lines.txt* theo thứ tự để tạo thành cuộc hội thoại

Ví dụ

u0 +++\$+++ u2 +++\$+++ m0 +++\$+++ ['L870', 'L871', 'L872']

Có nghĩa tại phim có mã 'm0' nhân vật 'u0' và 'u2' nói chuyện với nhau bằng các lời thoại ['L870', 'L871', 'L872']

2.2 Tiền xử lý dữ liệu

1. Gộp thông tin

Dựa vào thông tin trên file *movie_conversations.txt* và *movie_line.txt*, ta có gộp lại và tạo thành các câu trong đoạn hội thoại

Ví dụ: *File: movie_line.txt*

> b"L872 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Okay—you're gonna need to learn how to lie.\n"
> b'L871 +++\$+++ u2 +++\$+++ m0 +++\$+++ CAMERON +++\$+++ No\n'
> b'L870 +++\$+++ u0 +++\$+++ m0 +++\$+++ BIANCA +++\$+++ Iḿ kidding. You know how sometimes you just become this "persona"? And you donť know how to quit?\n'

File: movie_conversations.txt

> u0 +++\$+++ u2 +++\$+++ m0 +++\$+++ ['L870', 'L871', 'L872']

Sẽ tạo thành đoạn hội thoại sau:

> (1) Like my fear of wearing pastels?
> (2) I'm kidding. You know how sometimes you just become this "persona"? And you don't know how to quit?
> (3) No

2. Làm sạch dữ liệu

Đầu tiên chuyển unicode sang ASCII.

$$\textit{I am Heró} \mapsto \textit{I am Hero}$$

Tiếp đó chuyển tất cả các chữ thành chữ thường và loại bỏ các kí tự không phải chữ cái (ngoại trừ các dấu câu).

$$\textit{I am Hero} \mapsto \textit{i am hero}$$

Cuối cùng để hỗ trợ training convergence, lọc ra những câu có độ dài vượt ngưỡng 10

3. Xây dựng cặp câu vấn đáp

Với mỗi đoạn hội thoại, lấy 2 câu kế tiếp nhau tạo thành một cặp.
Như đoạn hội thoại sau

(1) Like my fear of wearing pastels?
(2) I'm kidding. You know how sometimes you just become this "persona"? And you don't know how to quit?
(3) No

Câu (1)-(2) thành một cặp (2)-(3) thành một cặp.

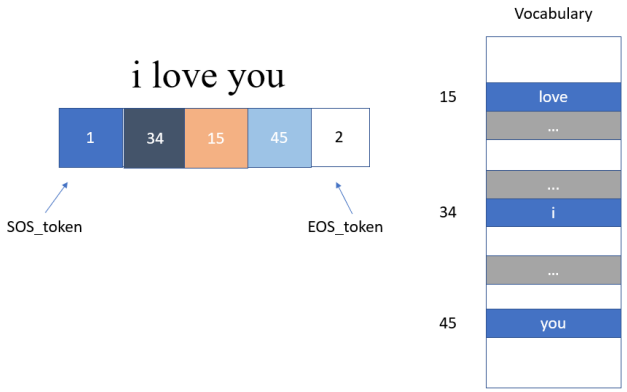
4. Xây dựng từ điển

Việc tiếp theo là tạo ra vocabulary và tải các từ vựng và các cặp câu đôi thoại vào bộ nhớ.

Lưu ý rằng chúng ta đang xử lý các chuỗi từ, không có một ánh xạ ngầm đến một không gian số rời rạc. Như vậy, chúng ta phải tạo ra một bằng cách ánh xạ từng từ duy nhất mà chúng ta bắt gặp trong tập dữ liệu của mình thành một giá trị chỉ số. Với một bộ dữ liệu khác nhau thì việc ánh xạ sẽ khác nhau.

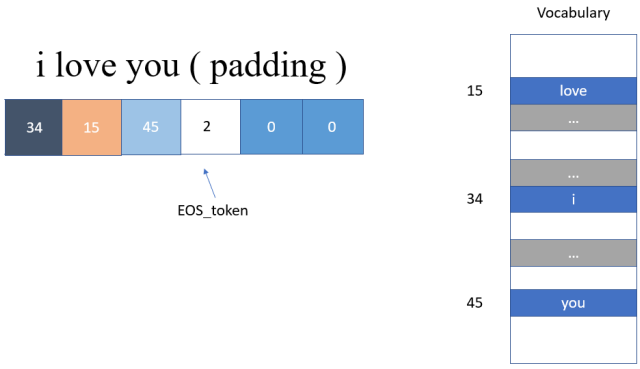
Việc sẽ bắt đầu xây dựng từ điển bằng việc với 3 ký tự đặc biệt:
PAD_token $\mapsto$ 0 *Dùng cho các câu ngắn*
SOS_token $\mapsto$ 1 *Đánh dấu bắt đầu câu*
EOS_token $\mapsto$ 2 *Đánh dấu kết thúc câu. Dùng trong thiết lập đầu vào của Decoder*

Sau đó với mỗi từ trong cặp hội thoại ta thêm từ đó vào từ điển bằng cách kiểm tra từ điển có từ hay chưa. Nếu có thì tăng chỉ số đếm từ là 1, còn chưa thì lưu lại từ và ánh xạ tới số tương ứng.
Trong một số trường hợp câu quá ngắn không phù hợp với model thì chêm PAD_token cho đủ



Hình. 2.1: Mô tả từ vựng.

độ dài.
Một kĩ thuật khác có lợi để đạt được convergence nhanh hơn trong thời gian training là xóa bỏ các từ không được sử dụng nhiều trong vocabulary. Giảm không gian tính năng cũng sẽ làm giảm bớt độ khó của chức năng model phải học cách gần đúng :



Hình. 2.2: Padding.

1) Xóa bỏ những từ có số lần sử dụng dưới ngưỡng

2) Lọc các cặp đối thoại có từ đã được xóa bỏ trong vocabulary

Model

3

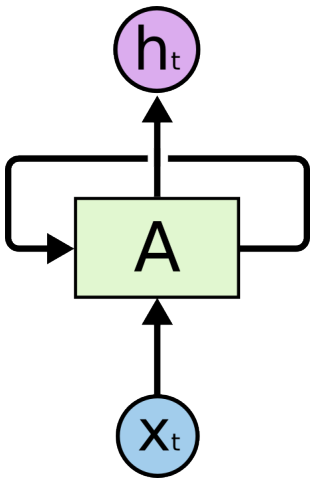
3.1 Recurrent Neural Network

Recurrent neural network

Con người không bắt đầu suy nghĩ từ đầu mỗi giây. Khi bạn đọc bài luận này, bạn hiểu từng từ dựa trên sự hiểu biết của bạn về các từ trước đó. Bạn không nên ném mọi thứ đi và bắt đầu suy nghĩ lại từ đầu. Suy nghĩ của bạn có sự lưu lại.

Mạng lưới thần kinh truyền thống có thể làm được điều này, và nó có vẻ như là một thiếu sót lớn. Ví dụ, hãy tưởng tượng bạn muốn phân loại loại sự kiện nào đang diễn ra tại mọi thời điểm trong phim. Nó không rõ làm thế nào một mạng lưới thần kinh truyền thống có thể sử dụng lý lẽ của nó về các sự kiện trước đó trong phim để thông báo cho những sự kiện sau này.

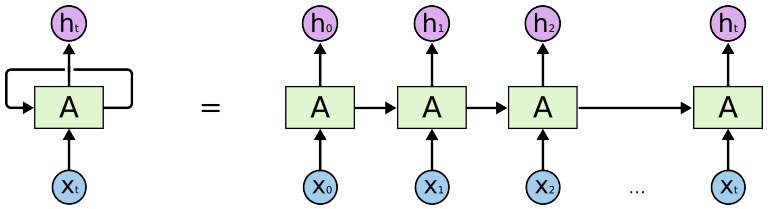
Recurrent neural network giải quyết vấn đề này. Chúng là các mạng có các vòng lặp trong đó, cho phép thông tin tồn tại.



Hình. 3.1: Recurrent Neural Network có vòng lặp.

Trong sơ đồ trên, một đoạn của mạng thần kinh, A, xem xét một số x_t đầu vào và xuất ra một giá trị h_t.Một vòng lặp cho phép thông tin được truyền từ một bước của mạng sang bước tiếp theo.

Những vòng lặp này làm cho Recurrent neural network có vẻ như bí ẩn. Tuy nhiên, nếu bạn suy nghĩ nhiều hơn một chút, hóa ra họ không phải là một mạng lưới thần kinh bình thường. Recurrent neural network có thể được coi là nhiều bản sao của cùng một mạng, mỗi bản tin truyền cho một người kế nhiệm. Xem xét những gì xảy ra nếu chúng ta bỏ vòng lặp:



Hình. 3.2: Recurrent Neural Network đã được trải ra.

Bản chất giống như chuỗi này cho thấy các Recurrent neural network có liên quan mật thiết đến các chuỗi và danh sách. Nó sử dụng kiến trúc tự nhiên của mạng neuron để sử dụng cho dữ liệu đó.

Và chúng chắc chắn được sử dụng! Trong vài năm qua, đã có những thành công đáng kinh ngạc khi áp dụng RNN cho nhiều vấn đề khác nhau: nhận dạng giọng nói, mô hình ngôn ngữ, dịch thuật, chú thích hình ảnh.

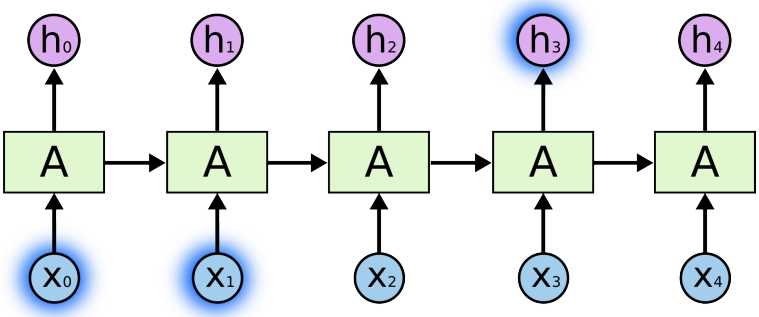
Điều cần thiết cho những thành công này là việc sử dụng "LSTM", một loại Recurrent neural network rất đặc biệt, hoạt động, cho nhiều tác vụ, tốt hơn nhiều so với phiên bản tiêu chuẩn. Hầu như tất cả các kết quả thú vị dựa trên các mạng thần kinh tái phát đều đạt được với chúng. Nó có những LSTM mà bài tiểu luận này sẽ khám phá.

Vấn đề phụ thuộc xa

Một trong những lời kêu gọi của RNN là ý tưởng rằng họ có thể kết nối thông tin trước đó với tác vụ hiện tại, chẳng hạn như sử dụng các khung video trước đó có thể thông báo cho sự hiểu biết về khung hiện tại. Nếu RNN có thể làm điều này, thì họ cực kỳ hữu ích. Nhưng họ có thể? Không hẳn.

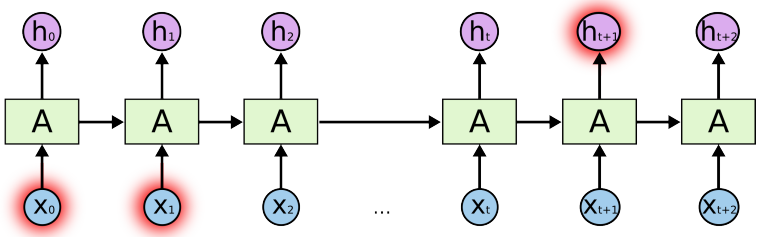
Đôi khi, chúng ta chỉ cần nhìn vào thông tin gần đây để thực hiện nhiệm vụ hiện tại. Ví dụ, hãy xem xét một mô hình ngôn ngữ đang cố gắng dự đoán từ tiếp theo dựa trên các từ trước đó. Nếu chúng ta đang cố gắng dự đoán từ cuối cùng trong "các đám mây trên *bầu trời*", thì chúng ta không cần bất kỳ bối cảnh nào nữa - đó là một điều khá rõ ràng, từ tiếp theo sẽ là *bầu trời*. Trong những trường hợp như vậy, khi khoảng cách giữa thông tin liên quan và địa điểm mà nó cần là nhỏ, RNN có thể học cách sử dụng thông tin trong quá khứ.

Nhưng cũng có những trường hợp chúng ta cần nhiều bối cảnh hơn. Cân nhắc việc cố gắng dự đoán từ cuối cùng trong văn bản. "Tôi lớn lên ở Việt Nam. Tôi nói tiếng trôi chảy tiếng *Việt*". Thông tin gần đây cho thấy từ tiếp theo có lẽ là tên của một ngôn ngữ, nhưng nếu chúng ta muốn thu hẹp ngôn ngữ nào, chúng ta cần thu hẹp ngôn ngữ nào bối cảnh của Việt Nam, từ



phía trước. Nó hoàn toàn có thể cho khoảng cách giữa thông tin liên quan và điểm cần thiết để trở nên rất lớn.

Thật không may, khi khoảng cách đó tăng lên, các RNN trở nên không thể học cách kết nối thông tin.



Về lý thuyết, các RNN hoàn toàn có khả năng xử lý các phụ thuộc dài hạn như vậy. Một người có thể cẩn thận chọn các tham số cho họ để giải quyết các vấn đề về đồ chơi theo hình thức này. Đáng buồn thay, trong thực tế, RNNs don lồng dường như có thể học chúng. Vấn đề đã được khám phá sâu bởi Hochreiter (1991) [German] và Bengio, et al. (1994), người đã tìm thấy một số lý do khá cơ bản tại sao nó có thể khó khăn.

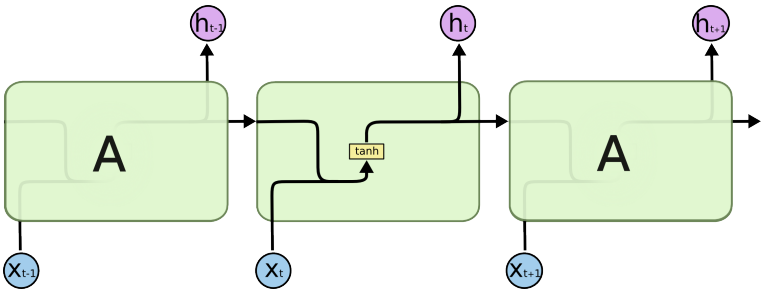
Rất may, LSTMs không có vấn đề này!

Mạng LSTM

Long Short Term Memory (mạng bộ nhớ dài ngắn hạn) - thường được gọi là LSTM của - - là một loại RNN đặc biệt, có khả năng học các phụ thuộc xa. Chúng được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được nhiều người tinh chỉnh và phổ biến. Chúng hoạt động rất tốt trong nhiều vấn đề lớn, và hiện đang được sử dụng rộng rãi.

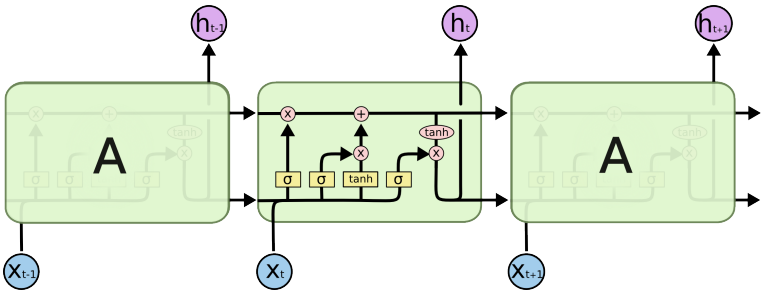
Các LSTM được thiết kế rõ ràng để tránh vấn đề phụ thuộc dài hạn. Ghi nhớ thông tin trong thời gian dài thực tế là hành vi mặc định của nó, không phải là thứ khó khăn để học!

Tất cả các mạng thần kinh tái phát có dạng một chuỗi các module lặp lại của mạng thần kinh. Trong các RNN tiêu chuẩn, module lặp lại này sẽ có cấu trúc rất đơn giản, chẳng hạn như một lớp $tanh$ duy nhất.

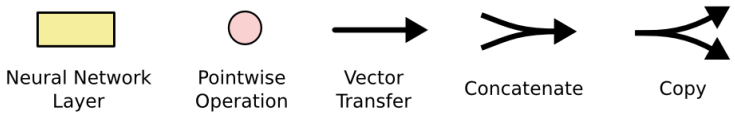


Hình. 3.3: Module lặp trong RNN chuẩn chứa một lớp duy nhất.

LSTMs cũng có cấu trúc chuỗi, nhưng các module lặp có một cấu trúc khác. Thay vì có một lớp mạng thần kinh duy nhất,nó có bốn lớp, tương tác theo một cách rất đặc biệt.



Hình. 3.4: Module lặp trong LSTM chứa 4 lớp tương tác.



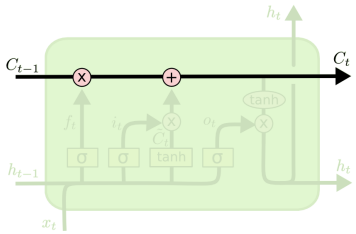
Hình. 3.5: Các ký hiệu trong LSTM.

Trong sơ đồ trên, mỗi dòng mang toàn bộ một vector, từ đầu ra của một nút đến đầu vào của các nút khác. Các vòng tròn màu hồng đại diện cho các phép toán, như phép cộng vector, trong khi các hình chữ nhật màu vàng biểu thị các mạng thần kinh để học. Các dòng hợp nhất biểu thị việc ghép nối, trong khi một dòng phân tách biểu thị nội dung của nó được sao chép và các bản sao đi đến các vị trí khác nhau.

Ý tưởng chính của LSTM

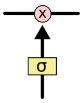
Ý tưởng chính của LSTM là ô trạng thái, đường ngang chạy qua đỉnh sơ đồ.

Dòng trạng thái giống như một băng chuyền. Nó chạy thẳng xuống toàn bộ chuỗi, chỉ với một số tương tác tuyến tính nhỏ dọc bên cạnh, dễ dàng cho thông tin truyền theo.



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

Cổng là một cấu trúc điều khiển thông tin thông qua. Chúng được cấu tạo từ một lớp lưới thần kinh $sigmold$ và một phép toán nhân.



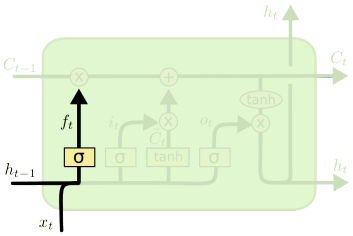
Đầu ra của các lớp $sigmoid$ có giá trị $[0, 1]$, mô tả mức độ cho qua. Giá trị bằng 0 có nghĩa là không để bất cứ thứ gì qua, trong khi giá trị của 1 nghĩa là có thể cho phép mọi thứ thông qua!

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

Các bước LSTM hoạt động

Bước đầu tiên trong LSTM là quyết định thông tin nào đi ra khỏi ô trạng thái. Quyết định này được đưa ra bởi một lớp sigmoid được gọi là lớp "cổng quên". Nó dựa vào giá trị của h_{t-1} và x_t, và đưa ra một số từ 0 đến 1 tương ứng với mỗi số ô trạng thái c_{t-1}. Số 1 có nghĩa là giữ lại toàn bộ thông tin trong khi đó số 0 nghĩa là hãy quên nó đi.

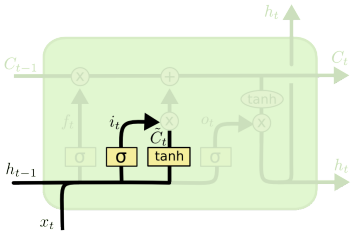
Hãy xem ví dụ của chúng ta về một mô hình ngôn ngữ đang cố gắng dự đoán từ tiếp theo dựa trên tất cả các từ trước đó. Trong một vấn đề như vậy, ô trạng thái có thể bao gồm vai về của ngữ hiện tại, để có thể sử dụng các động từ một cách chính xác. Khi có một chủ ngữ mới, nó sẽ quên đi vai về của chủ ngữ cũ.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Bước tiếp theo là quyết định những thông tin mới sẽ lưu trữ trong ô trạng thái. Việc này có hai phần. Đầu tiên, một lớp sigmoid được gọi là lớp "cổng đầu vào" quyết định giá trị nào sẽ cập nhật. Tiếp theo, một lớp $tanh$ tạo ra một vector các giá trị ứng cử viên mới, C_t, có thể được thêm vào ô trạng thái. Sau đó sẽ kết hợp cả hai để tạo ra một bản cập nhật cho trạng thái.

Trong ví dụ về mô hình ngôn ngữ, chúng tôi muốn thêm vai về của chủ ngữ mới vào ô trạng thái, để thay thế chủ ngữ đã quên.

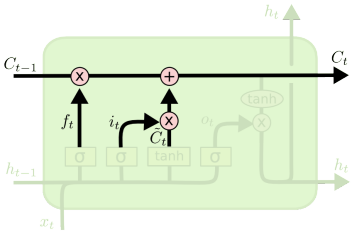


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Bây giờ, sẽ cập nhật ô trạng thái cũ, C_{t-1}, sang ô trạng thái mới C_t. Các bước trước đã quyết định phải quên và nhớ những gì, giờ là lúc thực hiện nó.

Nhân ô trạng thái cũ với f_t, quên đi những điều quyết định quên trước đó. Sau đó, cộng với $i_t * \tilde{C}_t$. Đây là giá trị ứng viên mới, được tính theo mức độ cập nhật từng giá trị trạng thái.

Trong trường hợp của mô hình ngôn ngữ, đây là lúc thực sự bỏ thông tin về vai về và thêm thông tin mới, như đã quyết định trong các bước trước. Cuối cùng là quyết định những gì sẽ xuất ra.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Đầu ra sẽ được lọc dựa vào ô trạng thái. Đầu tiên, chạy một lớp $sigmoid$ quyết định phần nào của ô trạng thái mà sẽ xuất ra. Sau đó, đưa ô trạng thái qua hàm $tanh$ (để đẩy các giá trị nằm trong khoảng -1 đến 1) và nhân nó với đầu ra của cổng $sigmoid$, do đó chỉ đưa ra các dữ liệu đã quyết định

Đối với ví dụ về mô hình ngôn ngữ, vì nó chỉ nhìn thấy một chủ ngữ, nó có thể muốn đưa ra thông tin có liên quan đến một động từ, trong trường hợp đó là những gì sắp diễn ra. Ví dụ, nó có thể xuất ra vai về của chủ ngữ , để biết được cách dùng từ với chủ ngữ đó.

Mạng GRU

Ý tưởng của GRU cũng khá giống với LSTM:

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$

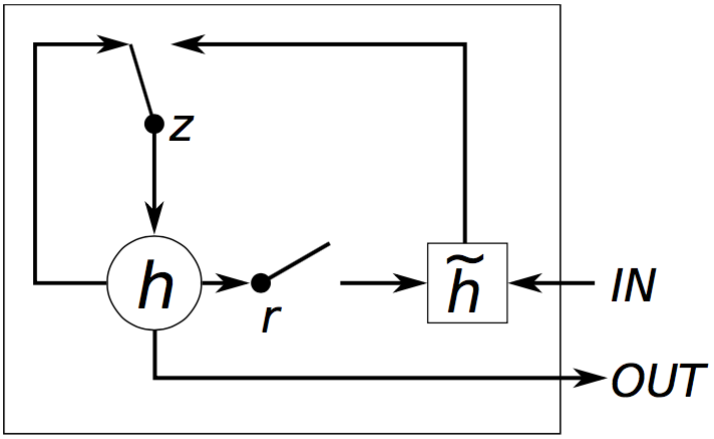
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$

$$h = \tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$

$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

GRU chỉ có 2 cổng: cổng thiết lập lại r và cổng cập nhập z. Cổng thiết lập lại sẽ quyết định cách kết hợp giữa đầu vào hiện tại với bộ nhớ trước, còn cổng cập nhập sẽ chỉ định có bao nhiêu thông tin về bộ nhớ trước nên giữa lại. Như vậy RNN thuần cũng là một dạng đặc biệt của GRU, với đầu ra của cổng thiết lập lại là 1 và cổng cập nhập là 0. Cùng chung ý tưởng sử dụng cơ chế cổng điều chỉnh thông tin, nhưng chúng khác nhau ở mấy điểm sau:

- GRU có 2 cổng, còn LSTM có tới 3 cổng.
- GRU không có bộ nhớ trong c_t và không có cổng ra như LSTM.
- 2 cổng vào và cổng quên được kết hợp lại thành cổng cập nhập z và cổng thiết lập lại r sẽ được áp dụng trực tiếp cho trạng thái ẩn trước.
- GRU không sử dụng một hàm phi tuyến tính để tính đầu ra như LSTM.



Hình. 3.6: Các cổng GRU.

3.2 Seq2seq

Lịch sử

Lê Viết Quốc khiến bạn không khỏi bất ngờ khi anh chính là một nhân vật quan trọng trong lĩnh vực trí tuệ nhân tạo tại Google. Quốc được biết đến với "Google Brain". Vào năm 2014, Quốc đề xuất trình tự chuỗi (Seq2seq) học với nhà nghiên cứu Google Ilya Sutskever và Oriol Vinyals. Nó là một khung công cụ - một thư viện các mã lệnh (framework) giải mã bộ mã hóa có mục đích đào tạo các mô hình để chuyển đổi các chuỗi từ một tên miền này sang miền khác, chẳng hạn như chuyển đổi các câu sang các ngôn ngữ khác nhau.

Seq2seq learning đòi hỏi ít sự lựa chọn trong thiết kế kỹ thuật hơn và cho phép hệ thống dịch của Google hoạt động hiệu quả và chính xác trên các tệp dữ liệu khổng lồ. Nó chủ yếu được sử dụng cho các hệ thống dịch máy và được chứng minh là có thể ứng dụng được ở nhiều mảng hơn, bao gồm tóm tắt văn bản, các cuộc hội thoại với trí tuệ nhân tạo, và trả lời câu hỏi.

Sau đó, Quốc tiếp tục phát minh ra Doc2vec – một thuật toán không giám sát sử dụng cho việc hiển thị các nội dung có độ dài cố định từ các đoạn văn bản có độ dài biến đổi, chẳng hạn như câu, đoạn văn và các tài liệu. Doc2vec là phần mở rộng của Word2vec, được giới thiệu vào năm 2013 bởi nghiên cứu sinh của Google, Tomas Mikolov. Ý tưởng của nó là mỗi từ có thể được biểu diễn bằng một vec-tơ, có thể được tự động học từ một tập hợp văn bản. Quốc sử dụng vector cho các đoạn văn để mô hình có thể tạo ra sự hiển thị - trình chiếu của tài liệu, bất chấp độ dài của nó.

Các ứng dụng của Seq2seq

Seq2seq tự nằm đằng sau nhiều hệ thống mà bạn phải sử dụng hằng ngày. Chẳng hạn, mô hình seq2seq hỗ trợ các ứng dụng như Google Dịch, thiết bị hỗ trợ giọng nói và chat-bot trực tuyến. Nói chung, các ứng dụng này bao gồm:

- Dịch máy - một bài báo năm 2016 của Google cho thấy cách tiếp cận chất lượng dịch thuật mô hình seq2seq hoặc vượt qua tất cả các kết quả được công bố hiện tại.

- Nhận dạng giọng nói - một bài báo khác của Google so sánh các mô hình seq2seq hiện có về nhiệm vụ nhận dạng giọng nói.

- Chú thích video - một bài báo năm 2015 cho thấy một seq2seq mang lại kết quả tuyệt vời như thế nào khi tạo mô tả phim.

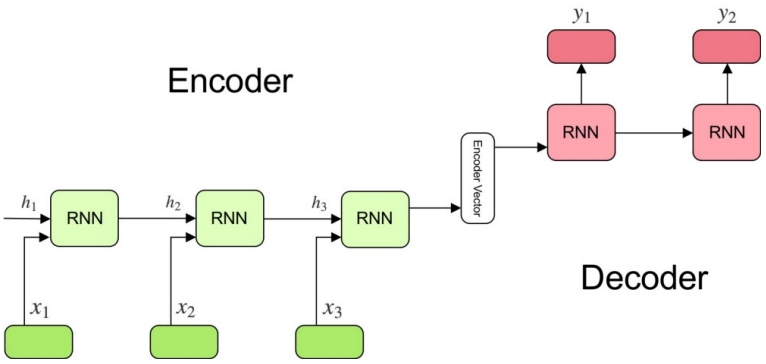
S2VT: A herd of zebras are walking in a field.

- Và đặc biệt tại Google I/O 2019, Google đã trình làng Translatotron, một công cụ cho phép dịch sang ngôn ngữ khác mà vẫn giữ được tông giọng của người nói

Đây chỉ là một số ứng dụng mà seq2seq được xem là giải pháp tốt nhất. Mô hình này có thể được sử dụng như một giải pháp cho bất kỳ vấn đề dựa trên tuần tự nào, đặc biệt là các vấn đề đầu vào và đầu ra có dữ liệu khác nhau.

Cách Seq2seq hoạt động

Chúng ta sẽ xem qua hình minh họa ở dưới để hiểu cách hoạt động



Hình. 3.7: Encoder-decoder seq2seq.

Model gồm 3 phần: encoder, intermediate (encoder) vector và decoder.

Encoder

- Một chồng các đơn vị recurrent (các ô LSTM hoặc GRU để có hiệu suất tốt hơn) trong đó mỗi đơn vị tiếp nhận một phần tử duy nhất của chuỗi đầu vào, thu thập thông tin cho phần tử đó và đẩy nó về phía trước.

- Trong vấn đề chatbox, chuỗi đầu vào là tập hợp tất cả các từ trong câu đầu vào. Mỗi từ được biểu diễn dưới dạng x_i trong đó i là thứ tự của từ đó.

- Các trạng thái ẩn h_i được tính bằng công thức:

$$h_t = f(W^{hh}h_{t-1} + W^{hx}x_t)$$

Công thức này đại diện cho kết quả của một Recurrent neural network thông thường. Có thể thấy, áp dụng các trọng số phù hợp cho trạng thái ẩn trước đó h_{t-1} và vector đầu vào x_t.

Encoder Vector

- Đây là trạng thái ẩn cuối cùng được tạo ra từ phần **Encoder** của mô hình. Nó được tính bằng công thức trên.

- Vectoc này nhằm mục đích tổng hợp thông tin đầu vào để giúp **Decoder** đưa ra dự đoán chính xác.

- Nó hoạt động như trạng thái ẩn ban đầu của **Decoder** của model.

Decoder

- Một chồng các đơn vị recurrent trong đó mỗi đơn vị dự đoán một đầu ra y_t tại một bước thời gian t.

- Mỗi đơn vị định kỳ chấp nhận một trạng thái ẩn từ đơn vị trước đó và tạo ra và đầu ra cũng như trạng thái ẩn của chính nó.

- Trong vấn đề chatbot, chuỗi đầu ra là tập hợp tất cả các từ trong câu đáp. Mỗi từ được biểu diễn dưới dạng y_i trong đó i là thứ tự của từ đó.

- Bất kỳ trạng thái ẩn h_i nào cũng được tính bằng công thức:

$$h_t = f(W^{hh}h_{t-1})$$

Như vậy có thế thấy rằng dùng trạng thái ẩn phía trước để tính trạng thái ẩn hiện tại

- Đầu ra y_t tại bước thời gian t được tính bằng công thức:

$$y_t = softmax(W^s h_t)$$

Tính toán các đầu ra bằng cách sử dụng trạng thái ẩn ở bước thời gian hiện tại cùng với trọng số tương ứng $W(S)$. Softmax được sử dụng để tạo một vector xác suất sẽ giúp xác định đầu ra cuối cùng (ví dụ: từ trong bài toán chatbot).

Sức mạnh của mô hình này nằm ở chỗ nó có thể ánh xạ các chuỗi có độ dài khác nhau với nhau. Như bạn có thể thấy đầu vào và đầu ra không tương quan và độ dài của chúng có thể khác nhau. Điều này mở ra một loạt các vấn đề mới mà bây giờ có thể được giải quyết bằng kiến trúc như vậy.

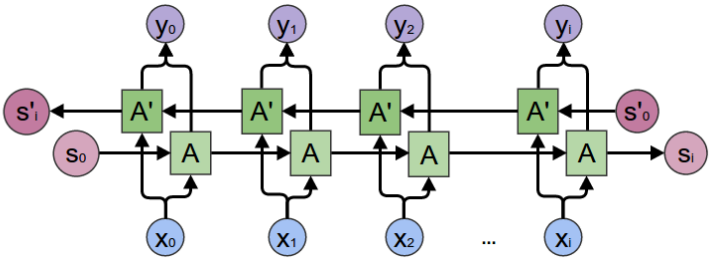
3.3 Model bài toán

Cài đặt

Chatbot được cài đặt dựa trên Pytorch và chạy trên Google colab để tăng tốc độ train.

Encoder

Trong bài toán này, chúng ta sẽ sử dụng **Encoder** bao gồm nhiều lớp Gated Recurrent Unit, được phát minh Cho et al. vào năm 2014. Chúng ta sẽ sử dụng GRU hai chiều, nghĩa là sẽ có 2 mạng RNN độc lập: một mạng thì đưa dữ liệu theo chiều thuận, mạng còn lại đưa theo chiều người lại. Đầu ra của mỗi mạng sẽ được cộng lại ở mỗi bước thời gian. Sử dụng GRU hai chiều giúp mã hóa được ngữ cãnh trong tương lai và quá khứ



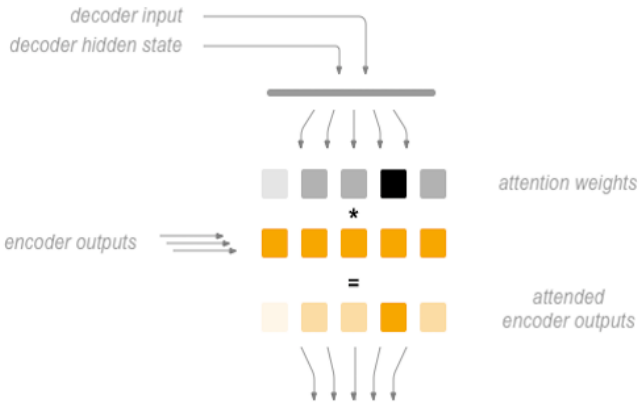
Hình. 3.8: Recurrent Neural Network 2 chiều.

Encoder

Một vấn đề phổ biến với **Decoder** seq2seq thuần túy là chỉ dựa vào trên vector bối cảnh để mã hóa toàn bộ chuỗi đầu vào có nghĩa là, có khả năng sẽ bị mất thông tin. Điều này đặc biệt xảy ra khi xử lý các chuỗi đầu vào dài, hạn chế rất nhiều khả năng của **Decoder** .

Để chống lại điều này, Bahdanau và các cộng sự đã tạo ra một cơ chế **Attention**, cho phép bộ giải mã chú ý đến một số phần nhất định của câu đầu vào, thay vì sử dụng toàn bộ bối cảnh cố định ở mỗi bước.

Ở mức cao, sự chú ý được tính bằng cách sử dụng trạng thái ẩn hiện tại của **Decoder** và các đầu ra của **Encoder**. Các trọng số **Attention** đầu ra có hình dạng giống như câu đầu vào, cho phép nhân chúng với các đầu ra của **Encoder**, cho chúng ta một tổng trọng số cho biết các phần của đầu ra **Encoder** cần chú ý.



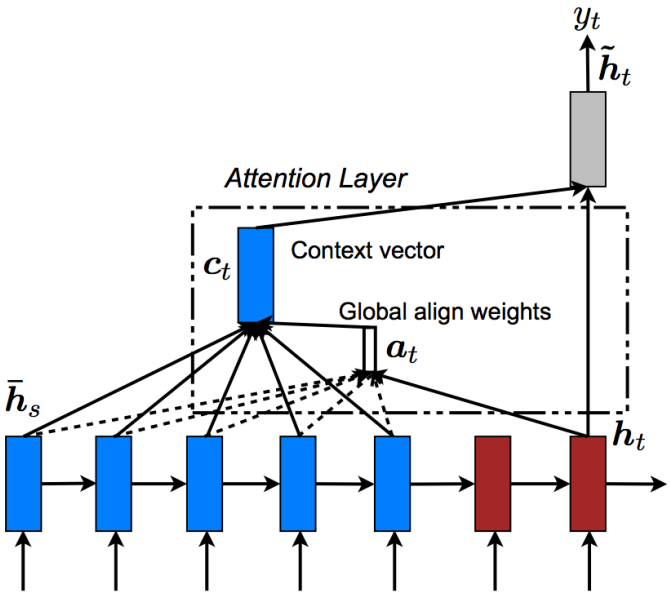
Hình. 3.9: Attention.

Lương Minh Thắng và cộng sự đã cải thiện dựa trên nền tảng của Bahdanau và cộng sự bằng cách tạo ra **Global attention**. Sự khác biệt chính là với **Global attention**, xem xét tất cả các trạng thái ẩn của **Encoder**, trái ngược với **Local attention** của Bahdanau và cộng sự. Một điểm khác biệt nữa là với **Global attention** là tính toán trọng số chú ý hoặc năng lượng, chỉ sử dụng trạng thái ẩn của **Encoder** từ bước thời gian hiện tại. **Local attention** của Bahdanau và cộng sự, đòi hỏi phải có thông tin về trạng thái **Decoder** từ bước trước. Ngoài ra, Thắng còn cung cấp các phương pháp khác nhau để tính toán năng lượng chú ý giữa đầu ra **Encoder** và đầu ra **Decoder** được gọi là các hàm điểm số của điểm số:

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W_a} \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_a}[\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

Trong đó h_t là trạng thái của decoder hiện tại, $\bar{h_s}$ là trạng thái của encoder

Tóm lại cơ chế **Global attention** có thể biểu diễn bằng hình sau



Hình. 3.10: Global attention.

3.4 Phương pháp đánh giá

Sau khi train xong, chắc hẳn sẽ muốn nói chuyện với bot. Vì vậy ta phải đề ra cách model giải mã từ **Encoder**

Greedy Encoding
Với mỗi bước thời gian, chúng ta chỉ cần chọn từ có giá trị softmax cao nhất. Phương pháp giải mã này là tối ưu trên mức bước thời gian.

4 Kết quả đạt được

4.1 Thử nghiệm

Một số đoạn hội thoại với chủ đề mở

> hello !
Bot: hello .
> how are you ?
Bot: okay .
> what 's your name ?
Bot: pris .
> when were you born ?
Bot: tomorrow .
> what year were you born ?
Bot: what ?
> where are you from ?
Bot: california .
> are you a man or a woman ?
Bot: no .
> why are we here ?
Bot: i m not leaving .
> okay, bye !
Bot: okay okay .
> see you later!
Bot: i m sorry .

> is sky blue or black
Bot: no .
> does a cat have a tail
Bot: no it s not a problem .
> does a cat have a wing
Bot: i have a problem .
> can a cat fly

Bot: can t you say that ?
> how many legs does a cat have
Bot: four hundred five thousand .
> how many legs does a spider have
Bot: four .
> how many legs does a centipede have
Error: Encountered unknown word.
> what is the color of the sky
Bot: what do you mean ?
> what is the color of water
Bot: i don t know .
> what is the color of blood
Bot: i don t know .
> what is the color of a yellow car
Bot: what s that ?
> how much is two plus two
Bot: two hundred .
> how much is ten minus two
Bot: four hundred .

> what is your job
Bot: i don t know .
> what do you do
Bot: i m sorry .
> what is the purpose of life
Bot: i don t know .
> what is the purpose of living
Bot: i don t know .
> what is the purpose of existence
Bot: i don t know .
> where are you now
Bot: i m in the bathroom .
> what is the purpose of dying
Bot: i don t know .
> what is the purpose of being intelligent
Bot: i don t know .

4.2 Đánh giá

Model có thể thực hiện các hình thức chào hỏi thông thường. Nhưng các câu hỏi hầu hết đều bị bot bỏ qua . Nếu một từ không có trong bộ từ điển thì chatbot khôngthể trả lời được

Model ngôn ngữ đơn giản dựa trên seq2seq này có thể được sử dụng để huấn luyện một chatbot. Kết quả cho thấy rằng có thể tạo ra các cuộc hội thoại đơn giản và cơ bản, và trích xuất kiến thức từ một bộ dữ liệu tên mở nhưng bị nhiễu. Mặc dù model có những hạn chế rõ ràng, nhưng thật đáng ngạc nhiên đối với chúng tôi rằng cách tiếp cận dựa trên dữ liệu mà không có bất kỳ quy tắc nào có thể tạo ra câu trả lời thích hợp cho nhiều loại câu hỏi. Tuy nhiên, mô hình có thể yêu cầu sửa đổi đáng kể để có thể cung cấp các cuộc hội thoại thực tế.

Tài liệu tham khảo

[DL11] Cristian Danescu-Niculescu-Mizil and Lillian Lee. „Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.“ In: *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*. 2011.

[Hải01] Đỗ Minh Hải. *[RNN] Cài đặt GRU/LSTM*. 201. URL: https://dominhhai.github.io/vi/2017/10/implement-gru-lstm/ (visited on June 13, 2019).

[LPM15] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025.

[Ola15] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/ (visited on June 13, 2019).

[Pas+17] Adam Paszke, Sam Gross, Soumith Chintala, et al. „Automatic differentiation in PyTorch“. In: *NIPS-W*. 2017.

[VL15] Oriol Vinyals and Quoc Le. *A Neural Conversational Model*. 2015. arXiv: 1506.05869.

Danh sách các hình