**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài**

XÂY DỰNG HỆ THỐNG TRẢ LỜI TỰ ĐỘNGDỰA TRÊN MẠNG GAN VỚI HỌC TĂNG CƯỜNG

**Sinh viên: Nguyễn Văn Đặng**

**Mã số: B1507224**

**Khóa: K41**

**Cần Thơ, 3/2019**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC CẦN THƠ**

**KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

**LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Đề tài**

XÂY DỰNG HỆ THỐNG TRẢ LỜI TỰ ĐỘNGDỰA TRÊN MẠNG GAN VỚI HỌC TĂNG CƯỜNG

**Người hướng dẫn Sinh viên thực hiện**

**TS Lâm Nhựt Khang Nguyễn Văn Đặng**

**Mã số: B1507224**

**Khóa: K41**

***Cần Thơ, 3/2019***

# LỜI CẢM ƠN

Để hoàn thành luận văn Xây dựng hệ thống trả lời tự động dựa trên mạng GAN và học tăng cường em bày tỏ lòng cảm ơn đến cô TS. Lâm Nhựt Khang đã tận tình hướng dẫn, hỗ trợ em trong suốt quá trình thực hiện đề tài luận văn.

Đồng thời em xin chân thành cám ơn quý Thầy, Cô ở Khoa Công nghệ Thông tin & Truyền thông đã tận tình dạy dỗ và truyền đạt kiến thức, kinh nghiệm quý báu trong những học kỳ vừa qua. Với vốn kiến thức đó, nền tảng kiến thức trong quá trình học tập là cơ sở để em thực hiện luận văn này.

Mặc dù đã cố gắng rất nhiều để thực hiện luận văn, nhưng vẫn không thể tránh khỏi những thiếu sót mà em chưa hoàn thiện được. Rất mong sự góp ý của quý Thầy Cô.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Nguyễn Văn Đặng

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

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

Cần Thơ, ngày tháng năm 2019

Giáo viên hướng dẫn

TS. Lâm Nhựt Khang

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc7274411)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 2](#_Toc7274412)

[MỤC LỤC 3](#_Toc7274413)

[DANH MỤC HÌNH ẢNH 5](#_Toc7274414)

[DANH MỤC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT 6](#_Toc7274415)

[TÓM TẮT 7](#_Toc7274416)

[ABSTRACT 8](#_Toc7274417)

[CHƯƠNG 1: GIỚI THIỆU 9](#_Toc7274418)

[1. Tính cấp thiết và lý do chọn đề tài 9](#_Toc7274419)

[2. Mục tiêu của đề tài 9](#_Toc7274420)

[3. Đối tượng và phạm vi nghiên cứu 9](#_Toc7274421)

[4. Cơ sở nghiên cứu khoa học 9](#_Toc7274422)

[5. Bố cục luận văn 11](#_Toc7274423)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 13](#_Toc7274424)

[1. Chatbot 13](#_Toc7274425)

[1.1. Khái niệm 13](#_Toc7274426)

[1.2. Phân loại 13](#_Toc7274427)

[1.3. Hoạt động 13](#_Toc7274428)

[2. Mạng Nơ-ron nhân tạo 14](#_Toc7274429)

[2.1. Tổng quan về mạng nơ-ron nhân tạo 14](#_Toc7274430)

[2.2. Quá trình xử lý thông tin của một ANN 15](#_Toc7274431)

[2.3. Mạng Nơ-ron hồi quy (RNN) 16](#_Toc7274432)

[2.4. Vấn đề phụ thuộc xa: 17](#_Toc7274433)

[2.5. Mạng LSTM 19](#_Toc7274434)

[2.6. Mạng GRU 23](#_Toc7274435)

[2.7. Mạng hierarchical encoder decoder 25](#_Toc7274436)

[3. Mô hình sequence to sequence 27](#_Toc7274437)

[4. Kỹ thuật Attention 29](#_Toc7274438)

[5. Reinforcement Learning 30](#_Toc7274439)

[5.1. Tổng quát 30](#_Toc7274440)

[5.2. Policy Gradient 32](#_Toc7274441)

[6. Mạng GAN 33](#_Toc7274442)

[6.1. Tổng quát về GAN 33](#_Toc7274443)

[6.2. Quá trình hoạt động của GAN 33](#_Toc7274444)

[6.3. Quá trình huấn luyện 34](#_Toc7274445)

[6.4. Ứng dụng của GAN 36](#_Toc7274446)

[7. Monte Carlo Tree Search 37](#_Toc7274447)

[7.1. Định nghĩa 37](#_Toc7274448)

[7.2. Nguyên tắc hoạt động 37](#_Toc7274449)

[CHƯƠNG 3: PHƯƠNG PHÁP THỰC HIỆN 39](#_Toc7274450)

[1. Tổng quát 39](#_Toc7274451)

[2. Các giai đoạn xử lý 39](#_Toc7274452)

[3. Xây dựng mô hình 40](#_Toc7274453)

[3.1. Giai đoạn huấn luyện 40](#_Toc7274454)

[3.2. Giai đoạn sinh câu trả lời 46](#_Toc7274455)

[CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ 47](#_Toc7274456)

[1. Kết quả thực nghiệm 47](#_Toc7274457)

[2. Đánh giá mô hình 48](#_Toc7274458)

[CHƯƠNG 5: KẾT LUẬN 50](#_Toc7274459)

[1. Kết quả đạt được và hạn chế 50](#_Toc7274460)

[2. Hướng phát triển 50](#_Toc7274461)

[TÀI LIỆU THAM KHẢO 51](#_Toc7274462)

# DANH MỤC HÌNH ẢNH

[**Hình *1*:** Quá trình hoạt động chatbot 14](#_Toc7274502)

[**Hình *2*:** Kiến trúc mạng nơ-ron nhân tạo 14](#_Toc7274503)

[**Hình *3***: Quá trình xử lý thông tin một ANN 15](#_Toc7274504)

[**Hình *4***: Một mạng nơ-ron hồi quy 17](#_Toc7274505)

[**Hình *5*:** Vấn đề phụ thuộc xa trong RNN 18](#_Toc7274506)

[**Hình *6*:** Khoảng cách giữa các RNN lớn dần 18](#_Toc7274507)

[**Hình *7*:** Mô-đun lặp lại trong RNN tiêu chuẩn chứa một lớp 19](#_Toc7274508)

[**Hình *8*:** Mô-đun lặp lại trong LSTM chứa bốn lớp tương tác 20](#_Toc7274509)

[**Hình *9*:** Kí hiệu trong LSTM 20](#_Toc7274510)

[**Hình *10*:** Trạng thái tế bào 21](#_Toc7274511)

[**Hình *11*:** Các cổng trạng thái 21](#_Toc7274512)

[**Hình *12*:** Hàm ft trong LSTM 22](#_Toc7274513)

[**Hình *13*:** Hàm it trong LSTM 22](#_Toc7274514)

[**Hình *14*:** Hàm Ct trong LSTM 23](#_Toc7274515)

[**Hình *15*:** Hàm Ot trong LSTM 23](#_Toc7274516)

[**Hình *16*:** Ví dụ về cách kích hoạt hàm ẩn GRU 24](#_Toc7274517)

[**Hình *17*:** Mạng RNN với chuỗi câu đầu vào 25](#_Toc7274518)

[**Hình *18*:** RNN encoder decoder 25](#_Toc7274519)

[**Hình *19*:** Ví dụ một chuỗi câu liên tục 26](#_Toc7274520)

[**Hình *20*:** Mô hình hierarchical encoder decoder trong tiếng Việt 26](#_Toc7274521)

[**Hình *21*:** Mô hình phát sinh văn bản 27](#_Toc7274522)

[**Hình *22*:** Mô hình seq2seq 28](#_Toc7274523)

[**Hình *23*:** Mô hình đối thoại seq2seq 29](#_Toc7274524)

[**Hình *24*:** Kịch bản của học tăng cường 31](#_Toc7274525)

[**Hình *25*:** Mô hình hoạt động của GAN 33](#_Toc7274526)

[**Hình *26*:** Huấn luyện Discriminator network, cố định Generator network 35](#_Toc7274527)

[**Hình *27*:** Huấn luyện Generator network, cố định Discriminator network 35](#_Toc7274528)

[**Hình *28*:** Dịch văn bản thành hình ảnh 36](#_Toc7274529)

[**Hình *29*:** Tạo ra độ phân giải ảnh cao 36](#_Toc7274530)

[**Hình *30*:** Sinh ra ảnh từ một ảnh mô hình vẽ con mèo 37](#_Toc7274531)

[**Hình 31:** Cây tìm kiếm Monte Carlo 37](#_Toc7274532)

[**Hình *32*:** Các quá trình xử lý 39](#_Toc7274533)

[**Hình *33*:** Xử lý trước khi đưa vào mô hình với Pyvi 42](#_Toc7274534)

[**Hình *34*:** Các bước huấn luyện mô hình 43](#_Toc7274535)

[**Hình *35*:** Seq2seq GRU với kỹ thuật Attention trong tiếng Việt 44](#_Toc7274536)

[**Hình *36*:** Chức năng của D xác định xác suất câu từ G 45](#_Toc7274537)

[**Hình *37*:** Giai đoạn sinh câu trả lời với mô hình Generator 46](#_Toc7274538)

# DANH MỤC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Từ chuẩn** | **Diễn giải** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| ANN | Artificial Neural Network | Mạng nơ-ron nhân tạo |
| CPU | Central Processing Unit | Bộ xử lý trung tâm |
| GAN | Generative adversarial network | Mạng chống đối tạo sinh |
| GPU | Graphics Processing Unit | Bộ xử lý chuyên dụng nhận nhiệm vụ tăng tốc, xử lý đồ họa cho bộ xử lý trung tâm CPU |
| ML | Machine Learning | Máy học |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| NLU | Natural Language Understanding | Hiểu ngôn ngữ tự nhiên |
| seq2seq | Sequence-to-Sequence | Mô hình seq2seq |
| RL | Reinforcement Learning | Học tăng cường |

# TÓM TẮT

Những thành tựu từ trí tuệ nhân tạo, máy học trong đó có xử lý ngôn ngữ tự nhiên, đang thúc đẩy sự phát triển thời đại của công nghệ số - chatbot một chương trình máy tính có thể thực hiện một số tác vụ, từ lên lịch họp, dự báo thời tiết đến hỗ trợ người dùng mua một đôi giày hay tâm sự với con người. Nhận thấy tiềm năng to lớn này, các doanh nghiệp bắt đầu đầu tư mạnh mẽ vào nền kinh tế đang phát triển, “bot”. Trong khi cuộc cách mạng “bot” vẫn còn đang ở trong giai đoạn sơ khởi, nhiều người tin rằng những năm sắp tới sẽ là những năm mà những hội thoại tương tác này bùng nổ.

Xây dựng miền đối thoại mở đang là thách thức trong cuộc cách mạng “bot” đòi hỏi “bot” phải tạo ra phản hồi mạch lạc và có ý nghĩa không chung chung, buồn tẻ, lặp đi lặp lại nhiều lần một phản hồi.

Để tiếp tục phát triển hệ thống trả lời tự động của học kì trước, luận văn này nghiên cứu và xây dựng miền đối thoại mở cho tiếng Việt trên nền mạng GAN và học tăng cường. Ý tưởng của sự thành công trong trong việc tạo ra hình ảnh của mạng GAN với mô hình gồm 2 mô hình nhỏ đó là một mô hình Generator được huấn luyện để sinh ra câu trả lời từ một chuỗi đầu vào tương ứng và một mô hình phân biệt Discriminator có chức năng xem phản hồi nào được tạo ra từ Generator có xác suất gần với phản hồi từ tập dữ liệu thật và gán nhãn cho chúng. Trong quá trình huấn luyện nhãn này được dùng để xem xét gán phần thưởng, cuối quá trình huấn luyện phần thưởng được dùng để cập nhật lại mô hình Generator. Mặc dù phương pháp này còn mới lạ trong xây dựng hội thoại nhưng lại đem kết quả khả quan cho hướng đi mới này.

# ABSTRACT

# CHƯƠNG 1: GIỚI THIỆU

## Tính cấp thiết và lý do chọn đề tài

Nếu như vài năm trước, chatbot vẫn còn là một khái niệm xa vời thì đến nay nó đã phát triển một cách vượt bật trong xã hội công nghệ 4.0 không chỉ ở các nước phát triển mà còn ở các nước đang phát triển trong đó có Việt Nam. Chatbot có nhiều ứng dụng trong thực tiễn như đặt vé máy bay, phòng khách sạn, trò chuyện với con người. Chatbot được mọi người sử dụng ở nhiều nơi và có khả năng tạo ra những phản hồi rành mạch, có ý nghĩa. Các phương pháp tiếp cận trước đây mặc dù đã tạo ra được một hệ thống đối thoại nhưng vẫn còn đâu đó vài bất cập khó khăn như không quan tâm đến ngữ cảnh của câu nói đầu vào cũng như không tạo ra một phản hồi tự nhiên như con người được.

Luận văn này sẽ tìm hiểu và xây dựng một hệ thống đối thoại có thể tạo ra những phản hồi mạch lạc có ý nghĩa trong ngữ cảnh phù hợp một cách tự nhiên như con người đang giao tiếp với nhau dựa trên mạng GAN và học tăng cường.

## Mục tiêu của đề tài

Luận văn này sẽ nghiên cứu xây dựng mô hình đối thoại cho tiếng Việt, dựa trên nền mạng GAN và học tăng cường, để sinh ra câu trả lời từ mỗt chuỗi đầu vào tương ứng.

## Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là mô hình đối thoại được xây dựng trên nền mạng GAN[[1]](#footnote-1) và học tăng cường.

Phạm vi nghiên cứu của đề tài là tập dữ liệu tiếng Việt.

## Cơ sở nghiên cứu khoa học

Chatbot đã không còn là khái niệm xa lạ trong cuộc cách mạng công nghệ 4.0. Các công ty công nghệ hàng đầu trên thế giới như Google, Microsoft, IBM, Apple, Amazon,… đã và đang cạnh tranh vị trí dẫn đầu bằng cách xây dựng nền tảng cho những nhà phát triển khác sử dụng để tạo nên chatbot của riêng họ. Việc nghiên cứu tạo để tạo ra những bước đột phá có ý nghĩa trong thực tiễn cũng như thúc đẩy sự phát triển của công nghệ thông tin trong xử lý ngôn ngữ tự nhiên.

Tạo ra các phản hồi cho đoạn hội thoại đã được Ritter, Alan, Colin Cherry, and William B. Dolan [1] bằng cách sử dụng các kỹ thuật dịch máy thống kê (Statistical Machine Translation - SMT)[[2]](#footnote-2). Tuy nhiên hệ thống này gặp khó khăn trong việc giải quyết thách thức là tạo ra các phản hồi phù hợp với bối cảnh của cuộc trò chuyện. Kế thừa từ hệ thống của Ritter nên Sordoni và các cộng sự của mình [2] đã kết hợp một mô hình mạng thần kinh với ngữ cảnh trước đó trên tập dữ liệu gồm 127 triệu phản hồi từ Twitter FireHose trong khoảng thời gian 3 tháng từ tháng 6 năm 2012 đến tháng 8 năm 2012. Phương pháp tiếp cận được Sordoni và cộng sự giới thiệu hai mô hình tạo phản hồi theo ngữ cảnh sử dụng kiến trúc mô hình ngôn ngữ mạng thần kinh hồi quy (Recurrent Language Model - RLM) [3]. Những mô hình này trước tiên mã hóa thông tin ở trước trong một hidden đại diện liên tục, sau đó được RLM giải mã để tạo ra các phản hồi hợp lý với ngữ cảnh. Mô hình đạt được 0.11 điểm BLEU[[3]](#footnote-3) trong cài đặt Machine Translation (MT) và 0.24 điểm BLEU trong cài đặt Information Retrieval (IR).

Những tiến bộ gần đây nhờ vào sự ra đời mô hình SEQ2SEQ [4] đã xây dựng các hệ thống đối thoại với mô hình SEQ2SEQ. Serban và cộng sự [5] đã xây dựng hệ thống đối thoại sử dụng mạng HRED (Hierarchical Recurrent Encoder-Decoder) sử dụng một số lượng lớn data MovieTriples với tập câu hỏi và câu trả lời tương ứng thu được kết quả với perplexity[[4]](#footnote-4) là 26.81. Yi Luan và cộng sự [6] đã mô hình hội thoại dựa trên LSTM trên tập dữ liệu Ubuntu Dialogue Corpus[[5]](#footnote-5) theo đó mô hình có điểm perplexity là 46.75.

Cùng với RL được áp dụng trong xây dựng hệ thống đối thoại đã được Jiwei Li và cộng sự của mình [7] bằng cách mô phỏng cuộc trò chuyện của hai tác nhân ảo và sử dụng phương thức policy gradient để thưởng cho cho câu đạt ba yêu cầu: có thông tin, mạch lạc và dễ trả lời. Mô hình này sử dụng 10 triệu messages từ OpenSubtitles dataset và 0,8 triệu câu với khả năng tạo ra phản hồi “i don’t know what you are taking about” thấp nhất để đảm bảo đầu vào ban đầu dễ phản hồi. Điểm đa dạng (Diversity scores) cho mô hình RL với unigram là 0.017 và bigram là 0.041 cao hơn mô hình SEQ2SEQ với unigram là 0.0062 và bigram là 0.015.

Ý tưởng của mạng GAN đã đạt được những thành công lớn trong thị giác máy tính. Huấn luyện theo mô hình này được xem như một trò chơi trong đó mô hình Generator được huấn luyến để tạo đầu ra đánh lừa Discriminator; kỹ thuật này đã áp dụng cực kỳ thành công trong việc tạo ra hình ảnh. Tuy nhiên nó đã không đạt được thành công như mong đợi trong xử lý ngôn ngữ tự nhiên. Khó khăn này được hiểu là việc tạo ra câu văn là trừu tượng khiến cho lỗi xuất phát từ Discriminator khó có thể lan truyền ngược lại Generator. Gần đây một số công việc đã bắt đầu giải quyết vấn đề này [8] đề xuất cung cấp cho Discriminator các vector ẩn trung gian thay vì các đầu ra chuỗi của nó. Một chiến lược như vậy làm cho hệ thống trở nên khác biệt và đạt được kết quả đầy hứa hẹn trong các nhiệm vụ như mô hình hóa ngôn ngữ cấp độ ký tự và tạo ra chữ viết tay. Yu và cộng sự [9] đã sử dụng policy gradient của RL để lan truyền ngược đầu ra từ discriminator, sau đó cải thiện generator như tạo thơ, tạo lời nói, tạo ra những bản nhạc,.... Bên cạnh đó Chen [10] đã áp dụng ý tưởng huấn luyện đối thủ trong phân tích tình cảm và Zhang [11] cũng áp dụng ý tưởng này. Công việc gần đây chính thức hóa việc tạo chuỗi đầu ra như là một vấn đề hành động trong RL . Ranzato [12] huấn luyện decoder RNN trong mô hình SEQ2SEQ sử dụng policy gradient và có kết quả rất cạnh tranh. Bahdanau [13] đã có bước tiến bằng cách đào tạo mô hình RL một actor critic cho dịch máy. Shen [14] và Wiseman [15] cũng đã giải quyết các vấn đề exposure bias và đánh giá loss-evaluation trong dịch máy thần kinh.

## Bố cục luận văn

Bố cục luận văn được chia thành các chương như sau:

* Chương 1 - Giới thiệu: trình bày nguyên nhân chọn đề tài, tính cấp thiết, mục tiêu đề tài, cơ sở và phương pháp nghiên cứu.
* Chương 2 - Cơ sở lý thuyết: trình bày các lý thuyết được áp dụng.
* Chương 3 - Phương pháp thực hiện: trình bày các bước xử lý để xây dựng mô hình.
* Chương 4 - Thực nghiệm và đánh giá kết quả: kết quả thực nghiệm và đánh giá kết quả đạt được.
* Chương 5 - Kết luận: tổng kết các việc đã làm được và chưa trong luận văn, cũng như hướng phát triển.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Chương này nêu lên các khái niệm cơ bản về chatbot, mạng nơ-ron nhân tạo, mạng nơ-ron hồi quy, mạng GRU, mạng hierarchical, kỹ thuật attention, mô hình seq2seq, học tăng cường cũng như mạng GAN.

## Chatbot

### Khái niệm

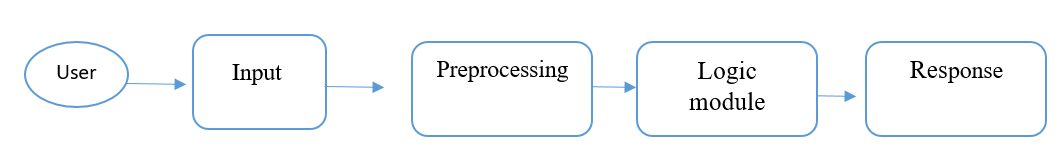
Chatbot là một chương trình máy tính tương tác với người dùng bằng ngôn ngữ tự nhiên dưới một giao diện đơn giản, âm thanh hoặc dưới dạng tin nhắn[[6]](#footnote-6). Chatbot thường được sử dụng trong các hệ thống hộp thoại cho các mục đích thực tế khác nhau bao gồm cả dịch vụ khách hàng hoặc thu thập thông tin. Một số cuộc trò chuyện sử dụng các hệ thống xử lý ngôn ngữ tự nhiên (natural language processing - NLP), nhưng nhiều hệ thống đơn giản sẽ lấy dữ liệu đầu vào, sau đó rút ra câu trả lời phù hợp nhất hoặc tương tự nhất từ cơ sở dữ liệu.

### Phân loại

Theo đúng cách mà chúng tương tác với người dùng, các chatbot thường được chia thành 2 loại:

* Audiotory (âm thanh):
* Siri[[7]](#footnote-7) (Apple)
* Google Assistant[[8]](#footnote-8) (Google)
* Cortana[[9]](#footnote-9) (Microsoft)
* Textual (tin nhắn):
* Thương mại điện tử: bán hàng, chăm sóc khách hàng.
* Trong các lĩnh vực y tế, tài chính, giáo dục, thể thao, du lịch,..

### Hoạt động



**Hình *1*:** Quá trình hoạt động chatbot

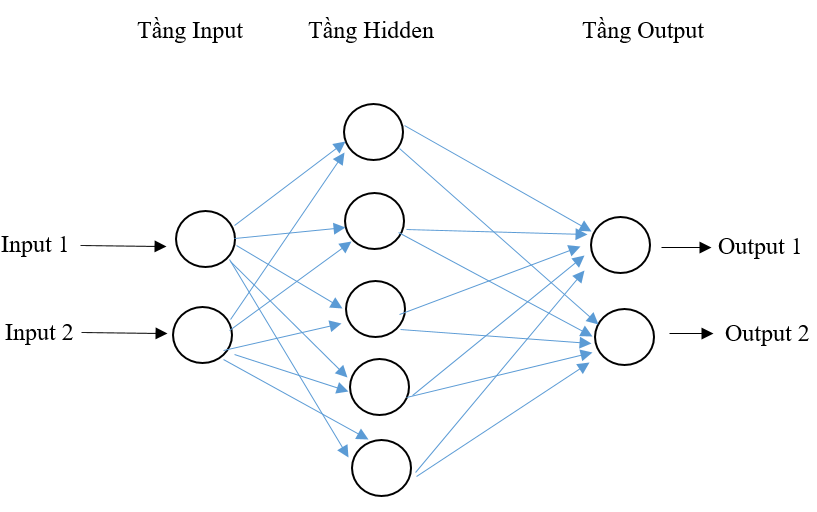
Chatbot tương tác với con người qua âm thanh hoặc văn bản và sử dụng các platform để giao tiếp với bot. Quá trình hoạt động của chatbot bao gồm:

* Input: câu đầu vào do người dùng user nhập vào.
* Preprocessing: dịch yêu cầu của người dùng, giúp máy tính hiểu được yêu cầu mình cần thực hiện.
* Logic module: tiếp nhận câu đã được xử lý từ bước trên và xử lý yêu cầu.
* Response: nhận output và đóng gói gửi trả lại messenger platform, trả lại cho người dùng kết quả.

## Mạng Nơ-ron nhân tạo

### Tổng quan về mạng nơ-ron nhân tạo

Mạng noron nhân tạo[[10]](#footnote-10) (Artifical Neural Networks - ANN) mô phỏng lại mạng nơ-ron sinh học là một cấu trúc khối gồm các đơn vị tính toán đơn giản được liên kết chặt chẽ với nhau trong đó các liên kết giữa các noron quyết định chức năng của mạng.

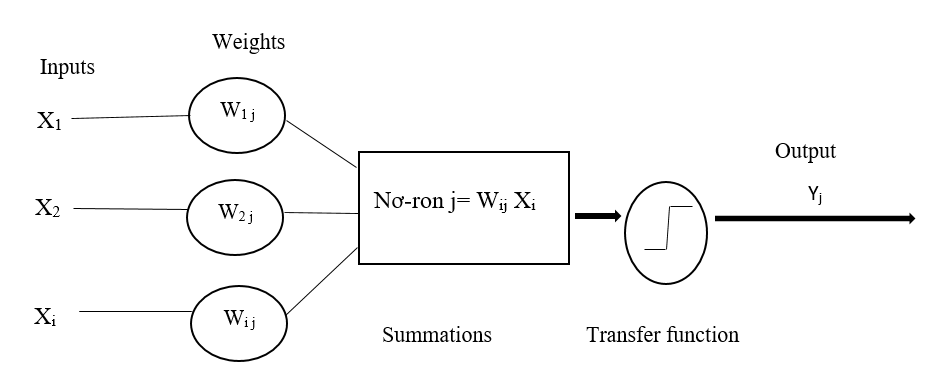


**Hình *2*:** Kiến trúc mạng nơ-ron nhân tạo

Kiến trúc chung của ANN gồm 3 thành phần đó là: input layer, hidden layer (lớp ẩn) và output layer (Hình 2)

Trong đó các Processing Elements (PE) của ANN gọi là nơ-ron, mỗi nơ-ron nhận các dữ liệu vào (inputs) xử lý chúng và cho ra một kết quả (output) duy nhất. Kết quả xử lý của một nơ-ron có thể làm input cho các nơ-ron khác.

### Quá trình xử lý thông tin của một ANN



**Hình *3***: Quá trình xử lý thông tin một ANN

Quá trình xử lý:

* Inputs (dữ liệu vào): mỗi Input tương ứng với 1 thuộc tính (attribute) của dữ liệu (patterns).
* Output (kết quả): kết quả của một ANN là một giải pháp cho một vấn đề.
* Connection Weights (trọng số liên kết) : đây là thành phần rất quan trọng của một ANN, nó thể hiện mức độ quan trọng (độ mạnh) của dữ liệu đầu vào đối với quá trình xử lý thông tin (quá trình chuyển đổi dữ liệu từ layer này sang layer khác). Quá trình học (Learning Processing) của ANN thực ra là quá trình điều chỉnh các trọng số (Weight) của các input data để có được kết quả mong muốn.
* Summation Function (hàm tổng): tính tổng trọng số của tất cả các input được đưa vào mỗi nơ-ron (phần tử xử lý PE). Hàm tổng của một nơ-ron đối với n input được tính theo công thức sau:
* Transfer Function (hàm chuyển đổi): hàm tổng của một nơ-ron cho biết khả năng kích hoạt (Activation) của nơ-ron đó còn gọi là kích hoạt bên trong (internal activation). Các nơ-ron này có thể sinh ra một output hoặc không trong ANN (nói cách khác rằng có thể output của 1 Nơron có thể được chuyển đến layer tiếp trong mạng nơ-ron hoặc không). Mối quan hệ giữa Internal Activation và kết quả (output) được thể hiện bằng hàm chuyển đổi (Transfer Function).

Việc lựa chọn Transfer Function có tác động lớn đến kết quả của ANN. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong ANN là sigmoid (logical activation) function.

Trong đó :

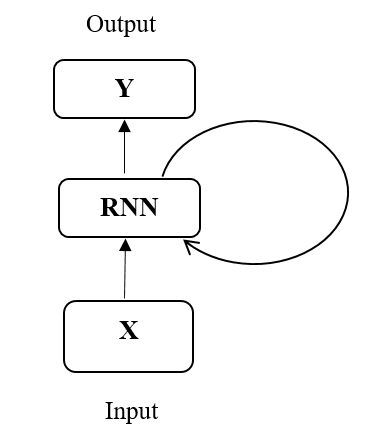
* YT: Hàm chuyển đổi
* Y: Hàm tổng

Kết quả của Sigmoid Function thuộc khoảng [0,1] nên còn gọi là hàm chuẩn hóa (Normalized Function).

Kết quả xử lý tại các nơ-ron (Output) đôi khi rất lớn, vì vậy Transfer Function được sử dụng để xử lý output này trước khi chuyển đến layer tiếp theo. Đôi khi thay vì sử dụng Transfer Function người ta sử dụng giá trị ngưỡng (Threshold value) để kiểm soát các output của các nơ-ron tại một layer nào đó trước khi chuyển các output này đến các layer tiếp theo. Nếu output của một nơ-ron nào đó nhỏ hơn giá trị ngưỡng thì nó sẽ không được chuyển đến layer tiếp theo.

### Mạng Nơ-ron hồi quy (RNN)

Ý tưởng chính của RNN[[11]](#footnote-11) (Recurrent Neural Network) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron 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. Ví dụ, nếu muốn đoán từ tiếp theo có thể xuất hiện trong một câu thì ta cũng cần biết các từ trước đó xuất hiện lần lượt thế nào chứ nhỉ? RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, 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 (ta cùng bàn cụ thể vấn đề này sau) mà thôi. Về cơ bản một mạng RNN có dạng như sau:

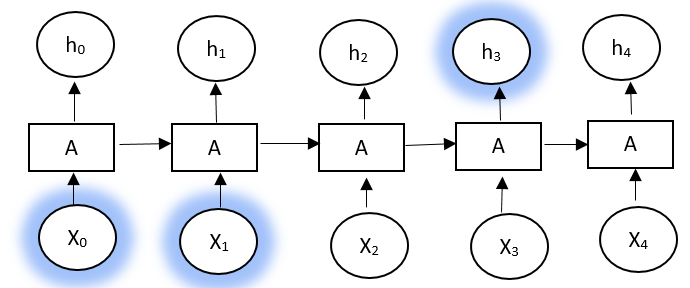


**Hình *4***: Một mạng nơ-ron hồi quy

### Vấn đề phụ thuộc xa:

Một điểm nổi bật của RNN chính là ý tưởng kết nối các thông tin phía trước để dự đoán cho hiện tại. Việc này tương tự như ta sử dụng các cảnh trước của bộ phim để hiểu được cảnh hiện thời.

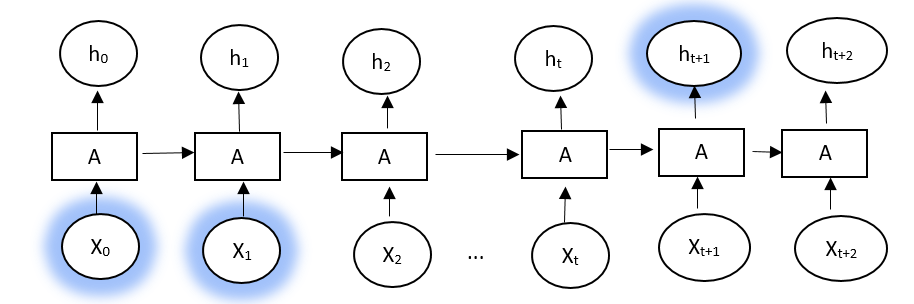
Đôi lúc chỉ cần xem lại thông tin vừa có thôi là đủ để biết được tình huống hiện tại. Ví dụ, ta có câu: “buổi sáng thật trong lành” thì ta chỉ cần đọc tới “buổi sáng thật trong” là đủ biết được chữ tiếp theo là “lành” rồi. Trong tình huống này, khoảng cách tới thông tin có được cần để dự đoán là nhỏ, nên RNN hoàn toàn có thể học được.



**Hình *5*:** Vấn đề phụ thuộc xa trong RNN

Nhưng trong nhiều tình huống buộc phải sử dụng nhiều ngữ cảnh hơn để suy luận. Ví dụ, dự đoán chữ cuối cùng trong đoạn: “Tôi sinh ra và lớn lên ở Mỹ... Tôi có thể nói lưu loát Tiếng Anh”. Rõ ràng là các thông tin gần (“Tôi có thể nói lưu loát”) chỉ có phép biết được đằng sau nó sẽ là tên của một ngôn ngữ nào đó, còn không thể nào biết được đó là tiếng gì. Muốn biết là tiếng gì thì cần phải có thêm ngữ cảnh “Tôi sinh ra và lớn lên ở Mỹ” nữa mới có thể suy luận được. Rõ ràng là khoảng cách thông tin lúc này có thể đã khá xa rồi.

Thật không may là với khoảng cách càng lớn dần thì RNN bắt đầu không thể nhớ và học được nữa.



**Hình *6*:** Khoảng cách giữa các RNN lớn dần

Về mặt lý thuyết, rõ ràng là RNN có khả năng xử lý các phụ thuộc xa (long-term dependencies). Chúng ta có thể xem xét và cài đặt các tham số sao cho khéo là có thể giải quyết được vấn đề này. Tuy nhiên, đáng tiếc trong thực tế RNN có vẻ không thể học được các tham số đó. Để giải quyết vấn đề phụ thuộc xa của RNN, bộ nhớ dài ngắn LSTM được giới thiệu.

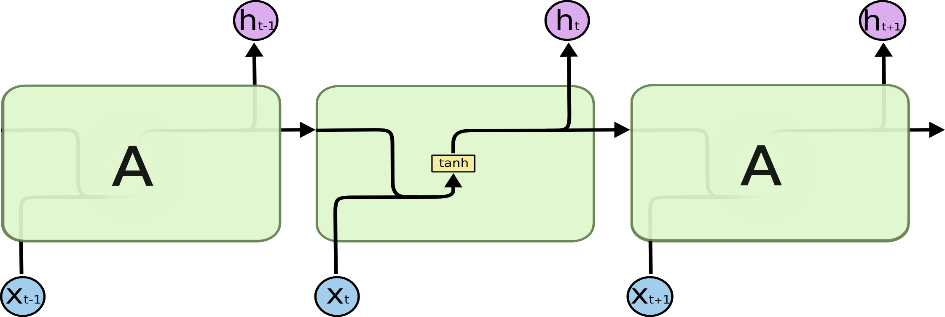
### Mạng LSTM

Về cơ bản mô hình của LSTM[[12]](#footnote-12) không khác mô hình truyền thống của RNN, nhưng chúng sử dụng hàm tính toán khác ở các trạng thái ẩn. Bộ nhớ của LSTM được gọi là tế bào (Cell) và bạn có thể tưởng tượng rằng chúng là các hộp đen nhận đầu vào là trạng thái phía trước  ht-1 và đầu vào hiện tại xt ​. Bên trong hộp đen này sẽ tự quyết định cái gì cần phải nhớ và cái gì sẽ xoá đi. Sau đó, chúng sẽ kết hợp với trạng thái phía trước, nhớ hiện tại và đầu vào hiện tại. Vì vậy mà ta ta có thể truy xuất được quan hệ của các từ phụ thuộc xa nhau rất hiệu quả.

Mạng bộ nhớ dài-ngắn (Long Short Term Memory networks), thường được gọi là LSTM - là một dạng đặc biệt của RNN, nó có khả năng học được các phụ thuộc xa. LSTM được giới thiệu bởi Hochreiter và Schmidhuber [16] , và sau đó đã được cải tiến và phổ biến bởi rất nhiều người trong ngành. Chúng hoạt động cực kì hiệu quả trên nhiều bài toán khác nhau nên dần đã trở nên phổ biến như hiện nay.

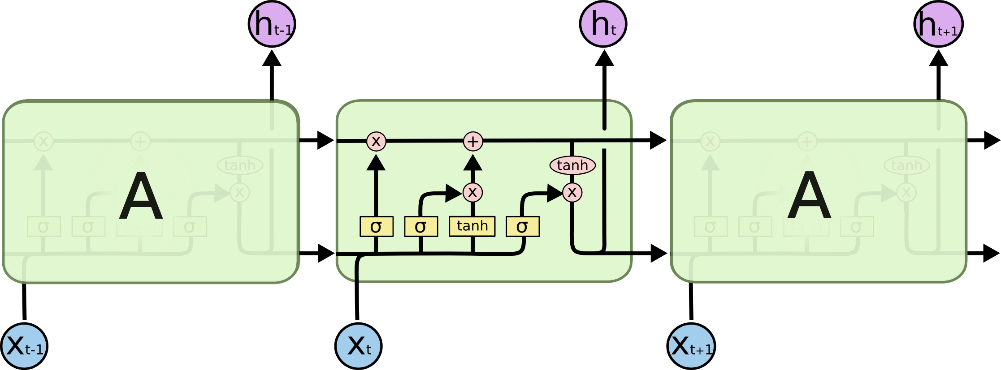
LSTM được thiết kế để tránh được vấn đề phụ thuộc xa (long-term dependency). Việc nhớ thông tin trong suốt thời gian dài là đặc tính mặc định của chúng, chứ ta không cần phải huấn luyện nó để có thể nhớ được. Tức là ngay nội tại của nó đã có thể ghi nhớ được mà không cần bất kì can thiệp nào.

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



**Hình *7*:** 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 *8*:** Mô-đun lặp lại trong LSTM chứa bốn lớp tương tác

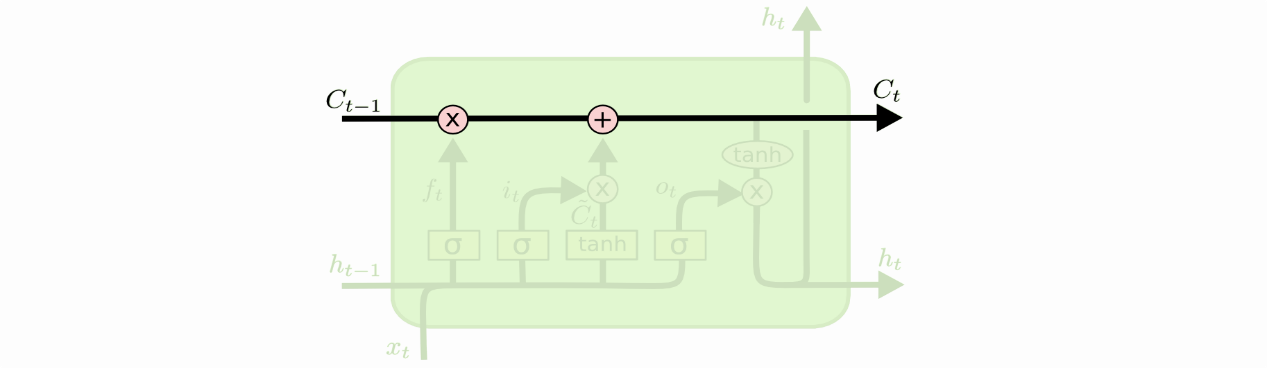
Kí hiệu sẽ sử dụng ở dưới đây:

****Hình *9*:** Kí hiệu trong LSTM

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.

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 *10*:** Trạng thái tế bào

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.

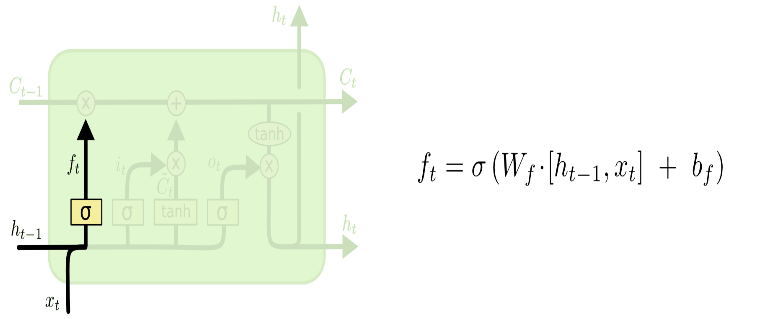


**Hình *11*:** Các cổng trạng thái

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

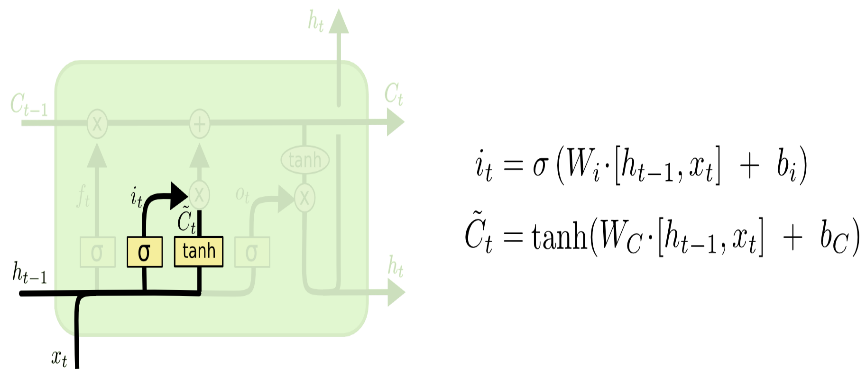
**Quá trình hoạt động của LSTM:**

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 ht-1 ​ và xt ​ rồi đưa ra kết quả là một số trong khoảng [0, 1] cho mỗi số trong trạng thái tế bào Ct-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.



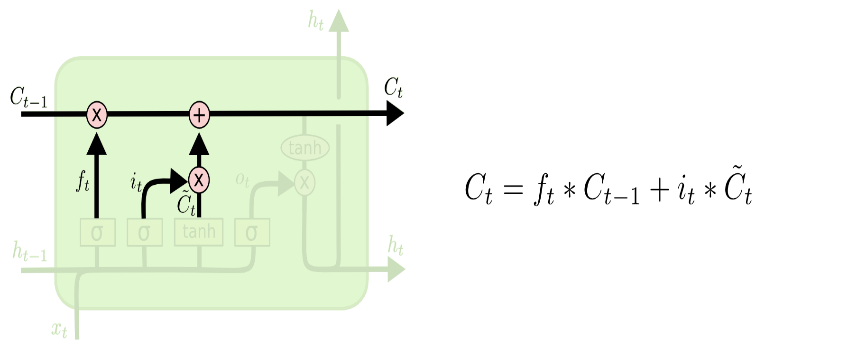
**Hình *12*:** Hàm ft trong LSTM

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 tanhtạo ra một véc-tơ cho giá trị mới Ĉt 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.



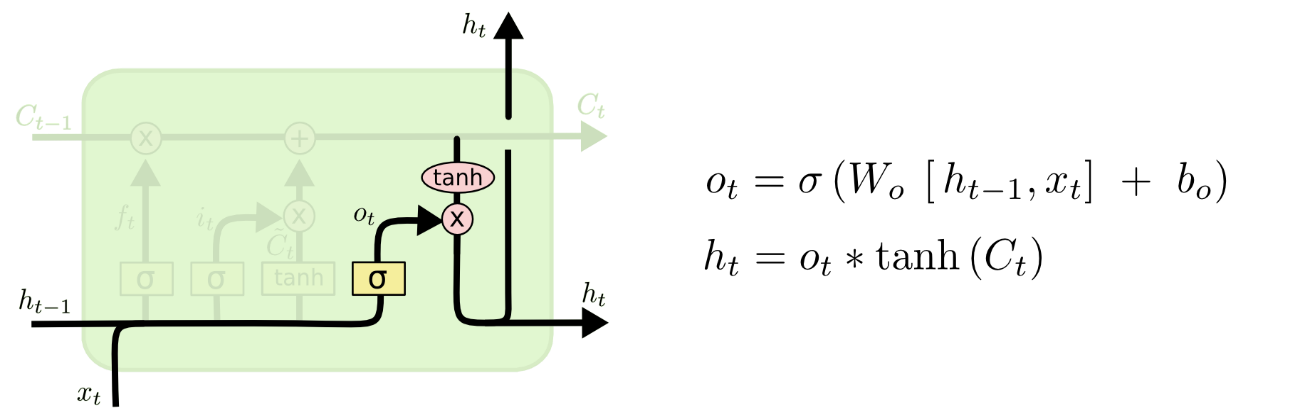
**Hình *13*:** Hàm it trong LSTM

Kế tiếp là cập nhập trạng thái tế bào cũ  *Ct-1*​ thành trạng thái mới *Ct*. 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 *ft*​ để bỏ đi những thông tin ta quyết định quên lúc trước. Sau đó cộng thêm it \* Ĉt ​​. Trạng thái mơi thu được này phụ thuộc vào việc ta quyết định cập nhập mỗi giá trị trạng thái ra sao.



**Hình *14*:** Hàm Ct trong LSTM

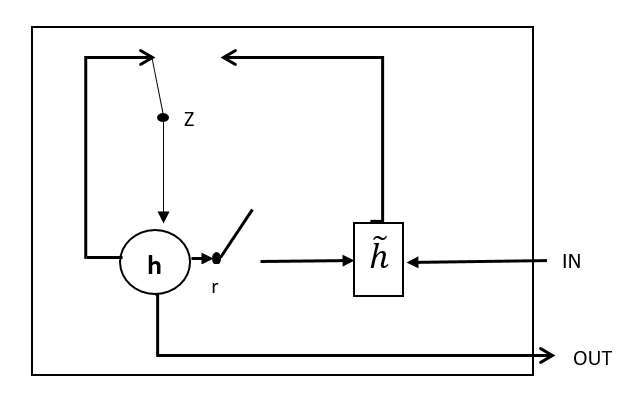
Cuối cùng, cần quyết định xem muốn đầu ra là gì. Giá trị đầu ra sẽ dựa vào trạng thái tế bào, nhưng sẽ được tiếp tục sàng lọc. Đầu tiên, chạy một tầng sigmoid để quyết định phần nào của trạng thái tế bào ta muốn xuất ra. Sau đó, đưa nó trạng thái tế bảo qua một hàm tanh tanh để co giá trị nó về khoảng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



**Hình *15*:** Hàm Ot trong LSTM

### Mạng GRU

GRU (Gated Recurrent Unit) được giới thiệu bởi [17] nhằm giải quyết vấn đề biến mất gradient đi kèm với mạng RNN. GRU là một biến thể của mạng LSTM vì cả hai có thiết kế tương tự nhau có đặc điểm dùng ít cổng hơn hẳn mạng LSTM và không có thành phần nhớ riêng biệt.



**Hình *16*:** Ví dụ về cách kích hoạt hàm ẩn GRU

Với các tham số đó là:

Trong đó:

* Z là cổng cập nhật tại thời điểm t. Cổng cập nhật giúp mô hình xác định được lượng thông tin trong quá khứ (thông tin ở bước t-1) cần chuyển đến tương lai (bước t). Điều này thực sự hữu ích bởi vì mô hình có thể quyết định copy tất cả thông tin từ quá khứ và loại bỏ nguy cơ mất mát đạo hàm.
* r là cổng đặt lại, cổng này được sử dụng từ mô hình để quyết định lượng thông tin trong quá khứ bị quên đi.
* h’ là nội dung nhớ hiện tại, nó sử dụng cổng reset r để lưu trữ thông tin có liên quan đến quá khứ.
* h là bộ nhớ tại thời điểm hiện tại chứa thông tin ở thời điểm hiện tại t và truyền nó đi. Để thực hiện điều này, cần có cổng update. Nó xác định nội dung thu thập từ bộ nhớ hiện tại ht' và những gì từ các bước trước đó ht-1.

GRU chỉ có 2 cổng: cổng thiết lập lại r và cổng cập nhập z. Cổng thiết lập lại sẽ quyết định cách kết hợp giữa đầu vào hiện tại với bộ nhớ trước, còn cổng cập nhập sẽ chỉ định có bao nhiêu thông tin về bộ nhớ trước nên giữ lại. Như vậy RNN thuần cũng là một dạng đặc biệt của GRU, với đầu ra của cổng thiết lập lại là 1 và cổng cập nhập là 0.

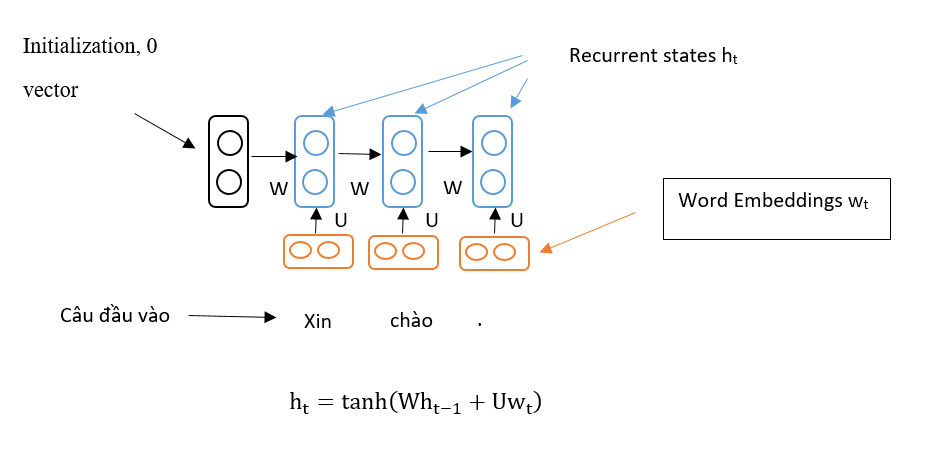
Những điểm khác nhau giữa GRU và LSTM:

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

### Mạng hierarchical encoder decoder

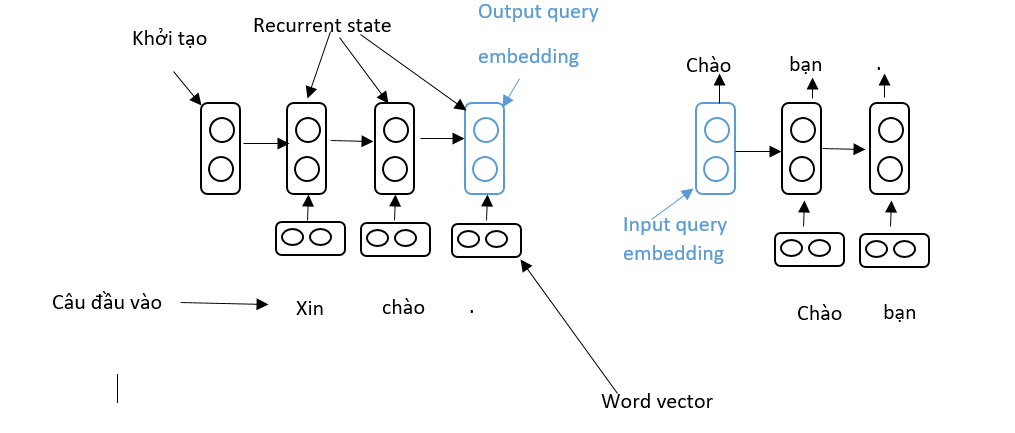
Chúng ta có Session (S) là quá trình làm việc dự đoán chuỗi tiếp theo từ chuỗi đầu vào với M chuỗi U. S = . Với mỗi ta có m là chiều dài của câu.

Mạng RNN được dùng cho chuỗi câu đầu vào:



**Hình *17*:** Mạng RNN với chuỗi câu đầu vào

Trong đó trọng số ma trận U và W không thay đổi trong thời gian.

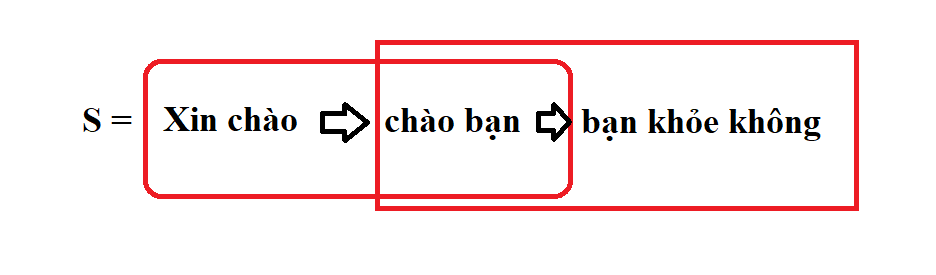


**Hình *18*:** RNN encoder decoder

Recurrent state cuối cùng chuỗi được gọi là query embedding.

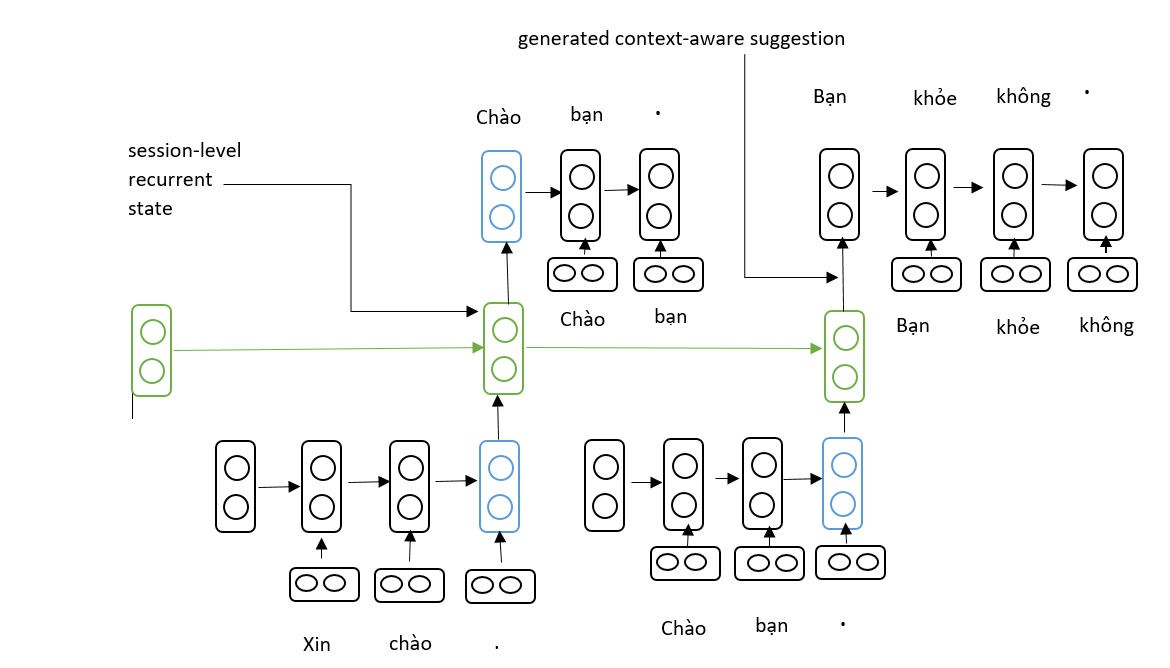
Query embedding của encoder RNN này là input query embedding của decoder RNN.

Vấn đề ở mô hình RNN encoder decoder hoàn toàn theo cặp sẽ dẫn đến thiếu ngữ cảnh chung cho nhiều chuỗi liên tục ví dụ:



**Hình *19*:** Ví dụ một chuỗi câu liên tục

Để giải quyết vấn đề này ta cần thêm một mạng RNN bổ sung vào mô hình RNN encoder decoder trong quá trình xử lý. Mô hình này đã được Sordoni và cộng sự [18] đã giới thiệu gọi là hierarchical recurrent encoder decoder.



**Hình *20*:** Mô hình hierarchical encoder decoder trong tiếng Việt

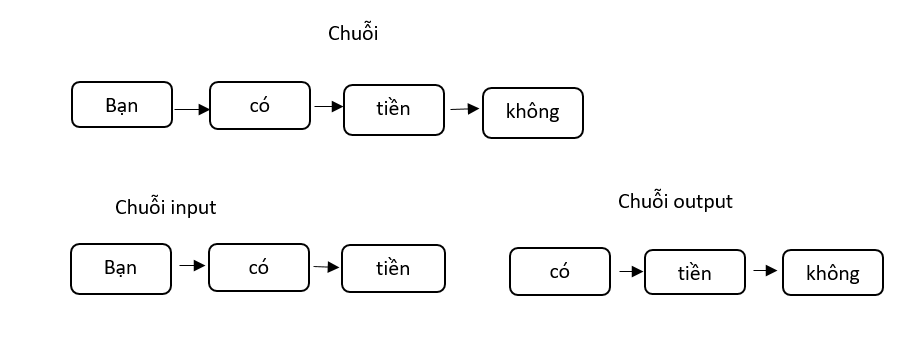
Trong hình 20, session-level recurrent state là trạng thái của quá trình làm việc cho câu đầu vào và dự đoán câu tiếp theo sẽ luôn được cập nhật. Mô hình mã hóa thông tin tại câu đầu và dự đoán câu tiếp theo. Quá trình này lặp lại trong quá trình làm việc của câu đầu vào.

## Mô hình sequence to sequence

Ứng dụng của mô hình ngôn ngữ cho phép nhận chuỗi các phần tử đầu vào, nhìn vào từng phần tử của chuỗi và cố gắng dự đoán phần tử tiếp theo của chuỗi văn bản. Có thể mô tả quá trình này bằng phương trình hàm số sau đây:

Trong đó, Yt là phần tử chuỗi ở thời điểm t, Yt-1 là phần tử ở thời điểm trước đó và f là hàm ánh xạ các phần tử trước đó của chuỗi sang phần tử tiếp theo của chuỗi. Trong các ứng dụng hiện đại, f đại diện cho mạng nơ-ron mà có thể dự đoán được phần tử tiếp theo của chuỗi, được cho trước bởi một phần tử hiện tại trong chuỗi đó. Trong mô hình ngôn ngữ có thể sinh sản, khi được huấn luyện thì chúng có thể được sử dụng để sinh ra các chuỗi thông tin bằng cách cho kết quả đầu ra ở bước trước làm đầu vào của mô hình.

Ví dụ, xét câu “Bạn có tiền không”. Một input là một lát cắt của chuỗi từ phần tử đầu tiên đến phần tử gần cuối. Chuỗi output là một lát cắt của chuỗi từ phần tử thứ 2 đến phần tử cuối cùng. Với mỗi lát cắt là từ được split sau khi đã tách từ.

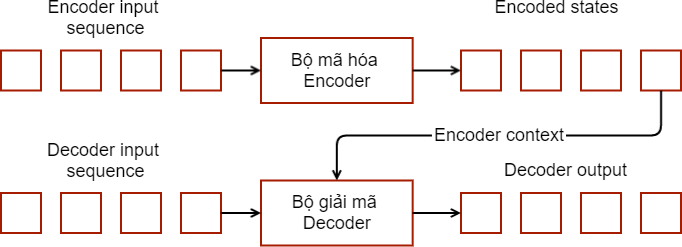


**Hình *21*:** Mô hình phát sinh văn bản

Trong quá trình huấn luyện, mô hình cố gắng dự đoán phần tử tiếp theo của chuỗi output được cho bởi phần tử hiện tại của chuỗi output. Trong quá trình phát sinh, mô hình sẽ lấy kết quả đã được sinh ra ở bước trước làm đầu vào cho lần dự đoán tiếp theo. Không giống với các mô hình ngôn ngữ đơn giản là chỉ dự đoán xác suất cho tử tiếp theo khi được cho bởi từ hiện tại, LSTM/GRU được ứng dụng sẽ “chụp” lại toàn bộ bối cảnh của chuỗi đầu vào và dự đoán xác suất tạo ra các từ tiếp theo dựa trên các từ hiện tại, cũng như các từ trước. LSTM/GRU có thể được ứng dụng như là mô hình ngôn ngữ cho việc sinh các phần tử của chuỗi sau khi được huấn luyện bởi các chuỗi tuần tự.

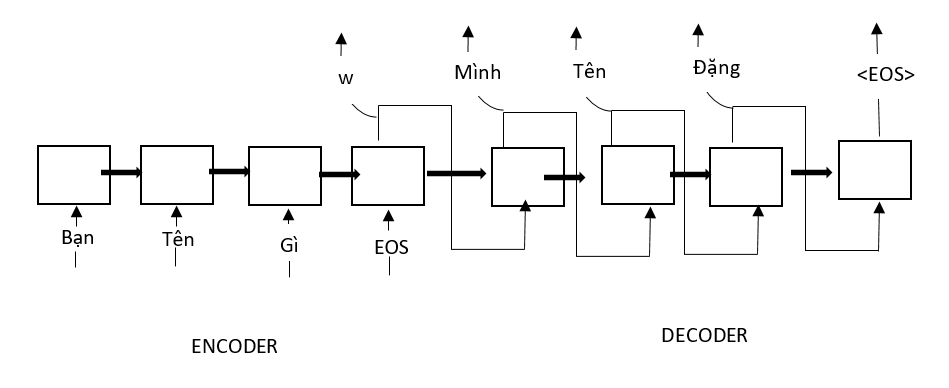
RNN có thể được sử dụng như là mô hình ngôn ngữ cho việc dự đoán các phần tử của một chuỗi khi cho bởi các phần tử trước đó của một chuỗi. Tuy nhiên, chúng vẫn còn thiếu các thành phần cần thiết cho việc xây dựng các mô hình đối thoại, hay các mô hình máy dịch, bởi vì chúng chỉ có thể thao tác trên một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

Các mô hình chuỗi sang chuỗi được xây dựng bên trên mô hình ngôn ngữ bằng việc thêm vào một bộ mã hóa Encoder và một bộ giải mã Decoder.



**Hình *22*:** Mô hình seq2seq

Mô hình seq2seq [4] được Quoc V.Le, et al giới thiệu vào năm 2014, kể từ đó đã trở thành mô hình phổ biến cho các hệ thống đối thoại và dịch máy. Mô hình seq2seq cơ bản gồm 2 mạng nơ-ron thành phần là bộ mã hóa và bộ giải mã để sinh chuỗi đầu ra từ một chuỗi đầu vào. Theo đó, bộ mã hóa nhận chuỗi đầu vào và mã hóa thành một vector a có độ dài cố định, mục tiêu của nó là chuyển đổi một chuỗi các phần tử vào một vectơ đặc trưng có kích thước cố định mà nó chỉ mã hóa thông tin quan trọng trong chuỗi và bỏ qua các thông tin không cần thiết. Bộ giải mã sẽ lần lượt sinh ra từng từ trong chuỗi đầu ra dựa trên vector a và những từ được dự đoán trước đó cho tới khi gặp kí tự kết thúc câu EOS (End of Sentence).



**Hình *23*:** Mô hình đối thoại seq2seq

Mỗi hidden state ảnh hưởng đến hidden state tiếp theo và hidden state cuối cùng được xem như tích lũy tóm tắt về chuỗi. Trạng thái này được gọi là bối cảnh hay vector suy diễn, vì nó đại diện cho ý định của chuỗi. Từ bối cảnh đó, bộ giải mã tạo ra một chuỗi, một phần tử tại một thời điểm. Ở đây, tại mỗi bước, bộ giải mã bị ảnh hưởng bởi bối cảnh và các phần tử được sinh ra trước đó.

Mô hình seq2seq 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. Kỹ thuật attention được đưa ra để giải quyết vấn đề đó.

## Kỹ thuật Attention

Kỹ thuật Attention được đề xuất bởi Bahdanau và cộng sự [18] năm 2014, lấy cảm hứng từ mô hình visual attention trong ngành thị giác máy tính. Nếu phần giao tiếp giữa mã hóa và giải mã chỉ có một vector ngữ cảnh duy nhất, vector đó sẽ phải mã hóa cho nguyên chuỗi đầu vào có thể dẫn đến bị tan biến khi nó xử lý từ dài. Thay vào đó, kỹ thuật Attention cho phép bộ giải mã tập trung vào một phần khác nhau từ đầu ra của bộ mã hóa bằng cách tìm ra chuỗi vector ngữ cảnh của mỗi bước mã hóa từ 1 đến n như sau:

Trong đó:

* ci là vector ngữ cảnhtại bước i,
* αij là trọng số attention,
* hj là trạng thái ẩn.

Trọng số attention được tính bởi công thức sau:

Trong đó:

* Si-1 là trạng thái ẩn của bước i-1 tại decoder.

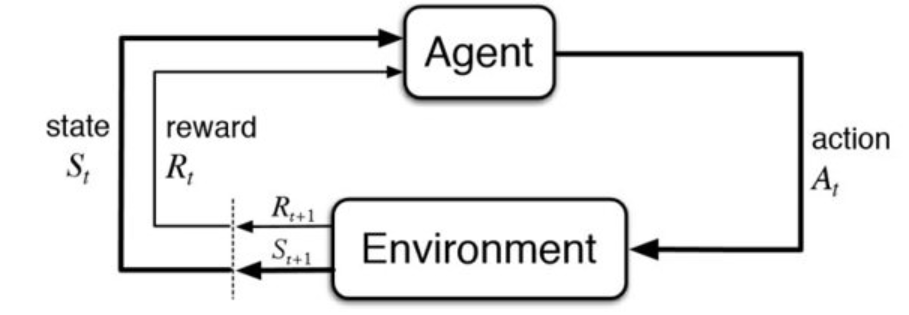
## Reinforcement Learning

### Tổng quát

Học tăng cường[[13]](#footnote-13) (Reinforcement Learning) là một lĩnh vực con của học máy, nghiên cứu cách thức một agent trong một môi trường nên chọn thực hiện các hành động nào để cực đại hóa một khoản thưởng (reward) nào đó về lâu dài. Các thuật toán học tăng cường cố gắng tìm một chiến lược ánh xạ các trạng thái của thế giới tới các hành động mà agent nên chọn trong các trạng thái đó.

Môi trường thường được biểu diễn dưới dạng một quá trình quyết định Markov trạng thái hữu hạn (Markov decision process - MDP[[14]](#footnote-14)), và các thuật toán học tăng cường cho ngữ cảnh này có liên quan nhiều đến các kỹ thuật quy hoạch động. Các xác suất chuyển trạng thái và các xác suất thu lợi trong MDP thường là ngẫu nhiên nhưng lại tĩnh trong quá trình của bài toán (stationary over the course of the problem).

Khác với học có giám sát, trong học tăng cường không có các cặp dữ liệu vào/kết quả đúng, các hành động gần tối ưu cũng không được đánh giá đúng sai một cách tường minh. Hơn nữa, ở đây hoạt động trực tuyến (on-line performance) được quan tâm, trong đó có việc tìm kiếm một sự cân bằng giữa khám phá (lãnh thổ chưa lập bản đồ) và khai thác (tri thức hiện có). Trong học tăng cường, sự được và mất giữa khám phá và khai thác đã được nghiên cứu chủ yếu qua bài toán multi-armed bandit.



**Hình *24*:** Kịch bản của học tăng cường

Các thuật ngữ trong học tăng cường[[15]](#footnote-15):

* Agent: một thực thể giả định thực hiện các hành động trong một môi trường để đạt được phần thưởng,
* State (s): tập các trạng thái của môi trường,
* Environment : là không gian mà agent tương tác,
* Action (a): tập các hành động,
* Reward (R): phần thưởng tương ứng từ môi trường mà agent nhận được khi thực hiện một action,
* Policy (π): chiến lược mà agent sử dụng để xác định hành động tiếp theo dựa trên trạng thái hiện tại,
* Episode: một chuỗi các trạng thái và hành động cho đến trạng thái kết thúc
* Accumulative Reward: phần thưởng tích lũy tổng phần thưởng tích lũy từ 1 state đến state cuối cùng. Như vậy, tại state s agent tương tác với environment với hành động a dẫn đến state mới tương ứng với phần thưởng . Vòng lặp như thế cho đến trạng thái cuối cùng.

### Policy Gradient

Mục tiêu của học tăng cường là tìm ra một chính sách hành vi tối ưu cho agent để có được phần thưởng tối ưu[[16]](#footnote-16). Gọi   là hàm phân bố xác suất của state tiếp theo khi agent tại state s và thực hiện action a. Gọi   là chuỗi từ state   đến state . Xác suất xảy ra chuỗi :

Hàm mục tiêu của bài toán học tăng cường: cần phải tìm tốt nhất để tối đa hóa . Sau khi trải qua N episodes khác nhau ta thu được N mẫu τ khác nhau

Cuối cùng ta thu được gradient của hàm mục tiêu:

Sau khi trải qua N episodes gradient của hàm mục tiêu:

Ta có thuật toán REINFORCE:

* Lấy một tập N chuỗi dựa theo policy
* Tính gradient
* Update

Việc tối ưu hàm mục tiêu cũng đồng nghĩa với việc tăng xác suất để đi theo chuỗi

τ cho accumulative reward cao.

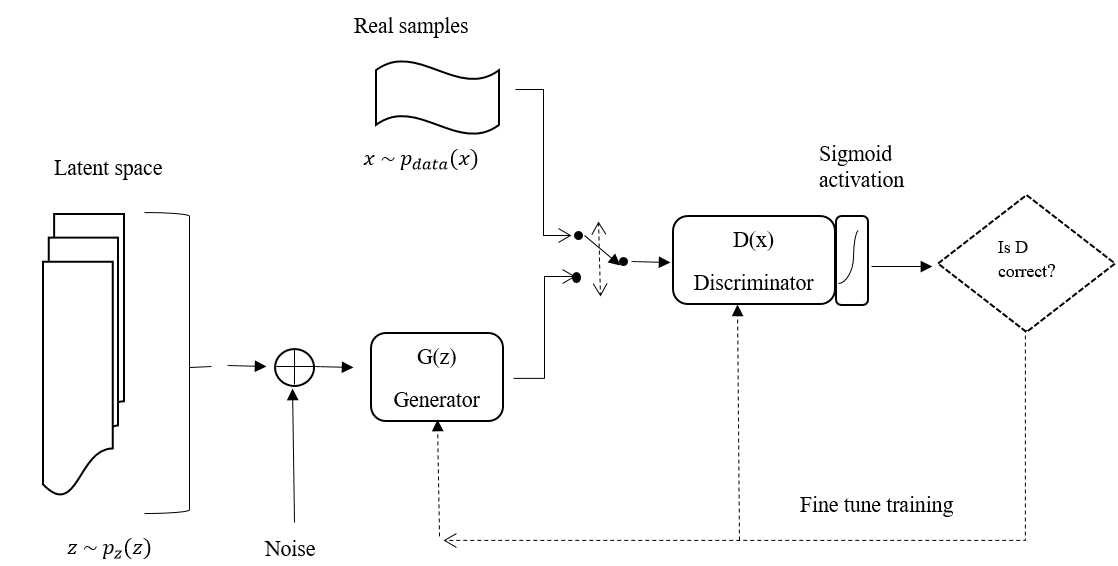
## Mạng GAN

### Tổng quát về GAN

GAN (Generative adversarial network) là một thuật toán học không giám sát (Unsupersived Learning) được Ian Goodfellow [19] giới thiệu vào năm 2014 tại hội nghị NIPS, trong đó bao gồm hai thành phần chính là Generator và Discriminator:

* **Generator** (G) nhận nhiệm vụ học ra cách áp xạ từ một không gian tìm ẩn Z  vào một không gian với phân phối từ dữ liệu cho trước.
* **Discriminator** (D) nhận nhiệm vụ phân biệt dữ liệu được tạo ra từ G và dữ liệu cho trước.

### Quá trình hoạt động của GAN



**Hình *25*:** Mô hình hoạt động của GAN

Generator lấy một random input và cố gắng sinh ra được sample của data. Như mô tả ở hình trên, generator G(z) lấy input z từ pz(z), với z là sample thuộc phân phối xác xuất p(z), được sinh ngẫu nhiên từ latent space, sau đó gán thêm nhiễu (noise). Sample sinh ra từ G(z) được nạp vào Discriminator network D(x). Công việc của Discriminator network là lấy input từ tập train (real sample) và sample được sinh ra từ G (generated sample) và xác định xem sample nào mới là thật. Real sample x được lấy từ phân phối xác suất pdata(x).

D(x) xử lý vấn đề binary classification bằng cách sử dụng hàm sigmoid, trả về kết quả khoảng từ 0 đến 1, với xác suất đầu ra càng cao thì khả năng sample đó là thật (sample lấy từ tập data) càng lớn, và ngược lại.

D được huấn luyện để tối đa xác suất gán đúng nhãn cho sample, đồng thời G lại được huấn luyện để tối thiểu khả năng phát hiện của D, tương đương tối thiểu:

Nói cách khác, việc huấn luyện D và G tương ứng với trò chơi minimax giữa hai người cho hàm số:

Trong đó:

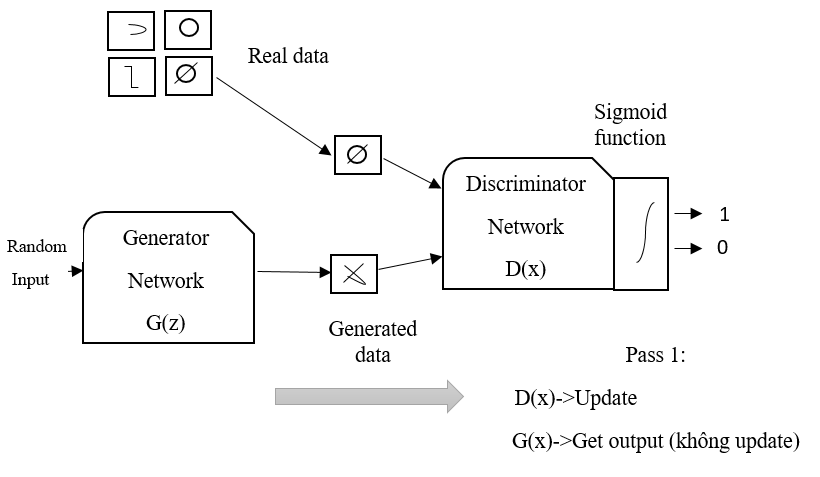
* là giá trị kỳ vọng khả năng sample từ phân phối training được D đánh giá dữ liệu thật. Giá trị này càng cao thì khả năng đánh giá dữ liệu training của D càng chính xác.
* là giá trị kỳ vọng khả năng sample từ G (được sinh ra từ phân phối pz) được D đánh giá là dữ liệu giả. Giá trị này càng cao khả năng đánh giá dữ liệu sinh của D càng chính xác.

Tổng thể, D cố gắng maximize V(D,G) trong khi G thì ngược lại. Quá trình huấn luyện sẽ hội tụ khi   D không thể phân biệt được 1 sample là thật hay giả (xác suất đều là 0.5).

### Quá trình huấn luyện

Nhìn chung, quá trình training của GAN được chia thành phần, thực hiện theo thứ tự:

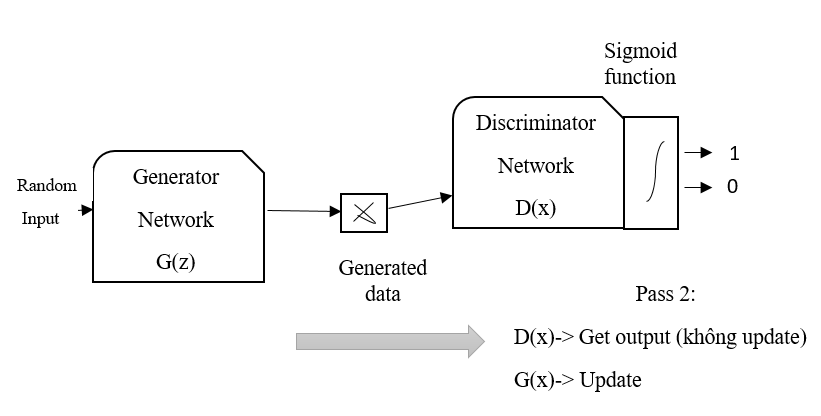
a. Huấn luyện Discriminator network, cố định Generator network (cố định ở đây nghĩa là chỉ feed-forward, không backpropagation)



**Hình *26*:** Huấn luyện Discriminator network, cố định Generator network

D được update Stochastic gradient sau mỗi bước train bằng cách công thêm giá trị

b. Huấn luyện Generator network, cố định Discriminator network



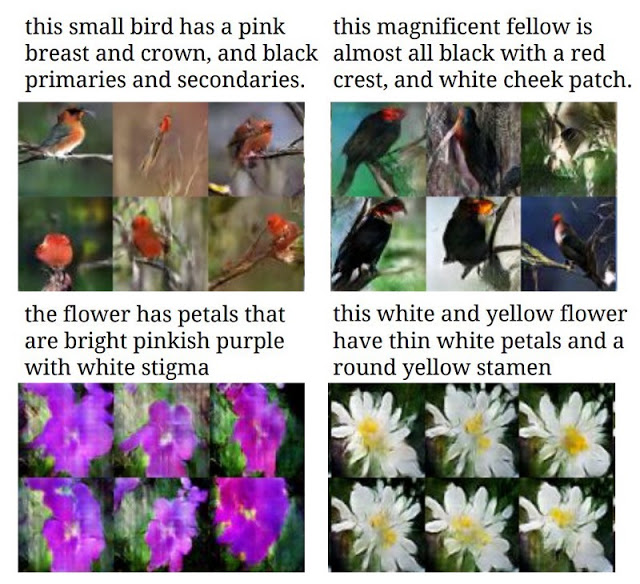
**Hình *27*:** Huấn luyện Generator network, cố định Discriminator network

G được update Stochastic gradient bằng cách trừ đi giá trị:

### Ứng dụng của GAN

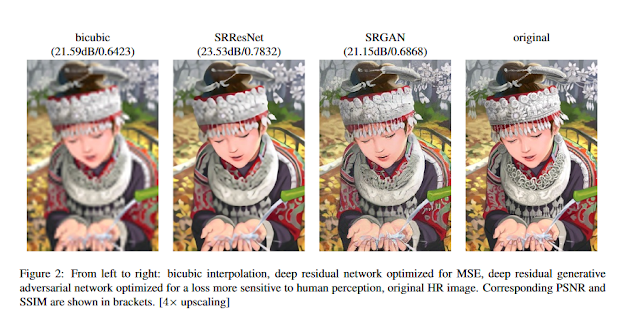
GAN cùng với các biến thể được phát triển của có đem lại rất nhiều ứng dụng hữu ích:

* Dịch văn bản thành hình ảnh

[](https://4.bp.blogspot.com/-JEdNoxX5OPU/Wbj7JW3TtYI/AAAAAAAAADA/OFmaCd2GIHE1Bev4IZpKFWf38xEyB3lLACLcBGAs/s1600/0-1hRx-30eiCL4ZP5k-.jpg)

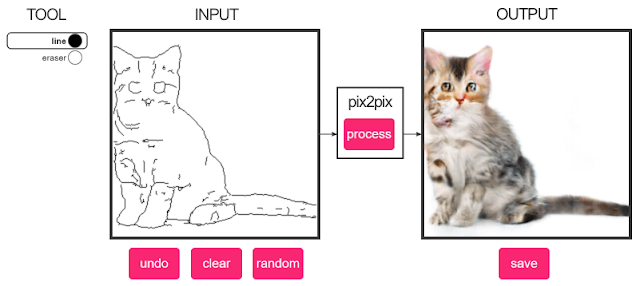
**Hình *28*:** Dịch văn bản thành hình ảnh

* Tăng độ phân giải cho hình ảnh

[](https://2.bp.blogspot.com/-xMSKRBdACl0/Wbj-VSy39gI/AAAAAAAAADM/QXnaZsJbefg2S4vjWpPEo0vr7V3xw4rDACLcBGAs/s1600/a3.png)

**Hình *29*:** Tạo ra độ phân giải ảnh cao

* Dịch mô hình ảnh thành ảnh thật

[](https://4.bp.blogspot.com/-mfDhzHw4opY/WbkBkoamu2I/AAAAAAAAADg/e2XW7NboCtYxxqwiD6o1QPKdbNMel_6EgCLcBGAs/s1600/b2.PNG)

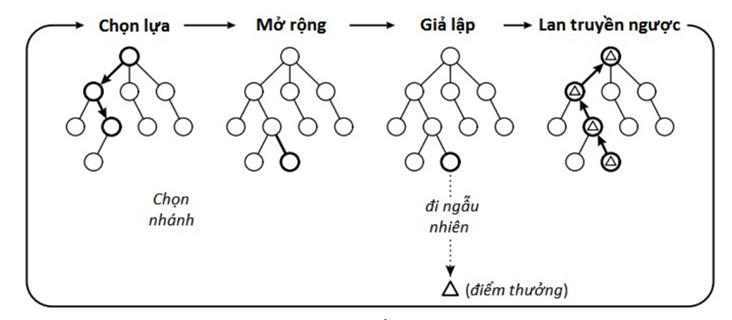
**Hình *30*:** Sinh ra ảnh từ một ảnh mô hình vẽ con mèo

## Monte Carlo Tree Search

### Định nghĩa

Tìm kiếm cây Monte Carlo ( MCTS ) là một thuật toán tìm kiếm heuristic cho một số loại quá trình quyết định , đặc biệt là những quá trình được sử dụng trong chơi trò chơi[[17]](#footnote-17). MCTS đã được sử dụng trong nhiều thập kỷ trong các chương trình máy tính Go. Nó đã được sử dụng trong các trò chơi khác như cờ vua và shogi, trò chơi có thông tin không đầy đủ như cầu và bài xì phé, cũng như trong các trò chơi video thời gian thực (như Total War: Rome IIViệc triển khai trong chiến dịch cấp cao AI ). Nổi tiếng nhất là AlphaGo.

### Nguyên tắc hoạt động



**Hình 31:** Cây tìm kiếm Monte Carlo

Trọng tâm của tìm kiếm cây Monte Carlo là phân tích các bước đi hứa hẹn nhất, mở rộng cây tìm kiếm dựa trên việc lấy mẫu ngẫu nhiên của không gian tìm kiếm.

MCTS là một quá trình lặp đi lặp lại bốn bước trong một khoảng thời gian hữu hạn:

* Chọn lựa, từ một nút gốc (trạng thái hiện hành) cho đến nút lá, vì vậy sẽ có nhiều hướng đi được mang ra đánh giá.
* Mở rộng, thêm một nút con vào nút lá của hướng được chọn trong bước chọn lựa, việc mở rộng không thực hiện trừ khi bài toán kết thúc tại nút lá.
* Giả lập, bài toán sẽ được giả lập được chơi từ nút mở rộng, sau đó kết quả của bài toán sẽ được xác định.
* Lan truyền ngược, kết quả sẽ được cập nhật cho tất cả các nút của hướng được chọn theo cách lan truyền ngược.

# CHƯƠNG 3: PHƯƠNG PHÁP THỰC HIỆN

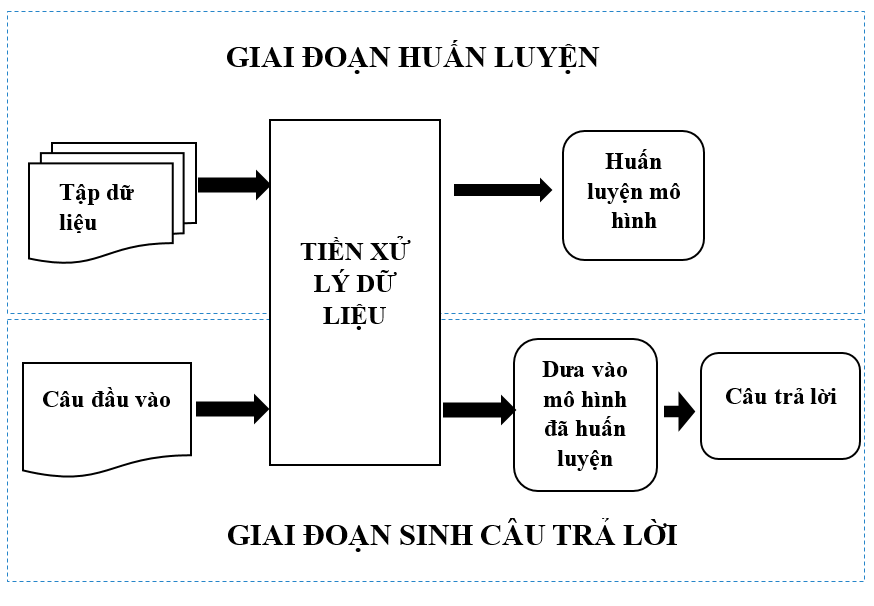
Trong chương này trình bày các bước xử lý để xây dựng mô hình.

## Tổng quát

Xây dựng hệ thống trả lời tự động dựa trên mạng chống đối tạo sinh GAN yêu cầu xây dựng hai mô hình thành phần là Generator và Discriminator. Mô hình Generator là mô hình kết hợp giữa seq2seq, GRU và kỹ thuật Attention có nhiệm vụ đọc chuỗi đầu vào, đọc từng từ để biểu diễn được vecto có kích thước cố định đưa vào mô hình kết hợp này để trích xuất đầu ra từ vecto đó. Mô hình Discriminator là một phân lớp nhị phân có nhiệm vụ gán nhãn cho chuỗi đầu vào là cặp câu hỏi và trả lời từ mô hình Generator so với dữ liệu thật từ tập data. Huấn luyện hai mô hình thành phần riêng biệt trước khi đưa vào huấn luyện theo mạng chống đối tạo sinh GAN có kết hợp thêm học RL làm cân bằng hai mô hình thành phần Generator và Discriminator liên tục cải thiện trong quá trình huấn luyện.

## Các giai đoạn xử lý

Để xây dựng mô hình, luận văn đề ra các giai đoạn:



**Hình *32*:** Các quá trình xử lý

Giai đoạn huấn luyện: sử dụng tập dữ liệu tiếng Việt sub từ các bộ phim gồm 32.000 dòng gồm câu hỏi và câu trả lời được lưu thành file text đã được làm sạch bỏ những câu vô nghĩa, khoảng cách, dấu câu,...Sau đó sử dụng công cụ Pyvi tách từ Tiếng Việt tạo tập điển và lưu lại. Sau đó đưa vào không gian vecto bằng mô hình word embedding. Huấn luyện mô hình được chia thành các các bước nhỏ: huấn luyện mô hình Generator [20] là seq2seq kết hợp GRU và Attention, huấn luyện mô hình Discriminator là một phân lớp nhị phân lấy chuỗi đầu vào (x,y) với câu đầu vào x, phản hồi y và gán nhãn xem y là cách nói tự nhiên của con người hay cách nói một cách máy móc. Sau khi huấn luyện 2 mô hình nhỏ trên ta bắt đầu huấn luyện trên nền mạng GAN sẽ được chi tiết trong xây dựng mô hình.

Giai đoạn sinh câu trả lời: nhận chuỗi đầu vào sau đó tiến hành phân tách theo mô hình đã huấn luyện ở trên và sinh ra câu trả lời.

## Xây dựng mô hình

Phần này trình bày phương pháp xây dựng ở từng giai đoạn trong mô hình.

* 1. Giai đoạn huấn luyện

**3.1.1. Tập dữ liệu**

Mục tiêu thu thập dữ liệu đầu vào cho mô hình, dữ liệu này sẽ quyết định khả năng học hỏi, xử lý. Giống như con người mô hình càng được học nhiều thì càng thông minh, tính chính xác câu mô hình càng cao. Các điều kiện bắt buộc cho dữ liệu đầu vào là khối dữ liệu đủ lớn để cho mô hình học, ngôn ngữ phải là Tiếng Việt và cần phải là những cuộc đối thoại.

Sử dụng tập data tiếng Việt, gồm hơn 32.000 dòng câu hội thoại của các nhân vật trong các bộ phim. Output của bước này là một tập tin chứa các cặp câu hỏi đáp của các nhận vật trong các bộ phim.

Dữ liệu sau khi thu thập là các dữ liệu thô có thể chứa các tạp từ, việc tiền xử lý dữ liệu sẽ chọn lọc các thuộc tính phù hợp với mô hình và làm sạch dữ liệu thô được thu thập ở bước thu thập.

Các công đoạn làm sạch xử lý dữ liệu đã thực hiện qua các bước sau:

* Loại bỏ các ký tự đặt biệt (@#$%♪, …). Ví dụ:

|  |
| --- |
| Sao giờ lại không chứ.. @@, Em có đủ can đảm để nghe..umhm..mm @##@, … |

* Xóa bỏ các ký tự phân tách câu (.,?!, …). Ví dụ:

|  |
| --- |
| thế còn anh bạn của chúng ta?, quay ra đằng sau!, … |

* Xóa bỏ các bình luận, chú thích, … Ví dụ:

|  |
| --- |
| DCBS (Tổ chức bảo tồn di tích thủ đô ), Toán học Pythagore (Pitago), … |

* Xóa bỏ các thẻ html. Ví dụ: <i>, <b>, …
* Xóa bỏ các câu tiếng Anh còn lẫn
* Xóa các đoạn bị lỗi font. Ví dụ: B#7897; #273;#7891; c#7911;a t#244;i...
* Chỉnh sửa hoặc xóa các đoạn có các từ bị mất khoảng trắng. Ví dụ:

|  |
| --- |
| Cónghĩalà ôngbiết, Chúngtôiđangtruylùng, … |

* Loại bỏ các câu không ý nghĩa. Ví dụ:

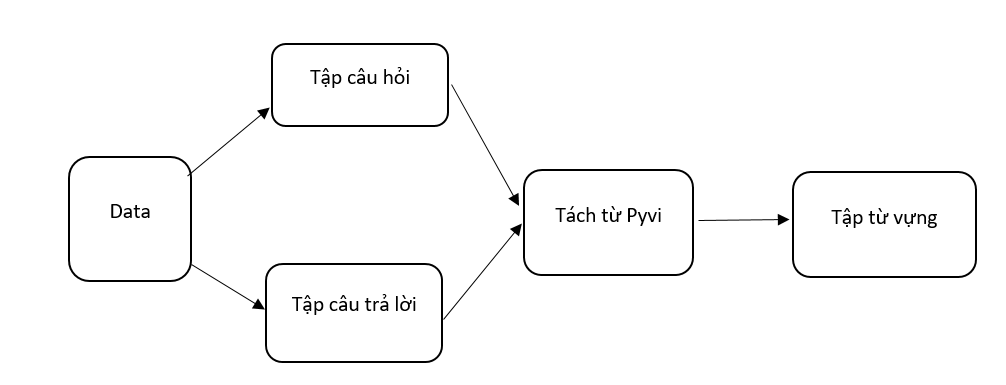
|  |
| --- |
| Phụ đề upload by ---nguoisat---, Dịch phụ đề: QKK, … |

Output của bước này là một file dữ liệu đã được xử lý tất cả các câu không có ý nghĩa, những kí tự bất thường.

Dữ liệu 32.000 câu sau bước tiền xử lý được tách ra thành 2 tập một tập câu hỏi và một tập câu trả lời tương ứng với tập câu hỏi. Ví dụ ở file tập câu hỏi: “Bạn tên gì ?” thì ở file tập câu trả lời: “Tôi tên A”.

Xử lý ngôn ngữ tự nhiên là một trong những thách thức của máy học nói chung và chatbot nói riêng. So với tiếng Anh thì tiếng Việt phức tạp hơn ở nhiều khía cạnh. Ở tiếng Anh, định nghĩa từ và kết hợp từ đơn giản trong khi tiếng Việt thì lại khác. Ví dụ câu “Welcome to CanTho” và “Chào mừng tới Cần Thơ”, cùng một ý nghĩa nhưng cách xử lý của 2 câu lại khác nhau. Trong câu “Welcome to CanTho” việc tách từ rất đơn giản, chỉ cần tách các từ theo kiểu “word by word” dựa trên khoảng trắng, các từ được tách ra là “Welcome”, “to” và “CanTho” hoàn toàn có thể đứng độc lập mà vẫn mang ý nghĩa. Trong khi đó, tiếng Việt việc tách từ không chỉ dừng lại ở mức “word by word” mà từ tách ra cần phải có ý nghĩa và có thể hiểu được. Theo đó, câu ví dụ trên sẽ tách ra là: “Chào\_mừng”, “tới”, “Cần\_Thơ”.

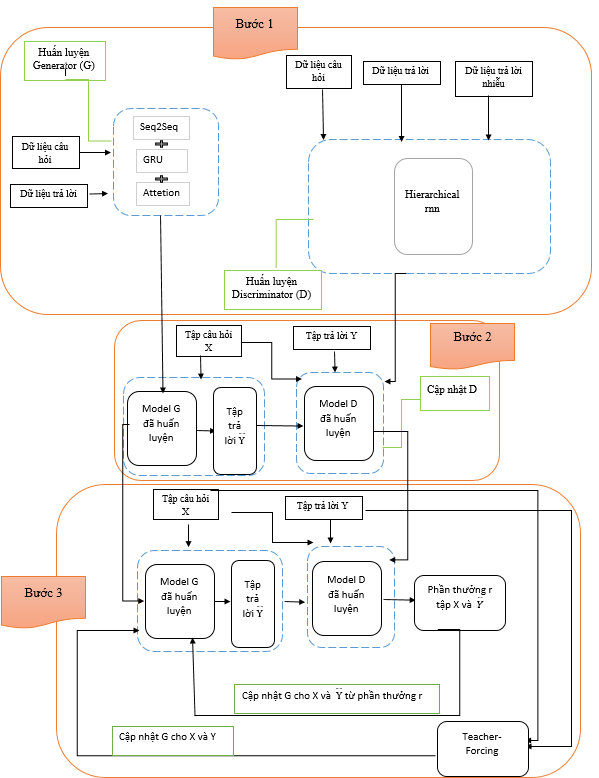
Sau khi trải qua bước tiền xử lí, dữ liệu sẽ được xử lí một lần nữa trước khi đưa vào mô hình.



**Hình *33*:** Xử lý trước khi đưa vào mô hình với Pyvi

Ở đây ta dùng công cụ Pyvi tách các từ riêng biệt cho tiếng Việt .Dữ liệu sau bước tiền xử lí sẽ được chia thành tập câu hỏi và tập câu trả lời để thuận tiện cho việc học, sử dụng các câu trong tập đó để tách các từ riêng biệt bằng công cụ Pyvi. Ví dụ, câu “Hôm nay anh sao ” sẽ được tách thành “Hôm\_nay, anh, sao”. Bước tiếp theo sẽ chọn ra 8.268 từ thông dụng nhất để xây dựng tập từ vựng, là tiền đề để xây dựng các tập đặc trưng đưa vào mô hình để huấn luyện.

* + 1. Huấn luyện mô hình



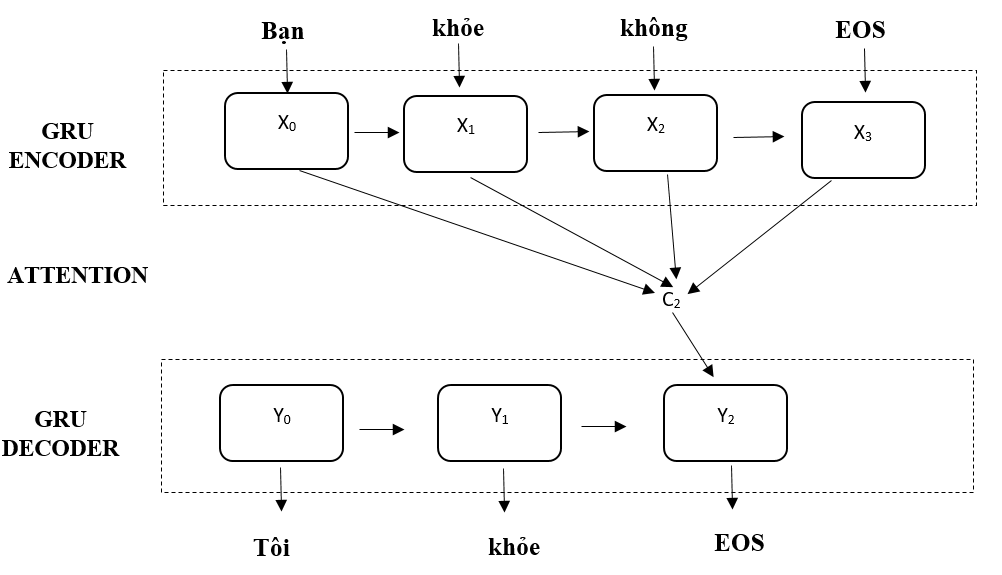
**Hình *34*:** Các bước huấn luyện mô hình

Các bước chính của giai đoạn huấn luyện mô hình:

* Bước 1: Huấn luyện Generator và Discriminator.
* Bước 2: Huấn luyện Discriminator, cố định Generator.
* Bước 3: Huấn luyện Generator, cố định Discriminator.

Phần tiếp theo đây sẽ mô tả chi tiết các bước thực hiện mô hình huấn luyện của xây dựng hệ thống trả lời tự động dựa trên nền mạng GAN và học tăng cường.

* + - 1. Huấn luyện Generator và Discriminator

Generator là mô hình sinh hội thoại kết hợp giữa seq2seq, GRU và kỹ thuật Attention.

**Hình *35*:** Seq2seq GRU với kỹ thuật Attention trong tiếng Việt

Đây là mô hình tạo sinh trong nền mạng GAN có nhiệm vụ đọc chuỗi đầu vào, đọc từng từ để biểu diễn được vecto có kích thước cố định đưa vào mô hình kết hợp giữa GRU và kỹ thuật Attetion này để tính toán trích xuất đầu ra từ vecto đó.

Discriminator :phân lớp nhị phân lấy đầu vào là một cặp câu đối thoại (x, y) và nhãn đầu ra cho biết liệu đầu vào được tạo bởi cách nói tự nhiên của con người hay một cách máy móc. Câu đầu vào được mã hóa thành vecto đại diện sử dụng hierarchical encoder [5] [21], sau đó đưa vào hàm softmax 2 lớp, trả về xác suất là một câu đối thoại được tạo ra bằng máy móc (kí hiệu ) hoặc câu thoại do con người nói một cách tự nhiên (kí hiệu ). Huấn luyện mô hình này tăng khả năng phát hiện và gán nhãn cho câu y được tạo ra từ Generator.

* + - 1. Huấn luyện Discriminator, cố định Generator

Bước này dùng để cập nhật dữ liệu mẫu từ Generator cho Discriminator để tăng khả năng phát hiện và gán nhãn cho chuỗi đầu vào từ Generator để cập nhật Generator trong bước 3.

Đầu tiên lấy dữ liệu thật gồm tập câu hỏi và trả lời (X,Y). Sau đó ta lấy tập câu hỏi X đưa vào mô hình Generator để sinh ra tập câu trả lời gọi là .

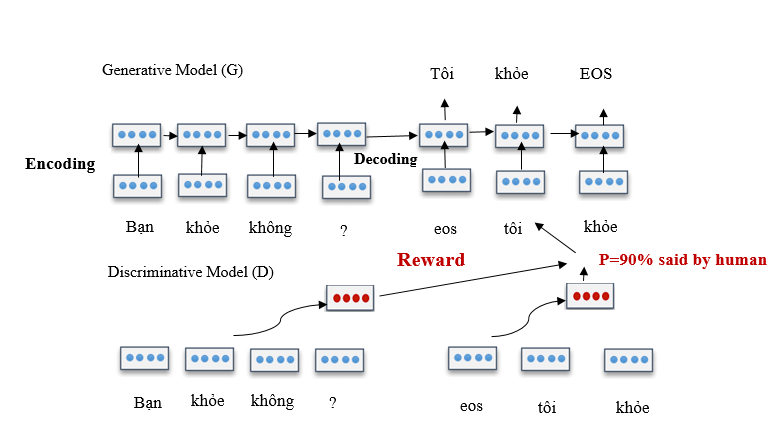
Sau đó cập nhật Discriminator sử dụng tập câu hỏi và trả lời (X,Y) là tập dữ liệu tích cực và (X, ) là tập dữ liệu tiêu cực.

* + - 1. Huấn luyện Generator, cố định Discriminator

Giống như bước 2 đầu tiên ta lấy dữ liệu thật gồm tập câu hỏi và trả lời (X,Y). Sau đó ta lấy tập câu hỏi X đưa vào mô hình Generator để sinh ra tập câu trả lời gọi là sử dụng tìm kiếm Monte Carlo search.

Tính toán phần thưởng r cho (X,) dựa trên mô hình Discriminator.

Cập nhật Generator trên (X,) bằng cách sử dụng phần thưởng r đã được tính toán sử dụng Policy Gradient Training.



**Hình *36*:** Chức năng của D xác định xác suất câu từ G

Policy Gradient Training với thuật toán REINFORCE

Như đã được giới thiệu trong phần cơ sở lý thuyết thuật toán REINFORCE được áp dụng vào mô hình. Điểm số của các phản hồi hiện tại được xem do người nói đúng một cách tự nhiên được chỉ định bởi discriminator () được sử dụng làm phần thưởng cho Generator, Generator được huấn luyện để tối đa hóa phần thưởng của các phản hồi được tạo bằng cách sử dụng thuật toán REINFORCE:

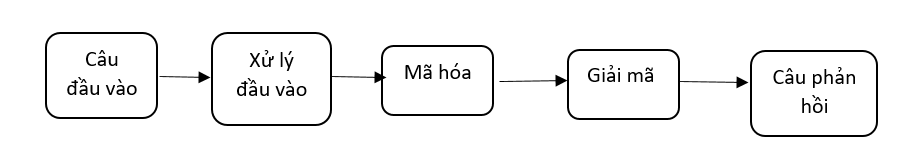
Gradient được tính:

Trong đó là giá trị baseline giảm phương sai của estimate.

Mô hình Generator chỉ có thể tiếp xúc một cách gián tiếp với các chuỗi tiêu chuẩn vàng thông qua phần thưởng được trao lại từ Discriminator. Phần thưởng đó được sử dụng để khuyến khích hoặc không khuyến khích bản thân Generator tạo ra chuỗi.

Để bắt buộc Generator đưa ra câu trả lời đúng ta cập nhật Generator một lần nữa với phần thưởng tuyệt đối là 1 bởi những câu đã được Discriminator phát hiện là câu trả lời gần đúng đây được gọi là Teacher-Forcing.

### Giai đoạn sinh câu trả lời



**Hình *37*:** Giai đoạn sinh câu trả lời với mô hình Generator

Câu đầu vào ở giai đoạn này sẽ được xử lý tách từ và dựa vào tập từ điển chúng sẽ được mã hóa thành một chuỗi các con số có độ dài cố định, trong trường hợp một từ không có trong tập từ điển chúng sẽ được gán là kí tự UNK. Bộ giải mã sẽ sinh ra câu trả lời dựa vào chuỗi đó cho tới khi gặp dấu hiệu kết thúc câu.

Output của giai đoạn này là câu phản hồi được tạo ra từ mô hình đã huấn luyện ở đây là mô hình Generator. Mô hình Generator đã được cập nhật bằng cách huấn luyện trên mạng GAN. Khi câu đầu vào không có trong bộ dữ liệu thì mô hình Generator sẽ phân tích ngữ cảnh để sinh ra câu phản hồi phù hợp. Mặc dù vậy vẫn còn nhiều bất cập được đề cập trong chương thực nghiệm và đánh giá kết quả.

CHƯƠNG 4: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

Chương này trình bày kết quả thực nghiệm được huấn luyện và thử nghiệm trên máy tính cá nhân và máy ảo, đánh giá kết quả đạt được:

## Kết quả thực nghiệm

Huấn luyện mô hình xây dựng hệ thống trả lời tự động dựa trên nền mạng GAN và học tăng cường được thực hiện với dữ liệu trên máy tính cấu hình:

* 32.000 dòng dữ liệu
* OS: Ubuntu 16.04
* RAM: 12GB
* CPU: 2 nhân 4 luồng @ 2.2GHz (up 2.7GHz)
* Tensorflow with GPU (GTX 950M 4GB)

Ta xây bộ từ vựng từ dữ liệu bằng công cụ Underthesea trước. Sau đó huấn luyện mô hình, sau thời gian huấn luyến mô hình ta thấy thời gian huấn luyện riêng cho 2 mô hình nhỏ là mô hình Generator là 45 phút/epoch và mô hình Discriminatorlà 45 phút/epoch. Sau đó ta huấn luyện theo bước 2 và bước 3 đã được trình bày trong phần đã được đề cập ở chương 3. Thời gian cập nhật lại 2 mô hình Generator và Discriminator trung bình 60 phút/epoch.

Ta so sánh kết quả mô hình là

|  |  |
| --- | --- |
|  |  |
| You: chào em gái  Bot: chào bác sỹ  You:em tên gì  Bot: brad  You:em đẹp quá  Bot: thật không  You: thật chứ  Bot: vâng  You:em ở đâu  Bot: em đây  You:chúc mừng năm mới  Bot: bạn cũng vậy  You: anh hỏi nhà em ở đâu  Bot: em không biết  You: hôm nay trời thế naò  Bot: cơm b quá | You:cuộc sống bạn ổn không  Bot: đẹp lắm  You: cuộc sống bạn thế naò  Bot: chưa giống như như vậy  You: cậu có biết mặt trời không  Bot: mờ ạ ạ ạ ạ  You: cậu sống ở đâu  Bot: ở bến tàu  You: cậu ở đâu  Bot: ở quanh đây  You: cậu có tiền không  Bot: có  You: tớ với cậu đi hát nhé  Bot: tớ không muốn cafe  You: điện thoại đẹp quá  Bot: cảm ơn cậu lại thích chú nhiều  You: bao lâu rồi chúng ta không gặp nhau  Bot: gì cơ |

## Đánh giá mô hình

Đánh giá mô hình đối thoại là một trong những thử thách. Nó được xem là chuẩn mực xem chất lượng của mô hình đã huấn luyện. Đánh giá mô hình xây dựng hội thoại này chủ yếu kết quả đầu ra thế nào để có phương pháp điều chỉnh các thông số lại cho mô hình tốt hơn.

Kiểm tra khả năng đánh giá của máy: Lỗi độ tin cậy của máy(ERE) là độ lệch trung bình của lỗi đánh giá đối thủ từ lỗi tiêu chuẩn vàng trong các trường hợp được thử, với trọng số bằng nhau cho mỗi tác vụ. Lỗi càng nhỏ, máy càng đáng tin cậy.

Kết quả sau khi huấn luyện mô hình ta có ERE là 0.26606522

Kết quả sau khi huấn luyện mô hình ta có đánh giá BLEU là 0.06

CHƯƠNG 5: KẾT LUẬN

Chương này trình bày kết quả thực nghiệm được huấn luyện và thử nghiệm trên máy tính cá nhân và đánh giá kết quả đạt được

1. Kết quả đạt được và hạn chế

Luận văn này đã xây dựng được mô hình Chatbot có thể chat được bằng tiếng Việt dựa trên mạng GAN và học tăng cường. Tuy với mô hình hiện tại bot có thể đáp ứng được các yêu cầu: có khả năng hiểu tiếng Việt và sinh câu trả lời tự động với độ dài câu hỏi bất kì, nhưng kết quả của câu trả lời còn chưa được đúng ngữ cảnh của câu đầu vào có đôi khi lặp lại một số từ trong câu trả lời làm cho nó trở nên khó hiểu. Một phần do dữ liệu huấn luyến mô hình còn hạn chế nên những câu đầu vào mới không. Tập dữ liệu huấn luyện chưa đủ lớn chưa đủ bao quát tất cả các ngữ cảnh có thể xảy ra trong cuộc sống.

1. Hướng phát triển

* Mở rộng tập dữ liệu huấn luyện
* Giới hạn số từ trong một câu để mô hình tốt hơn
* Điều chỉnh các thông số đầu vào sao cho 2 mô hình Generator và Discriminator tối ưu nhất.
* Xóa bỏ các câu trả lời ngắn giống như uh, hmm

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | Ritter, Alan, Colin Cherry, and William B. Dolan, "Data-driven response generation in social media," *Proceedings of the conference on empirical methods in natural language processing,* pp. 583-593, 2011. |
| [2] | Sordoni, Alessandro, et al., "A neural network approach to context-sensitive generation of conversational responses," *A neural network approach to context-sensitive generation of conversational responses,* pp. 1-11, 2015. |
| [3] | Mikolov, Tomáš, et al., "Recurrent neural network based language model," *Recurrent neural network based language model,* pp. 1-4, 2010. |
| [4] | Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le, "Sequence to Sequence Learning with Neural Networks," *Advances in neural information processing systems,* 2014. |
| [5] | Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., & Pineau, J. , "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models," *Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models,* pp. 3776-3783, 2016. |
| [6] | Luan, Yi, Yangfeng Ji, and Mari Ostendorf. , "LSTM based conversation models," *LSTM based conversation models,* 2016. |
| [7] | Li, Jiwei, et al. , "Deep reinforcement learning for dialogue generation," *Deep reinforcement learning for dialogue generation,* 2016. |
| [8] | Lamb, A. M., Goyal, A. G. A. P., Zhang, Y., Zhang, S., Courville, A. C., & Bengio, Y. , "Professor forcing: A new algorithm for training recurrent networks," *Advances In Neural Information Processing Systems,* pp. 4601-4609, 2016. |
| [9] | Yu, L., Zhang, W., Wang, J., & Yu, Y., "Seqgan: Sequence generative adversarial nets with policy gradient," *Thirty-First AAAI Conference on Artificial Intelligence,* 2017. |
| [10] | Chen, X., Sun, Y., Athiwaratkun, B., Cardie, C., & Weinberger, K. , "Adversarial deep averaging networks for cross-lingual sentiment classification.," *Adversarial deep averaging networks for cross-lingual sentiment classification.,* 2016. |
| [11] | Zhang, Y., Barzilay, R., & Jaakkola, T., "Aspect-augmented adversarial networks for domain adaptation," *Transactions of the Association for Computational Linguistics 5 ,* pp. 515-528, 2017. |
| [12] | Ranzato, M. A., Chopra, S., Auli, M., & Zaremba, W., "Sequence level training with recurrent neural networks," *Sequence level training with recurrent neural networks,* 2015. |
| [13] | Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., ... & Bengio, Y., "An actor-critic algorithm for sequence prediction," *An actor-critic algorithm for sequence prediction,* 2016. |
| [14] | Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., & Liu, Y. , "Minimum risk training for neural machine translation," *Minimum risk training for neural machine translation,* 2015. |
| [15] | Wiseman, S., & Rush, A. M. , "Sequence-to-sequence learning as beam-search optimization," *Sequence-to-sequence learning as beam-search optimization,* 2016. |
| [16] | Hochreiter, Sepp, and Jürgen Schmidhuber, "Long short-term memory," *Neural computation,* pp. 1735-1780, 1997. |
| [17] | Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. , "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014. |
| [18] | Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," 2014. |
| [19] | Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y., "Generative adversarial nets," *Generative adversarial nets,* pp. 1-9, 2014. |
| [20] | Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., & Bengio, Y., "Attention-Based Models for Speech Recognition," *Advances in neural information processing systems,* pp. 577-585, 2015. |
| [21] | Li, J., Luong, M. T., & Jurafsky, , "A hierarchical neural autoencoder for paragraphs and documents," *A hierarchical neural autoencoder for paragraphs and documents,* 2015. |
| [22] | Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, Dan Jurafsky, "Adversarial Learning for Neural Dialogue Generation," *Adversarial Learning for Neural Dialogue Generation,* pp. 1-13, 2017. |
| [23] | Bengio, S., Vinyals, O., Jaitly, N., & Shazeer, N. , "Scheduled sampling for sequence prediction with recurrent neural networks," *Scheduled sampling for sequence prediction with recurrent neural networks,* pp. 1-9, 2015. |
| [24] | Luong, M. T., Le, Q. V., Sutskever, I., Vinyals, O., & Kaiser, L. , "Multi-task sequence to sequence learning," *Multi-task sequence to sequence learning,* pp. 1-10, 2015. |
| [25] | A. M. Turing, "Computing machinery and intelligence. In Parsing the Turing Test," *Computing machinery and intelligence. In Parsing the Turing Test,* pp. 23-65, 2009. |
| [26] | Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. , "Generating Sentences from a Continuous Space," *Generating Sentences from a Continuous Space,* pp. 1-12, 2016. |
| [27] | Kannan, A., & Vinyals, O., "Adversarial Evaluation of Dialogue Models," *Adversarial Evaluation of Dialogue Models,* pp. 1-4, 2017. |
| [28] | Bowman, Samuel R., et al, "Generating sentences from a continuous space," *Generating sentences from a continuous space,* 2015. |
| [29] | Đ. V. Chiến, "Xây dựng hệ thống trả lời tự động dựa trên mô hình SEQ2SEQ với LSTM," *Xây dựng hệ thống trả lời tự động dựa trên mô hình SEQ2SEQ với LSTM,* 2018. |
| [30] | N. T. Kiệt, "Xây dựng hệ thống trả lời tự động dựa trên mô hình SEQ2SEQ với GRU," *Xây dựng hệ thống trả lời tự động dựa trên mô hình SEQ2SEQ với GRU,* 2018. |
| [31] | Radford, Alec, Luke Metz, and Soumith Chintala., "Unsupervised representation learning with deep convolutional generative adversarial networks," *Unsupervised representation learning with deep convolutional generative adversarial networks,* 2015. |
| [32] | Chen, Xi, et al., "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems,* pp. 2172-2180, 2016. |
| [33] | Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X., "Improved techniques for training gans," *Advances in neural information processing systems,* pp. 2234-2242, 2016. |
| [34] | Turing, A. M., "Computing machinery and intelligence," *Parsing the Turing Test,* pp. 23-65, 2009. |
| [35] | Denton, E. L., Chintala, S., & Fergus, R. , "Deep generative image models using a laplacian pyramid of adversarial networks," *Advances in neural information processing systems,* pp. 1486-1494, 2015. |
| [36] | Williams, R. J. , "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning,* pp. 229-256, 1992. |
| [37] | Glynn, P. W., "Likelihood ratio gradient estimation for stochastic systems," *Communications of the ACM,* pp. 75-84, 1990. |
| [38] | Aleksandrov, V. M., Sysoyev, V. I., & SHEMENEV. VV. , "Stochastic optimization," *Engineering Cybernetics,,* pp. 11-+, 1968. |

1. <https://en.wikipedia.org/wiki/Generative_adversarial_network> [↑](#footnote-ref-1)
2. <https://en.wikipedia.org/wiki/Statistical_machine_translation> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/BLEU> [↑](#footnote-ref-3)
4. <https://en.wikipedia.org/wiki/Perplexity> [↑](#footnote-ref-4)
5. <http://dataset.cs.mcgill.ca/ubuntu-corpus-1.0/> [↑](#footnote-ref-5)
6. <https://viblo.asia/p/tat-ca-nhung-gi-ban-can-biet-ve-chatbot-Az45bnNg5xY> [↑](#footnote-ref-6)
7. <https://en.wikipedia.org/wiki/Siri> [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/Google_Assistant> [↑](#footnote-ref-8)
9. <https://en.wikipedia.org/wiki/Cortana> [↑](#footnote-ref-9)
10. <http://bis.net.vn/forums/p/482/898.aspx> [↑](#footnote-ref-10)
11. <https://dominhhai.github.io/vi/2017/10/what-is-rnn/> [↑](#footnote-ref-11)
12. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> [↑](#footnote-ref-12)
13. <https://skymind.ai/wiki/deep-reinforcement-learning> [↑](#footnote-ref-13)
14. <https://en.wikipedia.org/wiki/Markov_decision_process> [↑](#footnote-ref-14)
15. <https://www.datacamp.com/community/tutorials/introduction-reinforcement-learning> [↑](#footnote-ref-15)
16. <http://rail.eecs.berkeley.edu/deeprlcourse/static/homeworks/hw2.pdf> [↑](#footnote-ref-16)
17. <https://en.wikipedia.org/wiki/Monte_Carlo_tree_search> [↑](#footnote-ref-17)