TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

DỰ ÁN CÔNG NGHỆ THÔNG TIN 2

Ứng dụng chatbot vào hệ thống tư vấn học đường

Giảng viên hướng dẫn:
Vũ Đình Hồng

Sinh viên thực hiện:
Bành Đại Nam - 51503279
Huỳnh Thị Liễu - 51503078

Ngày 11 tháng 1 năm 2019

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

DỰ ÁN CÔNG NGHỆ THÔNG TIN 2

Ứng dụng chatbot vào hệ thống tư vấn học đường

Giảng viên hướng dẫn:
Vũ Đình Hồng

Sinh viên thực hiện:
Bành Đại Nam - 51503279
Huỳnh Thị Liễu - 51503078

Ngày 11 tháng 1 năm 2019

LỜI CẢM ƠN

Trên thực tế không có sự thành công nào mà không gắn liền với những sự hỗ trợ, giúp đỡ dù ít hay nhiều, dù trực tiếp hay gián tiếp của người khác. Trong suốt thời gian từ khi bắt đầu học tập ở giảng đường đại học đến nay, em đã nhận được rất nhiều sự quan tâm, giúp đỡ của quý thầy cô, gia đình và bạn bè. Với lòng biết ơn sâu sắc nhất, em xin gửi đến quý thầy cô ở Khoa Công nghệ thông tin – Trường Đại Học Tôn Đức Thắng đã cùng với tri thức và tâm huyết của mình để truyền đạt vốn kiến thức quý báu cho chúng em trong suốt thời gian học tập tại trường. Và đặc biệt, trong môn *Nhập môn xử lý ảnh*, thầy đã tổ chức cho chúng em được tiếp cận với môn học mà theo em là rất hữu ích đối với sinh viên ngành **Công nghệ phần mềm** cũng như tất cả các sinh viên thuộc khoa **Công nghệ thông tin**. Em xin chân thành cảm ơn thầy *Vũ Đình Hồng* đã tận tâm hướng dẫn chúng em qua từng buổi học trên lớp cũng như những buổi nói chuyện, thảo luận về lĩnh vực sáng tạo trong nghiên cứu khoa học. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì em nghĩ bài thu hoạch này của em rất khó có thể hoàn thiện được. Một lần nữa, em xin chân thành cảm ơn thầy. Bài thu hoạch được thực hiện trong khoảng thời gian gần 10 tuần. Bước đầu đi vào thực tế, tìm hiểu về lĩnh vực sáng tạo trong nghiên cứu khoa học, kiến thức của em còn hạn chế và còn nhiều bỡ ngỡ. Do vậy, không tránh khỏi những thiếu sót là điều chắc chắn, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô và các bạn học cùng lớp để kiến thức của em trong lĩnh vực này được hoàn thiện hơn.

Lời cảm tạ thầy *Vũ Đình Hồng*. Sau cùng, em xin kính chúc quý thầy cô trong khoa **Công nghệ thông tin** thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau.

ĐỒ ÁN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG
ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm đồ án/luận văn (xem mẫu trên sakai) của riêng chúng tôi và được sự hướng dẫn của thầy/cô. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫnvà chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm
Sinh viên (Ký và ghi rõ họ tên):

Đánh giá của giáo viên

Phần xác nhận của GV hướng dẫn

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Tp.Hồ Chí Minh, ngày ..... tháng.... năm.....
(Ký và ghi rõ họ tên)

Mục lục

Danh sách bảng

BẢNG **Trang**

Danh sách hình vẽ

HÌNH **Trang**

Chương 1

Tổng quan về chatbot

Chatbot là một chương trình máy tính hoặc một trí tuệ nhân tạo tiến hành một cuộc trò chuyện thông qua phương pháp giao tiếp như văn bản hay giọng nói. Các chương trình này mô phỏng hành vi giao tiếp của con người với con người thông qua các phép thử Turning. Chatbots thường được sử dụng trong các hệ thống hộp thoại cho các mục đích thực tế khác nhau bao gồm dịch vụ khách hàng hoặc thu thập thông tin. Một số chatterbots sử dụng hệ thống xử lý ngôn ngữ tự nhiên phức tạp, nhưng nhiều hệ thống đơn giản hơn quét từ khóa trong đầu vào, sau đó kéo trả lời với từ khóa phù hợp nhất hoặc mẫu từ ngữ tương tự nhất từ cơ sở dữ liệu.

Thuật ngữ "ChatterBot" ban đầu được đặt ra bởi Michael Mauldin (tác giả của Verbot , Julia) đầu tiên vào năm 1994 để mô tả các chương trình đàm thoại này. Ngày nay, hầu hết các chatbots đều được truy cập thông qua các trợ lý ảo như Trợ lý Google và Amazon Alexa , thông qua các ứng dụng nhắn tin như Facebook Messenger hoặc WeChat hoặc thông qua các ứng dụng và trang web của các tổ chức riêng lẻ. Với sự ra mắt của chương trình messenger và các nền tảng như slack, bots đã nhận được một tăng. Các chương trình của Facebook đã phát triển từ 34000 chương trình trong tháng 11 năm 2016 lên 100.000 bot vào tháng 4 năm 2017 . Khi truyền thông xã hội mở rộng sự thâm nhập của nó, các công ty sẽ chuyển trọng tâm của họ sang các bot để tiếp cận và phục vụ người dùng. Điều này là do sự hiện diện trên nền tảng truyền thông xã hội giúp người dùng dễ dàng truy cập hơn. Nó cũng có nghĩa là ít tài nguyên hơn để đầu tư cho công ty và người dùng. Người dùng cũng không phải giải quyết rắc rối khi tải xuống ứng dụng. Họ chỉ có thể nhắn tin cho bot tương ứng trên nền tảng truyền thông xã hội ưa thích của họ. Nó cũng có nghĩa là dịch vụ có sẵn 24/7. Điều này đã được cho một dịch vụ khách hàng của con

người làm việc, nó sẽ chi phí cách nhiều hơn để thuê và đào tạo con người. Nó sẽ có nghĩa là có người trong nhiều ca để cung cấp dịch vụ 24/7. Bots vẫn chưa được chính xác và hiểu như con người nhưng họ có thể hoàn thành công việc cơ bản và khi họ bị mắc kẹt, con người luôn có thể tiếp quản [4]. Vì những lợi ích này, bot được xem là một miền có tiềm năng lớn cho việc tiếp cận khách hàng, thúc đẩy doanh số bán hàng, tin tức, cá nhân, năng suất, mua sắm, xã hội, thể thao, du lịch và tiện ích.

Tuy nhiên, hầu hết các chương trình ngày nay là các bot dựa trên quy tắc cung cấp cho người dùng một menu và người dùng điều hướng qua menu như hệ thống đặt chỗ khiếu nại qua điện thoại nhưng bằng văn bản. Thứ hai, hầu hết các chương trình đều là các bot có tên miền khép kín có nghĩa là chúng tập trung vào một nhiệm vụ cụ thể và chỉ được huấn luyện cho trường đó. Alexa, Siri là ví dụ về các bot miền mở. Tuy nhiên, các bot dành cho đặt chỗ nhà hàng là các bot miền bị đóng. Các bot miền đóng có thể là cả hai, được hỗ trợ bởi trí tuệ nhân tạo hoặc dựa trên quy tắc. Trong khi người dùng mong đợi các bot miền mở thông minh trong mọi khía cạnh, mục đích của một hệ thống bot không chỉ là vậy. Mục tiêu của một hệ thống bot vẫn là để tự động hóa một dịch vụ bằng cách sử dụng một giao diện đàm thoại cho phép người dùng truy cập dịch vụ từ các nền tảng mà họ thường truy cập. Điều này đi để nói rằng một trong những không phải là dissuaded từ xây dựng một bot dựa trên quy tắc mà thực sự làm việc vì lý do mà bot không thông minh như một siêu người. Điều đó đang được nói, chúng ta phải theo đuổi việc xây dựng các hệ thống thông minh vì họ càng thông minh về mặt hiểu biết về người dùng cuối, sẽ càng hữu ích.

Chúng ta sẽ thảo luận về các loại bot và nơi mà mỗi chương trình đều hữu ích, theo sau là kiến trúc chung của một hệ thống đàm thoại. Tiếp theo chúng ta thảo luận về các nền tảng có thể được sử dụng để xây dựng một bot và so sánh chúng với các tác vụ chúng thực hiện. Tiếp theo đó chúng tôi thảo luận về các khả năng tương lai cho nghiên cứu trong lĩnh vực các cuộc hội thoại dựa trên văn bản.

1.1 Những loại chatbot

Chatbots có thể được phân loại thành hai loại: tên miền mở và miền đóng. Ở đây, chúng ta thấy cách chatbots có thể được phân loại bằng cách sử dụng các tham số khác quá như mức độ tương tác, và phương pháp tạo ứng dụng.

- Knowledge domain - Miền tri thức.
- Service Provided - Cung cấp dịch vụ.
- Goal.
- Phương pháp tạo phản hồi.

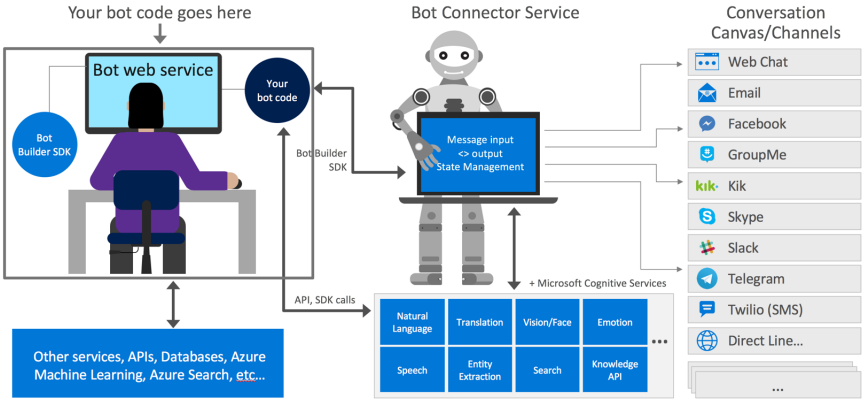
1.1.1 Knowledge domain - Miền tri thức

Các chatbots được phân loại dựa trên kiến thức mà chúng có được hoặc lượng dữ liệu mà chúng được đào tạo:

1. Open Domain: Các bot như vậy có thể nói về các chủ đề chung và phản hồi một cách thích hợp
2. Close Domain: Các bot như vậy tập trung vào một miền tri thức cụ thể và có thể không thành công trả lời các câu hỏi khác. Ví dụ, một nhà hàng đặt phòng bot sẽ không nói bạn là tên của tổng thống da đen đầu tiên của nước Mỹ. Nó có thể cho bạn biết một trò đùa hoặc trả lời như thế nào là ngày của bạn nhưng nó không phải là dự kiến sẽ làm như vậy kể từ khi công việc của nó là để đặt một bảng và cung cấp cho bạn thông tin về nhà hàng.

1.1.2 Chat based/conversational

Những chương trình này nói chuyện với người dùng, giống như một người khác. Mục tiêu của họ là trả lời chính xác câu mà họ đã được đưa ra. Do đó, chúng thường được xây dựng với mục đích tiếp tục trò chuyện với người dùng dựa trên các kỹ thuật như cross questioning, evasion, deferenceo. Ví dụ: Siri, Alexa,...Chúng sử dụng các thuật toán như named entity extraction - trích xuất thực thể, truy xuất thông tin, đối sánh chuỗi, phát hiện mức độ liên quan.



Hình 1.1: Chatbot kết nối với các dịch vụ

1.1.3 Task based

Chúng thực hiện một nhiệm vụ nhất định như đặt vé máy bay hoặc giúp bạn duyệt một cửa hàng. Phần lớn thời gian, các hành động cần thiết để thực hiện một nhiệm vụ được xác định trước, dòng chảy của các sự kiện bao gồm các trường hợp ngoại lệ cũng được quyết định. Các bot thông minh trong bối cảnh yêu cầu thông tin và hiểu đầu vào của người dùng.

1.1.4 Input Processing and Response generation method

Loại này có phụ thuộc đến phương thức xử lý đầu vào và tạo ra phản hồi. Các hệ thống thông minh thực sự tạo ra các phản hồi và sử dụng hiểu biết ngôn ngữ tự nhiên để hiểu được truy vấn. Các hệ thống này được sử dụng khi miền hẹp và dữ liệu phong phú có sẵn để đào tạo một hệ thống.

Các hệ thống dựa trên quy tắc sử dụng đối sánh mẫu và chứng minh. Đây có thể được sử dụng khi số lượng kết quả có thể được cố định và các kịch bản có thể tưởng tượng được về số lượng.

1.1.4.1 Dạng lai - Hybrid

Các hệ thống này là sự kết hợp giữa các quy tắc và học máy. Một ví dụ sẽ là một hệ thống sử dụng biểu đồ lưu lượng để quản lý hướng của cuộc hội thoại nhưng cung cấp câu trả lời được tạo bằng cách sử dụng xử lý ngôn ngữ tự nhiên.

Bots không phải độc quyền thuộc về một thể loại hoặc khác. Các loại này tồn tại trong mỗi bot với tỷ lệ khác nhau. Ví dụ, tất cả các bot sẽ yêu cầu một số loại khả năng chat, có lẽ một bot cho một cửa hàng sẽ cần phải sử dụng khai thác thông tin khi nói đến các câu hỏi thường gặp và tìm kiếm trang web khi đưa ra kết quả sản phẩm. Một dịch vụ mà bot cung cấp do đó có thể bao gồm cả ba loại thuật toán. Lấy một ví dụ khác, bot sẽ bị đóng miền nhưng sẽ được lập trình để nói chuyện nhỏ hoặc bot sẽ là miền mở và intrapersonal như bot năng suất nhưng nó cũng sẽ tập trung vào các cuộc hội thoại liên quan đến năng suất.

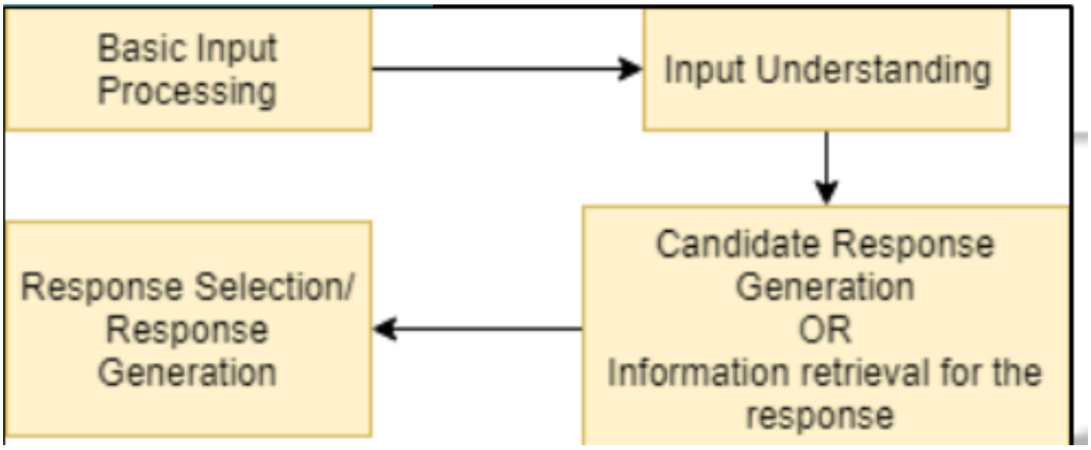
Phân loại này giúp cho người dùng biết điều gì sẽ xảy ra và xây dựng hình ảnh của bot trong tâm trí người dùng làm giảm khoảng cách giữa kỳ vọng của người dùng và bot được phân phối. Tiếp theo chúng ta thảo luận về kiến trúc của chatbots và các module khác nhau của bot cho phép các tính năng của bot.

1.2 PIPELINE & TOOLS

Như đã trình bày ở trên ở trên, có thể có nhiều chatbot. Chatbot sẽ là kết hợp của một hoặc nhiều chatbot con lại tạo thành một hệ thống chatbot khổng lồ. Ví dụ, nó có thể mang tính thông tin và dựa trên quy tắc hoặc có thể dựa trên nhiệm vụ nhưng sử dụng các kỹ thuật sinh tự động . Tuy nhiên, tất cả các chatbots theo một dòng chảy chung hoặc một kiến trúc chung, một đường ống dẫn. Đối với mỗi cấp trong đường ống chung, đối với một danh mục nhất định, các thuật toán hoặc phương pháp liên quan đến danh mục - tên miền đó đó sẽ được sử dụng.

1.2.1 Các nhiệm vụ khác nhau bao gồm trong các giai đoạn khác nhau như sau

Chunking, Sentence Boundary Detection, Sentence Parsing, một phần của Speech Tagging là một số thuật toán xử lý đầu vào cơ bản luôn diễn ra. Là một trong những thư viện sử dụng, các nhiệm vụ này được che giấu trong các chức năng. Tiếp theo khai thác thực thể được đặt tên, phát hiện ý định, phát hiện mơ hồ, phát hiện ngữ cảnh, phân tích tình cảm, phân loại truy vấn, khái niệm và phát hiện từ đồng nghĩa, vv diễn ra để hiểu đầu vào.



Hình 1.2: Chatbot pipeline

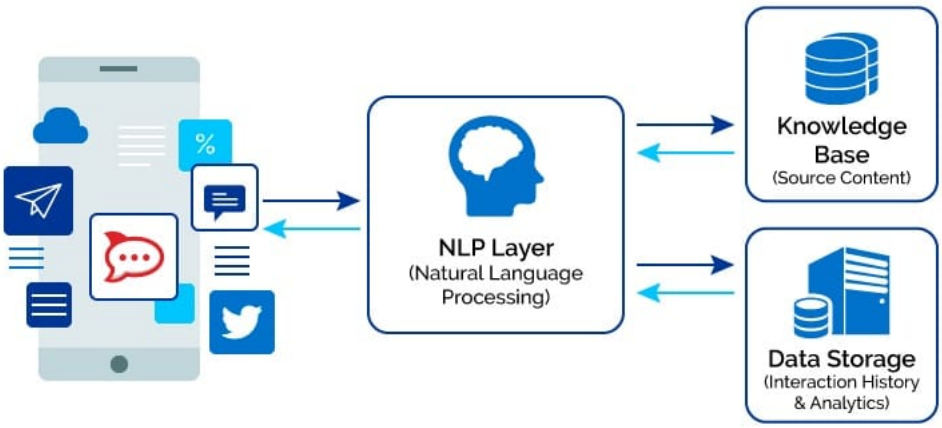
Khi truy vấn hoặc đầu vào được cấu trúc thành dạng mong muốn, phản hồi được tạo ra. Nó có thể được tạo ra thông qua một định dạng được xác định trước hoặc thông qua học máy. Các kỹ thuật tổng hợp và đơn giản hóa cũng có thể được sử dụng nếu đáp ứng được lấy từ một nguồn văn bản. Một khi có một tập hợp các phản hồi của ứng cử viên, các câu trả lời có thể được kiểm tra tính thích hợp và có thể trả lời phù hợp.

Như đã thấy trong phần trên, chatbots có hai loại dựa trên hành động chúng thực hiện trong thế hệ phản hồi:

1. Mô hình dựa trên truy xuất: các quy tắc dựa trên quy tắc hoặc tìm nạp câu trả lời từ một bộ câu trả lời được xác định trước. Mô hình Dựa trên truy xuất có thể được sử dụng khi dữ liệu bị giới hạn và miền cuộc hội thoại được giới hạn trong một vài trường hợp hội thoại. Do đó, nếu tất cả các cuộc hội thoại có thể cho một ca sử dụng với bot có thể tưởng tượng được, thì mô hình dựa trên truy xuất hoạt động tốt. Ngoài ra, khi bot không được mong đợi để hiển thị thông minh trong tất cả các kịch bản nhưng có thể đủ khả năng từ chối biết câu trả lời, các mô hình dựa trên truy xuất đó hoạt động tốt. Hệ thống đặt chỗ, hệ thống Hỏi đáp hoặc bất kỳ hệ thống nào khác tìm nạp thông tin có thể hoạt động tốt ngay cả với bot dựa trên truy xuất.

2. Generative models: Các hệ thống này, chẳng hạn như trong đó dự định duy trì cấu trúc phân cấp của ý nghĩa cơ bản của ngôn ngữ, có thể được sử dụng khi một lượng lớn dữ liệu là có sẵn và hệ thống có thể được đào tạo về dữ liệu đó. Họ thường sử dụng thuật toán NLP và NLU để xử lý đầu vào và tạo

câu. Mạng nơron nhân tạo, LSTM, SVM, các mô hình trình tự, Thuật toán sinh học, vv là một số thuật toán được sử dụng trong mô hình này. Những hệ thống tiên tiến nhất như Alexa, Siri và Cortana đều dựa trên quy tắc bán thống nhất.



Hình 1.3: NLP layer chatbot

Tuy nhiên, không phải tất cả điều này phải được thực hiện từ đầu. Các công cụ và nền tảng khác nhau có sẵn cung cấp mức trích xuất để thực hiện tác vụ. Tiếp theo, chúng tôi thảo luận về các công cụ khác nhau có sẵn giúp việc trò chuyện trở nên dễ dàng hơn.

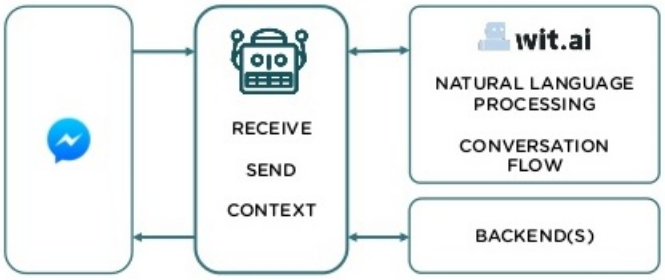
1.3 Truy xuất tên thực thể và phát hiện ý định

1.3.1 Recast.ai

Nó cho phép người dùng tạo các truy vấn tùy chỉnh và đào tạo bot trên chúng. Ngoài việc cung cấp các dịch vụ học máy để trích xuất các thực thể và mục đích, nó cung cấp mọi thứ từ các mẫu để phân tích cho các bot được triển khai. Nó cung cấp lưu trữ và tích hợp bot trên các nền tảng khác nhau. Người ta cũng có thể thêm tích hợp bên ngoài và tạo logic tùy chỉnh trong bot. Nó đã được xây dựng trong hỗ trợ cho phân tích tình cảm và nhiều ngôn ngữ. Ngoài việc cho phép thêm truy vấn tùy chỉnh, nó cũng có giao diện kéo và thả để thiết kế luồng hội thoại.

1.3.2 Wit.ai

Wit.ai giúp phân loại ý định và trích xuất các thực thể có tên từ một đầu vào đã cho. Sau đó, nó sẽ trả về nhãn cho ứng dụng hoặc bot. Nó cho phép một người trích xuất thông tin như vị trí, thời gian, ngày tháng, thời tiết và cũng cho phép người ta tạo ra ý định của riêng mình. Nó phân tích cú pháp người dùng nhập vào dữ liệu có cấu trúc và cũng giúp trong cuộc hội thoại bằng cách gửi văn bản. Tuy nhiên, để thực hiện các hành động trên dữ liệu có cấu trúc, đầu ra của wit.ai phải được gửi đến ứng dụng hoặc máy chủ có phản hồi wit.ai rồi trả về nền tảng hội thoại như messenger.



Hình 1.4: Sơ đồ luồng chatbot trên Messager

1.4 Tạo luồng dựa trên câu chuyện và các trò chuyện dựa trên nguyên tắc

1.4.1 Chatfuel

Chatfuel cung cấp giao diện kéo và thả để tạo bot dựa trên quy tắc. Module thông minh nhân tạo của nó cho phép bạn đào tạo bot để ánh xạ các câu lệnh đầu vào tới đầu ra nhưng so với các mô đun recast.ai và các mô-đun khác nó khá cứng nhắc về luồng hội thoại.Nó cũng cho phép các lời nhắc phản hồi và tích hợp với các dịch vụ như email và IFTTT.Nó cũng cho phép tích hợp json để chứa logic tùy chỉnh vào bot.Điểm hấp dẫn nhất của dịch vụ là cực kỳ đơn giản để xây dựng bot dựa trên quy tắc phù hợp cho các doanh nghiệp nhỏ. Nó cũng cho phép người dùng chọn tham gia hoặc không tham gia danh sách gửi thư và có thể gửi tin nhắn quảng bá. Nó chỉ cần tích hợp với trang faceboook và không cần bất kỳ thiết lập nào khác.

1.4.2 Pandora bot

Pandora bot cung cấp trí tuệ nhân tạo như một dịch vụ cùng với một nền tảng để xây dựng bot.Nó sử dụng AIML và cung cấp một IDE để xây dựng bot cũng như AIaaS API. Nó cho phép mã hóa trong AIML, nơi các lập trình viên có nghĩa vụ phải đề cập đến các phản hồi đầu vào như một khuôn khổ. Mô hình này sau đó có thể được đào tạo về dữ liệu và được sử dụng.

1.4.3 Text Processing Platforms

Bên cạnh đó, các nền tảng đám mây khác nhau như amazon, IBM, Google, Microsoft đã phát hành API đàm thoại riêng cho phép các lập trình viên phân tích dữ liệu, chuyển đổi lời nói thành văn bản và ngược lại và tạo ra các phản hồi thích hợp. Lợi ích của việc sử dụng các nền tảng này nằm trong thực tế là việc tích hợp chúng dễ dàng hơn với cơ sở hạ tầng đám mây hiện có. Các mô hình này được xây dựng dựa trên số lượng lớn dữ liệu và sử dụng các khả năng cho các nền tảng một độ chính xác mà lập trình viên không thể đạt được với các tài nguyên đơn giản. Ví dụ: Microsoft bot framework, LUIS, Amazon Lex và Poly, IBM Watson, API Google Speech, API ngôn ngữ tự nhiên của Google.

1.4.4 For training on your own data

Người ta có thể xây dựng mô hình nguyên mẫu của họ ngay từ đầu sau đó kiểm tra chúng trên tập dữ liệu.Họ có thể được đào tạo bằng cách sử dụng các nền tảng như Tensorflow và Weka.Mô hình sau đó có thể được thu nhỏ bằng cách sử dụng máy chủ của riêng mình hoặc các dịch vụ học máy được cung cấp bởi nền tảng như Amazon, Google ...

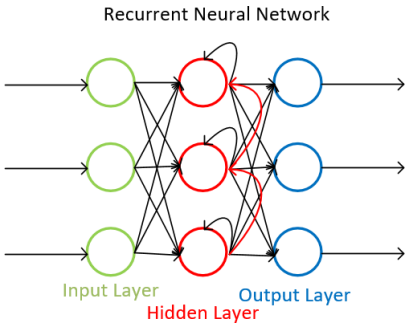
Chương 2

Ứng dụng mô hình seq2seq-RNN vào chatbot

Mạng neural hồi quy (RNN) được áp dụng rộng rãi cho các bài toán xử lý ngôn ngữ tự nhiên do RNN mô hình hóa được bản chất của dữ liệu trong NLP.Việc áp dụng RNN có thể coi là một bước đột phá trong NLP.
Mô hình seq2seq áp dụng cho các bài toán khi ta cần sinh ra một chuỗi đầu ra từ một câu đầu vào cho trước nó còn có tên gọi khác là Encoder-Decoder framework

2.1 Khái niệm RNN

Recurrent Neural Networks giữ kết quả trước đó (output) tại một thời điểm (t-1) để là input tại thời điểm t và ngoài ra còn có input tại thời điểm t. Nhu vậy RNN lưu những thông tin trước đó (history information) và cả thông tin tại thời điểm hiện tại là t. RNN hữu ích vì giá trị trung gian (trạng thái) của chúng có thể lưu trữ thông tin về các yếu tố đầu vào trong một thời gian mà không cần quan tâm đến độ ưu tiên của thông tin đó.



Hình 2.1: Mạng neural hồi quy (RNN) đầy đủ

2.1.1 Hàm softmax

2.1.2 Khái niệm

- Là một mô hình xác xuất cao với mỗi input x, a_i thể hiện xác xuất để x rơi vào class i. Như vậy, điều kiện cần là a_i phải dương và tổng của chúng bằng 1.
- Để có thể thỏa mãn điều kiện này, chúng ta cần nhìn vào mọi giá trị z_i và dựa trên quan hệ giữa các z_i.

2.1.3 Công thức

$$a_i = \frac{\exp(z_j)}{\sum_{j=1}^{c}} \ \forall \ \text{i=1,2,3...c}$$

- Hàm số này, tính tất cả các a_i dựa vào tất cả các z_i thõa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1,giữ được thứ tự của z_i.Hàm số này được gọi là softmax function. Chú ý rằng với cách định nghĩa này, không có xác suất a_i nào tuyệt đối bằng 0 hoặc tuyệt đối bằng 1, mặc dù chúng có thể rất gần 0 hoặc 1 khi z_i rất nhỏ hoặc rất lớn khi so sánh với các z_j , j≠i
- Để có thể thỏa mãn điều kiện này, chúng ta cần nhìn vào mọi giá trị z_i và dựa trên quan hệ giữa các z_i.

- P(y_k = i | W) = a_i

2.2 Hàm mất mát

2.2.1 Khái niệm

- Hàm mất mát trả về một số thực không âm thể hiện sự chênh lệch giữa hai đại lượng: y', label được dự đoán và y, label đúng. Hàm mất mát giống như một hình thức để bắt model đóng phạt mỗi lần nó dự đoán sai, và số mức phạt này tỉ lệ thuận với độ trầm trọng của sai sót.Trong mọi bài toán supervised learning, mục tiêu của ta luôn bao gồm giảm thiểu tổng mức phạt phải đóng. Trong trường hợp lý tưởng,tức là y'=y,hàm mất mát sẽ trả về giá trị cực tiểu, bằng 0.

2.2.2 Công thức

$$w = \sum_n [(\frac{e^{w_i^T \cdot x_n}}{\sum_j e^{w_i^T \cdot x_n}}) - y_n] \cdot x_n$$

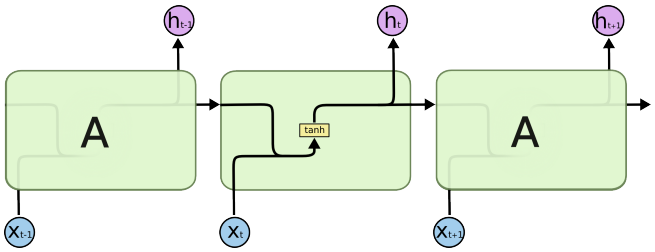
2.3 LSTM

2.3.1 Khái niệm của 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. 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 tanh.

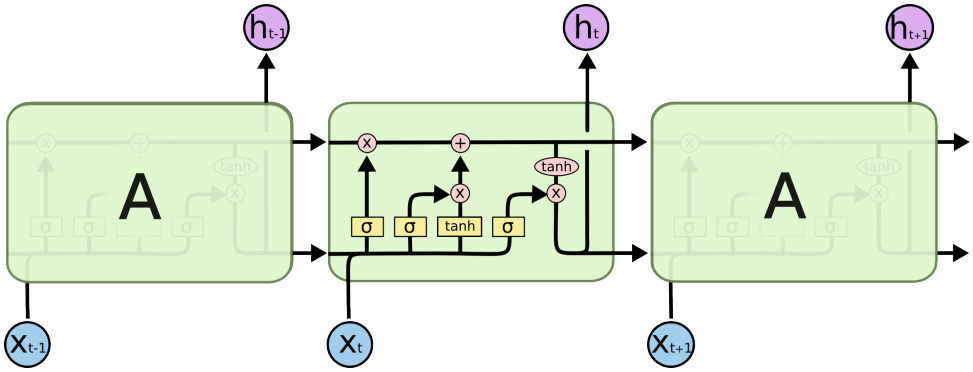
2.3.2 Sự khác nhau giữa LSTM và RNN

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 hay hàm tanh.

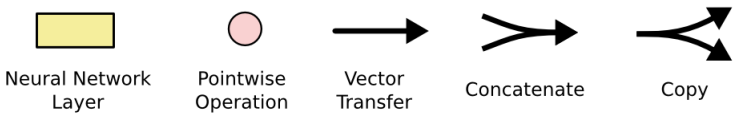


Hình 2.2: Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp.

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.



Hình 2.3: Mô-đun lặp lại trong LSTM chứa bốn lớp tương tác.



Hình 2.4: Chú ý ký hiệu

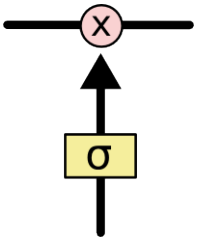
Ở sơ đồ trên, mỗi một đường mang một véc-tơ từ đầu ra của một nút tới đầu vào của một nút khác. Các hình trong màu hồng biểu diễn các phép toán như phép

cộng véc-tơ chẳng hạn, còn các ô màu vàng được sử dụng để học trong các từng mạng nơ-ron. Các đường hợp nhau kí hiệu việc kết hợp, còn các đường rẽ nhánh ám chỉ nội dung của nó được sao chép và chuyển tới các nơi khác nhau.

2.3.3 Ý tưởng cốt lõi của LSTM

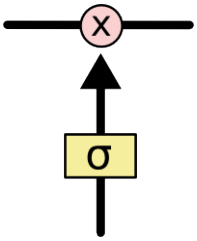
Chìa khóa của LSTM là trạng thái tế bào (cell state) - chính đường chạy thông ngang phía trên của sơ đồ hình vẽ.

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.



Hình 2.5: Trạng thái tế bào cell state

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.



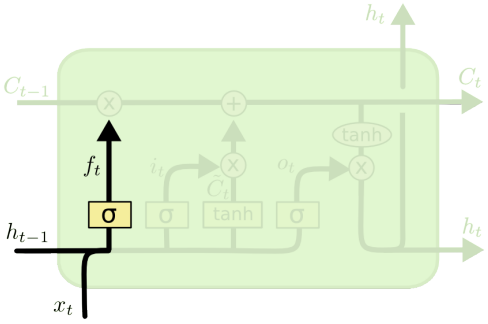
Hình 2.6: Nạp thông tin

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 0 0 thì có nghĩa là không cho thông tin nào qua cả, còn khi là 1 1 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.

2.3.4 Bên trong LSTM

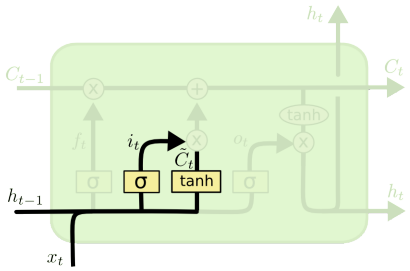
Bước đầu tiên của LSTM là quyết định xem thông tin nào cần bỏ đi từ trạng thái tế bào. Quyết định này được đưa ra bởi tầng sigmoid - gọi là "tầng cổng quên" (forget gate layer). Nó sẽ lấy đầu vào là h_{t-1} và x_t và đưa ra kết quả trong khoảng [0-1] cho mỗi số trong trạng thái tế bào C_{t-1}. Đầu ra là 1 thể hiện rằng nó giữ toàn bộ thông tin lại, còn 0 chỉ rằng toàn bộ thông tin sẽ bị bỏ đi. Quay trở lại với ví dụ mô hình ngôn ngữ dự đoán từ tiếp theo dựa trên tất cả các từ trước đó, với những bài toán như vậy, thì trạng thái tế bào có thể sẽ mang thông tin về giới tính của một nhân vật nào đó giúp ta sử dụng được đại từ nhân xưng chuẩn xác. Tuy nhiên, khi đề cập tới một người khác thì ta sẽ không muốn nhớ tới giới tính của nhân vật nữa, vì nó không còn tác dụng gì với chủ thế mới này.



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

Hình 2.7: Công thức sigmoid

Bước tiếp theo là quyết định xem thông tin mới nào ta sẽ lưu vào trạng thái tế bào. Việc này gồm 2 phần. Đầu tiên là sử dụng một tầng sigmoid được gọi là "tầng cổng vào" (input gate layer) để quyết định giá trị nào ta sẽ cập nhập. Tiếp theo là một tầng $tanh$ tạo ra một véc-tơ cho giá trị mớiC_t nhằm thêm vào cho trạng thái. Trong bước tiếp theo, ta sẽ kết hợp 2 giá trị đó lại để tạo ra một cập nhập cho trạng thái. Chẳng hạn với ví dụ mô hình ngôn ngữ của ta, ta sẽ muốn thêm giới tính của nhân vật mới này vào trạng thái tế bào và thay thế giới tính của nhân vật trước đó.



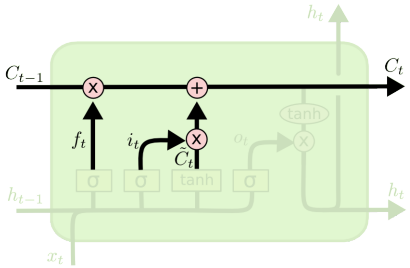
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Hình 2.8: Cập nhật trạng thái tiếp theo

Giờ là lúc cập nhập trạng thái tế bào cũ $C_t - 1$ thành trạng thái mới C_t. Ở các bước trước đó đã quyết định những việc cần làm, nên giờ ta chỉ cần thực hiện là xong.

Ta sẽ nhân trạng thái cũ với f_t để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm $i_t * C_t$. Trạng thái mơi thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.

Với bài toàn mô hình ngôn ngữ, chính là việc ta bỏ đi thông tin về giới tính của nhân vật cũ, và thêm thông tin về giới tính của nhân vật mới như ta đã quyết định ở các bước trước đó.



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

Cuối cùng, ta cần quyết định xem ta muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, ta chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, ta đưa nó trạng thái tế bảo qua một hàm *tanh* để co giá trị nó về khoảng [-1 1], và nhân nó với đầu ra của cổng sigmoid để được giá trị đầu ra ta mong muốn.

```
ERROR_THRESHOLD = 0.25
def classify(sentence):
    results = model.predict([bow(sentence, words)])[0]
    results = [[i,r] for i,r in enumerate(results) if r>ERROR_THRESHOLD]
```

```
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append((classes[r[0]], r[1]))
    return return_list

def response(sentence, userID='1', show_details=False):
    results = classify(sentence)
    print()
    if results:
        while results:
            for i in intents['intents']:
                if i['tag'] == results[0][0]:
                    if 'context_set' in i:
                        context[userID] = i['context_set']
                    if not 'context_filter' in i or (userID in context and
                    ↪  'context_filter' in i and i['context_filter']
                    ↪ == context[userID]):
                        if show_details: print('tag:', i['tag'])
                        list_talk = i['responses']
                        talk = random.choice(list_talk)
                        res = str(talk)
                        f = open("talk.txt",'w',encoding='utf-8')
                        f.write(res+"")
                        f.close()
                        return talk

            results.pop(0)
```

2.4 Setup

Chúng tôi sẽ xây dựng một RNN đơn giản để ghi nhớ dữ liệu đầu vào và sau đó lặp lại dữ liệu đó sau một vài bước thời gian. Trước tiên, hãy để cho bộ cài đặt một số hằng số mà chúng tôi cần, ý nghĩa của chúng sẽ trở nên rõ ràng trong giây lát.

```
from __future__ import print_function, division

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

num_epochs = 100

total_series_length = 50000

truncated_backprop_length = 15

state_size = 4

num_classes = 2

echo_step = 3

batch_size = 5

num_batches = total_series_length//batch_size//truncated_backprop_length
```

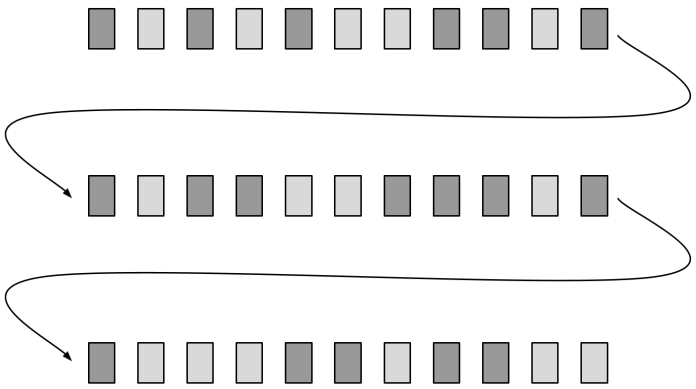
2.5 Tạo tập dữ liệu

Bây giờ tạo dữ liệu đào tạo, đầu vào về cơ bản là một vectơ nhị phân ngẫu nhiên. Đầu ra sẽ là các echo echo của các đầu vào, chuyển các bước echo_step sang phải.

```
def generateData():

    x = np.array(np.random.choice(2,
    total_series_length, p=[0.5, 0.5]))
```

```
    y = np.roll(x, echo_step)

    y[0:echo_step] = 0

    x = x.reshape((batch_size, -1))

    y = y.reshape((batch_size, -1))

    return (x, y)
```

Lưu ý việc định hình lại dữ liệu thành một ma trận với các hàng batch_size. Mạng lưới thần kinh được đào tạo bằng cách xấp xỉ độ dốc của hàm mất đối với trọng lượng nơ-ron, bằng cách chỉ nhìn vào một tập hợp con nhỏ của dữ liệu, còn được gọi là một lô nhỏ. Lý do lý thuyết để làm điều này được xây dựng thêm trong câu hỏi này. Việc định hình lại lấy toàn bộ dữ liệu và đưa nó vào một ma trận, sau đó sẽ được chia thành các lô nhỏ này.



Hình 2.9: Sơ đồ của ma trận dữ liệu được định hình lại, các đường cong mũi tên hiển thị các bước thời gian liền kề kết thúc trên các hàng khác nhau. Hình chữ nhật màu xám nhạt tượng trưng cho một số 0 không màu xám và màu xám đậm là một bản một.

2.6 Xây dựng đồ thị tính toán

TensorFlow hoạt động bằng cách đầu tiên xây dựng một biểu đồ tính toán, trong đó chỉ định những hoạt động sẽ được thực hiện. Đầu vào và đầu ra của biểu đồ này thường là các mảng nhiều chiều, còn được gọi là các tenxơ. Biểu đồ hoặc các phần của nó sau đó có thể được thực thi lặp đi lặp lại trong một phiên, điều này có thể được thực hiện trên CPU, GPU hoặc thậm chí là tài nguyên trên máy chủ từ xa.

2.6.1 Variables và placeholders

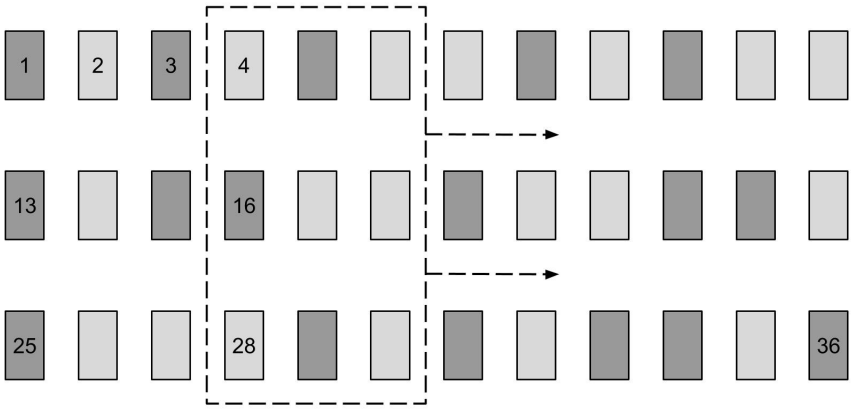
Hai cấu trúc dữ liệu TensorFlow cơ bản sẽ được sử dụng trong ví dụ này là Variables và placeholders. Trên mỗi lần chạy, dữ liệu hàng loạt được cung cấp cho các trình giữ chỗ, đó là các nút bắt đầu của Google, của biểu đồ tính toán. Ngoài ra, trạng thái RNN được cung cấp trong một trình giữ chỗ, được lưu từ đầu ra của lần chạy trước.

```
batchX_placeholder = tf.placeholder(tf.float32,

[batch_size, truncated_backprop_length])

batchY_placeholder = tf.placeholder(tf.int32,

[batch_size, truncated_backprop_length])

init_state = tf.placeholder(tf.float32, [batch_size,

state_size])
```

Các tính năng và tính năng của chúng tôi là một phần lớn trong số những thứ khác nhau.

Hình dưới đây cho thấy ma trận dữ liệu đầu vào và batchX_placeholder hiện tại nằm trong hình chữ nhật nét đứt. Như chúng ta sẽ thấy sau đó, cửa sổ lô hàng này, một trong số đó được trượt theo các bước rút ngắn về phía bên phải trong mỗi lần chạy, do đó là mũi tên. Trong ví dụ của chúng tôi bên dưới batch_size = 3, truncated_backprop_length = 3 và Total_series_length = 36. Lưu ý rằng những

con số này chỉ nhằm mục đích trực quan hóa, các giá trị khác nhau trong mã. Chỉ số thứ tự chuỗi được hiển thị dưới dạng số trong một vài điểm dữ liệu.



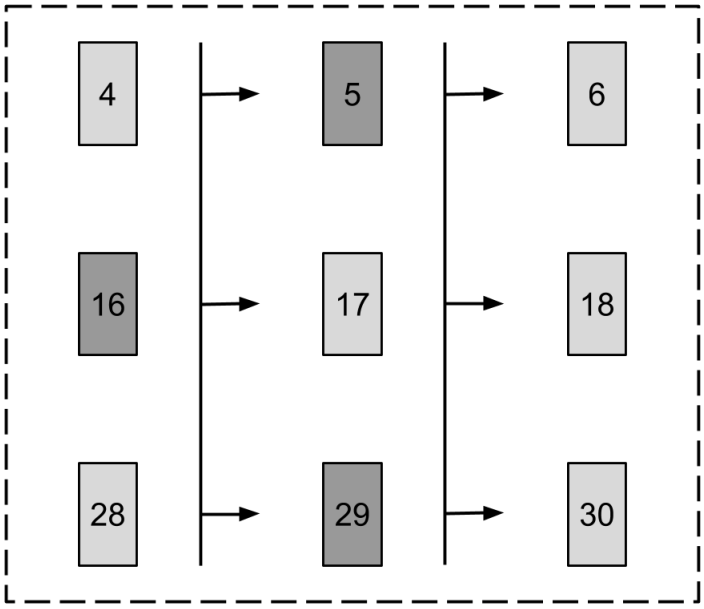
Hình 2.10: Sơ đồ dữ liệu đào tạo, lô hiện tại được cắt ra trong hình chữ nhật nét đứt. Chỉ số bước thời gian của datapoint được hiển thị.

2.7 Unpacking

Bây giờ đã có thời gian để xây dựng một phần của biểu đồ giống với tính toán RNN thực tế, trước tiên chúng tôi muốn chia dữ liệu lô thành các bước thời gian liền kề.

```
# Unpack columns
inputs_series = tf.unpack(batchX_placeholder, axis=1)

labels_series = tf.unpack(batchY_placeholder, axis=1)
```

Như bạn có thể thấy trong hình bên dưới được thực hiện bằng cách giải nén các cột (trục = 1) của lô vào danh sách Python. RNN sẽ đồng thời được đào tạo về các phần khác nhau trong chuỗi thời gian; các bước 4 đến 6, 16 đến 18 và 28 đến 30 trong ví dụ về lô hiện tại. Lý do cho việc sử dụng các tên biến số nhiều số nhiều nhất là sê-ri là để nhấn mạnh rằng biến đó là một danh sách đại diện cho một chuỗi thời gian với nhiều mục nhập ở mỗi bước.



Hình 2.11: Caption

Thực tế là việc đào tạo được thực hiện đồng thời ở ba nơi trong chuỗi thời gian của chúng tôi, đòi hỏi chúng tôi phải lưu ba trường hợp trạng thái khi tuyên truyền về phía trước. Điều đó đã được tính đến, như bạn thấy rằng trình giữ chỗ init_state có các hàng batch_size.

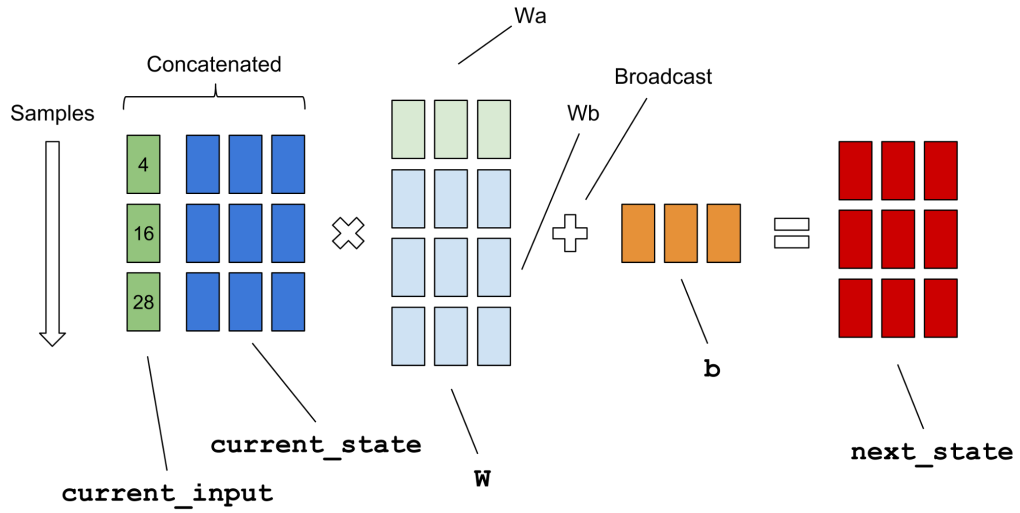
2.8 Forward pass

Tiếp theo, hãy xây dựng một phần của biểu đồ thực hiện tính toán RNN thực tế.

```
# Forward pass
current_state = init_state
states_series = []
for current_input in inputs_series:
    current_input = tf.reshape(current_input, [batch_size, 1])
    input_and_state_concatenated = tf.concat(1, [current_input,
       ↪ current_state]) # Increasing number of columns
```

```
next_state = tf.tanh(tf.matmul(input_and_state_concatenated, W) + b)
    ↪  # Broadcasted addition
states_series.append(next_state)
current_state = next_state
```

Lưu ý cách nối trên dòng 6, điều chúng ta thực sự muốn làm là tính tổng của hai biến đổi affine current_input * Wa + current_state * Wb trong hình bên dưới. Bằng cách ghép hai tenxơ đó, bạn sẽ chỉ sử dụng một phép nhân ma trận. Việc bổ sung sai lệch b được phát trên tất cả các mẫu trong lô.



Hình 2.12: Sơ đồ tính toán của các ma trận trên dòng 8 trong ví dụ mã ở trên, arctan biến đổi phi tuyến tính được bỏ qua.

Bạn có thể tự hỏi tên biến truncated_backprop_length có nghĩa là gì. Khi một RNN được đào tạo, nó thực sự được coi là một mạng lưới thần kinh sâu với các trọng số định kỳ trong mỗi lớp. Các lớp này sẽ không được kiểm soát vào đầu thời gian, điều này sẽ quá tốn kém về mặt tính toán, và do đó bị cắt ngắn ở một số bước giới hạn. Trong sơ đồ mẫu của chúng tôi ở trên, lỗi được sao lưu ba bước trong lô của chúng tôi.

2.8.1 Tính hàm mất mát

Đây là phần cuối cùng của biểu đồ, một lớp softmax được kết nối đầy đủ từ trạng thái đến đầu ra sẽ làm cho các lớp được mã hóa một lần nóng, sau đó tính toán tổn thất của lô.

```
logits_series = [tf.matmul(state, W2) + b2 for state in states_series]

predictions_series = [tf.nn.softmax(logits) for logits in logits_series]

losses = [tf.nn.sparse_softmax_cross_entropy_with_logits(logits, labels)
    ↪  for logits, labels in zip(logits_series,labels_series)]

total_loss = tf.reduce_mean(losses)

train_step = tf.train.AdagradOptimizer(0.3).minimize(total_loss)
```

Dòng cuối cùng là thêm chức năng đào tạo, TensorFlow sẽ tự động thực hiện truyền lại cho chúng tôi - biểu đồ tính toán được thực hiện một lần cho mỗi lô nhỏ và trọng số mạng được cập nhật tăng dần.

423/5000 Lưu ý lệnh gọi API tới spzzy_softmax_cross_entropy_with_logits, nó tự động tính toán softmax bên trong và sau đó tính toán entropy chéo. Trong ví dụ của chúng tôi, các lớp là loại trừ lẫn nhau (chúng là 0 hoặc 1), đó là lý do để sử dụng các Spid-softmax, bạn có thể đọc thêm về API trong API. Việc sử dụng là để havelogits có hình dạng [batch_size, num_groupes] và nhãn có hình dạng [batch_size].

2.8.2 Hình dung về đào tạo

Có một chức năng trực quan hóa để chúng ta có thể nắm bắt những gì đang diễn ra trong mạng khi chúng ta đào tạo. Nó sẽ vẽ biểu đồ tổn thất theo thời gian, hiển thị đầu vào đào tạo, đầu ra đào tạo và các dự đoán hiện tại của mạng trên các chuỗi mẫu khác nhau trong một đợt đào tạo.

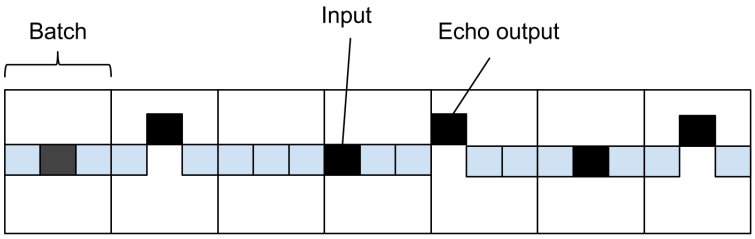
```
def plot(loss_list, predictions_series, batchX, batchY):
    plt.subplot(2, 3, 1)
    plt.cla()
    plt.plot(loss_list)

    for batch_series_idx in range(5):
```

```
        one_hot_output_series = np.array(predictions_series)[:,
            ↪ batch_series_idx, :]
        single_output_series = np.array([(1 if out[0] < 0.5 else 0) for
            ↪ out in one_hot_output_series])

        plt.subplot(2, 3, batch_series_idx + 2)
        plt.cla()
        plt.axis([0, truncated_backprop_length, 0, 2])
        left_offset = range(truncated_backprop_length)
        plt.bar(left_offset, batchX[batch_series_idx, :], width=1, color
            ↪ ="blue")
        plt.bar(left_offset, batchY[batch_series_idx, :] * 0.5, width=1,
            ↪ color="red")
        plt.bar(left_offset, single_output_series * 0.3, width=1, color="
            ↪ green")

    plt.draw()
    plt.pause(0.0001)
```

2.9 Bắt đầu chạy trainning

Nó có thời gian để kết thúc và huấn luyện mạng, trong TensorFlow, đồ thị được thực thi trong một phiên. Dữ liệu mới được tạo trên mỗi kỷ nguyên (không phải cách thông thường để làm điều đó, nhưng nó hoạt động trong trường hợp này vì mọi thứ đều có thể dự đoán được).

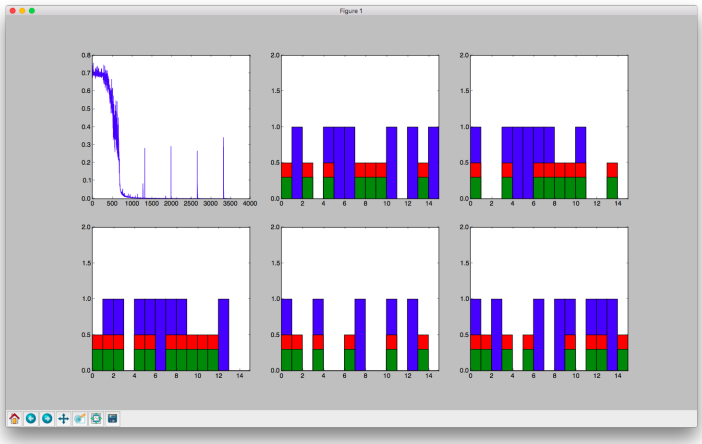
Bạn có thể thấy rằng chúng tôi đang di chuyển các bước truncated_backprop_length về phía trước trên mỗi lần lặp (dòng 15 dòng19), nhưng có thể có những bước tiến khác nhau. Chủ đề này được xây dựng thêm trong bài viết này. Nhược điểm của việc này là truncated_backprop_length cần phải lớn hơn đáng kể so với phụ thuộc thời gian (ba bước trong trường hợp của chúng tôi) để đóng gói dữ liệu đào tạo có liên quan. Nếu không, có thể có rất nhiều người bỏ lỡ, như bạn có thể thấy trong hình bên dưới.



Hình 2.13: Chuỗi thời gian của hình vuông, hình vuông màu đen nâng cao tượng trưng cho đầu ra echo, được kích hoạt ba bước từ đầu vào echo (hình vuông màu đen). Cửa sổ lô trượt cũng sải ba bước trong mỗi lần chạy, trong trường hợp mẫu của chúng tôi có nghĩa là không có lô nào sẽ đóng gói phụ thuộc, vì vậy nó không thể đào tạo.

Cũng nhận ra rằng đây chỉ là ví dụ đơn giản để giải thích cách RNN hoạt động, chức năng này có thể dễ dàng được lập trình chỉ trong một vài dòng mã. Mạng sẽ có thể tìm hiểu chính xác hành vi tiếng vang nên không cần kiểm tra dữ liệu.

Chương trình sẽ cập nhật cốt truyện khi tiến trình đào tạo, được hiển thị trong hình dưới đây. Thanh màu xanh biểu thị tín hiệu đầu vào đào tạo (nhị phân), thanh màu đỏ hiển thị tiếng vang trong đầu ra đào tạo và thanh màu xanh là tiếng vang mà mạng đang tạo. Các ô thanh khác nhau hiển thị loạt mẫu khác nhau trong lô hiện tại. Thuật toán của chúng tôi sẽ khá nhanh chóng tìm hiểu nhiệm vụ. Biểu đồ ở góc trên bên trái cho thấy đầu ra của hàm mất, nhưng tại sao lại có các gai trong đường cong? Hãy suy nghĩ về nó một lúc, câu trả lời dưới đây.



Hình 2.14: Trực quan hóa dữ liệu đào tạo mất, đầu vào và đầu ra (màu xanh, đỏ) cũng như dự đoán (màu xanh lá cây).

Lý do cho sự tăng đột biến là chúng tôi đang bắt đầu vào một kỷ nguyên mới và tạo ra dữ liệu mới. Vì ma trận được định hình lại, phần tử đầu tiên trên mỗi hàng nằm liền kề với phần tử cuối cùng trong hàng trước đó. Một vài thành phần đầu tiên trên tất cả các hàng (trừ đầu tiên) có các phụ thuộc sẽ không được đưa vào trạng thái, do đó, mạng sẽ luôn hoạt động kém trong lô đầu tiên.

Chương 3

Ứng dụng mô hình Question & Answer Systems vào chatbot

3.1 Tổng quan về hệ thống hỏi đáp

3.1.1 Đặt vấn đề

- Với số lượng ngày càng tăng nhanh chóng của tri thức trên Web, các máy tìm kiếm cần có nhiều trí thông minh hơn. Trong một vài trường hợp người sử dụng chỉ cần một phần chính xác của thông tin thay vì một danh sách các tài liệu.Thay vì bắt người sử dụng phải đọc toàn bộ tài liệu, nó thường được ưa chuộng hơn bằng cách đưa cho người sử dụng câu trả lời chính xác và ngắn gọn.Các hệ thống hỏi đáp (Question Answering systems-QA) phải cung cấp các phần thông tin chính xác cho các câu hỏi tương ứng. Một hệ thống hỏi đáp miền mở có thể trả lời được các câu hỏi viết bằng ngôn ngữ tự nhiên giống như con người

- Một trong các thành phần đóng vai trò quan trọng trong hệ thống hỏi đáp là phân loại câu hỏi.Nhiệm vụ của phân loại câu hỏi như sau: Cho 1 câu hỏi, ánh xạ câu hỏi đó tới một trong k lớp, các lớp đó cung cấp một gợi ý ngữ nghĩa về câu trả lời sau khi được tìm kiếm. Mục đích của sự phân loại này là giảm thiểu các câu trả lời không có tiềm năng, giai đoạn này được xử lý tại quá trình hạ lưu để lựa chọn câu trả lời chính xác từ một lượng các câu trả lời có tiềm năng

- Phân loại câu hỏi trong hệ thống hỏi đáp có 2 yêu cầu chính. Thứ nhất , nó

cung cấp các gợi ý về loại câu trả lời mà cho phép tiếp tục xử lý để xác định vị trí chính xác và xác minh câu trả lời. Thứ hai, nó cung cấp thông tin trong quá trình xử lý hạ lưu được sử dụng để lựa chọn các chiến lược cho từng câu trả lời cụ thể. Ví dụ như với câu hỏi: Who was the first woman killed in the Vietnam War?, các nhà nghiên cứu không muốn kiểm tra mọi cụm danh từ trong tài liệu để xem liệu có nó cung cấp một câu trả lời hay không. Mà ít nhất họ muốn biết rằng mục đích của câu trả lời này là về "person", do đó mà làm giảm thiểu đáng kể không gian các câu trả lời tiềm năng.

- Chúng ta cùng xem xét một số ví dụ tiếp theo được lấy từ bộ câu hỏi TREC 100, thể hiện một số khía cạnh của quan điểm này.

- Q:What is a prism? Câu hỏi này xác định rằng, mục đích mà nó hướng tới là "định nghĩa" (definition), chiến lược cụ thể cho các "definitions" (chẳng hạn như sử dụng các mẫu câu định sẵn: Prism is... hoặc Prism is defined as...) có thể trở nên hữu ích.

- Các ví dụ trên cho thấy rằng, các loại câu trả lời khác nhau được đưa ra tương ứng với câu hỏi có thể được tìm kiếm sử dụng nhiều chiến lược khác nhau, một phân loại tốt có thể giúp tìm ra câu trả lời tốt trong nhiệm vụ hỏi đáp. Hơn nữa, việc xác định loại ngữ nghĩa cụ thể của câu trả lời cũng có thể giúp ích trong việc tìm ra câu trả lời. Ví dụ với 2 câu hỏi: What Canadian city has the largest population?và Which country gave New York the Statue of Liberty? ta biết rằng mục tiêu là loại "city" hoặc "country" sẽ hữu ích hơn là chỉ biết chúng thuộc loại "localtions"

3.1.2 Hệ thống hỏi đáp (Question Answering System)

3.1.2.1 Giới thiệu

- QA system: là một hệ thống đóng vai trò phổ biến trong việc tìm kiếm thông tin chính xác và hiệu quả. Nhiệm vụ của nó là đưa ra câu trả lời đầy đủ và chính xác ứng với yêu cầu của người dùng và câu trả lời được thể hiện bằng ngôn ngữ tự nhiên. Người dùng nhanh chóng lấy được thông tin cần thiết thay vì tìm kiếm thông tin trong một khối lượng lớn các văn bản.

- Hầu hết các nghiên cứu trước đây chủ yếu là các hệ thống hỏi đáp trong một lĩnh vực đặc biệt hoặc là có sự giới hạn trong việc hỏi đáp. Do thiếu kiến thức

để cung cấp câu trả lời cho câu hỏi miền mở, các nghiên cứu về hệ thống hỏi đáp nằm im trong vài thập kỷ cho đến khi sự xuất hiện của các trang web. Với số lượng lớn của các dữ liệu trên web, cần phải thực hiện các truy vấn web, do đó các nhiệm vụ về hỏi đáp lại được tập trung nghiên cứu. Sự tập trung nghiên cứu về hỏi đáp đặc biệt tăng khi hội nghị truy hồi văn bản (Text REtrieval Conference-Trec) bắt đầu một chủ đề về hỏi đáp vào năm 1990.

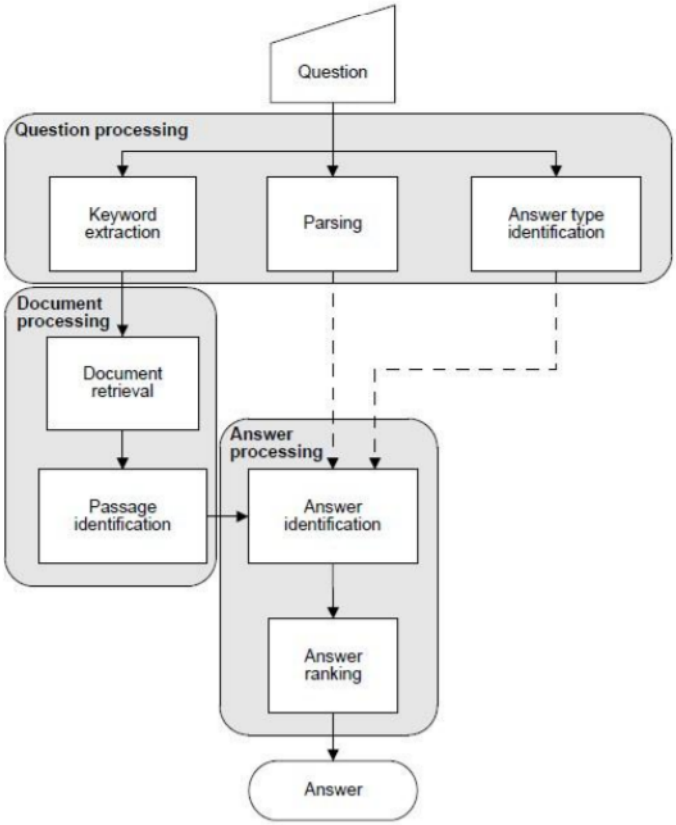
- Có 2 loại hệ thống hỏi đáp:

  + Hệ thống hỏi đáp miền đóng (Closed-domain Question Answering): hệ thống này liên quan đến các câu hỏi trong một lĩnh vực cụ thế, chẳng hạn như lĩnh vực y học.

  + Hệ thống hỏi đáp miền mở (Open-domain Question Answering): hệ thống này liên quan đến các câu hỏi gần như về tất cả mọi thứ.

3.1.2.2 Cấu trúc của một hệ thống hỏi đáp

Có nhiều hệ thống QA đã được đưa ra, nhưng hầu hết chúng đều tuân theo một khung làm việc chung. Thông thường, một hệ thống hỏi đáp xử lý 3 nhiệm vụ sau: xử lý câu hỏi, xử lý tài liệu và xử lý câu trả lời. Hình 3.2 dưới đây là kiến trúc tổng quan về một hệ thống hỏi đáp.



Hình 3.1: Kiến trúc tổng quan về một hệ thống hỏi đáp

3.2 ChatBot trong lĩnh vực học máy

Trong lĩnh vực học máy, ChatBot hay các trợ lý ảo đều được quy chung về một loại đó là Question and Answering system. Đối với các hệ chuyên gia như vậy, công việc bạn cần làm bao gồm:

- Phân loại câu hỏi.
- Mapping câu trả lời (Trích chọn tài liệu liên quan).
- Trích xuất câu trả lời.

Việc phân loại câu hỏi là bước khó nhất trong hệ thống hỏi đáp, Tuy nhiên, như chính cái tên của nó, bạn hoàn toàn có thể chuyển bài toán này về các bài toán phân lớp đã biết. Đến đây, có lẽ các bạn đã hình dung ra được các bước xây dựng chatBot không hề khó như những gì ta nghĩ phải không?

3.3 Chuẩn bị dữ liệu

Một bộ dữ liệu training đủ nhiều và chuẩn chỉnh có thể nâng cao độ chính xác lên rất nhiều. Mặc dù trong bài viết này dữ liệu nhóm em sử dụng là không nhiều nhưng một phần nào đó có thể giúp mọi người hình dung cấu trúc của dữ liệu training mà bạn cần có. Bạn có thể định nghĩa cấu trúc dữ liệu của bạn theo dạng json dưới đây:


```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Em chào thầy cô", "Xin chào", "Cho em hỏi", "Xin chào", "Thưa thầy","Thưa cô", "Em tên là *", "Chúng
         "responses": ["Chúng tôi có thể giúp gì cho em", "Em muốn hỏi chuyện gì", "Chào em - Em muốn hỏi chuyện gì"],
         "context_set": ""
        },
        {
          "tag":"conversation",
          "patterns":["Em thấy trong chương trình đào tạo có rất nhiều nhóm môn học khác nhau. Nó có ý nghĩa như thế nào?",
                      "Chương trình mình gồm những môn học nào"
                      ],
          "responses":["Môn học bắt buộc  Môn học bắt buộc theo lựa chọn chuyên ngành hoặc hướng chuyên môn  Môn học và nhó
          "context_set": "Em còn hỏi gì nữa không"
        },{
          "tag":"conversation_1",
          "patterns": ["Môn bắt buộc là gì","không hiểu môn bắt buộc là gì","Môn bắt buộc có ý nghĩa gì","Môn bắt buộc có ợ
                       "Môn bắt buộc có ảnh hưởng như thế nào","Môn bắt buộc em không học có được không","Bỏ môn học bắt b
          "responses": ["Đây là các môn học trong chương trình đào tạo chứa đựng những nội dung chính yếu của ngành và chuy
        },
        {
         "responses": "Môn học bắt buộc. Đây là các môn học trong chương trình đào tạo chứa đựng những nội dung chính yếu c
         "patterns": "Em thấy trong chương trình đào tạo có rất nhiều nhóm môn học khác nhau. Nó có ý nghĩa như thế nào?",
         "tag": "conversation_1"
```


Hình 3.2: Data dùng để huấn luyện

Việc quan trọng trong hệ thống chatbot là xác định được câu hỏi hay nội dung mà người dùng nhập.

- Bạn cần một **tag** (nhãn lớp cho nội dung nhập của người dùng)
- **patterns** - mẫu câu đầu vào được training để phân lớp
- **responses** - các câu trả lời (bot) được mapping để hồi đáp những request trước đó.

3.4 Training model

Chúng ta sẽ sử dụng TFlearn - High level API của Tensorflow, và dĩ nhiên ngôn ngữ sử dụng là Python. Đầu tiên, hãy import các thư viện cần thiết, đặc biệt là các

thư viện được xử dụng trong NLP như NLTK, tensorflow...

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

import numpy as np
import tflearn
import random
```

Loading data training. Vì dữ liệu traning của chúng ta đang ở dạng json nên đừng quên import json.

```
import json
with open('data/intents.json',encoding="utf-8") as json_data:
    intents = json.load(json_data)
```

Với intents.json vừa được đưa vào bộ training, bạn cần tổ chức lại nó, Xác định cho công cụ của bạn biết đâu là documents dùng để training, các từ, các classes để phân lớp. Bạn nên sử dụng thêm bộ công cụ của xử lý ngôn ngữ tự nhiên nltk để tiền xử lý dữ liệu. Bộ công cụ này cho phép bạn thực hiện các quá trình tokenizer, POS stagging, word segmentation, remove stopword....

```
words = []
classes = []
documents = []
ignore_words = ["em","vi"]

for intent in intents['intents']:
    for pattern in intent['patterns']:

        w = nltk.word_tokenize(pattern)
        words.extend(w)
        documents.append((w, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
```

```
words = [stemmer.stem(w.lower()) for w in words if w not in ignore_words
    ↪ ]
words = sorted(list(set(words)))
```

Bổ sung thêm stopwords vào stop_words array hoặc tạo một file text để list danh sách các stopwords .stopwords là danh sách các từ xuất hiện nhiều trong văn bản nhưng lại không có giá trị trong việc phân lớp. Vì vậy trước khi training, chúng ta cần làm sạch văn bản, loại bỏ những từ ngữ không có ý nghĩa này để tránh overfiting ảnh hưởng đến kết quả training.

Nếu chúng ta sử dụng dự liệu dạng word như vậy thì sẽ không thể hoạt động được với tensorflow, vì vậy việc cần thiết bây giờ là chuyển dữ liệu này sang dạng tensor number

```
#Create training data
training = []
output = []

output_empty = [0] * len(classes)

for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [stemmer.stem(word.lower()) for word in
        ↪ pattern_words]

    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])
```

```
random.shuffle(training)
```

Lưu ý rằng, dữ liệu của chúng ta bị xáo trộn. Tensorflow sẽ lấy một số dữ liệu trong tập intents.json để làm dữ liệu thử nghiệm để đo độ chính xác cho mô hình. Sau khi chuyển word -> number, bạn có thể thấy các "bag-of-words" array dạng như sau:

- **train_x example**: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1...]
- **train_y example**: [1, 0, 0, 0, 0, 0, 0, 0]

Với dữ liệu như trên, tensorflow đã có thể hiểu được nội dung bạn muốn training và bây giờ cần phải build neural network để training model ngay thôi. Ở đây tôi dùng hàm activation là softmax và optimizer function là Adam. Bạn có thể thay đổi 2 hàm này để có thể đưa ra kết quả training cao nhất và phù hợp nhất.

```
tf.reset_default_graph()

net = tflearn.input_data(shape=[None, len(train_x[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(train_y[0]), activation='softmax
    ↪ ')
net = tflearn.regression(net, optimizer='adam', loss='
    ↪ categorical_crossentropy')

model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')

model.fit(train_x, train_y, n_epoch=1000, batch_size=32, show_metric=
    ↪ True)
model.save('models/model.tflearn')
```

3.5 Xây dựng chatbot

Ở bước này chúng ta sẽ xây dựng hệ thống phản hồi của Chatbot, sử dụng intents model đã được training ở bước trên. Sau khi import các thư viện giống như bước

training ở trên, Bạn cần un-pickle model và documents, cũng như cần phải load lại intents.json (cái này để lấy các response đã được định nghĩa trước đó)

```
# restore our data structures
import pickle
data = pickle.load( open( "training_data", "rb" ) )
words = data['words']
classes = data['classes']
train_x = data['train_x']
train_y = data['train_y']

# import intents file
import json
with open('intents.json') as json_data:
    intents = json.load(json_data)
```

Hình 3.3: Xây dựng hệ thống phản hồi của Chatbot

Load tensorflow model

```
model.load('./model.tflearn')
```

Hình 3.4: Load tensorflow model

Cũng giống như khi training dữ liệu, với các ý định người dùng nhập vào bạn cũng cần phải tiền xử lý, tokenizer, hay chuyển sang bag-of-words để hệ thống có thể hiểu và phân loại về đúng lớp của nó

```
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
    sentence_words = [stemmer.stem(word.lower()) for word in sentence_words]
    return sentence_words

# bag of words
def bow(sentence, words, show_details=False):
    sentence_words = clean_up_sentence(sentence)
    # bag of words
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)

    return(np.array(bag))
```

Hình 3.5: chuyển sang bag-of-words

Hãy kiểm tra đoạn code trên với dữ liệu người dùng có thể nhập vào lúc order

```
bow('I want to special food', words)
>>> array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0])
```

Hình 3.6: Kiểm tra đoạn code trên với dữ liệu người dùng có thể nhập vào lúc order

Vậy là dữ liệu đầu vào đã xử lý xong. Và bây giờ là bộ xử lý phản hồi của chatbot

```
# data structure to hold user context
context = {}

ERROR_THRESHOLD = 0.25
def classify(sentence):
    results = model.predict([bow(sentence, words)])[0]
    results = [[i,r] for i,r in enumerate(results) if r>ERROR_THRESHOLD]
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append((classes[r[0]], r[1]))
    return return_list

def response(sentence, userID='1', show_details=False):
    results = classify(sentence)
    if results:
        while results:
            for i in intents['intents']:
                if i['tag'] == results[0][0]:
                    if 'context_set' in i:
                        if show_details: print ('context:', i['context_set'])
                        context[userID] = i['context_set']
                    if not 'context_filter' in i or \
                        (userID in context and 'context_filter' in i and i['context_filter'] == context[userID]):
                        if show_details: print ('tag:', i['tag'])
                        return print(random.choice(i['responses']))

            results.pop(0)
```

Hình 3.7: Bộ xử lý phản hồi của chatbot

Với mỗi câu hỏi (hay request) người dùng nhập vào, tôi sẽ sử dụng hàm model.predict() để xác định request đó thuộc loại nào. Sau đó sẽ đưa ra các phản hồi tiềm ẩn, có khả năng và phù hợp nhất với các request trước đó.

Chương 4

Tạo chatbot trên Facebook Messenger

Gần đây, Facebook đã mở nền tảng Messenger của họ để cho phép các bot trò chuyện với người dùng thông qua Ứng dụng Facebook và trên Trang Facebook.Chúng ta có thể đọc tài liệu của nhóm Messenger để tạo 1 chatbot trên Facebook .

Messenger bot sử dụng một máy chủ web để xử lý các tin nhắn mà nó nhận được hoặc để tìm ra những tin nhắn cần gửi. Bạn cũng cần phải xác thực bot để cuộc nói chuyện với máy chủ web và bot được Facebook chấp thuận để nói chuyện công khai.

4.1 Xây dựng máy chủ

Xây dựng máy chủ gồm các bước sau:

- **1**.Cài đặt công cụ Heroku từ trang https://toolbelt.heroku.com để khởi chạy, dừng và theo dõi các trường hợp. Đăng ký miễn phí tại https://www.heroku.com nếu chưa có tài khoản.
- **2**.Cài đặt Node từ trang https://nodejs.org, đây sẽ là môi trường máy chủ. Sau đó mở Terminal hoặc Prompt và đảm bảo bạn đã có phiên bản npm mới nhất bằng cách cài đặt lại:

```
sudo npm install npm -g
```

Hình 4.1: Cài đặt npm

- **3**.Tạo một thư mục mới ở đâu đó và tạo một dự án Node mới.Để khởi tạo một dự án node.js với npm chúng ta sử dụng lệnh sau. Nhấn Enter để chấp nhận mặc định.

```
npm init
```

Hình 4.2: Khởi tạo một dự án node.js với npm

- **4**.Cài đặt các phụ thuộc Node bổ sung. Express là dành cho máy chủ, yêu cầu là gửi thông báo và body-parser là để xử lý thông báo.

```
npm install express request body-parser --save
```

Hình 4.3: Cài đặt các phụ thuộc Node bổ sung

- **5**.Tạo một tệp index.js trong thư mục và sao chép tệp này vào đó. Chúng ta sẽ bắt đầu bằng cách xác thực bot.

```
'use strict'
const express = require('express')
const bodyParser = require('body-parser')
const request = require('request')
const app = express()

app.set('port', (process.env.PORT || 5000))
// Process application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: false}))
// Process application/json
app.use(bodyParser.json())
// Index route
app.get('/', function (req, res) {
    res.send('Hello world, I am a chat bot')
})
// for Facebook verification
app.get('/webhook/', function (req, res) {
    if (req.query['hub.verify_token'] === 'my_voice_is_my_password_verify_me') {
        res.send(req.query['hub.challenge'])
    }
    res.send('Error, wrong token')
})
// Spin up the server
app.listen(app.get('port'), function() {
    console.log('running on port', app.get('port'))
})
```

Hình 4.4: Dữ liệu cần có trong file index.js

- **6**.Tạo một tệp có tên Procfile và gõ dòng bên dưới hình vào.Để Heroku có thể biết nên chạy file nào.

```
web: node index.js
```

Hình 4.5: Dữ liệu giúp Heroku biết chạy file nào

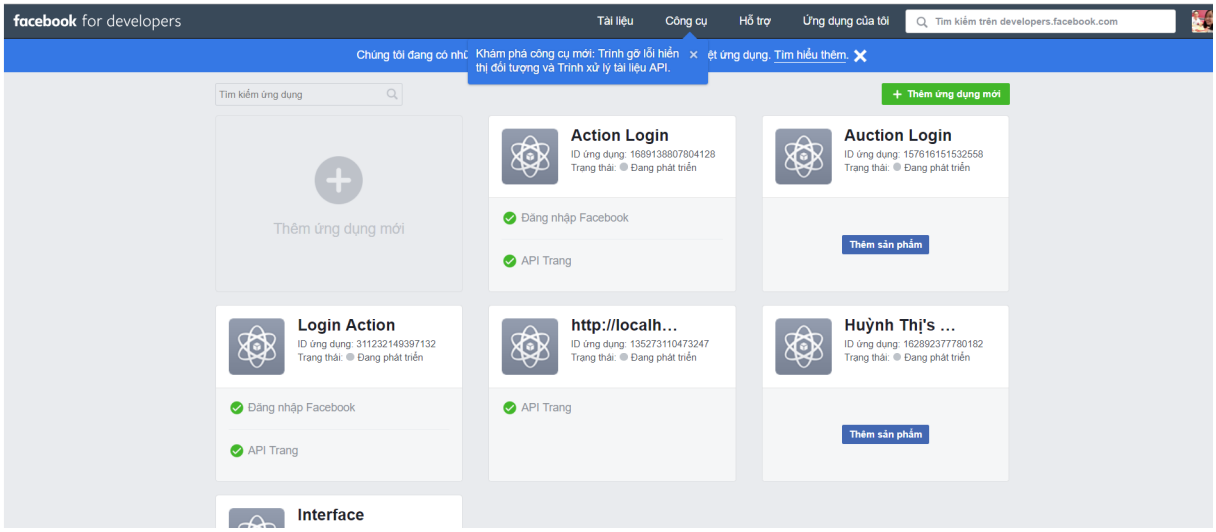
- **7**.Commit tất cả mã nguồn lên Git sau đó tạo một cá thể Heroku mới và đẩy mã nguồn lên đám mây.

```
git init
git add .
git commit --message "hello world"
heroku create
git push heroku master
```

Hình 4.6: Các câu lệnh để commit tất cả mã nguồn và tạo Heroku

4.2 Cài đặt ứng dụng facebook

- **1**.Tạo hoặc cấu hình Ứng dụng hoặc page của Facebook tại đây https://developers.facebook.com/apps/



Hình 4.7: Trang web để tạo hoặc cấu hình ứng dụng hoặc page facebook

- **2**.Trong ứng dụng, chuyển đến tab Messenger, sau đó nhấp vào Cài đặt Webhook. Tại đây bạn sẽ đặt URL của máy chủ Heroku và mã token.Hãy đảm bảo kiểm tra tất cả các trường đăng ký.
- **3**.Nhận mã thông báo truy cập trang và lưu nó ở đâu đó.
- **4**.Quay trở lại Terminal và gõ lệnh này để kích hoạt ứng dụng Facebook để gửi tin nhắn. Hãy nhớ sử dụng mã thông báo bạn yêu cầu trước đó.

- **7**.Commit tất cả mã nguồn lên Git sau đó tạo một cá thể Heroku mới và đẩy mã nguồn lên đám mây.

```
git init
git add .
git commit --message "hello world"
heroku create
git push heroku master
```

Hình 4.8: Các câu lệnh để commit tất cả mã nguồn và tạo Heroku

4.3 Cài đặt bot

Bây giờ Facebook và Heroku có thể nói chuyện với nhau, chúng tôi có thể mã hóa bot.

- Thêm một điểm cuối API vào index.js để xử lý tin nhắn. Hãy nhớ cũng bao gồm mã thông báo mà chúng tôi đã nhận được trước đó.

```
app.post('/webhook/', function (req, res) {
    let messaging_events = req.body.entry[0].messaging
    for (let i = 0; i < messaging_events.length; i++) {
        let event = req.body.entry[0].messaging[i]
        let sender = event.sender.id
        if (event.message && event.message.text) {
            let text = event.message.text
            sendTextMessage(sender, "Text received, echo: " + text.substring(0, 200))
        }
    }
    res.sendStatus(200)
})

const token = "<PAGE_ACCESS_TOKEN>"
```

Hình 4.9: Trang web để tạo hoặc cấu hình ứng dụng hoặc page facebook

- Cài đặt token facebooks

```
function sendTextMessage(sender, text) {
    let messageData = { text:text }
    request({
            url: 'https://graph.facebook.com/v2.6/me/messages',
            qs: {access_token:token},
            method: 'POST',
```

```
            json: {
                recipient: {id:sender},
                    message: messageData,
            }
        }, function(error, response, body) {
            if (error) {
                console.log('Error sending messages: ', error)
            } else if (response.body.error) {
                console.log('Error: ', response.body.error)
          }
    })
}
```

- Commit code lên Heroku

```
git add .
git commit -m 'updated the bot to speak'
git push heroku master
```

Chương 5

Kết Luận

Chatbots sẽ không bị giới hạn trong cửa sổ nhắn tin. Kiến trúc được đề cập ở trên có thể được khái quát cho tất cả các hệ thống tương tác. Cuối cùng, đó là về việc sử dụng ngôn ngữ để hiểu cấu trúc của thế giới. Các hình thức ngôn ngữ sẽ thay đổi khi thời gian tiến triển, từ văn bản đến biểu tượng cảm xúc sang thần kinh, nhưng điều không thay đổi là sự thể hiện của thế giới trong tâm trí và nhu cầu của chúng ta để truyền đạt suy nghĩ của chúng ta. Bots là khởi đầu của một giao diện giữa con người và trí thông minh chung nhân tạo. Kiến trúc và các công cụ mà chúng tôi khảo sát giúp người ta có ý tưởng về loại bot sẽ được xây dựng cho hệ thống của họ, những gì mong đợi từ bot và sử dụng công cụ nào để xây dựng nó. Chúng tôi cũng hy vọng rằng điều này cung cấp một hướng cho những đổi mới và lĩnh vực nghiên cứu có thể có trong các giao diện đàm thoại dựa trên văn bản.

Tài liệu tham khảo

[1] Ketakee Nimavat Prof. Tushar Champaneria - UG student Assistant Professor *Chatbots: An overview Types, Architecture, Tools and Future Possibilities*, [*Third Edition*].

[2] Pender - author *https:/ /github.com/pender/chatbot-rnn*

[3] Ilya Sutskever - Oriol Vinyals - Quoc V. Le Sequence to Sequence Learning with Neural Networks

[4] M. Auli, M. Galley, C. Quirk, and G. Zweig.Joint language and translation modeling with recurrent neural networks. In EMNLP, 2013