TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

ỨNG DỤNG HỌC SÂU XÂY DỰNG MÔ HÌNH ĐỐI THOẠI

ĐỒ ÁN III

Chuyên ngành: Toán Tin

Chuyên sâu: Tin học

Giảng viên hướng dẫn: **TS. VŨ THÀNH NAM**

Sinh viên thực hiện: **NGUYỄN MINH QUÂN**

Lớp: **KSTN Toán Tin - K61**

HÀ NỘI, 01/2021

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Mục đích và nội dung của đồ án

2. Kết quả đạt được

3. Ý thức làm việc của sinh viên

Hà Nội, ngày tháng 01 năm 2021

Giảng viên hướng dẫn
(Ký và ghi rõ họ tên)

Mục lục

Lời mở đầu

Bài toán xây dựng hệ thống trả lời tự động, Chatbot là một bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên. Với lịch sử nghiên cứu và phát triển lâu đời của lĩnh vực trí tuệ nhân tạo nói chung và bài toán Chatbot nói riêng đã có rất nhiều thuật toán và phương pháp xây dựng được mô hình Chatbot. Với mục tiêu xây dựng một Chatbot thông minh có thể hiểu và nói chuyện như con người, song các phương pháp tiếp cận này đều có những hạn chế.

Trong những năm gần đây với sự phát triển của lĩnh vực trí tuệ nhân tạo ứng dụng vào thực tế và cùng với sự phát triển của các hệ thống máy tính, các nghiên cứu ra đời nhằm ứng dụng các thuật toán học máy và các mô hình học sâu vào bài toán Chatbot. Bài báo cáo này tập trung chủ yếu giới thiệu về mạng neuron hồi tiếp và mô hình Sequence-to-Sequence là các kiến thức trong học sâu để xây dựng một mô hình Chatbot sinh hội thoại đơn giản.

1. **Chương 1 :** Tổng quan về hệ thống trả lời tự động.
2. **Chương 2 :** Cơ sở mạng neuron nhân tạo.
3. **Chương 3 :** Mô hình đối thoại với mạng neuron.
4. **Chương 4 :** Cài đặt chương trình và kết quả thực nghiệm.

Để hoàn thành bài báo cáo đồ án 3 này, em xin gửi lời cảm ơn tới thầy giáo TS. Vũ Thành Nam đã có những nhận xét, góp ý trong phần trình bày của em để bài báo cáo của em được hoàn thiện hơn. Trong quá trình thực hiện do kiến thức còn nhiều hạn chế và thời gian có giới hạn, nên bài báo cáo chắc chắn vẫn còn những thiếu sót, em rất mong những nhận xét và góp ý của thầy cô và các bạn sinh viên để bài báo cáo được hoàn thiện hơn.

Hà Nội, ngày 01 tháng 01 năm 2021

Danh mục kí hiệu và các thuật ngữ

Từ viết tắt	Từ chuẩn	Diễn giải
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Máy học
DL	Deep Learning	Học sâu
Chatbot	Chatbot	Hệ thống đối thoại tự động
Retrieval-based	Retrieval-based	Mô hình truy xuất thông tin
Generation-based	Generation-based	Mô hình sinh hội thoại
NN	Neural Network	Mạng neuron nhân tạo
RNN	Recurrent Neural Network	Mạng neuron hồi tiếp
biRNN	Bidirectional RNN	Mạng RNN hai chiều
LSTM	Long Short Term Memory	Mạng bộ nhớ dài ngắn hạn
Seq2Seq	Sequence-to-Sequence	Mô hình chuỗi sang chuỗi
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
Python	Python Programming Language	Ngôn ngữ lập trình Python
API	Application Programming Interface	Giao diện lập trình ứng dụng

Danh sách hình vẽ

Danh sách bảng

Chương 1

Tổng quan về hệ thống trả lời tự động

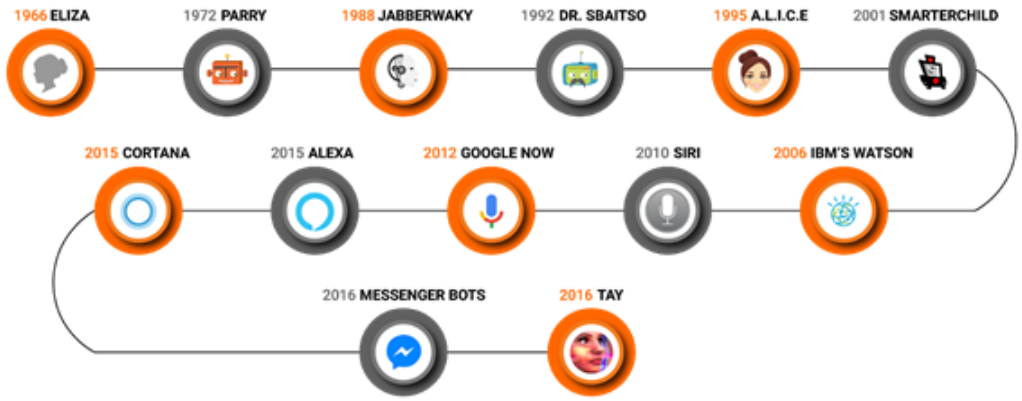
Bài toán xây dựng hệ thống trả lời tự động là một bài toán khó thuộc lĩnh vực xử lý ngôn ngữ tự nhiên (**Natural Language Processing - NLP**). Bởi vì tính nhập nhằng, đa nghĩa, đa ngữ cảnh của ngôn ngữ tự nhiên. Bài toán đặt ra nhiều thách thức để phát hiện ra được câu trả lời phù hợp nhất, thông tin hữu ích nhất. Chương này sẽ giới thiệu tổng quan về hệ thống đối thoại người máy, tìm hiểu lịch sử hình thành và phát triển cũng như các phương pháp tiếp cận của các nghiên cứu. Các kiến thức trong chương này được tham khảo từ [1], [2] và [3].

1.1 Hệ thống đối thoại người và máy

Các hệ thống đối thoại người máy (Dialogue systems), còn được gọi là trợ lý tương tác hội thoại, trợ lý ảo và đôi khi được gọi với thuật ngữ là Chatbot, được sử dụng rộng rãi trong các ứng dụng khác nhau, từ các dịch vụ kỹ thuật cho đến các công cụ có thể học ngôn ngữ và giải trí. Định nghĩa một cách đơn giản nhất, Chatbot là một chương trình tương tác với người dùng bằng ngôn ngữ tự nhiên với một giao diện đơn giản, âm thanh hoặc dưới dạng tin nhắn. Chatbot có rất nhiều ứng dụng trên các lĩnh vực khác nhau như: giải trí, y tế, thương mại, tự động hóa,...

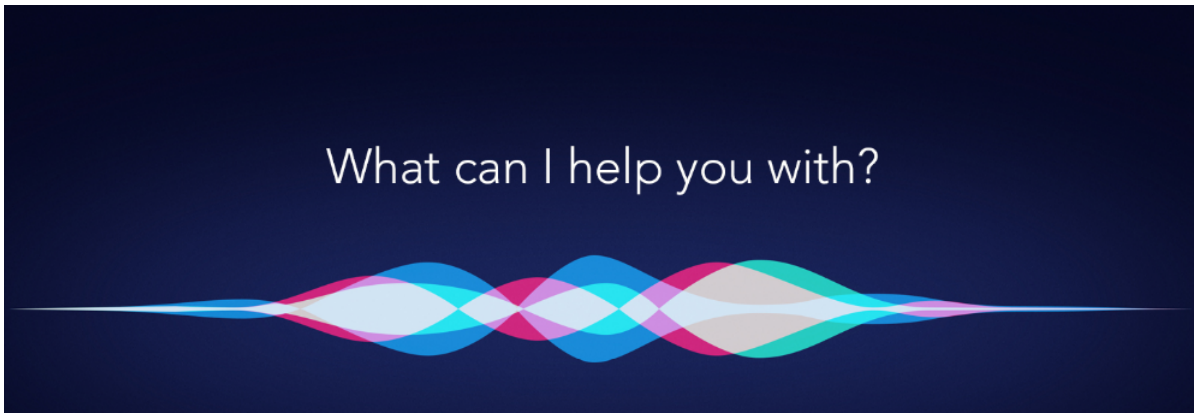
Lịch sử phát triển của Chatbot gắn liền với lịch sử phát triển của lĩnh vực trí tuệ nhân tạo. Vào năm 1950, Alan Turing viết báo cáo với tựa đề "Computing Machinery and Intelligence", xây dựng các phép thử Turing. Về cơ bản, các phép thử Turing là một loạt các phép thử dựa trên việc phân tích câu trả lời của một "máy tính". Các phép thử Turing được thực hiện để xác định xem một chương trình máy tính có thể phân biệt được máy tính với con người trong một cuộc trò chuyện chỉ có văn bản thuần túy hay không? Bằng cách gõ câu hỏi cho cả hai đối tượng thử nghiệm, người thẩm vấn sẽ cố gắng xác định đối tượng nào là máy tính và đối tượng nào là con người. Máy tính sẽ vượt qua các phép thử Turing nếu người thẩm vấn không thể nói sự khác biệt giữa chủ thể con người và máy tính. Năm 1966, chương trình trò chuyện đầu tiên được tạo

ra Joseph Weizenbaum xuất bản chương trình ELIZA, được coi là một trong những chương trình Chatbot đầu tiên trên thế giới. Sau đó lần lượt các hệ thống Chatbot ra đời với các mục đích khác nhau: Chatbot PARRY (1972), Dr.SBAITSO (1992) và Chatbot A.L.I.C.E (1995).



Hình 1.1: Lịch sử hình thành và phát triển của Chatbot[1].

Trong thập kỷ đầu tiên của thế kỷ 21, đã có những đột phá trong lĩnh vực học máy và trí thông minh nhân tạo, tạo điều kiện cho sự phát triển của các sản phẩm ứng dụng AI nói chung và hệ thống Chatbot nói riêng. Giao đoạn 2010-2016 chứng kiến sự bùng nổ của các trợ lý cá nhân ảo: Siri (2010), Google Now (2012), Alexa (2015), Cortana (2015) và Google Assistant (2016). Với khả năng phân tích và xử lý ngôn ngữ tự nhiên, các trợ lý này kết nối với các dịch vụ web để trả lời các câu hỏi và đáp ứng các yêu cầu của người dùng.



Hình 1.2: Trợ lý ảo Siri của Apple.

[1]Nguồn ảnh: `https://automationedge.com/`

Hiện nay, Chatbot phát triển dựa trên sự kết hợp của các kịch bản có trước và tự học trong quá trình tương tác. Người dùng sẽ tương tác với Chatbot qua tin nhắn. Với các câu hỏi được đặt ra, Chatbot sử dụng các hệ thống xử lý ngôn ngữ tự nhiên để phân tích dữ liệu sau đó chúng lựa chọn các thuật toán học máy để đưa ra các loại phản hồi khác nhau, chúng sẽ dự đoán và phản hồi chính xác nhất có thể. Chatbot sử dụng nhiều hệ thống quét các từ khoá do người dùng nhập vào, sau đó bot khởi động một hành động, kéo một câu trả lời với các từ khóa phù hợp nhất và trả lời thông tin từ một cơ sở dữ liệu hoặc một API. Nếu tình huống đó chưa xảy ra hoặc không có trong dữ liệu, Chatbot sẽ bỏ qua nhưng sẽ đồng thời có thể tự học để áp dụng cho các cuộc trò chuyện về sau. Một trong các yếu tố làm nên sức mạnh của Chatbot là khả năng tự học hỏi. Càng được sử dụng, tương tác với người dùng nhiều, nền tảng Chatbot càng "thông minh". Chatbot thông minh có khả năng tự học hỏi dựa trên các dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Mặc dù có những tiến bộ như vậy nhưng các chuyên gia AI vẫn chưa thể phát triển một Chatbot có thể tư duy và giao tiếp thực sự với con người. Thay vào đó, tập trung vào việc xây dựng Chatbot cho các mục đích cụ thể, chủ yếu là các trợ lý ảo có khả năng truy cập dữ liệu và trả lời các câu hỏi. Như vậy, nếu phân loại theo hướng tiếp cận, chúng ta có thể đưa ra các mô hình Chatbot:

- **Mô hình truy xuất thông tin**: Xây dựng các kịch bản, định nghĩa các câu trả lời từ trước. Khi có một câu hỏi đầu vào, hệ thống sẽ sử dụng các kỹ thuật heuristic, kết hợp với các thuật toán học máy để phân loại câu hỏi và đưa ra đáp án từ tập dữ liệu cố định.
- **Mô hình sinh hội thoại**: Mô hình này không dựa trên tập dữ liệu được định nghĩa từ trước. Mô hình này học các hội thoại có sẵn bằng các thuật toán học máy, sau đó với một câu hỏi đầu vào, mô hình sẽ sinh ra câu trả lời tương ứng.
- **Mô hình kết hợp**: Mô hình kết hợp cả hai mô hình: mô hình truy xuất thông tin và mô hình sinh hội thoại. Đây là hướng tiếp cận xây dựng Chatbot tương đối mới.

Trong phạm vi của bài báo cáo này, chúng ta chỉ đề cập đến mô hình truy xuất thông tin và mô hình sinh hội thoại. Cả hai mô hình này đều có những ưu diểm và hạn chế sẽ được đề cập vào trong phần sau của bài báo cáo.

1.2 Xử lý ngôn ngữ tự nhiên cho Chatbot

Xử lý ngôn ngữ tự nhiên là một lĩnh vực của trí tuệ nhân tạo cho phép máy tính để phân tích và hiểu ngôn ngữ của con người.

Để xây dựng Chatbot, chúng ta cần biết các phương pháp cơ bản của xử lý ngôn ngữ tự nhiên. Những phương pháp này giúp chúng ta chia đầu vào thành các phần nhỏ hơn và hiểu rõ hơn về nó. Nếu có thể xử lý văn bản đầu vào một cách hiệu quả, Chatbot có thể đưa ra những phản hồi tốt cho người dùng. Dưới đây là một số phương pháp thường dùng trong xử lý ngôn ngữ tự nhiên:

- **Gán nhãn từ loại (Part of speech-POS)**: Quá trình đọc một số văn bản và chỉ định các thành phần của câu như danh từ, động từ, tính từ,... Gắn thẻ POS trở nên cực kỳ quan trọng khi muốn xác định một số thực thể trong một câu đã cho. Bước đầu tiên là thực hiện gắn thẻ POS và xem văn bản xét chứa những gì.

- **Rút gọn (Stemming)** và **chuẩn hóa (Lemmatization)**: Rút gọn kỹ thuật dùng để biến đổi một từ về dạng từ gốc của nó. Chẳng hạn trong tiếng Anh biến đổi từ "walking" về dạng gốc là từ "walk". Chuẩn hóa là kỹ thuật xử lý thông minh hơn bằng một bộ từ điển hoặc một bộ ontology nào đó. Điều này sẽ đảm bảo rằng các từ như “goes“, “went” và “go” sẽ chắc chắn có kết quả trả về là như nhau.

- **Nhận dạng thực thể có tên (Named Entity Recognition- NER)**: Quá trình tìm kiếm và phân loại các thực thể được đặt tên tồn tại trong văn bản đã cho vào các danh mục được xác định trước. Chẳng hạn, "Albert Einstein" là tên của một người (person), "Google" là tên của một công ty (ORG).

- **Các từ xuất hiện nhiều (Stop Word)**: Các từ dừng là những từ có tần suất cao như a, an, the, to (trong Tiếng Anh) và đôi khi chúng ta muốn lọc ra khỏi tài liệu trước khi xử lý thêm. Các từ dừng thường có ít nội dung và không có nhiều ý nghĩa.

- **Phân tích sự phụ thuộc về cú pháp (Dependency Parsing)**: Tìm hiểu xem làm thế nào tất cả các từ trong câu của chúng ta liên quan đến nhau. Hay nói cách khác là tìm hiểu xem các từ trong câu của chúng ta liên quan đến nhau như thế nào.

- **Tìm sự tương đồng (Finding Similarity)**: Tìm sự giống nhau giữa hai từ. Đôi khi, việc tìm xem hai từ có phải là một hay không. **GloVe** là một thuật toán học tập không giám sát để lấy các biểu diễn vectơ cho từ ngữ. Thuật toán GloVe sử dụng thống kê đồng xuất hiện tương ứng từ với từ trên toàn cầu được tổng hợp từ một kho dữ liệu để đào tạo mô hình.

1.3 Mô hình truy xuất thông tin

Mô hình truy xuất thông tin (**Retrieval-based**) là mô hình trong đó Chatbot đưa ra những phản hồi được chuẩn bị trước hoặc tuân theo những kịch bản nhất định. Mô hình này khác với mô hình tự động sinh câu trả lời, trong đó câu trả lời của Chatbot được tự động sinh ra bằng việc học từ một tập dữ liệu các đoạn hội thoại. Trước khi tìm hiểu mô hình Chatbot truy xuất thông tin, chúng ta sẽ điểm qua một số các thuật ngữ quan trọng được sử dụng khi xây dựng Chatbot.

Ý định (Intent): Khi người dùng tương tác với một hệ thống Chatbot, cần xác định ý định sử dụng hệ thống của người dùng là gì. Chẳng hạn khi người sử dụng thông báo "Tôi muốn đặt một vé xem phim ", thì chúng ta hiểu rằng người dùng muốn đặt một vé xem phim. Nội dung ấy có thể được tóm tắt lại bằng từ khó "đặt vé xem phim". Chúng ta có thể tự đặt tên cho các ngữ cảnh hay ý định mà chúng ta mong muốn.

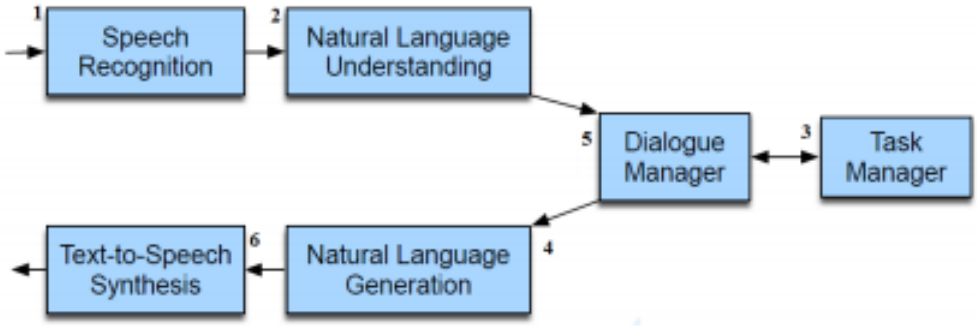
Thực thể (Entities): Ý định mang dữ liệu (metadata) được gọi là các thực thể. Ví dụ trên khi người sử dụng thông báo " Tôi muốn đặt một vé xem phim" thì đặt vé là một "ý định" và "phim" là một thực thể và thực thể ở đây có thể là một cái gì đó khác như "chuyến bay" hay "một buổi hòa nhạc". Chúng ta có thể gán nhãn các thực thể trong xuyên suốt các ý định. Các thực thể có thể là số lượng, số đếm hoặc khối lượng. Ý định cũng có thể có nhiều thực thể. Một ý định cũng có thể bao gồm nhiểu thực thể. Chẳng hạn, khi người dùng cuối thông báo "Đặt cho tôi một đôi giày có size 42". Ở đây thông báo này bao gồm hai thực thể đó là phụ kiện là "giày" và kích thước là "42".

Các câu trả lời (Utterance): Các câu trả lời khác nhau cho cùng một ý định hay câu hỏi của người sử dụng. Chẳng hạn, khi người sử dụng thông báo "xin chào", thì cách tối ưu nhất trả lời "xin chào", hoặc đôi khi là "Bạn có khỏe không?".

Huấn luyện máy (Training the Bot): Huấn luyện cơ bản là một quá trình xây dựng mô hình học từ tập hợp các ý đinh/thực thể và cách câu trả lời đã xác định về cách phân loại các cách phát biểu mới và cung cấp điểm tin cậy cùng với nó.

Điểm tin cậy của mô hình: Mỗi khi bạn cố gắng tìm ra ý định của một câu nói nào, mô hình của bạn sẽ xuất hiện lên với điểm số tin cậy. Điểm số này cho bạn biết mức độ chính xác của mô hình là về việc xác định ý định của người dùng.

Trong mô hình Chatbot kiểu truy xuất thông tin, thành phần hiểu ngôn ngữ tự nhiên (Natural Language Uderstanding - NLU) sử dụng rất nhiều kỹ thuật xử lý ngôn ngữ tự nhiên để phân tích câu hội thoại, xác định ý định của người dùng, trích xuất những thông tin của người dùng cung cấp. Sau đó những thông tin này được chuyển tới cho thành phần quản lý hội thoại. Thành phần quản lý hội thoại sẽ xử lý thông tin và chuyển tới cho thành phần sinh ngôn ngữ để sinh ra câu hội thoại. Kiến trúc cơ bản của một hệ thống giao tiếp tự động được mô tả trong hình 1.3.



Hình 1.3: Kiến trúc cơ bản của một hệ thống giao tiếp tự động[2].

Như vậy để thiết kế mô hình Chatbot kiểu truy xuất thông tin có ba vấn đề chính cần phải giải quyết:

- Xác định ý định của người dùng (Intent Classification hay Intent Detection).
- Trích xuất thông tin (Information Extraction).
- Quản lý hội thoại (Dialog Management).

Trên thực tế, hầu hết các Chatbot đều tuân theo mô hình truy xuất thông tin và được áp dụng trong các lĩnh vực nhất định, điều này đảm bảo một Chatbot có thể vận hành tốt và đưa ra những câu trả lời thích hợp với người dùng cuối. Một số các nền tảng Chatbot phổ biến và tốt nhất có thể kể đến như:

1. `https://cloud.google.com/dialogflow`: Là một trong những nền tảng phổ biến nhất. Kết nối với người dùng trên Trợ lý Google, Amazon Alexa, Facebook Messenger và các nền tảng và thiết bị phổ biến khác.
2. `https://core.rasa.ai/`: Người dùng có thể triển khai các hành động mà bot của bạn có thể thực hiện bằng mã Python. Logic của bot dựa trên mô hình xác suất được đào tạo trên ví dụ cuộc trò chuyện.

[2]Nguồn ảnh: `https://techinsight.com.vn`

1.4 Mô hình sinh hội thoại

Hầu hết các nghiên cứu Chatbot trước đây đều tập trung vào việc sử dụng các phương pháp dựa trên truy xuất. Mô hình dựa trên truy xuất có lợi thế là trả về câu trả lời mạch lạc và đem lại nhiều thông tin trong các lĩnh vực khác nhau. Tuy nhiên, mô hình này gặp những vấn đề độ lớn dữ liệu khi nhà phát triển cần gán nhãn dữ liệu huấn luyện cho bộ phân lớp ý định. Trong nhiều các lĩnh vực, cần có kiến thức về chuyên môn nên công sức tạo ra những bộ dữ liệu huấn luyện là khá đắt đỏ. Chẳng hạn trong lĩnh vực y học, để xây dựng mô hình Chatbot chuẩn đoán bệnh phải cần những chuyên gia về y tế và bác sĩ hiểu biết về lĩnh vực này.

Trong những năm gần đây, với sự phát triển của các máy tính với hiệu suất cao và những nghiên cứu chuyên sâu, mô hình sinh hội thoại (**Generation-based**) bằng học sâu ra đời. Bắt nguồn từ bài báo "Sequence-to-Sequence Learning with Neural Networks" của nhóm tác giả của Google [4], sau đó là các bài báo "A Neural Conversational Model" [5], "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Model" [6],... tiếp cận theo mô hình sinh hội thoại bằng mạng neuron Sequence-to-Sequence. Mô hình Sequence-to-Sequence bao gồm 2 mạng neuron hồi tiếp. Mạng neuron thứ nhất đóng vai trò mã hoá câu hội thoại thành một vec-tơ số thực và mạng neuron thứ hai đóng vai trò giải mã và dự đoán các từ trong câu phản hồi. Ta có thể xây dựng bộ mã hóa và bộ giải mã bằng mạng neuron hồi tiếp với phiên bản là mạng LSTM hoặc mạng GRU. Mặc dù mô hình sinh hội thoại có thể đưa ra những phản hồi mới tuy nhiên một trong những vấn đề phổ biến của mô hình này là sinh ra những câu trả lời không mang đầy đủ thông tin cho câu hỏi và có thể mắc những lỗi ngữ pháp. Bên cạnh đó, mô hình sinh hội thoại yêu cầu số lượng hội thoại đủ lớn để học được mô hình sinh tốt. Sự khác biệt của hai mô hình này được thể hiện trong hình 1.4.

Vào năm 2017, kỹ thuật Attention được phát triển là một bước tiến quan trọng trong mô hình Sequence-to-Sequence và đã được chứng tỏ trong nhiều bài toán xử lý ngôn ngữ tự nhiên. Một số bài báo gần đây xuất hiện với mục tiêu là kết hợp cả hai mô hình, mô hình truy xuất thông tin và mô hình sinh hội thoại. Bài báo "A Hybrid Retrieval-Generation Neural Conversation Model" [7], đã xây dựng mô hình kết hợp hai mô hình nói trên thử nghiệm trên bộ dữ liệu của Twiter và Foursquare có kết quả tốt hơn cả hai mô hình được thử nghiệm trên cùng bộ dữ liệu.

Item	Retrieval-based methods	Generation-based methods
Main techniques	Retrieval models; Neural ranking models	Seq2Seq models
Diversity	Usually good if similar contexts have diverse responses in the repository	Easy to generate bland or universal responses
Response length	Can be very long	Usually short
Context property	Easy for similar context in the repository; Hard for unseen context	Easy to generalize to unseen context
Efficiency	Building index takes long time; Retrieval is fast	Training takes long time; Decoding is fast
Flexibility	Fixed response set once the repository is constructed	Can generate new responses not covered in history
Fluency	Natural human utterances	Sometimes bad or contain grammar errors
Bottleneck	Size and coverage of the repository	Specific responses; Long text; Sparse data
Informativeness	Easy to retrieve informative content	Hard to integrate external factual knowledge
Controllability	Easy to control and explain	Difficult to control the actual generated content

Hình 1.4: So sánh mô hình truy xuất thông tin và mô hình sinh hội thoại [3].

[3]Nguồn ảnh: [7]

Chương 2

Cơ sở mạng neuron nhân tạo

Chương này giới thiệu về cơ sở lý thuyết về mạng neuron nhân tạo, cách thức hoạt động của mạng neuron và các phiên bản mở rộng của mạng neuron nhân tạo như: Mạng neuron hồi tiếp, mạng cải tiến LSTM. Trong đó mạng neuron hồi tiếp là một thuật toán được chú ý rất nhiều trong thời gian gần đây bởi các kết quả tốt thu được trong lĩnh vực xử lý ngôn ngữ tự nhiên. Các kiến thức trong chương này này được tham khảo từ [1], [2] và [9] đó là kiến thức cơ sở được sử dụng trong các phần tiếp theo của bài báo cáo.

2.1 Mạng neuron nhân tạo

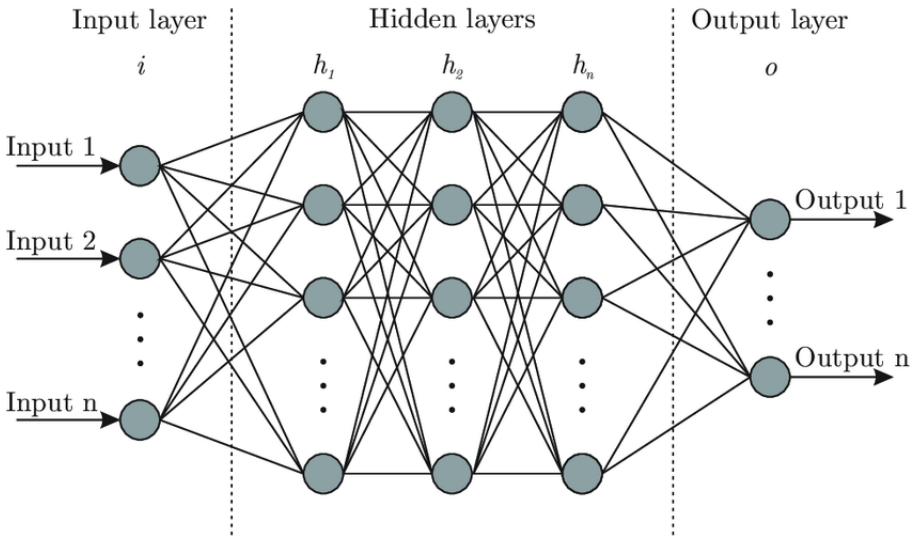
Mạng neuron nhân tạo (**Neural Network - NN**) là một mô hình lập trình rất đẹp lấy cảm hứng từ mạng neuron thần kinh. Kết hợp với các kĩ thuật học sâu (Deep Learning), mạng neuron đang trở thành một công cụ rất mạnh mẽ mang lại hiệu quả tốt nhất cho nhiều bài toán khó như nhận dạng ảnh, giọng nói hay xử lý ngôn ngữ tự nhiên. Sau đây, chúng ta sẽ tìm hiểu các thành phần cấu tạo nên một mạng neuron và cách thức hoạt động của một mạng neuron.

2.1.1 Kiến trúc mạng neuron nhân tạo

Mạng neuron bao gồm có 3 tầng chính, được thể hiện đơn giản trong hình 2.1:

- **Tầng vào** *(Input layer)*: Là tầng bên trái cùng của mạng thể hiện cho các đầu vào của mạng.
- **Tầng ra** *(Ouput layer)*: Là tầng bên phải cùng thể hiện cho các đầu ra của mạng.
- **Tầng ẩn** *(Hidden layer)*: Là tầng nằm giữa giữa tầng vào và tầng ra thể hiện cho việc suy luận logic của mạng.

Một mạng neuron chỉ có 1 tầng vào và 1 tầng ra, nhưng có thể không có hoặc có thể có nhiều tầng ẩn. Các đặc trưng của input sẽ được đổ vào tầng vào, mỗi nốt của tầng vào tương ứng với 1 đặc trưng của input. Sau đó được lan truyền tiến qua mạng để tính toán ra các đầu ra ở tầng ra, từ tầng đầu ra này, hàm mất mát đánh giá và lại lan truyền ngược lại để cập nhật lại các tham số mạng để cho mạng càng ngày càng dự đoán tốt cho mô hình. Các tham số của mạng chính là các ma trận trọng số liên kết giữa các tầng mạng với nhau. Trong mạng neuron, các nút mạng thường có một hàm kích hoạt nào đó và có thể khác nhau. Ở mỗi tầng, số lượng các nút mạng có thể khác nhau tuỳ thuộc vào bài toán và cách giải quyết. Mạng neuron giải quyết được các bài toán phân lớp không tuyến tính nhờ vào cơ chế của các hàm kích hoạt.



Hình 2.1: Kiến trúc của mạng neuron nhân tạo [1].

Có rất nhiều hàm kích hoạt trong xây dựng mạng neuron. Trong phạm vi của báo cáo này, chúng ta xét các hàm kích hoạt sau:

- Hàm sigmoid:
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- Hàm tanh:
$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$
- Hàm reLU:
$$\text{reLU}(x) = \max(0, x)$$

[1]Nguồn ảnh: `https://www.researchgate.net`

2.1.2 Hoạt động mạng neuron nhân tạo

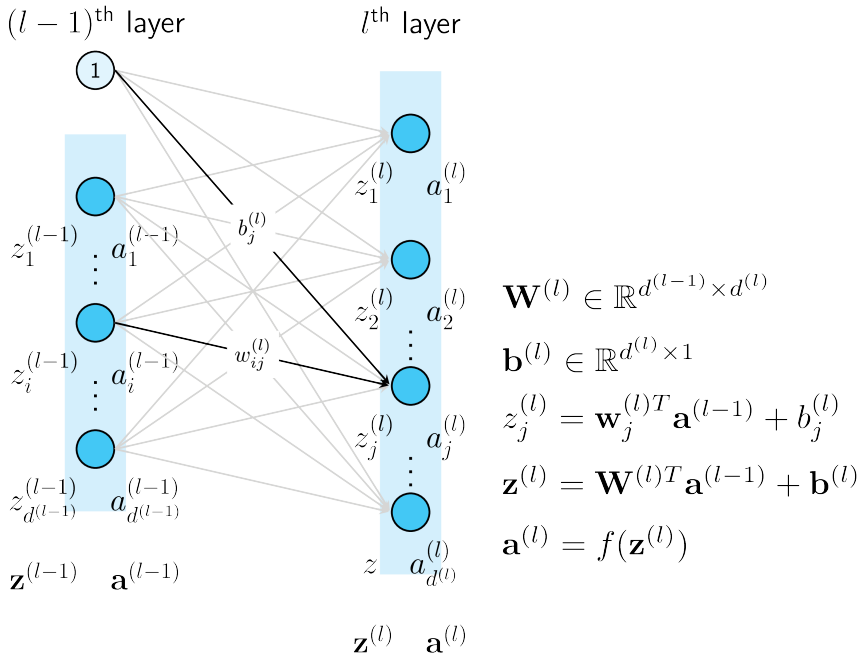
- **Lan truyền tiến** (*feedforward)*
  Ký hiệu:
  - $l^{(i)}$: Số node trong hidden layer thứ i.
  - $W^{(k)}$: Kích thước $l^{(k-1)} * l^{(k)}$ là ma trận hệ số giữa layer $k-1$ và layer k, trong đó $w_{ij}^{(k)}$ là hệ số kết nối từ node thứ i của layer $k-1$ đến node thứ j của layer k.
  - $b^{(k)}$: Kích thước $l^{(k)} * 1$ là hệ số bias của các node trong layer k, trong đó $b_i^{(k)}$ là bias của node thứ i trong layer k.
  - $a^{(l)}$: Các node mạng của tầng thứ l, trong đó $a_j^{(l)}$ là node mạng thứ j của tầng l.

  Việc lan truyền tiến trong mạng neuron thực hiện từ tầng vào đến tầng ra được thực hiện như sau:

$$z_i^{(l+1)} = \sum_{j=1}^{l^{(l)}} w_{ij}^{(l+1)} a_j^{(l)} + b_i^{(l+1)}$$
$$a_i^{(l+1)} = f(z_i^{(l+1)}).$$

  Với f là hàm kích hoạt, ta cũng có thể viết lại công thức trên như sau:

$$z^{(l+1)} = W^{(l+1)}.a^{(l)} + b^{(l+1)}$$
$$a^{(l+1)} = f(z^{(l+1)}).$$



Hình 2.2: Mô phỏng quá trình lan truyền tiến [2].

- **Lan truyền ngược** (*backpropagation*)
  Cũng tương tự như các bài toán học máy khác thì quá trình học vẫn là một mô hình tối thiểu hàm lỗi để có được các trọng số hợp lý nhất. Chúng ta cần tìm ra các trọng số của ma trận W và bias b. Hàm mất mát tuỳ vào bài toán mà lựa chọn, trong các bài toán phân lớp thì hàm mất mát thường là hàm *cross entropy*. Để tối ưu hàm mất mát, ta sử dụng các phương pháp như gradient descent hoặc biến thể mạnh mẽ của nó gần đây như: Adam [14], AMSGrad ...
  Sau khi lan truyền tiến thì cần tính đạo hàm hàm lỗi để lan truyền ngược lại và cập nhật lại các tham số của mô hình bằng các phương pháp tối ưu đã kể trên.

  Gọi J là hàm mất mát của mô hình. Quá trình lan truyền ngược được thực hiện như sau: Tính đạo hàm theo z ngược lại từ $l = (L-1) \to 2$ theo công thức:

$$\begin{aligned}\frac{\partial J}{\partial z^{(l)}} &= \frac{\partial J}{\partial z^{(l+1)}} \frac{\partial z^{(l+1)}}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \\ &= \left(\left(W^{(l+1)}\right)^T \frac{\partial J}{\partial z^{(l+1)}} \right) \frac{\partial a^{(l)}}{\partial z^{(l)}}.\end{aligned}$$

[2] Nguồn ảnh: `https://github.com/tiepvupsu/ebookMLCB`

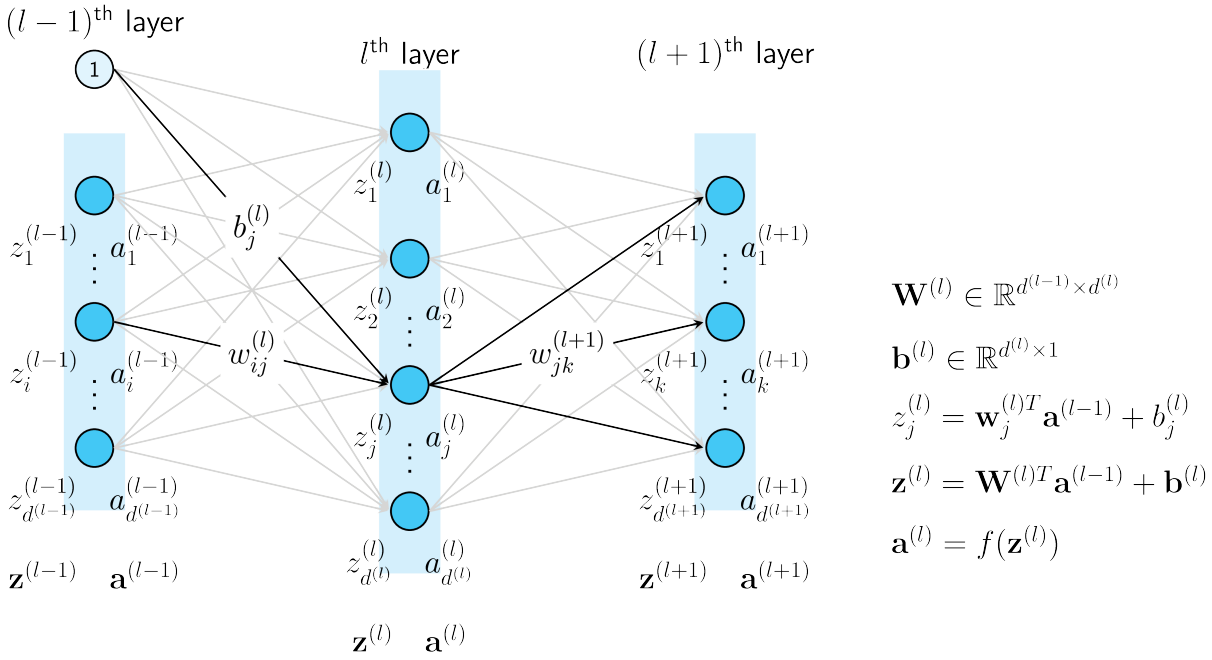
Tính đạo hàm hàm mất mát theo các tham số:

Tính đạo hàm theo tham số W bằng công thức:

$$\begin{aligned}\frac{\partial J}{\partial W^{(l)}} &= \frac{\partial J}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial W^{(l)}} \\ &= \frac{\partial J}{\partial z^{(l)}}\left(a^{(l-1)}\right)^T.\end{aligned}$$

Tính toán đạo hàm theo tham số b theo công thức:

$$\begin{aligned}\frac{\partial J}{\partial b^{(l)}} &= \frac{\partial J}{\partial z^{(l)}}\frac{\partial z^{(l)}}{\partial b^{(l)}} \\ &= \frac{\partial J}{\partial z^{(l)}}.\end{aligned}$$



Hình 2.3: Mô phỏng cách tính lan truyền ngược [3].

[3]Nguồn ảnh: `https://github.com/tiepvupsu/ebookMLCB`

2.2 Mạng neuron hồi tiếp và ứng dụng

Mạng neuron hồi tiếp (**Recurrent Neural Network - RNN**) được áp dụng rất rộng rãi trong các bài toán trong lĩnh xử lý ngôn ngữ tự nhiên bởi hai yếu tố chính. Thứ nhất, RNN mô hình hóa được bản chất của dữ liệu trong ngôn ngữ tự nhiên. Thứ hai, khi năng lực tính toán của máy tính ngày càng mạnh mẽ giúp hiện thực hóa việc huấn luyện mạng neuron hồi tiếp. Trong phần này, chúng ta sẽ tiếp cận các khái niệm cũng như ứng dụng của RNN trong các bài toán thực tế.

2.2.1 Mạng neuron hồi tiếp

Trong các mạng neuron truyền thống tất cả các đầu vào và cả đầu ra là độc lập với nhau, tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán. Trên thực tế, rất nhiều dữ liệu cũng như sự vật trong cuộc sống có mối tương quan chứ không hề độc lập với nhau. Chẳng hạn, một video bao gồm nhiều đoạn nối tiếp nhau, âm nhạc cũng là một dạng chuỗi tuần tự âm thanh, các văn bản xử lý cũng là dạng chuỗi tuần tự... Câu hỏi đặt ra là vậy làm thế nào chúng ta có thể làm cho mô hình của chúng ta có khả năng xử lý chuỗi theo như cách làm của con người.

Ta định nghĩa $x_{i:j}$ là một dãy các vec-tơ $x_i, ..., x_j$. Tổng quát RNN là một hàm nhận đầu vào là một dãy thứ tự có độ dài n các vec-tơ có chiều d_{in} $x_{1:n} = x_1, x_2, ..., x_n (x_i \in \mathbb{R}^{d_{in}})$ và đầu ra là một vec-tơ y_n có số chiều là d_{out} $(y_n \in \mathbb{R}^{d_{out}})$.

$$y_n = \text{RNN}(x_{1:n}) \tag{2.1}$$

$$x_i \in \mathbb{R}^{d_{in}}, y_n \in \mathbb{R}^{d_{out}}.$$

Định nghĩa y_i là vec-tơ đầu ra cho mỗi chuỗi $x_{1:i}$ của chuỗi $x_{1:n}$. Kí hiệu hàm RNN^* là hàm trả ra chuỗi này:

$$y_{1:n} = \text{RNN}^*(x_{1:n}) \tag{2.2}$$

$$y_i = \text{RNN}(x_{1:i})$$

$$x_i \in \mathbb{R}^{d_{in}}, y_n \in \mathbb{R}^{d_{out}}.$$

Đầu ra vec-tơ y_n sau đó được sử dụng để dự báo. Chẳng hạn, mô hình dự đoán xác suất có điều kiện của một sự kiện e với dãy $x_{1:n}$ có thể được định nghĩa là $p(e = j|x_{1:n}) = \text{softmax}(\text{RNN}(x_{1:n}.W + b)_{[j]})$ là phần tử thứ j trong vec-tơ đầu ra qua toán tử **softmax** và phép biến đổi tuyến tính.

Xem xét một cách chi tiết, RNN được định nghĩa đệ quy bởi hàm R với đầu vào là vec-tơ trạng thái s_{i-1}, vec-tơ đầu vào x_i và trả ra vec-tơ trạng thái mới là s_i . vec-tơ trạng thái mới đưa ra một vec-tơ đầu ra y_i bằng một hàm xác định $O(.)$ Thông thường, vec-tơ trạng thái ban đầu s_0 được bỏ qua hoặc là giả sử $s_0 = 0$.

RNN được định nghĩa một cách đệ quy bằng hàm đệ quy R nhận đầu vào là trạng thái trước và vec-tơ đầu vào hiện tại.

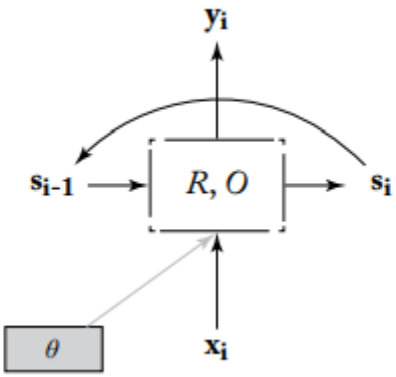
$$\text{RNN}^*(x_{1:n}; s_0) = y_{1:n} \tag{2.3}$$

$$y_i = O(s_i)$$

$$s_i = R(s_{i-1}, x_i)$$

$$x_i \in \mathbb{R}^{d_{in}}, y_n \in \mathbb{R}^{d_{out}}, s_i \in \mathbb{R}^{f(d_{out})}.$$

Các hàm R và O thực hiện một cách trình tự, nhưng RNN giữ theo dõi các trạng thái tính toán thông qua vectơ trạng thái được lưu giữ và chuyển qua lời gọi của hàm R.



Hình 2.4: Hình ảnh đệ quy của một mạng RNN[4].

Chẳng hạn, chúng ta triển khai công thức đệ quy 2.3 với $i = 4$, ta có:

$$s_4 = R(s_3, x_4) = R(R(s_2, x_3), x_4) = R(R(R(s_1, x_2), x_3), x_4)$$

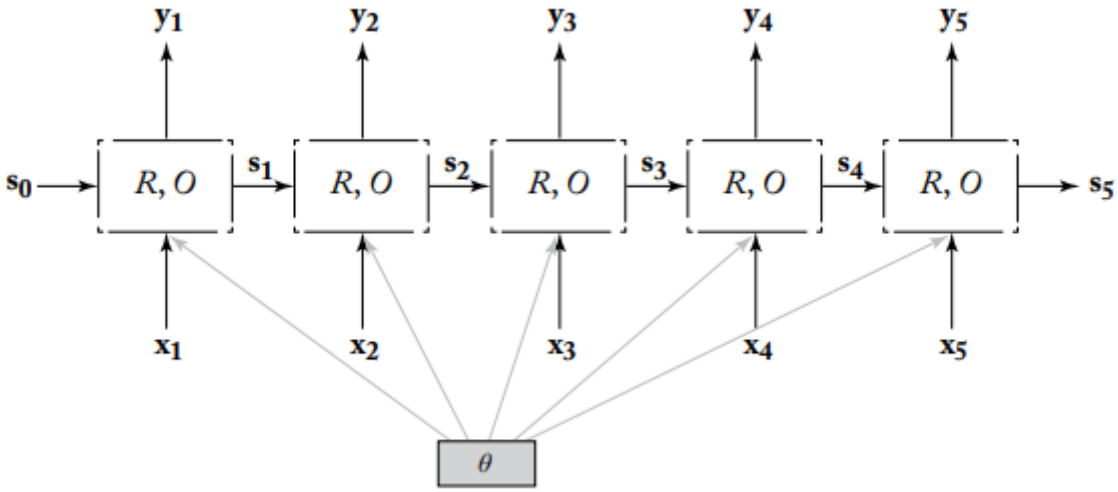
$$= R(R(R(R(s_0, x_1), x_2), x_3), x_4).$$

Vì vậy, s_n và y_n được coi là các mã hóa của toàn bộ chuỗi đầu vào.

Mạng neuron hồi tiếp được tạo ra với mục đích nắm bắt được thông tin dạng chuỗi, ví dụ một câu là một chuỗi gồm nhiều từ. RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó.

Hình vẽ 2.4 biểu diễn một mạng RNN theo định nghĩa đệ quy với chuỗi đầu vào tùy ý. Tuy nhiên, đối với một chuỗi đầu vào có kích thước hữu hạn có thể giải nén đệ quy và được cấu trúc mạng RNN hình vẽ 2.5.

[4]Nguồn ảnh: [1]
[5]Nguồn ảnh: [1]



Hình 2.5: Hình ảnh duỗi thẳng của một mạng RNN[5].

Simple Recurrent Network (S-RNN) được giới thiệu lần đầu bởi *Jeff Elman* năm 1990 và được sử dụng mô hình hóa ngôn ngữ bởi *Mikolov* năm 2010. Mô hình S-RNN được biểu diễn như sau:

$$s_i = R_{\text{SRNN}}(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^s + b) \tag{2.4}$$

$$y_i = O_{\text{SRNN}}(s_i) = s_i$$

$$s_i, y_i \in \mathbb{R}^{d_s}, x_i \in \mathbb{R}^{d_x}, W^x \in \mathbb{R}^{d_x \times d_s}, W^s \in \mathbb{R}^{d_s \times d_s}, b \in \mathbb{R}^{d_s}.$$

Trạng thái s_{i-1} và đầu vào x_i được biến đổi tuyến tính, kết quả (đã được cộng thêm bias) và sau đó được chuyển qua một hàm kích hoạt phi tuyến g (thông thường là hàm tanh hoặc hàm ReLU). Đầu ra tại vị trí i chính bằng trạng thái ẩn tại vị trí đó. S-RNN là một kiến trúc khá mạnh nhưng có nhược điểm là việc S-RNN không hiệu quả trong việc xử lý các phụ thuộc dài.

2.2.2 Huấn luyện mạng neuron hồi tiếp

Huấn luyện RNN tương tự như huấn luyện neuron truyền thống. Chúng ta cũng sử dụng đến thuật toán backpropagation (lan truyền ngược) nhưng có một chút tinh chỉnh. Gradient tại mỗi output không chỉ phụ thuộc vào kết quả tính toán của bước hiện tại mà còn phụ thuộc vào kết quả tính toán của các bước trước đó.

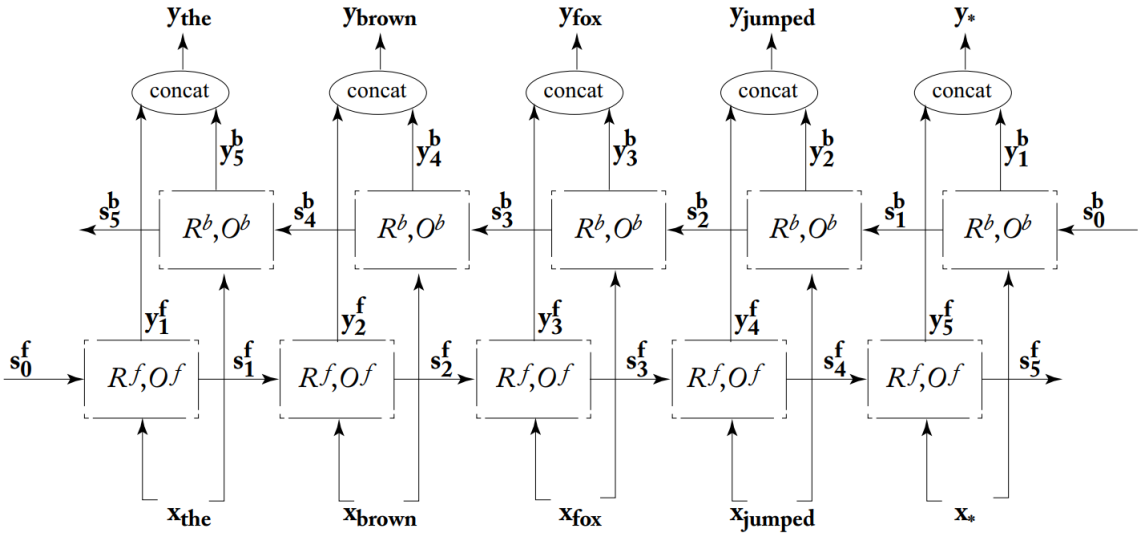
Ví dụ, để tính gradient tại thời điểm $t = 4$, ta cần backpropagation 3 bước trước đó và cộng dồn các gradient này lại với nhau. Kĩ thuật này gọi là Backpropagation Through Time (BPTT). Điểm hạn chế ở đây đó là lớp ẩn không có trí nhớ dài hạn. Vấn đề này còn gọi là vanishing/exploding gradient và kiến trúc mạng LSTM được sinh ra để giải quyết vấn đề này.

2.2.3 Các phiên bản mở rộng của RNN

1. **Mô hình RNN 2 chiều (Bidirectional RNN)**
   Một trong những phiên bản cái tiến của mạng RNN cổ điển là mô hình RNN hai chiều (biRNN). Giả sử ta có một chuỗi tuần tự $x_1, x_2, ..., x_n$, mạng RNN tính toán từ thứ i là x_i dựa trên các từ trước nó là $x_{1:i}$ và bao gồm chính nó. Tuy nhiên các từ $x_{i+2:n}$ cũng rất hữu ích cho việc dự đoán, như vậy có thể đưa ra một cách tiếp cận sử dụng một cửa số trượt kích thước k trung tâm là từ cần dự đoán. Mô hình biRNN cố định kích thước của cửa sổ trượt và cho phép dựa vào cả thành phần trước và thành phần sau trong chuỗi.

   Giả sử có chuỗi đầu vào $x_{1:n}$. biRNN hoạt động bằng cách duy trì hai trạng thái riêng biệt s_i^f và s_i^b tại mỗi vị trí đầu vào thứ i. Trạng thái tiến s_i^f dựa trên $x_1, x_2, ..., x_i$, trong khi trạng thái lùi s_i^b dựa trên $x_n, x_{n-1}, ..., x_i$. Trạng thái tiến và trạng thái lùi dựa trên hai mạng RNN khác nhau. Mạng RNN đầu tiên (R^f, O^f) được cấp cho chuỗi đầu vào nguyên trạng $x_{1:n}$, trong khi mạng RNN thứ hai (R^b, O^b) cung cấp cho chuỗi đầu vào theo trình tự ngược lại. Trạng thái s_i được tính toán dựa trên cả trạng thái tiến và trạng thái lùi. Đầu ra vị trí thứ i là sự kết nối của hai vec-tơ đầu ra $y_i = [y_i^f; y_i^b] = [O^f(s_i^f); O^b(s_i^b)]$, tính cả quá khứ lẫn tương lai. Nói cách khác, y_i, mã hóa biRNN của từ thứ i trong trình tự là sự kết hợp của hai RNN, một đọc trình tự từ đầu và một đọc nó từ cuối. Định nghĩa biRNN$(x_{1:n}, i)$ là vec-tơ đầu ra tương ứng với vị trí thứ i:

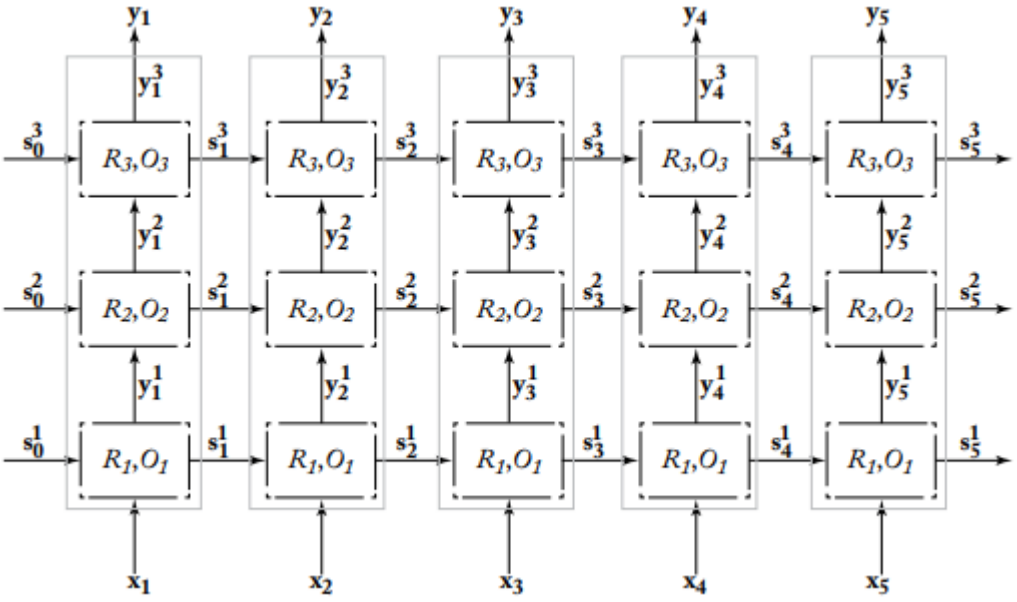
$$\text{biRNN}(x_{1:n}, i) = y_i = [\text{RNN}^f(x_{1:i}), \text{RNN}^b(x_{n:i})]$$



Hình 2.6: Mô phỏng mạng RNN hai chiều.[6].

[6]Nguồn ảnh: [1]

2. **Mô hình RNN sâu (Deep Bidirectional RNN)**
   Các RNN có thể được xếp chồng lên nhau thành từng lớp, tạo thành lưới. Giả sử có k RNNs là RNN_1, RNN_2,...,RNN_k, trong đó RNN thứ j có các trạng thái $s^j_{1:n}$ và các đầu ra $y^j_{1:n}$. Đầu vào của RNN đầu tiên là $x_{1:n}$, trong đó đầu vào của RNN thứ $j(j \geq 2)$ là đầu ra của RNN bên dưới nó, $y^{j-1}_{1:n}$. Đầu ra của toàn bộ hệ thống là đầu ra của RNN cuối cùng, $y^k_{1:n}$. Một kiến trúc nhiều lớp như vậy được gọi là mạng RNN sâu. Một số các bài báo đã cho kết quả các RNN sâu hoạt động tốt hơn các RNN nông hơn trên các bài toán về dịch máy.



Hình 2.7: Kiến trúc RNN sâu 3 tầng[7].

3. **Mạng LSTM (Long Short Term Memory)**
   LSTM là phiên bản mở rộng của RNN, được đề xuất vào năm 1997 bởi Sepp Horcheiter. LSTM được thiết kế để giải quyết các bài toán về phụ thuộc xa trong mạng RNN do bị ảnh hưởng của gradient biến mất. Hiểu một cách đơn giản, mạng RNN truyền thống trong thực tế không có khả năng ghi nhớ thông tin từ các bước khoảng cách xa và do đó phần tử đầu tiên trong chuỗi đầu vào không có nhiều ảnh đến kết quả tính toán dự đoán phần tử trong chuỗi đầu ra trong các bước sau. Mạng LSTM là mô hình tương đối phức tạp, chi tiết mô hình mạng này sẽ được làm rõ trong phần tiếp theo.

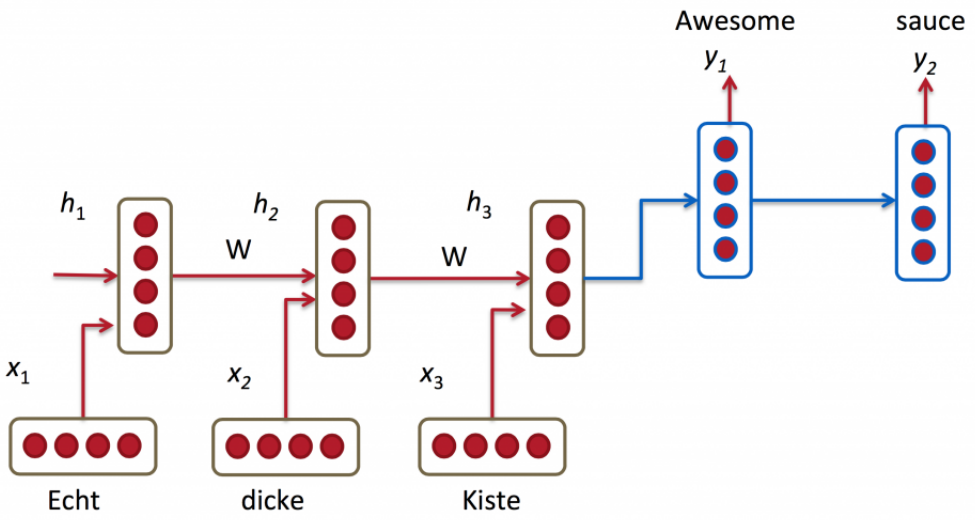
[7]Nguồn ảnh: [1]

2.2.4 Các ứng dụng của RNN

Các mô hình RNN hầu như được sử dụng trong lĩnh vực xử lí ngôn ngữ tự nhiên và ghi nhận tiếng nói. Một số các ứng dụng nổi bật:

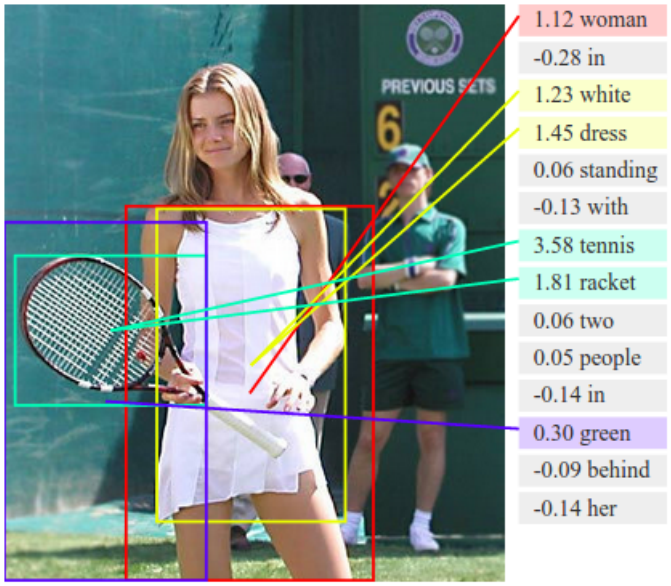
- **Mô hình hóa ngôn ngữ và sinh văn bản (Generating Text)**
  Mô hình ngôn ngữ cho phép ta dự đoán được xác xuất của một từ nào đó xuất hiện sau một chuỗi các từ đi liền trước nó. Do có khả năng ước lượng được độ tương tự của các câu nên nó còn được ứng dụng cho việc dịch máy. Một điểm lý thú của việc có thể dự đoán được từ tiếp theo là ta có thể xây dựng được một mô hình tự sinh từ cho phép máy tính có thể tự tạo ra các văn bản mới từ tập mẫu và xác xuất đầu ra của mỗi từ.

- **Dịch máy (Machine Translation)**
  RNN có thể được sử dụng để dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác. Hầu hết tất cả các hệ thống dịch đang được sử dụng ngày nay đều sử dụng một số phiên bản nâng cao của RNN. Đầu vào có thể là ngôn ngữ nguồn và đầu ra sẽ bằng ngôn ngữ đích mà người dùng muốn.



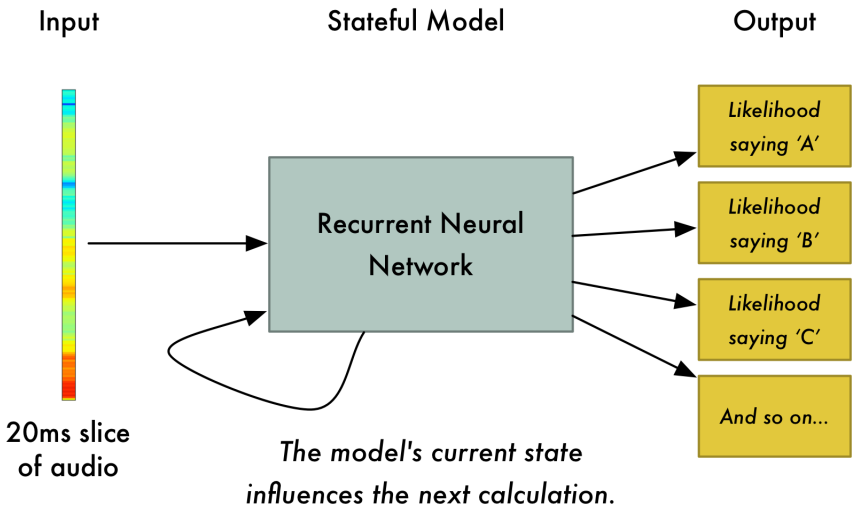
Hình 2.8: Ứng dụng RNN trong dịch máy[8].

- **Mô tả hình ảnh (Generating Image Descriptions)**
  Sự kết hợp giữa CNN (mạng neuron tích chập) và RNN được sử dụng để cung cấp mô tả chính xác những gì đang xảy ra bên trong một hình ảnh. CNN thực hiện phần phân đoạn và RNN sau đó sử dụng dữ liệu được phân đoạn để tạo mô tả.



Hình 2.9: Ứng dụng RNN trong mô tả ảnh [9].

- **Nhận dạng tiếng nói (Speech Recognition)**
  Đưa vào một chuỗi các tín hiệu âm thanh, ta có thể dự đoán được chuỗi các đoạn ngữ âm đi kèm với xác xuất của chúng.



Hình 2.10: Ứng dụng RNN trong nhận dạng tiếng nói [10].

[8]Nguồn ảnh: `http://cs224d.stanford.edu`
[9]Nguồn ảnh: `http://cs.stanford.edu`
[10]Nguồn ảnh: `http://cs.stanford.edu`

2.3 Mạng Long Short Term Memory

2.3.1 Vấn đề phụ thuộc dài

Ý tưởng ban đầu của RNN là kết nối những thông tin trước đó nhằm hỗ trợ cho các xử lý hiện tại. Nhưng đôi khi, chỉ cần dựa vào một số thông tin gần nhất để thực hiện tác vụ hiện tại. Ví dụ, trong mô hình hóa ngôn ngữ, chúng ta cố gắng dự đoán từ tiếp theo dựa vào các từ trước đó. Nếu chúng ta dự đoán từ cuối cùng trong câu "đám_mây bay trên bầu_trời", thì chúng ta không cần truy tìm quá nhiều từ trước đó, ta có thể đoán ngay từ tiếp theo sẽ là "bầu_trời". Trong trường hợp này, khoảng cách tới thông tin liên quan được rút ngắn lại, nạng RNN có thể học và sử dụng các thông tin quá khứ.

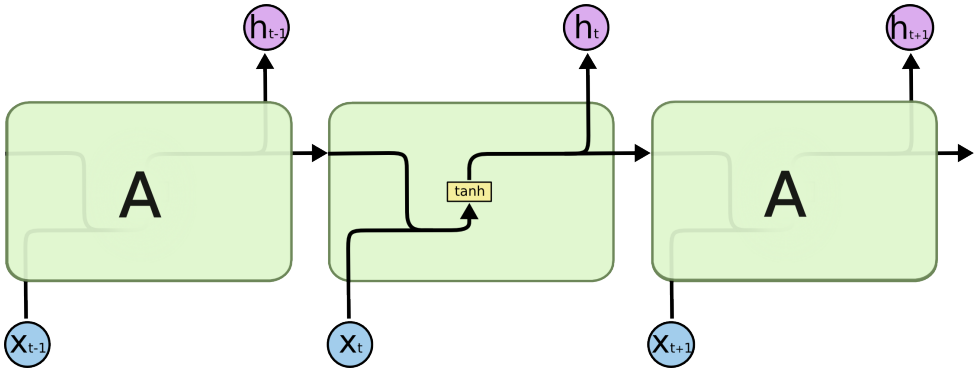
Nhưng cũng có trường hợp chúng ta cần nhiều thông tin hơn, nghĩa là phụ thuộc vào ngữ cảnh. Ví dụ nhưng khi dự đoán từ cuối cùng trong đoạn văn bản "Tôi sinh ra và lớn lên ở Việt_Nam ... Tôi có_thể nói thuần_thục Tiếng_Việt." Từ thông tin gần nhất cho thấy rằng từ tiếp theo là tên một ngôn ngữ, nhưng khi chúng ta muốn biết cụ thể ngôn ngữ nào, thì cần quay về quá khứ xa hơn, để tìm được ngữ cảnh Việt_Nam. Và như vậy, RRN có thể phải tìm những thông tin có liên quan và số lượng các điểm đó trở nên rất lớn. Không được như mong đợi, RNN không thể học để kết nối các thông tin lại với nhau.

Gradient biến mất (Vanishing Gradient) và gradient bùng nổ (Exploding Gradient) là những vấn đề gặp phải khi dựa trên kỹ thuật tối ưu hóa trọng số dựa trên gradient để huấn luyện mạng neuron. Các vấn đề này thường gặp phải do việc lựa chọn các hàm kích hoạt không hợp lý hoặc số lượng các lớp ẩn của mạng quá lớn. Đặc biệt các vấn đề này thường xuất hiện trong quá trình huấn luyện mạng neuron hồi tiếp. Trong thuật toán BPTT, khi chúng ta càng quay lui về các bước thời gian trước đó thì các giá trị gradient càng giảm dần, điều này làm giảm tốc độ hội tụ của các trọng số do sự thay đổi hầu như rất nhỏ. Trong một số trường hợp khác, các gradient có giá trị rất lớn khiến cho quá trình cập nhật các tọng số bị phân kỳ và vấn đề này được gọi là gradient bùng nổ. Có nhiều nghiên cứu đề xuất các giải pháp giải quyết vấn đề này như lựa chọn hàm kích hoạt hợp lý, thiết lập các kích thước cho mạng hợp lý hoặc khởi tạo các trọng số phù hợp khi huấn luyện.

2.3.2 Kiến trúc của mạng LSTM

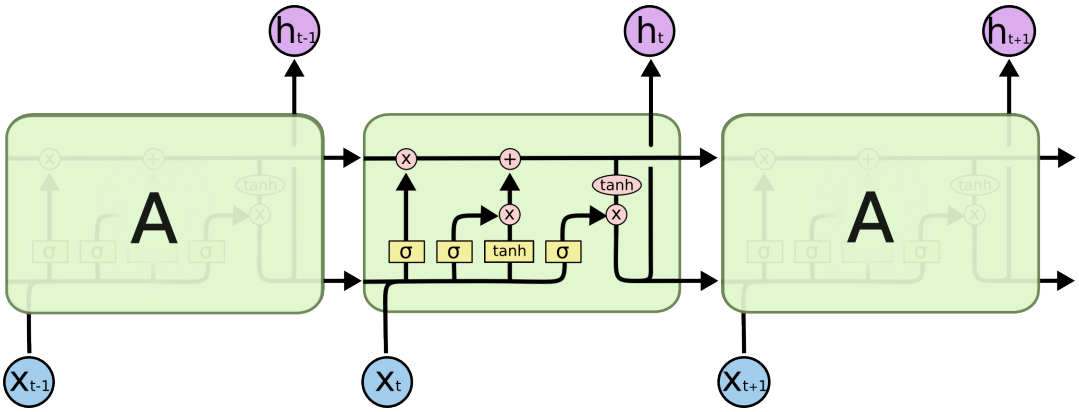
Long Short Term Memory Network (LSTM) là phiên bản mở rộng của RNN, được đề xuất vào năm 1997 bởi Hochreiter và Schmidhuber. LSTM được thiết kế để giải quyết các bài toán về phụ thuộc dài trong mạng RNN do bị ảnh hưởng bởi vấn đề gradient biến mất.

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

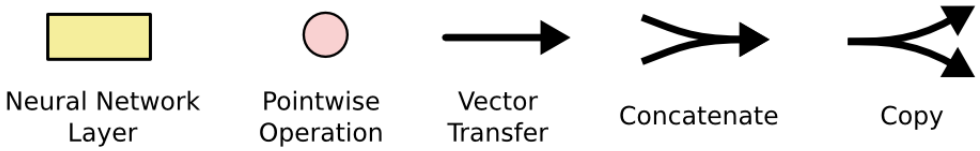


Hình 2.11: Các mô-đun lặp của mạng RNN chứa một layer[11].

LSTM cũng có cấu trúc mắt xích tương tự, nhưng các module lặp có cấu trúc khác hẳn. Thay vì chỉ có một layer neuron network, thì LSTM có tới bốn tầng, tương tác với nhau theo một cấu trúc cụ thể.



Hình 2.12: Các mô-đun lặp của mạng LSTM chứa bốn layer[12].



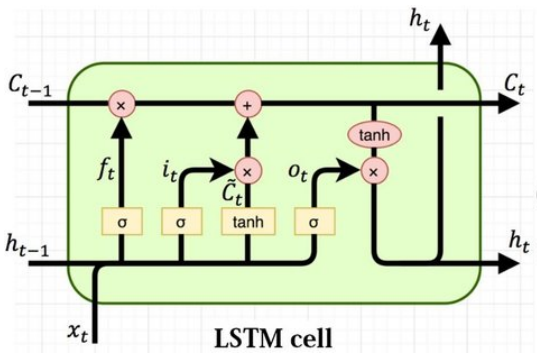
Hình 2.13: Các ký hiệu được sử dụng trong mạng LSTM[13].

Trong đó, các ký hiệu sử dụng trong mạng LSTM được giải nghĩa như hình 2.14 sau đây:

[11]Nguồn ảnh: `https://www.researchgate.net`
[13]Nguồn ảnh: `https://www.researchgate.net`

- Hình nền chữ nhật vàng là các lớp ẩn của mạng neuron.
- Hình tròn nền hồng biểu diễn toán tử Pointwise.
- Đường kẻ gộp lại với nhau biểu thị phép nối các toán hạng.
- Đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác.

Có lẽ sau khi quan sát mô hình thiết kế của LSTM, chúng ta nhận ra ngay, đây là một bảng mạch số, gồm các mạch logic và các phép toán logic trên đó. Thông tin, hay nói khác hơn là tần số của dòng điện di chuyển trong mạch sẽ được lưu trữ, lan truyền theo cách mà chúng ta thiết kế bảng mạch.



$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

Hình 2.14: Kiến trúc của LSTM và phương trình của các cổng[14].

Mạng LSTM bao gồm nhiều tế bào LSTM (LSTM memory cell) liên kết với nhau theo kiến trúc cụ thể. Ý tưởng của LSTM là bổ sung thêm trạng thái bên trong tế bào (cell internal state) C_{t-1} và ba cổng sàng lọc thông tin đầu vào và đầu ra cho tế bào bao gồm cổng quên (forget gate) f_t, cổng vào (input gate) i_t và cổng ra (output gate) o_t. Tại mỗi bước thời gian t, các cổng đều lần lượt nhận giá trị đầu vào x_t (đại diện cho một phần tử trong chuỗi đầu vào) và giá trị h_{t-1} có được từ đầu ra của tế bào từ bước thời gian trước đó $t-1$. Các cổng đều đóng vai trò có nhiệm vụ sàng lọc thông tin với mục đích khác nhau:

- Cổng quên: Có nhiệm vụ loại bỏ những thông tin không cần thiết nhận được khởi cell internal state.
- Cổng vào: Có nhiệm vụ chọn lọc những thông tin cần thiết nào được thêm vào cell internal state.
- Cổng ra: Có nhiệm vụ xác định những thông tin nào từ cell internal state được sử dụng như đầu ra.

[14]Nguồn ảnh: `https://www.researchgate.net`

Trước khi trình bày các phương trình mô tả cơ chế hoạt động của một tế bào LSTM, chúng ta thống nhất các quy ước sau:

- x_t là vec-tơ đầu vào tại mỗi bước thời gian t.
- C_t và $\tilde{C}_t$ lần lượt là vec-tơ đại diện cho tế bào trạng thái và thông tin tiềm năng.
- f_t, i_t, o_t lần lượt chứa các giá trị kích hoạt lần lượt cho các cổng quên, cổng vào và cổng ra.
- h_t là giá trị đầu ra của tế bào LSTM.
- W, U là các ma trận trọng số trong mỗi tế bào LSTM.

Quá trình hoạt động của LSTM được thông qua các bước cơ bản sau:

1. Ở bước đầu tiên, tế bào LSTM quyết định những thông tin nào cần được loại bỏ từ cell internal state ở bước thời gian trước đó C_{t-1}. Giá trị của f_t của cổng quên tại bước thời gian t được tính dựa trên giá trị đầu vào hiện tại x_t, giá trị đầu ra h_{t-1} từ tế bào LSTM trước đó và giá trị bias b_f. Hàm sigmoid biến đổi tất cả các giá trị về miền có giá trị trong khoảng từ 0 (hoàn toàn quên) và 1 (hoàn toàn ghi nhớ). Công thức được viết đơn giản là:
$$f_t = \sigma(x_t U^f + h_{t-1} W^f).$$
2. Ở bước thứ hai, tế bào LSTM quyết định những thông tin nào được thêm vào cell internal state C_t. Bước này gồm hai quá trình tính toán đối với $\tilde{C}_t$ và f_t.
   - Những thông tin tiềm năng cần được thêm vào cell internal sate được tính như sau:
   $$\tilde{C}_t = \tanh(x_t U^g + h_{t-1} W^g).$$
   - Giá trị của i_t của cổng vào cũng được tính như sau:
   $$i_t = \sigma(x_t U^i + h_{t-1} W^i).$$
3. Ở bước thứ ba, giá trị mới của cell internal state C_t được tính dựa trên kết quả thu được từ các bước trước với phép nhân từng phần tử ký hiệu là $*$:
$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t).$$
4. Ở bước cuối cùng giá trị đầu ra h_t của tế bào LSTM được tính toán dựa trên hai phương trình:
$$o_t = x_t U^o + h_{t-1} W^o$$
$$h_t = \tanh(C_t) * o_t.$$

Như vậy, chúng ta đã cơ bản tìm hiểu kiến trúc mạng LSTM- một trong những bước đột phá từ mô hình RNN. Mạng neuron hồi tiếp và mạng LSTM cũng là những kiến thức lý thuyết trọng tâm trong bài báo cáo này.

Chương 3

Mô hình đối thoại với mạng neuron

Chương này sẽ giới thiệu về mô hình ngôn ngữ có thể sinh ra văn bản: mô hình chuỗi tuần tự liên tiếp Sequence-to-Sequence. Chương này cũng giới thiệu kỹ thuật Attention, một bước tiến quan trọng trong mô hình Sequence-to-Sequence và đã chứng tỏ được hiệu quả trong nhiều bài báo. Các kiến thức trong chương này được tham khảo từ [4], [10], [11] và [12].

3.1 Mô hình Sequence-to-Sequence

RNN có thể được sử dụng như là mô hình ngôn ngữ cho việc dự đoán các phần tử của một chuỗi khi cho bởi các phần tử trước đó của một chuỗi. Tuy nhiên, chúng ta vẫn còn thiếu các thành phần cần thiết cho việc xây dựng các mô hình đối thoại, hay các mô hình dịch máy, bởi vì chúng ta chỉ có thể thao tác trên một chuỗi đơn, trong khi việc dịch hoạt động trên cả hai chuỗi – chuỗi đầu vào và chuỗi được dịch sang.

Mô hình **Sequence to Sequence (seq2seq)** được đến xuất bởi Ilya Sutskever et al. [4] vào năm 2014 gắn liền với bài toán dịch máy. Về cơ bản, mô hình này được chia thành hai pha:

- **Pha mã hoá (Encode)**: Trong phần này, dữ liệu đầu vào được mã hoá bằng việc sử dụng các biến thể của mạng neuron hồi tiếp RNN, như mạng LSTM hay mạng bi-RNN. Dữ liệu sẽ được mã hoá thành một vec-tơ có chiều cố định ở cuối mạng neuron. Vec-tơ này sẽ mang thông tin của dữ liệu đầu vào.
- **Pha giải mã (Decode)**: Trong phần này, vec-tơ được mã hoá trong pha mã hoá sẽ được giải mã bằng các biến thể của mạng RNN, vec-tơ mã hoá đóng vai trò như trạng thái ẩn đầu tiên của pha giải mã và mô hình hoạt động như một mô hình sinh ngôn ngữ.

Về mặt toán học mô hình sequence to sequence được mô tả như sau:

- Trước hết, từ một chuỗi nguồn $x_{1:n}$ ta cần sinh ra chuỗi mục tiêu $t_{1:m}$. Thành phần RNN Encoder mã hóa hay tóm tắt chuỗi đầu vào thành vector ngữ cảnh c có độ dài cố định, $c = \text{RNN}^{enc}(x_{1:n})$.

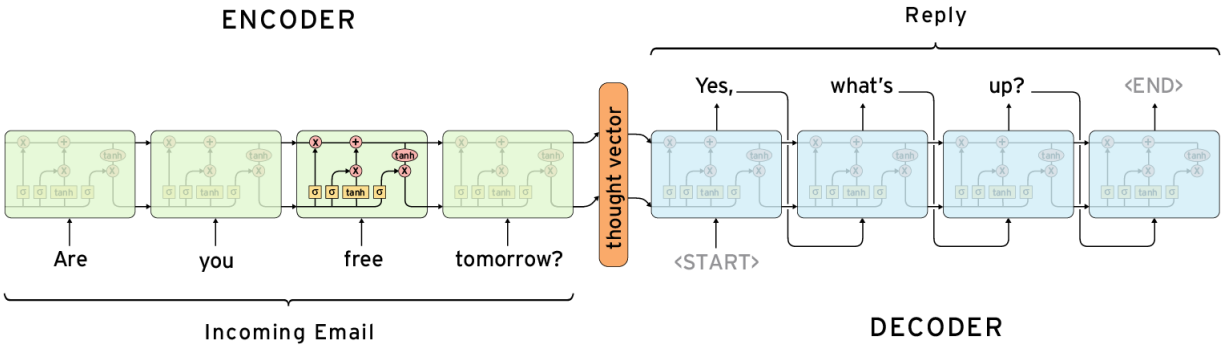
- Sau đó, thành phần RNN Decoder sẽ sinh lần lượt từng từ cho chuỗi đầu ra theo mô hình sinh có điều kiện (conditional generation):

$$p(t_{j+1} = k|\hat{t}_{1:j}, c) = f(O(s_{j+1}))$$

$$s_{j+1} = R(s_j, [\hat{t}_j; c])$$

$$\hat{t}_j \sim p(t_j|\hat{t}_{1:j-1}, c).$$

  Trong đó t_j là bước thời gian thứ j trong chuỗi đầu ra, $\hat{t}_j$ là giá trị dự đoán tại bước thứ j và s_j là các trạng thái ẩn.



Hình 3.1: Seq2seq sử dụng mạng LSTM cho bài toán sinh hội thoại[1].

[1]Nguồn ảnh: `https://medium.com/botsupply/`

3.2 Kỹ thuật Attention

Mô hình Sequence-to-Sequence cơ bản có nhược điểm là yêu cầu RNN Decoder sử dụng toàn bộ thông tin mã hóa từ chuỗi đầu vào cho dù chuỗi đó dài hay ngắn. Thứ hai, RNN Encoder cần phải mã hóa chuỗi đầu vào thành một vec-tơ duy nhất và có độ dài cố định. Ràng buộc này không thực sự hiệu quả vì trong thực tế, việc sinh ra từ tại một bước thời gian trong chuỗi đầu ra có khi phụ thuộc nhiều hơn vào một số những thành phần nhất định trong chuỗi đầu vào. Ví dụ, khi dịch một câu từ tiếng nước này sang tiếng nước khác, chúng ta thường quan tâm nhiều hơn đến ngữ cảnh xung quanh từ hiện tại so với các từ khác trong câu. Như vậy, chúng ta cần có một cơ chế để trong bước sinh ra đầu ra, tại mỗi bước ta cho RNN Decoder tập trung vào đầu vào một trong những phần nhất định của đầu vào được encode. Kỹ thuật attention được đưa ra để giải quyết vấn đề đó.

Kỹ thuật attention được đưa ra lần đầu vào năm 2014 bởi Bahdanau et al. [10] trong công trình nghiên cứu về dịch máy. Ở mức trừu tượng, kỹ thuật attention nới lỏng điều kiện rằng toàn bộ chuỗi đầu vào được mã hóa bằng một vector duy nhất. Thay vào đó các từ trong chuỗi đầu vào sẽ được RNN Encoder mã hóa thành một dãy các vector. Sau đó RNN Decoder áp dụng kỹ thuật attention mềm dẻo (soft attention) bằng cách lấy tổng có trọng số của dãy các vector mã hóa. Các trọng số trong mô hình này được tính bằng một mạng neuron truyền thẳng. RNN Encoder, RNN Decoder và các tham số trong kỹ thuật attention được huấn luyện đồng thời từ dữ liệu.

Mô tả toán học của mô hình này như sau:

- Thành phần Encode nhận đầu vào là một dãy các vec-tơ $\mathbf{x} = (x_1, ..., x_{T_x})$ thành một vec-tơ c.

$$h_t = f(x_t, h_{t-1}) \tag{3.1}$$

và

$$c = q(\{h_1, ..., h_{T_x}\}),$$

trong đó $h_t \in \mathbb{R}^n$ là trạng thái ẩn thứ t và c là vec-tơ được từ dãy các trạng thái ẩn. f và q là các hàm phi tuyến.

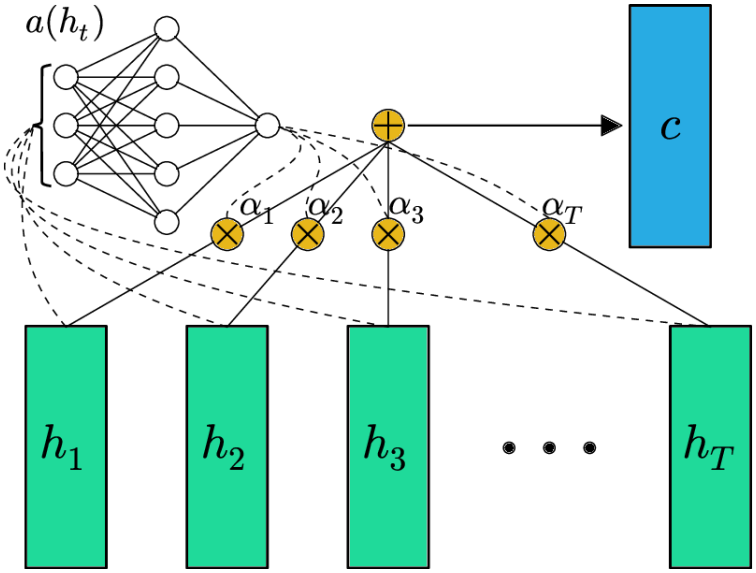
- Thành phần Decode được huấn luyện để dự đoán từ tiếp theo $y_{t'}$ từ vec-tơ ngữ cảnh c và tất cả các từ trước đó $\{y_1, ..., y_{t'-1}\}$. Nói cách khác, thành phần giải mã định nghĩa một xác suất có điều kiện thứ tự:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{3.2}$$

với $\mathbf{y} = (y_1, ..., y_{T_y})$. Với RNN, mỗi xác suất có điều kiện được mô hình hoá như sau

$$p(y_t|\{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c), \tag{3.3}$$

trong đó g là hàm phi tuyến, hàm đưa ra xác suất của y_t và s_t là trạng thái ẩn của RNN.



Hình 3.2: Cơ chế attention lan truyền thẳng[2].

Mô hình toán học của phiên bản attention mềm dẻo, được thể hiện như sau:

- Trong kiến trúc mô hình mới này, chúng ta định nghĩa xác suất có điều kiện trong phương trình 3.2:

$$p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(y_{i-1}, s_i, c_i), \tag{3.4}$$

  trong đó s_i là trạng thái ẩn thứ i, được tính:

$$s_i = f(s_{i-1}, y_{i-1}, c_i).$$

- Vec-tơ ngữ cảnh c_i phụ thuộc vào mỗi chuỗi các $(h_1, ..., h_{T_x})$ trong bộ mã hoá từ đầu vào. Vec-tơ c_i là tổng trọng số của các vec-tơ h_i:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{3.5}$$

  Các trọng số α_{ij} của mỗi h_j được tính toán:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{3.6}$$

[2]Nguồn ảnh: `https://arxiv.org/pdf/1512.08756.pdf`

trong đó

$$e_{ij} = a(s_{i-1}, h_j)$$

là mô hình **alignment model** mô tả sự tương ứng giữa giá trị đầu vào xung quanh vị trí j và đầu ra tại ví trí i.

Trong bài báo Bahdanau et al. [10] các tác giả đề xuất **alignment model** như sau:

$$\text{score}(s_{i-1}, h_j) = a(s_{i-1}, h_j) = v_a^T \tanh(W_a s_{i-1} + U_a h_j),$$

là một mạng neuron với $W_a \in \mathbb{R}^{n \times n}$, $U_a \in \mathbb{R}^n$ và $v_a \in \mathbb{R}^n$ là các ma trận trọng số.

- Pha mã hoá Encode: mạng neuron hồi tiếp hai chiều (biRNN).

  Với mạng RNN thông thường được mô tả trong phương trình 3.1, pha mã hoá Encode sẽ đọc chuỗi đầu vào $\mathbf{x}$ lần lượt theo các từ x_1 cuối cùng đến x_{T_x}. Tuy nhiên, việc tập trung mỗi từ để tóm tắt, không chỉ dựa trên các từ đứng trước mà còn dựa vào cả các từ đứng sau, vì vậy mô hình mạng neuron hồi tiếp hai chiều được đề xuất.

Dựa vào cách tính alignment score, người ta đưa ra các biến thể của cơ chế attention.

- Content-base Attention [11]:

$$\text{score}(s_{i-1}, h_j) = \text{consine}[s_{i-1}, h_j].$$

- Multiplicative Attention hay General Attention [12]:

$$\text{score}(s_{i-1}, h_j) = s_{i-1}^T W_a h_j.$$

- Dot Product [12]:

$$\text{score}(s_{i-1}, h_j) = s_{i-1}^T h_j.$$

Chương 4

Xây dựng mô hình đối thoại tự động

4.1 Kiến trúc của ứng dụng

Với các kiến thức lý thuyết đã được tìm hiểu trong chương 2 và chương 3, tôi để xuất mô hình xây dựng Chatbot như sau:

Thuật toán	Mô hình học sâu, mạng neuron hồi tiếp
Kỹ thuật chính	Mô hình sequence to sequence
Kỹ thuật cải tiến bộ giải mã	Mô hình LSTM, mô hình biRNN, cơ chế attention
Xây dựng ứng dụng web	HTLM, CSS, Javascript, Python Flask
Thuật toán tối ưu	Thuật toán Adam [14]
Hàm mất mát	Hàm Cross Entropy

Bảng 4.1: Kiến trúc của ứng dụng Chatbot.

Quy trình thực hiện bài toán:

- Hiển thị dữ liệu, phân tích dữ liệu và tiền xử lý dữ liệu.
- Cài đặt mô hình.
- Lựa chọn các tham số cho mô hình.
- Huấn luyện mô hình.
- Hiệu chỉnh các tham số.
- Cài đặt chương trình, cài đặt giao diện.

4.2 Tiền xử lý dữ liệu

Để đảm bảo xây dựng được mô hình học sâu có kết quả tốt việc hiểu dữ liệu và các tiền xử lý dữ liệu là một nhiệm vụ rất quan trọng. Trước khi đưa dữ liệu vào mô hình để huấn luyện, chúng ta lần lượt đi qua các bước: Hiểu về dữ liệu, biểu diễn dữ liệu bằng các công cụ trực quan và cuối cùng là bước tiền xử lý dữ liệu.

- **Giới thiệu về dữ liệu**
  - Dữ liệu: Cornell Movie-Dialogs Corpus, các hội thoại ngắn bằng Tiếng Anh trên phim ảnh. Nguồn dữ liệu: `https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html`
  - Tổng quan về dữ liệu: Dữ liệu gồm 220579 cuộc trò chuyện giữa 10292 cặp nhân vật trong phim.
  - 2 file dữ liệu chính: **movie_lines.txt** chứa văn bản thực tế của mỗi câu nói, **movie_conversations.txt** chứa cấu trúc của các cuộc trò chuyện.

```
There.
Where?

You got something on your mind?
I counted on you to help my cause. You and that thug are obviously failing. Aren't we ever going on our date?

You have my word.  As a gentleman
You're sweet.

How do you get your hair to look like that?
Eber's Deep Conditioner every two days. And I never, ever use a blowdryer without the diffuser attachment.

Sure have.
I really, really, really wanna go, but I can't.  Not unless my sister goes.

I really, really, really wanna go, but I can't.  Not unless my sister goes.
I'm workin' on it. But she doesn't seem to be goin' for him.
```

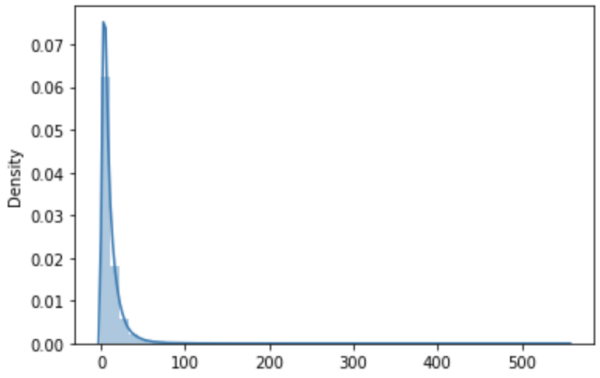
Hình 4.1: Các hội thoại Tiếng Anh ngắn trong phim.

- **Hiển thị dữ liệu bằng các công cụ trực quan**

  Việc giới hạn số lượng từ trong một câu hỏi là một nhiệm vụ quan trọng trước khi mã hoá dữ liệu. Bằng việc hiển thị dữ liệu và sử dụng các công cụ thống kê, chúng ta có thể biết được, các câu hỏi hay các câu trả lời trong dữ liệu với độ dài bao nhiêu là phổ biến nhất.

	Length	Count	Density (%)
4	4	38785	8.750496
5	5	35750	8.065753
1	1	34888	7.871273
3	3	32686	7.374468
6	6	31833	7.182018
2	2	29745	6.710932
7	7	27523	6.209615
8	8	23113	5.214651
9	9	19667	4.437180
10	10	16936	3.821024

(a)



(b)

Hình 4.2: Hiển thị độ dài các câu tập trung nhiều nhất trong bộ dữ liệu.

Kích thước từ điển	6286 từ
Số cặp câu hỏi và câu trả lời	23120 câu
Kích thước từ điển câu hỏi	9611 từ
Kích thước từ điển trả lời	9636 từ

Bảng 4.2: Các số liệu trong dữ liệu.

Các nhận xét đưa ra khi hiển thị dữ liệu và thống kê dữ liệu:

- Số câu trong hội thoại có độ dài là 4 chiếm tỉ trọng cao nhất trong tập dữ liệu.
- Dữ liệu tập trung nhiều ở các câu có số lượng 2 từ đến 6 từ trên một câu. Các câu có số từ là 20 trở lên chiếm tỉ trọng rất nhỏ.
- Như vậy, chúng ta sẽ giới hạn các câu có số từ là 2 đến 5 hoặc từ 2 đến 6 để sử dụng là dữ liệu huấn luyện.

- **Tiền xử lý dữ liệu**

  Bước 1. Đọc dữ liệu là các đoạn hội thoại, đưa ra các dãy id của cuộc trò chuyện.

  Bước 2. Đọc dữ liệu là các dòng thoại, đưa ra tập từ điển bao gồm id của nó và văn bản tương ứng.

  Bước 3. Đưa ra 2 tập là tập các câu hỏi và tập các câu trả lời, question-answer.

  Bước 4. Làm sạch văn bản: Đưa về văn bản chữ thường, thay thế và chuẩn hoá các từ viết tắt.

  Bước 5. Lọc ra các câu hỏi và câu trả lời quá ngắn và quá dài, lấy các câu có độ dài trong đoạn từ 2 đến 5 từ.

  Bước 6. Tạo tập từ điển vocab_dictionary bao gồm mỗi từ và số lượng của nó từ hai tập câu hỏi và câu trả lời. Tương tự tạo tập từ điển với tập câu hỏi và tập câu trả lời.

  Bước 7. Tạo tập từ điển vocab_index gồm các từ xuất hiện nhiều hơn 2 lần từ tập từ điển vocab_dictionary.

  Bước 8. Với dãy<EOS> (kết thúc câu), <PAD> (phần đệm bù), <GO> (bắt đầu> và <UNK> (từ không có trong từ điển> thêm vào tập từ điển vocab_index. Tương tự với tập từ điển của tập câu hỏi và tập câu trả lời.

  Bước 9. Tạo tập từ điển chỉ mục index_vocab từ tập vocab_index.

  Bước 10. Thêm tag <EOS> vào cuối mỗi câu trả lời.

```
def clean_text(text):...

def paired_data(movie_line, movie_convo):...

def data_shorting(max_length, min_length, clean_questions, clean_answers):...

def data_vocabs(shorted_q, shorted_a, threshold):...

def data_int(shorted_q, shorted_a, vocabs_to_index):...

def preparing_data(movie_line, movie_convo, max_length, min_length, threshold):...
```

Hình 4.3: Các hàm tiền xử lý dữ liệu.

4.3 Cài đặt mô hình

Dựa trên việc phân tích bài toán, kiến trúc của mô hình, tôi đề xuất các công cụ sau để xây dựng chương trình.

Công cụ	**Chú thích**	**Pha sử dụng**
Python	Ngôn ngữ lập trình	Ngôn ngữ lập trình
Flask	Thư viện của python	API
TensorFlow	Thư viện học máy của python	CORE
Re	Thư viện tiền xử lý dữ liệu	CORE
Numpy	Thư viện số học của python	CORE
HTML, CSS	Thiết kế giao diện	Giao diện của sản phẩm

Bảng 4.3: Các công cụ xây dựng chương trình.

Tôi chia các file và thư mục quản lý chương trình như sau:

Tên	**Thể loại**	**Chức năng**
Datasets	Folder	Chứa dữ liệu của bài toán
Model_Weight	Folder	Lưu trữ thông tin của mô hình
UI	Folder	Lưu trữ chương trình giao diện
config.py	File python	Cài đặt các tham số cho mô hình
preprocessing_data.py	File python	Tiền xử lý dữ liệu
model.py	File python	Cài đặt mô hình
train.py	File python	Huấn luyện dữ liệu cho mô hình
visualize_data.ipynb	File jupyter notebook	Hiển thị dữ liệu

Bảng 4.4: Kiến trúc file cài đặt của chương trình.

Giao diện là nơi tương tác trực tiếp với người sử dụng, đóng vai trò trung gian giữa người dùng và sever. Giao diện được thiết kế đơn giản bằng HTML5 và CSS3, được đóng gói vào trong thư mục UI.

4.4 Kết quả thực nghiệm

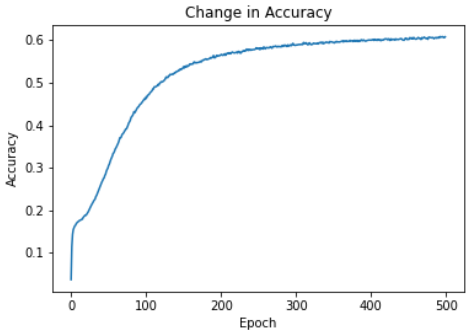
Trong phần kết quả thực nghiệm này, tôi sử dụng sẵn các mô hình đã được huấn luyện trong bài báo [8] với các bộ tham số đầu được cài đặt như chi tiết như sau:

Các bộ tham số	**Thứ nhất**	**Thứ hai**	**Thứ ba**
batch size	128	512	128
embedding size	128	512	128
rnn size	128	512	128
learning rate	0.001	0.001	0.001
epochs	500	100	200
keep_probability	0.75	0.75	0.75
max conversation length	5	5	6
min conversation length	2	2	2

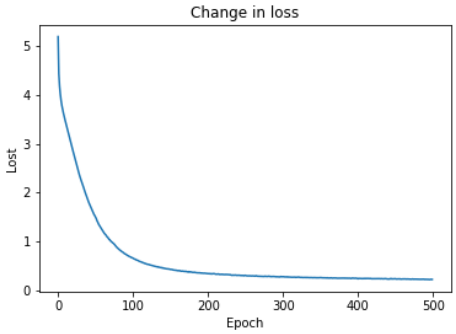
Bảng 4.5: Các tham số đầu vào của mô hình.

Dữ liệu được huấn luyện trên Google Colab GPU/TPU với thời gian của các bộ tham số như sau:

- Với bộ tham số thứ nhất thời gian huấn luyện mô hình là 11.5 giờ.
- Với bộ tham số thứ hai thời gian huấn luyện mô hình là 12.5 giờ.
- Với bộ tham số thứ ba thời gian huấn luyện mô hình là 10.5 giờ.

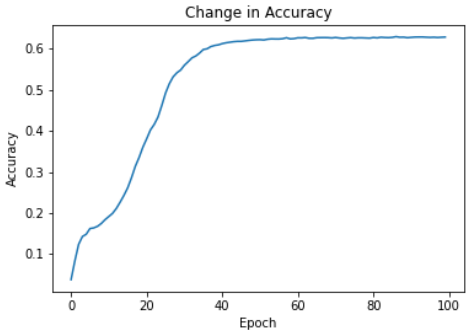


(a)

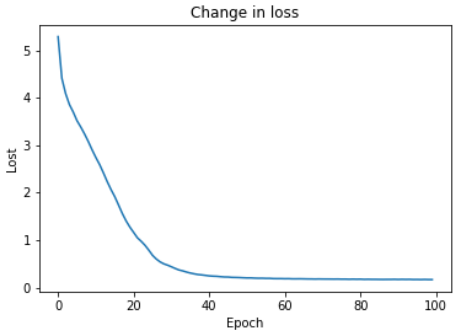


(b)

Hình 4.4: Độ chính xác và sai số của mô hình với bộ tham số thứ nhất.

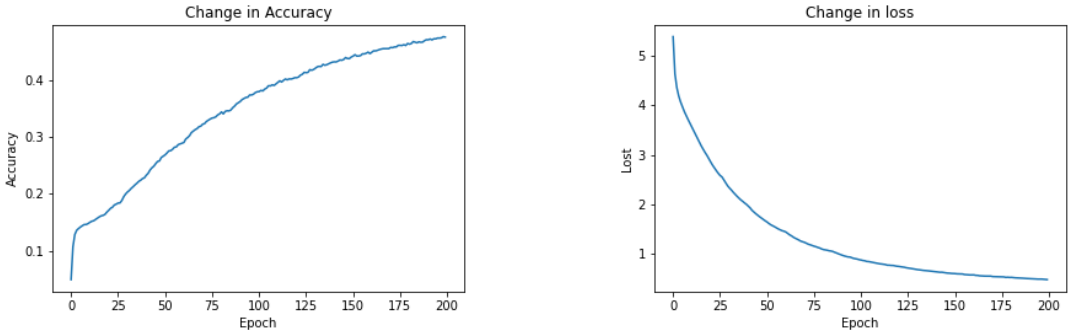


(a)



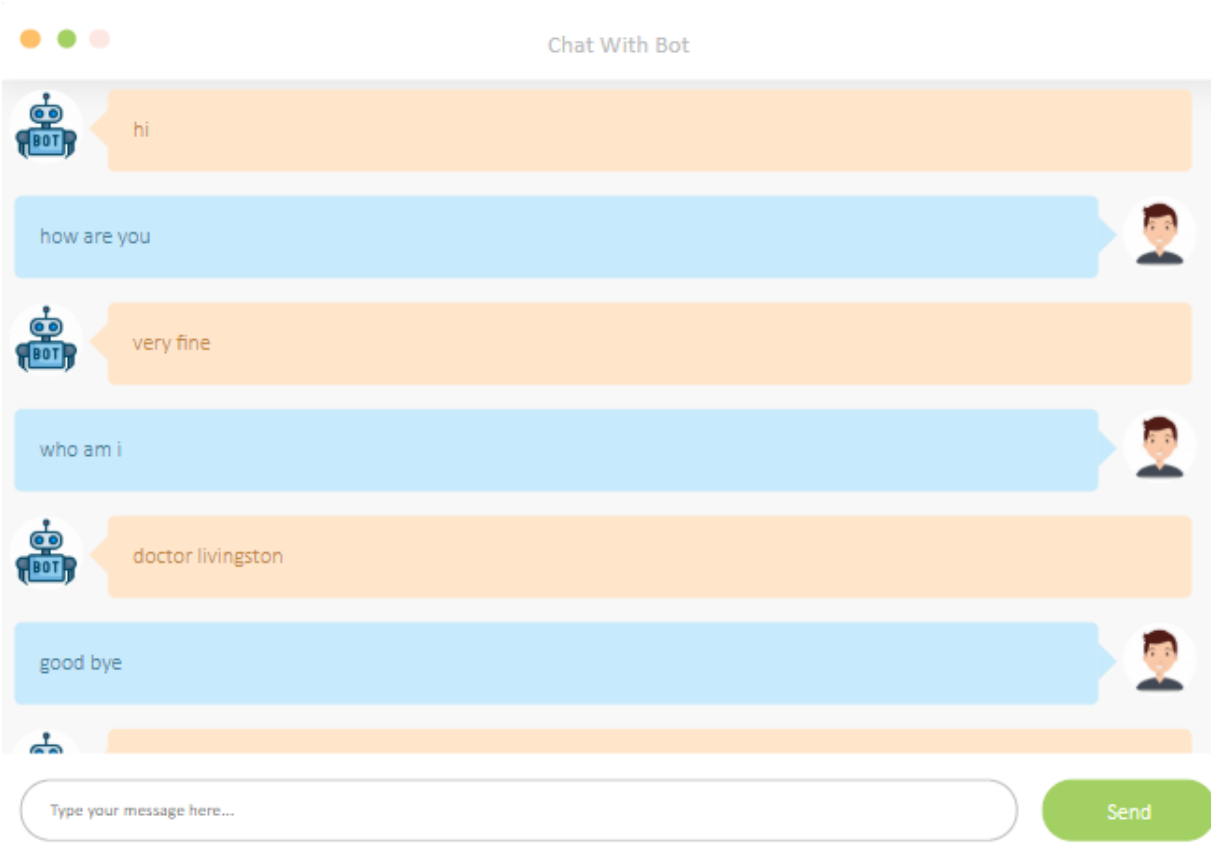
(b)

Hình 4.5: Độ chính xác và sai số của mô hình với bộ tham số thứ hai.



(a) (b)

Hình 4.6: Độ chính xác và sai số của mô hình với bộ tham số thứ ba.



Hình 4.7: Giao diện web người dùng với Chatbot.

Những hạn chế của mô hình:

- Thời gian huấn luyện mô hình lớn.
- Độ chính xác của mô hình còn thấp.
- Hiện tai, bài toán chỉ xử lý với những câu đầu vào ngắn.
- Chatbot không đưa ra các câu trả lời trọng tâm.

Các hướng cải tiến xây dựng mô hình:

- Thử nghiệm với các bộ tham số khác nhau.
- Thay đổi kiến trúc bên trong của mô hình.
- Bổ sung dữ liệu huấn luyện mô hình, chuyển đổi sang bộ dữ liệu Tiếng Việt.
- Sử dụng các cơ chế attention khác trong các bài báo [11], [12].
- Thử nghiệm với các mô hình huấn luyện hiện đại, kết hợp của nhiều mô hình trong bài báo [5] và [6].

Kết luận

Như vậy, bài báo cáo đã giới thiệu một số các khái niệm về hệ thống trả lời tự động, kiến trúc và cách thức hoạt động của các mô hình quan trọng trong học sâu như mô hình mạng neuron, mạng neuron hồi tiếp, mô hình sequence to sequence. Bài báo cáo đã trình bày về lý thuyết mô hình sinh hội thoại và quy trình xây dựng một ứng dụng đơn giản để tạo Chatbot. Cuối cùng chúng ta vẫn có thể thấy những ưu điểm và nhược điểm của mô hình này so với các mô hình cổ điển khác. Có rất nhiều cách cải tiến Chatbot trên như: sử dụng các thuật attention khác nhau, điều chỉnh các tham số trong mô hình phù hợp, cập nhật bổ sung bộ dữ liệu,...Trong quá trình thực hiện, vì thời gian và kiến thức có hạn, nên bài báo cáo chỉ xây dựng được một chương trình có thể kiểm tra và hiểu được lý thuyết, song rất khó ứng dụng vào thực tiễn. Trong các bài báo cáo tiếp theo, em hi vọng sẽ hoàn thiện, xây dựng một chương trình Chatbot thông minh, cũng như các đọc tài liệu, các cách cải tiến để xây dựng một chương trình có tính hoàn thiện và ứng dụng cao hơn trong thực tế. Một lần nữa, em xin cảm ơn thầy giáo đã có những nhận xét sâu sắc để bài báo cáo của em được hoàn thiện.

Tài liệu tham khảo

[1] Yoav Goldberg, *Neural Network Methods for Natural Language Processing*, Morgan & Claypool Publishers, 2017.

[2] Hobson Lane, Hannes Hapke, Cole Howard, *Natural Language Processing in Action*, Manning Publications, 2019.

[3] Sumit Raj, *Building Chatbots with Python-Using Natural Language Processing and Machine Learning*, Apress, 2019.

[4] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, 14 Dec 2014. "Sequence to Sequence Learning with Neural Networks".

[5] Oriol Vinyals, Quoc Le, 22 Jul 2015. "A Neural Conversational Model".

[6] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, Joelle Pineau, 16 Apr 2016. "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models".

[7] Liu Yang et al., 25 Aug 2019. "A Hybrid Retrieval-Generation Neural Conversation Model".

[8] Abonia Sojasingarayar, 2020. "Seq2Seq AI Chatbot with Attention Mechanism".

[9] Vũ Hữu Tiệp, *Machine learning cơ bản*, Nhà xuất bản khoa học và kỹ thuật, 2018.

[10] Dzmitry Bahdanau et al., 1 Sep 2014. "Neural Machine Translation by Jointly Learning to Align and Translate".

[11] Graves, A., Wayne, G. & Danihelka, I, 10 Dec 2014. "Neural Turing machines".

[12] Luong, M.-T., Pham, H., and Manning, C. D. , 20 Sep 2015. "Effective approaches to attentionbased neural machine translation".

[13] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. "Recurrent neural network based language model". In INTERSPEECH, pages 1045–1048, 2010.

[14] Diedelrik P.Kingma, Jimmy Lei Ba, 30 Jan 2017. "Adam: A method for stochastic optimization".