**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**======\*\*\*======**

BÁO CÁO PROJECT 2

**Đề tài: HỆ THỐNG CHATBOT VÀ ỨNG DỤNG TRONG ĐÀO TẠO TRỰC TUYẾN**

|  |  |
| --- | --- |
| GVHD: | Giảng viên Đỗ Phan Thuận |
| Sinh viên: | Trịnh Quang Huy  MSSV:20158996 |

Hà nội, Năm 2020

MỤC LỤC

[Danh sách biểu đồ - hình vẽ 5](#_Toc60945302)

[Mở đầu 7](#_Toc60945303)

[Chương 1. Tổng quan về hệ thống Chatbot 8](#_Toc60945304)

[1.1 Giới thiệu 8](#_Toc60945305)

[1.2 Cấu trúc các thành phần hệ thống chatbot 9](#_Toc60945306)

[1.3 Hiểu ngôn ngữ tự nhiên (NLU) 11](#_Toc60945307)

[1.3.1 Xác định ý định người dùng 13](#_Toc60945308)

[1.4 Quản lý hội thoại (DM) 14](#_Toc60945309)

[1.4.1 Mô hình máy trạng thái hữu hạn FSA 16](#_Toc60945310)

[1.4.2 Mô hình Frame-based 17](#_Toc60945311)

[1.5 Thành phần sinh ngôn ngữ (NLG) 18](#_Toc60945312)

[1.5.1 Template-based NLG 18](#_Toc60945313)

[1.5.2 Plan-based 19](#_Toc60945314)

[1.5.3 Class-based NLG 20](#_Toc60945315)

[1.6 Cấu tạo của chatbot 20](#_Toc60945316)

[1.7 Các loại chatbot 21](#_Toc60945317)

[1.8 Cách thức hoạt động 22](#_Toc60945318)

[1.9 Lợi ích và ưu điểm của chatbot 22](#_Toc60945319)

[Chương 2. Một số kỹ thuật sử dụng trong chatbot 24](#_Toc60945320)

[2.1 Kiến trúc mạng nơ ron nhân tạo 24](#_Toc60945322)

[2.2 Mạng nơ ron hồi quy RNN 29](#_Toc60945323)

[2.3 Mạng Long short Term Memory (LSTM) 31](#_Toc60945324)

[2.3.1 Vấn đề phụ thuộc quá dài 31](#_Toc60945325)

[2.3.2 Kiến trúc mạng LSTM 33](#_Toc60945326)

[2.3.3 Phân tích mô hình LSTM 34](#_Toc60945327)

[2.4 Word embeddings 37](#_Toc60945328)

[2.4.1 Word2vec 37](#_Toc60945329)

[2.4.2 Glove 39](#_Toc60945330)

[2.5 Ứng dụng RNN vào quản lý hội thoại 40](#_Toc60945331)

[2.5.1 Mô hình word-based DST 40](#_Toc60945332)

[2.5.2 Mô hình Global-Locally Self-Attentive DST (GLAD) 42](#_Toc60945333)

[Chương 3. Phân tích thiết kế hệ thống 44](#_Toc60945334)

[3.1 Khảo sát 44](#_Toc60945336)

[3.2 Các chức năng của hệ thống 44](#_Toc60945338)

[3.2.1 Yêu cầu chức năng của hệ thống 44](#_Toc60945339)

[3.2.2 Yêu cầu phi chức năng của hệ thống 44](#_Toc60945340)

[3.3 Phân tích use case 44](#_Toc60945341)

[3.3.1 Biểu đồ use case tổng quát 44](#_Toc60945342)

[3.3.2 Đặc tả use case 45](#_Toc60945343)

[Chương 4. Ứng dụng chatbot 48](#_Toc60945344)

[4.1 Ngôn ngữ lập trình Python 48](#_Toc60945346)

[4.1.1 Python là gì? 48](#_Toc60945347)

[4.2 Thư viện sử dụng 49](#_Toc60945348)

[4.2.1 Chatterbot 49](#_Toc60945349)

[4.2.2 Flask 51](#_Toc60945350)

[4.2.3 NLTK 52](#_Toc60945351)

[4.3 Chương trình ứng dụng 56](#_Toc60945353)

[4.3.1 Chương trình code 56](#_Toc60945354)

[4.3.2 Demo sản phẩm 57](#_Toc60945355)

[Kết luận 59](#_Toc60945357)

[Tài liệu tham khảo 60](#_Toc60945358)

# Danh sách biểu đồ - hình vẽ

[*Hình 1. 1: Cấu trúc các thành phần cơ bản hệ thống Chatbot* 8](#_Toc60851461)

[*Hình 1. 2: Mô hình các thành phần xử lý trong chatbot* 9](#_Toc60851462)

[*Hình 1. 3: Các bước xử lý chính trong pipeline của NLU* 10](#_Toc60851463)

[*Hình 1. 4: Mô hình các bước xác định ý định* 12](#_Toc60851464)

[*Hình 1. 5: Mô hình quản lý trạng thái và quyết định action trong hội thoại* 14](#_Toc60851465)

[*Hình 1. 6: Quản lý hội thoại theo mô hình máy trạng thái hữu hạn FSA* 15](#_Toc60851466)

[*Hình 1. 7: Frame cho chatbot hỏi thông tin khách hàng* 16](#_Toc60851467)

[*Hình 1. 8: Phương pháp sinh ngôn ngữ dựa trên tập mẫu câu trả lời* 17](#_Toc60851468)

[*Hình 1. 9: Phương pháp sinh ngôn ngữ Plan-based* 18](#_Toc60851469)

[*Hình 1. 10: Phương pháp sinh ngôn ngữ class-based* 19](#_Toc60851470)

[*Hình 2. 1: Kiến trúc mạng nơ ron nhân tạo* 23](#_Toc60851471)

[*Hình 2. 2: Quá trình xử lý thông tin của một mạng nơ-ron nhân tạo* 25](#_Toc60851472)

[*Hình 2. 3: Mạng RNN* 28](#_Toc60851473)

[*Hình 2. 4: Mạng RNN 2 chiều* 29](#_Toc60851474)

[*Hình 2. 5: Mạng RNN nhiều tầng* 30](#_Toc60851475)

[*Hình 2. 6: RNN phụ thuộc short-term* 31](#_Toc60851476)

[*Hình 2. 7: RNN phụ thuộc long-term* 31](#_Toc60851477)

[*Hình 2. 8: Các mô-đun lặp của mạng RNN chứa một layer* 32](#_Toc60851478)

[*Hình 2. 9: Các mô-đun lặp của mạng LSTM chứa bốn layer* 32](#_Toc60851479)

[*Hình 2. 10: Tế bào trạng thái LSTM giống như một băng truyền* 33](#_Toc60851480)

[*Hình 2. 11: Cổng trạng thái LSTM* 34](#_Toc60851481)

[*Hình 2. 12: LSTM focus f* 34](#_Toc60851482)

[*Hình 2. 13: LSTM focus I* 35](#_Toc60851483)

[*Hình 2. 14: LSTM focus c* 35](#_Toc60851484)

[*Hình 2. 15: LSTM focus o* 36](#_Toc60851485)

[*Hình 2. 16: Mô hình từ nhúng* 37](#_Toc60851486)

[*Hình 2. 17: Mô hình CBOW và Skip-Ngram* 37](#_Toc60851487)

[*Hình 2. 18: Mô hình word-based DST với mạng RNN* 40](#_Toc60851488)

[*Hình 2. 19: Mô hình Global-Locally Self-Attentive DST (GLAD)* 41](#_Toc60851489)

[*Hình 2. 20: Global-locally self-attentive encoder modul* 42](#_Toc60851490)

# Mở đầu

Ở nước ta, việc giải đáp thắc mắc của bộ phận chăm sóc khách hàng qua tin nhắn trực tuyến đang được ưa chuộng. Tuy nhiên, việc này còn thực hiện một cách thủ công và gặp nhiều khó khăn như: tốn rất nhiều thời gian và chi phí chi trả cho nhân viên chỉ để trả lời những câu hỏi đơn giản và giống nhau. Chính vì vậy, nhu cầu cấp thiết là cần một hệ thống điều khiển thông minh, tự động để mang lại hiệu quả cao hơn và Chatbot là một sự lựa chọn hoàn hảo.

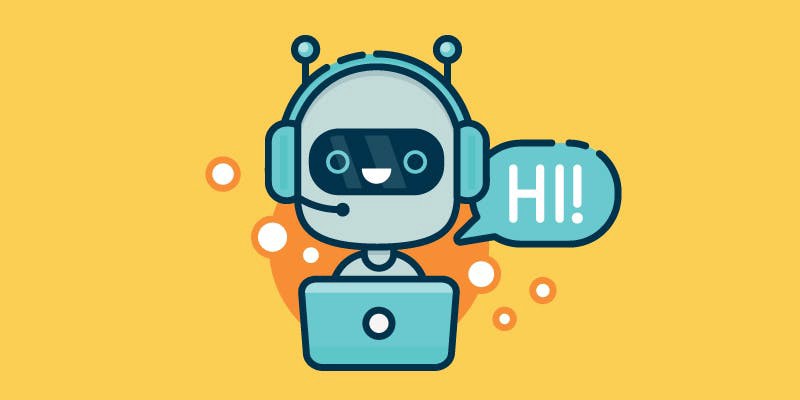
Hiện nay, các ứng dụng trò chuyện trực tuyến được mọi người sử dụng đang bắt đầu trở thành một phương tiện ưa thích để giao tiếp với các doanh nghiệp và giải quyết thắc mắc của khách hàng. Ứng dụng nhắn tin nhanh đã trở thành điểm đến hàng đầu cho mọi thương hiệu nhằm tiếp cận người tiêu dùng, bởi vậy không có gì đáng ngạc nhiên khi Chatbot ngày càng trở nên phổ biến.

Hiện nay, chúng ta đang ở trên đỉnh của cuộc cách mạng 4.0, chatbot đang rất quan trọng đối với nền văn hóa nhân loại. Chatbot bây giờ có mặt ở khắp mọi các lĩnh vực thương mại điện tử, ngân hàng, y tế, giáo dục, thư viện,… Vì vậy nhóm chúng em chọn và tìm hiểu đề tài bài tập lớn *“Hệ thống chatbot và ứng dụng trong đào tạo trực tuyến”.*

# Chương 1. Tổng quan về hệ thống Chatbot

## Giới thiệu

Hệ thống đối thoại người máy hay còn gọi với thuật ngữ là chatbot. ChatBot là một chương trình máy tính tiến hành cuộc trò chuyện thông qua nhắn tin nhanh, nó có thể tự động trả lời những câu hỏi hoặc xử lý tình huống. Phạm vi và sự phức tạp của ChatBot được xác định bởi thuật toán của người tạo nên chúng. ChatBot thường được ứng dụng trong nhiều lĩnh vực như thương mại điện tử, dịch vụ khách hàng, y tế, tài chính ngân hàng, các dịch vụ giải trí, ứng dụng giáo dục,…



Chatbot có thể được chia thành 2 loại:

- Hệ thống hướng mục tiêu trên một miền ứng dụng (Task-Oriented)

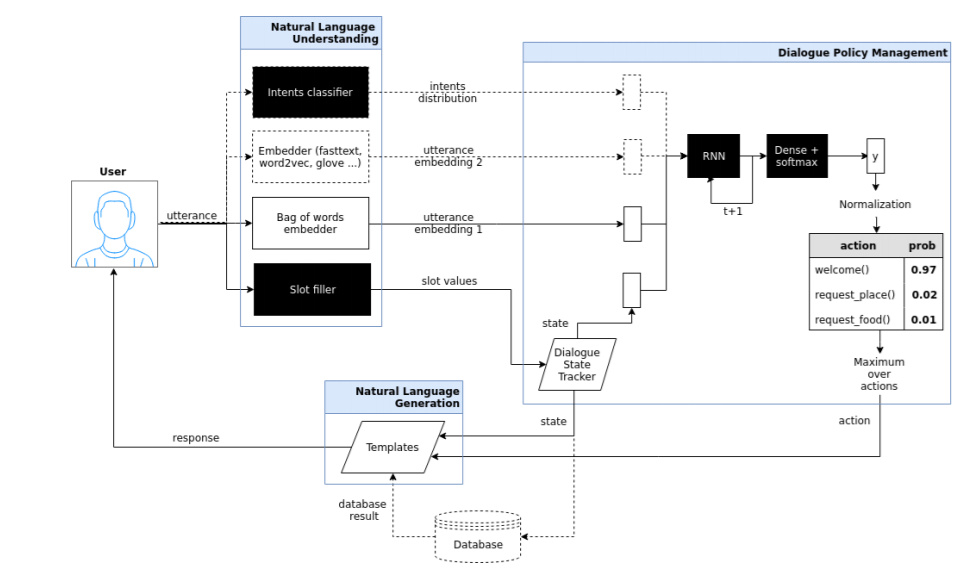
- Hệ thống không có định hướng mục tiêu (chit-chat)

Miền mở (Open Domain): Mô hình trả lời tự động trên miền mở cho phép người dùng có thể tham gia trò chuyện với một chủ đề bất kỳ, không nhất thiết phải có một mục tiêu rõ ràng hay một ý định cụ thể nào. Các cuộc trò chuyện trên mạng xã hội như Facebook, Twitter thường là miền mở, chúng có thể đi vào tất cả các chủ đề. Số lượng các chủ đề thảo luận được đề cập đến là không giới hạn, do đó, tri thức yêu cầu được tạo ra để trả lời các câu đối thoại thuộc miền mở trở nên khó hơn. Tuy nhiên, việc thu thập trích rút dữ liệu từ miền này khá phong phú và đơn giản.

Miền đóng (Close Domain): Mô hình trả lời tự động thuộc miền đóng thường tập trung vào trả lời các câu hỏi đối thoại liên quan đến một miền cụ thể, ví dụ như: Y tế, Giáo dục, Du lịch, Mua sắm, ..

Trong một miền đóng cụ thể, không gian các mẫu hỏi input và output là có giới hạn, bởi vì các hệ thống này đang cố gắng để đạt được một mục tiêu rất cụ thể. Hệ thống hỗ trợ kỹ thuật (Technical Customer Support) hay tư vấn và hỗ trợ mua hàng (Shopping Assistants) là các ứng dụng thuộc miền đóng. Các hệ thống này không thể đối thoại về “Chính trị” hay “Pháp luật”, chúng chỉ cần thực hiện các nhiệm vụ cụ thể một cách hiệu quả nhất có thể. Chắc chắn, người dùng vẫn có thể hỏi đáp bất cứ gì, nhưng hệ thống không yêu cầu phải xử lý những trường hợp ngoại lệ này.

## Cấu trúc các thành phần hệ thống chatbot



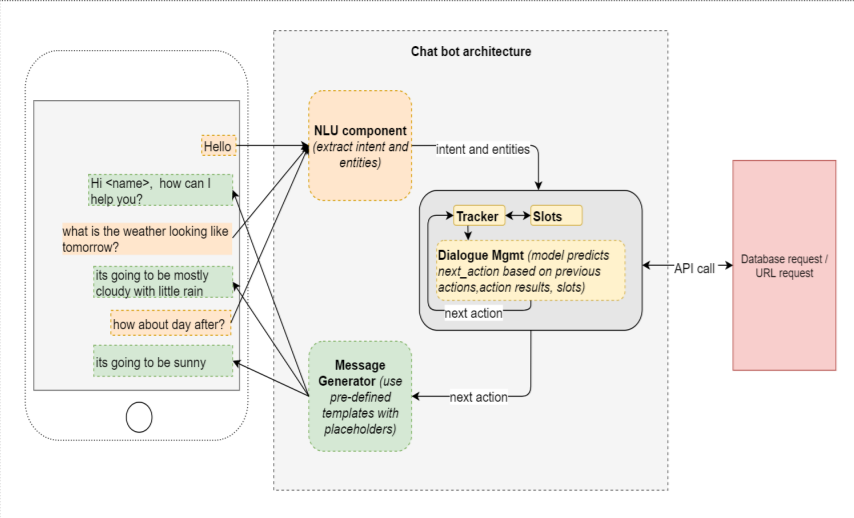
*Hình 1. 1: Cấu trúc các thành phần cơ bản hệ thống Chatbot*

Chatbot có 3 thành phần chính là hiểu ngôn ngữ tự nhiên (NLU), quản lý hội thoại (DM), thành phần sinh ngôn ngữ (NLG). Các thành phần nhận dạng giọng nói Speech Recognition (text to speech hay speech to text) là các thành phần tăng cường.

Mỗi thành phần trong chatbot đều có vai trò riêng:

* NLU: bao gồm việc xử lý ngôn ngữ tự nhiên (NLP) có nhiệm vụ xác định được ý định câu hỏi(intent classification) và trích chọn thông tin (slots filter).
* DM: Quản lý hội thoại có nhiệm vụ xác định được hành động (action) tiếp theo dựa vào trạng thái hành động trước đó hay ngữ cảnh hội thoại. Các ngữ cảnh này phải được đối chiếu trong các kịch bản dựng sẵn (history) đã đào tạo cho bot. Thành phần này cũng đảm nhiệm việc lấy dữ liệu từ hệ thống khác qua các API gọi trong action.
* NLG: là thành phần sinh ngôn ngữ dựa vào chính sách (policy) và hành động được xác định trong DM thông qua các tập hội thoại. NGL có thể được sinh ra 7 câu trả lời dựa vào tập mẫu câu trả lời (pre-defined template) đã đào tạo cho bot.

Để rõ chi tiết luồng xử lý tin nhắn từ các thành phần chatbot ta xem mô hình sau:



*Hình 1. 2: Mô hình các thành phần xử lý trong chatbot*

## Hiểu ngôn ngữ tự nhiên (NLU)

Đây có thể nói là thành phần quan trọng nhất của chatbot. Chatbot có thông minh hay không thì đây là thành phần quyết định. Mục tiêu của thành phần này là trích xuất ra 3 thành phần thông tin từ câu nói của người dùng:

* Đầu tiên là phân loại lĩnh vực (domain classification), nó có thể là lĩnh vực ngân hàng, y tế hay bảo hiểm. Nếu trong trường hợp chỉ có một lĩnh vực thì không cần thiết cho bước phân loại này.
* Tiếp đến là phân loại ý định (intent classification), ví dụ như xác định được ý định tra cứu thông tin tài khoản hoặc ý định kiểm tra số dư.
* Cuối cùng là bước trích xuất thông tin (slot fillter hay entity extraction) trong câu hỏi người dùng.
* NLU xử lý tin nhắn người dùng bằng một đường ống (pipeline) nơi mà cấu hình các bước xử lý liên tiếp theo tuần tự :

*Hình 1. 3: Các bước xử lý chính trong pipeline của NLU*

Trong đường ống này thì bạn có thể tùy chỉnh các thành phần từ bước tiền xử lý dữ liệu, mô hình hóa ngôn ngữ, các thuật toán dùng để tách từ và trích xuất thông tin thực thể…

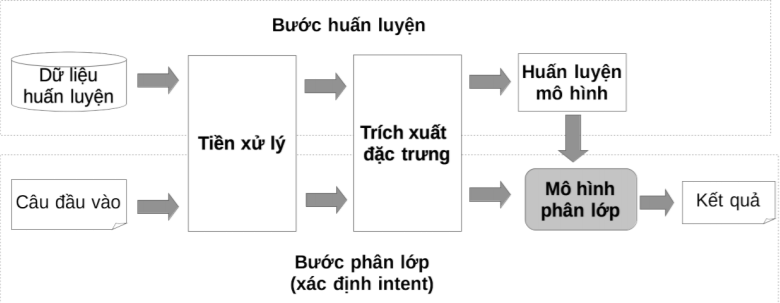
Để phân loại được ý định câu người dùng thì ta cần mô hình hóa ngôn ngữ tức là việc biểu diễn ngôn ngữ dưới dạng vector số học cho máy có thể hiểu được (vectorization). Phương pháp phổ biến nhất hiện tại là word embedding (nhúng từ). Tập nhúng từ là tên chung cho một tập hợp các mô hình ngôn ngữ và các phương pháp học đặc trưng trong xử lý ngôn ngữ tự nhiên (NLP), nơi các từ hoặc cụm từ từ vựng được ánh xạ tới vectơ số thực. Một số phương pháp biểu diễn phổ biến như Word2Vec, GloVe hay mới hơn là FastText sẽ được giới thiệu trong phần sau.

Sau khi mô hình hóa ngôn ngữ bao gồm dữ liệu đầu vào training cho bot thì việc xác định ý định người dùng từ câu hỏi người dùng dựa trên tập đã training là bước phân loại ý định (intent classification) hay phân loại văn bản. Ở bước này ta có thể dùng một số kỹ thuật như: Naive Bayes, Decision Tree (Random Forest), Vector Support Machine (SVM), Convolution Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM, Bi-LSTM). Hầu hết các chatbot hiện tại đều ứng dụng mô hình deep learning như RNN và LSTM để phân loại ý định người dùng. Bài toán thách thức lớn nhất cho các chatbot ở bước này là xác định nhiều ý định (multiple intents) trong một câu nói người dùng. Ví dụ nếu bạn nói “xin chào, mã sinh viên 2017605551 tên là gì” thì bot phải xác định được 2 ý định “chào hỏi” và “kiểm tra tên sinh viên” trong câu nói người dùng. Nếu bot có thể hiểu và trả lời được câu hỏi loại này sẽ giúp việc tương tác với bot trở nên tư nhiên hơn.

Tiếp đến là việc trích xuất thông tin trong câu hội thoại người dùng. Các thông tin cần trích xuất thường dưới dạng số, chuỗi hoặc thời gian và chúng phải được khai báo và huấn luyện trước.

Phân tách các từ (Tokenization hay word segmention): Tách từ là một quá trình xử lý nhằm mục đích xác định ranh giới của các từ trong câu văn, cũng có thể hiểu đơn giản rằng tách từ là quá trình xác định các từ đơn, từ ghép… có trong câu. Đối với xử lý ngôn ngữ, để có thể xác định cấu trúc ngữ pháp của câu, xác định từ loại của một từ trong câu, yêu cầu nhất thiết đặt ra là phải xác định được đâu là từ trong câu. Có một số thuật toán hỗ trợ giải quyết bài toán này như mô hình so khớp từ dài nhất (longest matching), so khớp cực đại (Maximum Matching), Markov ẩn (Hidden Markov Models- HMM) hay mô hình CRF (conditinal random field)…

### Xác định ý định người dùng



*Hình 1. 4: Mô hình các bước xác định ý định*

Hệ thống phân lớp ý định người dùng có một số bước cơ bản:

* Tiền xử lý dữ liệu
* Trích xuất đặc trưng
* Huấn luyện mô hình
* Phân lớp

Bước tiền xử lý dữ liệu chính là thao tác “làm sạch” dữ liệu như: loại bỏ các thông tin dư thừa, chuẩn hoá dữ liệu và chuyển các từ viết sai chính tả thành đúng chính tả, chuẩn hoá các từ viết tắt… Bước tiền xử lý dữ liệu có vai trò quan trọng trong hệ thống chatbot. Nếu dữ liệu đầu vào có xử lý ở bước này thì sẽ làm tăng khả năng năng độ chính xác cũng như sự thông minh cho bot.

Tiếp đến là bước trích xuất đặc trưng (feature extraction hay feature engineering) từ những dữ liệu đã được làm sạch. Trong mô hình học máy truyền thống (trước khi mô hình học sâu được áp dụng rộng rãi), bước trích xuất đặc trưng ảnh hưởng lớn đến độ chính xác của mô hình phân lớp. Để trích xuất được những đặc trưng tốt, chúng ta cần phân tích dữ liệu khá tỉ mỉ và cần cả những tri thức chuyên gia trong từng miền ứng dụng cụ thể.

Bước huấn luyện mô hình nhận đầu vào là các đặc trưng đã được trích xuất và áp dụng các thuật toán học máy để học ra một mô hình phân lớp. Các mô hình phân lớp có thể là các luật phân lớp (nếu sử dụng decision tree) hoặc là các vector trọng số tương ứng với các đặc trưng được trích xuất (như trong các mô hình logistic regression, SVM, hay mạng Neural).

Sau khi có một mô hình phân lớp intent, chúng ta có thể sử dụng nó để phân lớp một câu hội thoại mới. Câu hội thoại này cũng đi qua các bước tiền xử lý và trích xuất đặc trưng, sau đó mô hình phân lớp sẽ xác định “điểm số” cho từng intent trong tập các intent và đưa ra intent có điểm cao nhất

Để đưa ra hỗ trợ được chính xác, chatbot cần xác định được ý định (intent) đó của người dùng. Việc xác định ý định của người dùng sẽ quyết định hội thoại tiếp theo giữa người và chatbot sẽ diễn ra như thế nào. Vì thế, nếu xác định sai ý định người dùng, chatbot sẽ đưa ra những phản hồi không đúng, không hợp ngữ cảnh. Khi đó, người dùng có thể thấy chán ghét và không quay lại sử dụng hệ thống. Bài toán xác định ý định người dùng vì thế đóng vai trò rất quan trọng trong hệ thống chatbot.

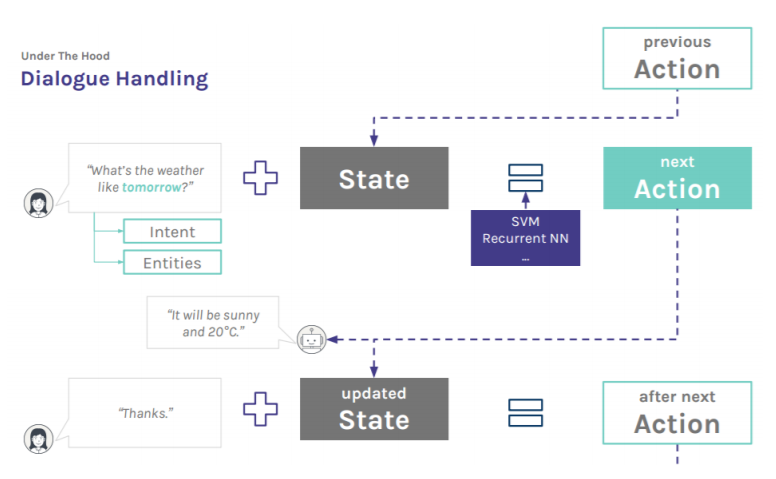
Đối với miền ứng dụng đóng, chúng ta có thể giới hạn số lượng ý định của người dùng nằm trong một tập hữu hạn những ý định đã được định nghĩa sẵn, có liên quan đến những nghiệp vụ mà chatbot có thể hỗ trợ. Với giới hạn này, bài toán xác định ý định người dùng có thể quy về bài toán phân lớp văn bản. Với đầu vào là một câu giao tiếp của người dùng, hệ thống phân lớp sẽ xác định ý định tương ứng với câu đó trong tập các intent đã được định nghĩa trước.

Để xây dựng một mô hình phân lớp intent, chúng ta cần một tập dữ liệu huấn luyện bao gồm các cách diễn đạt khác nhau cho mỗi intent.

## Quản lý hội thoại (DM)

Trong các phiên trao đổi dài (long conversation) giữa người và chatbot, chatbot sẽ cần ghi nhớ những thông tin về ngữ cảnh (context) hay quản lý các trạng thái hội thoại (dialog state). Vấn đề quản lý hội thoại (dialoge management) khi đó là quan trọng để đảm bảo việc trao đổi giữa người và máy là thông suốt.

Chức năng của thành phần quản lý hội thoại là nhận đầu vào từ thành phần NLU, quản lý các trạng thái hội thoại (dialogue state), ngữ cảnh hội thoại (dialogue context), và truyền đầu ra cho thành phần sinh ngôn ngữ (Natural Language Generation, viết tắt là NLG).

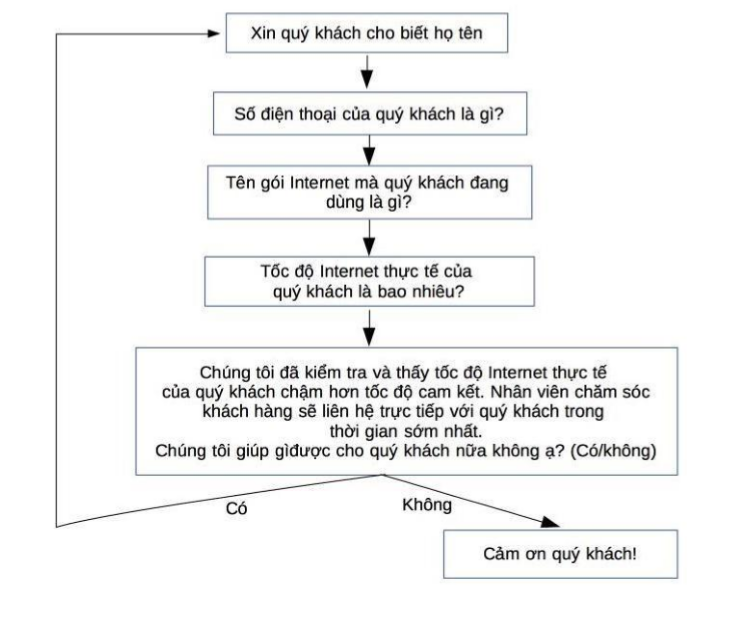


*Hình 1. 5: Mô hình quản lý trạng thái và quyết định action trong hội thoại*

Trạng thái hội thoại (dialog state) được lưu lại và dựa vào tập luật hội thoại (dialog policy) để quyết định hành động tiếp theo cho câu trả lời của bot trong một kịch bản hội thoại, hay hành động (action) chỉ phụ thuộc vào trạng thái (dialog state) trước của nó.

Ví dụ module quản lý dialogue trong một chatbot phục vụ đặt vé máy bay cần biết khi nào người dùng đã cung cấp đủ thông tin cho việc đặt vé để tạo một ticket tới hệ thống hoặc khi nào cần phải xác nhận lại thông tin do người dùng đưa vào. Hiện nay, các sản phẩm chatbot thường dùng mô hình máy trạng thái hữu hạn (Finite State Automata – FSA), mô hình Frame-based (Slot Filling), hoặc kết hợp hai mô hình này. Một số hướng nghiên cứu mới có áp dụng mô hình ANN vào việc quản lý hội thoại giúp bot thông minh hơn.

### Mô hình máy trạng thái hữu hạn FSA



*Hình 1. 6: Quản lý hội thoại theo mô hình máy trạng thái hữu hạn FSA*

Mô hình FSA quản lý hội thoại đơn giản nhất. Ví dụ hệ thống chăm sóc khách hàng của một công ty viễn thông, phục vụ cho những khách hàng than phiền về vấn đề mạng chậm. Nhiệm vụ của chatbot là hỏi tên khách hàng, số điện thoại, tên gói Internet khách hàng đang dùng, tốc độ Internet thực tế của khách hàng. Hình vẽ minh hoạ một mô hình quản lý hội thoại cho chatbot chăm sóc khách hàng. Các trạng thái của FSA tương ứng với các câu hỏi mà dialogue manager hỏi người dùng. Các cung nối giữa các trạng thái tương ứng với các hành động của chatbot sẽ thực hiện. Các hành động này phụ thuộc phản hồi của người dùng cho các câu hỏi. Trong mô hình FSA, chatbot là phía định hướng người sử dụng trong cuộc hội thoại.

Ưu điểm của mô hình FSA là đơn giản và chatbot sẽ định trước dạng câu trả lời mong muốn từ phía người dùng. Tuy nhiên, mô hình FSA không thực sự phù hợp cho các hệ thống chatbot phức tạp hoặc khi người dùng đưa ra nhiều thông tin khác nhau trong cùng một câu hội thoại. Trong ví dụ chatbot ở trên, khi người dùng đồng thời cung cấp cả tên và số điện thoại, nếu chatbot tiếp tục hỏi số điện thoại, người dùng có thể cảm thấy khó chịu.

### Mô hình Frame-based

Mô hình Frame-based (hoặc tên khác là Form-based) có thể giải quyết vấn đề mà mô hình FSA gặp phải. Mô hình Frame-based dựa trên các frame định sẵn để định hướng cuộc hội thoại. Mỗi frame sẽ bao gồm các thông tin (slot) cần điền và các câu hỏi tương ứng mà dialogue manager hỏi người dùng. Mô hình này cho phép người dùng điền thông tin vào nhiều slot khác nhau trong frame. Hình vẽ là một ví dụ về một frame cho chatbot ở trên.

|  |  |
| --- | --- |
| **Slot** | **Câu hỏi** |
| Họ tên | Xin quý khách cho biết họ tên |
| Số điện thoại | Số điện thoại của quý khách là gì? |
| Tên gói Internet | Gói Internet quý khách đang dùng là gì ạ? |
| Tốc độ Internet thực tế | Tốc độ Internet của quý khách hiện tại là bao nhiêu ạ? |

*Hình 1. 7: Frame cho chatbot hỏi thông tin khách hàng*

Thành phần quản lý dialogue theo mô hình Frame-based sẽ đưa ra câu hỏi cho khách hàng, điền thông tin vào các slot dựa trên thông tin khách hàng cung cấp cho đến khi có đủ thông tin cần thiết. Khi người dùng trả lời nhiều câu hỏi cùng lúc, hệ thống sẽ phải điền vào các slot tương ứng và ghi nhớ để không hỏi lại những câu hỏi đã có câu trả lời.

Trong các miền ứng dụng phức tạp, một cuộc hội thoại có thể có nhiều frame khác nhau. Vấn đề đặt ra cho người phát triển chatbot khi đó là làm sao để biết khi nào cần chuyển đổi giữa các frame. Cách tiếp cận thường dùng để quản lý việc chuyển điều khiển giữa các frame là định nghĩa các luật (production rule). Các luật này dựa trên một số các thành tố như câu hội thoại hoặc câu hỏi gần nhất mà người dùng đưa ra.

## Thành phần sinh ngôn ngữ (NLG)

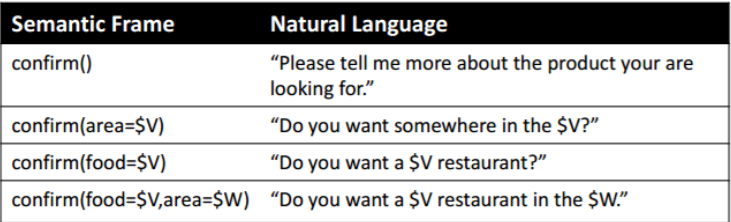
NLG là thành phần sinh câu trả lời của chatbot. Nó dựa vào việc ánh xạ các hành động của quản lý hội thoại vào ngôn ngữ tự nhiên để trả lời người dùng.



Có 4 phương pháp ánh xạ hay dùng là: Template-Base, Plan-based, Class-base, RNN-base.

### Template-based NLG

Phương pháp ánh xạ câu trả lời này là dùng những câu mẫu trả lời của bot đã được định nghĩa từ trước để sinh câu trả lời

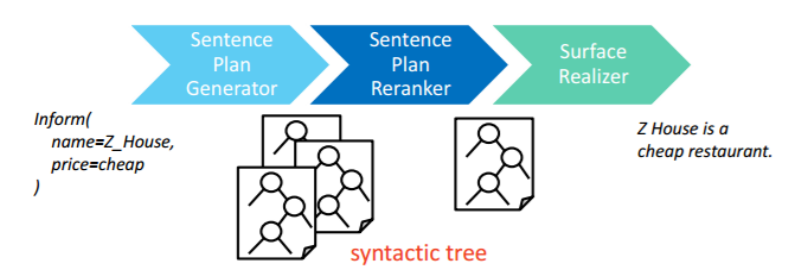


*Hình 1. 8: Phương pháp sinh ngôn ngữ dựa trên tập mẫu câu trả lời*

**Ưu điểm**: đơn giản, kiểm soát dễ dàng. Phù hợp cho các bài toán miền đóng.

**Nhược điểm**: tốn thời gian định nghĩa các luật, không mang tính tự nhiên trong câu trả lời. Đối với các hệ thống lớn thì khó kiểm soát các luật dẫn đến hệ thống cũng khó phát triển và duy trì.

### Plan-based

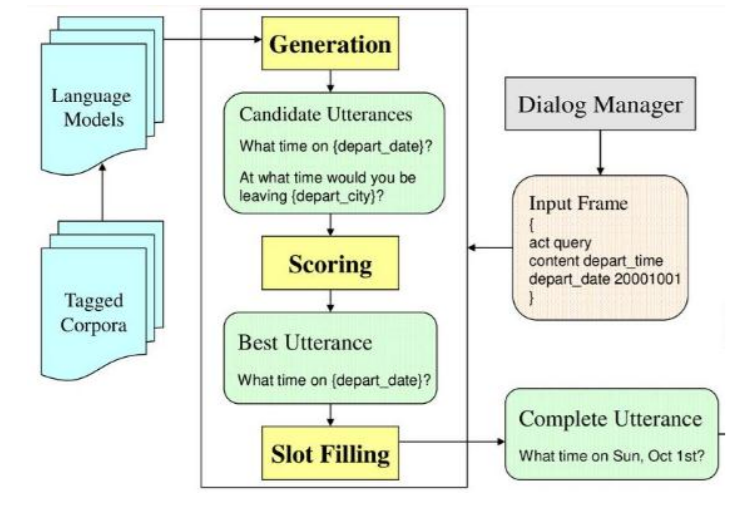


*Hình 1. 9: Phương pháp sinh ngôn ngữ Plan-based*

**Ưu điểm**: Có thể mô hình hóa cấu trúc ngôn ngữ phức tạp

**Nhược điểm**: Thiết kế nặng nề, đòi hỏi phải rõ miền kiến thức

### Class-based NLG



*Hình 1. 10: Phương pháp sinh ngôn ngữ class-based*

Phương pháp này dựa trên việc cho bot học những câu trả lời đầu vào đã được gán nhãn. Ứng với các hành động (action) và thông tin(slot) từ quản lý hội thoại thì bot sẽ đưa ra câu trả lời gần nhất dựa trên tập dữ liệu trả lời được đào tạo trước đó.

Ưu điểm: dễ dàng thực thi

Nhược điểm: phụ thuộc vào dữ liệu trả lời đã được gán nhãn đào tạo trước đó. Bên cạnh đó việc tính toán điểm số không hiệu quả cũng dân đến việc sinh câu trả lời sai

## Cấu tạo của chatbot

Cấu tạo cơ bản của ChatBot gồm có ba phần bao gồm: cơ sở dữ liệu, lớp ứng dụng, quyền truy cập vào các API và giao diện đồ họa người dùng. Nhờ những thành phần cơ bản mà nó có thể hoạt động được.

* Cơ sở dữ liệu: Cơ sở dữ liệu lưu trữ các thông tin, dữ liệu và nội dung.
* Tầng ứng dụng: Các giao thức của tầng ứng dụng thường được dùng để trao đổi dữ liệu giữa các chương trình chạy trên máy nguồn và máy đích. Tầng này đóng vai trò như cửa sổ dành cho hoạt động xử lý các trình ứng dụng, nó biểu diễn những dịch vụ hỗ trợ trực tiếp các ứng dụng người dùng, chẳng hạn như phần mềm chuyển tin, truy nhập cơ sở dữ liệu và email.v.v…
* Giao diện lập trình ứng dụng (API): là một giao diện mà một hệ thống máy tính hay ứng dụng cung cấp để cho phép các yêu cầu dịch vụ có thể được tạo ra từ các chương trình máy tính khác, và/hoặc cho phép dữ liệu có thể được trao đổi qua lại giữa chúng.

## Các loại chatbot

ChatBot viết kịch bản (Scripted chatbot): Đây là những ChatBot có hành vi được xác định bởi các quy tắc. Tại mỗi bước trong cuộc trò chuyện, người dùng sẽ cần chọn các tùy chọn rõ ràng để xác định bước tiếp theo trong cuộc trò chuyện. Các cách tùy chọn được trình bày cho người dùng ở mỗi bước trong cuộc hội thoại, tức là liệu họ cần phản hồi bằng một văn bản, giọng nói hoặc cảm ứng thì sẽ phụ thuộc vào các tính năng của nền tảng trò chuyện mà người dùng đang sử dụng và thiết kế của ChatBot.

ChatBot thông minh (Intelligent ChatBot): ChatBot thông minh là ChatBot được xây dựng với các kỹ thuật nhân tạo trí thông minh. Trí tuệ nhân tạo (AI) cho phép chúng cải thiện linh hoạt hơn về đầu vào người dùng mà chúng có thể thu nhận. Chúng có thể thu nhận đầu vào dạng tự do dưới hình thức trình bày bằng văn bản hoặc giọng nói và cũng không giới hạn các dạng đầu vào khác nếu nó có ý nghĩa.Ví dụ: Khi bạn muốn mua quần áo, Bot sẽ tự đông hỏi bạn một danh sách câu hỏi về phong cách quần áo ưa thích của bạn và sẽ đưa ra mẫu gợi ý và giá của mặt hàng.

## Cách thức hoạt động

ChatBot là sự kết hợp của các kịch bản có trước và tự học trong quá trình tương tác. Ta sẽ tương tác với ChatBot qua nền tảng tin nhắn. Với các câu hỏi được đặt ra,ChatBot sử dụng các hệ thống xử lý ngôn ngữ tự nhiên (Natural Language Processing) để phân tích dữ liệu sau đó chúng lựa chọn các thuật toán học máy để tạo ra các loại phản ứng khác nhau, chúng sẽ dự đoán và phản hồi chính xác nhất có thể.

ChatBot sử dụng nhiều hệ thống quét các từ khoá bên trong đầu 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 / API, hoặc bàn giao cho con người.

Nếu tình huống đó chưa xảy ra (không có trong dữ liệu), ChatBot sẽ bỏ qua nhưng sẽ đồng thời 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 (đó được gọi là phương pháp máy học-Machine Learning).

## Lợi ích và ưu điểm của chatbot

**Lợi ích của chatbot đối với người dùng:**

* Tiết kiệm thời gian
* Tiết kiệm chi phí thuê nhân lực
* Tự động gợi ý và tìm kiếm những thông tin liên quan đến sản phẩm khi khách hàng cần.
* Người dùng có thể tương tác với ChatBot bất cứ lúc nào.

**Ưu điểm của chatbot:**

* Độ tin cậy cao.
* Chatbot có thể có lợi thế hơn các tác nhân con người, chúng có thể vượt trội con người về mặt tốc độ và độ chính xác.
* Bất cứ ngôn ngữ lập trình nào cũng có thể sử dụng để phát triển nên ChatBot.
* Người dùng không có xu hướng tải ứng dụng vì nó sẽ làm nặng máy nên ChatBot sẽ dễ dàng tiếp cận được người dùng.
* ChatBot thay thế số lượng lớn về nhân lực.
* Tự tích lũy kiến thức nhận được và trở nên thông minh, linh hoạt hơn sau mỗi cuộc trò chuyện.
* Kỹ thuật thông minh hơn và luôn có sẵn 24 giờ, 7 ngày một tuần.

**Nhược điểm của chatbot:**

* Có rất nhiều nền tảng, với mức độ trưởng thành và ổn định khác nhau.
* Đôi khi GUI không quá dễ hiểu và khi logic chatbot trở nên phức tạp hơn, nó trở nên khó xử lý.
* Họ có ít hoặc không có khả năng xử lý ngôn ngữ tự nhiên. Ví dụ, một số nền tảng không thể thực hiện khai thác thông tin. Do đó, được đưa ra một cụm từ như là tôi ở Boston, họ không thể rút ra được thực tế là thành phố Boston (thực thể địa điểm) xảy ra.
* Chúng dường như không thích hợp cho các bot phức tạp.

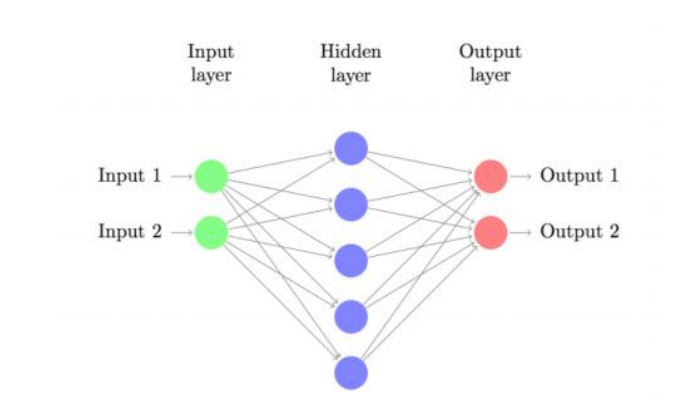
# Chương 2. Một số kỹ thuật sử dụng trong chatbot



## Kiến trúc mạng nơ ron nhân tạo

Mạng nơ ron nhân tạo (Artificial Neural Network – ANN) là một mô hình xử lý thông tin được mô phỏng dựa trên hoạt động của hệ thống thần kinh của sinh vật, bao gồm số lượng lớn các Nơ-ron được gắn kết để xử lý thông tin. ANN hoạt động giống như bộ não của con người, được học bởi kinh nghiệm (thông qua việc huấn luyện), có khả năng lưu giữ các tri thức và sử dụng các tri thức đó trong việc dự đoán các dữ liệu chưa biết (unseen data).

Một mạng nơ-ron là một nhóm các nút nối với nhau, mô phỏng mạng nơ-ron thần kinh của não người. Mạng nơ ron nhân tạo được thể hiện thông qua ba thành phần cơ bản: mô hình của nơ ron, cấu trúc và sự liên kết giữa các nơ ron. Trong nhiều trường hợp, mạng nơ ron nhân tạo là một hệ thống thích ứng, tự thay đổi cấu trúc của mình dựa trên các thông tin bên ngoài hay bên trong chạy qua mạng trong quá trình học.



*Hình 2. 1: Kiến trúc mạng nơ ron nhân tạo*

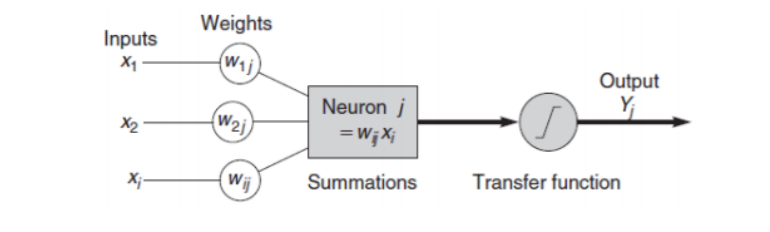
Kiến trúc chung của một ANN gồm 3 thành phần đó là Input Layer, Hidden Layer và Output Layer

Trong đó, lớp ẩn (Hidden Layer) gồm các nơ-ron, nhận dữ liệu input từ các Nơron ở lớp (Layer) trước đó và chuyển đổi các input này cho các lớp xử lý tiếp theo. Trong một mạng ANN có thể có nhiều Hidden Layer.

Lợi thế lớn nhất của các mạng ANN là khả năng được sử dụng như một cơ chế xấp xỉ hàm tùy ý mà “học” được từ các dữ liệu quan sát. Tuy nhiên, sử dụng chúng không đơn giản như vậy, một số các đặc tính và kinh nghiệm khi thiết kế một mạng nơ-ron ANN.

Phương pháp này là tính toán tỷ lệ chính xác dữ liệu đầu ra (output) từ dữ liệu đầu vào (input) bằng cách tính toán các trọng số cho mỗi kết nối (connection) từ các lần lặp lại trong khi “huấn luyện” dữ liệu cho Chatbot. Mỗi bước “huấn luyện” dữ liệu cho Chatbot sẽ sửa đổi các trọng số dẫn đến dữ liệu output được xuất ra với độ chính xác cao.

* Chọn mô hình: Điều này phụ thuộc vào cách trình bày dữ liệu và các ứng dụng. Mô hình quá phức tạp có xu hướng dẫn đền những thách thức trong quá trình học.
* Cấu trúc và sự liên kết giữa các nơ-ron.
* Thuật toán học: Có hai vấn đề cần học đối với mỗi mạng ANN, đó là học tham số của mô hình (parameter learning) và học cấu trúc (structure learning). Học tham số là thay đổi trọng số của các liên kết giữa các nơ-ron trong một mạng, còn học cấu trúc là việc điều chỉnh cấu trúc mạng bằng việc thay đổi số lớp ẩn, số nơ-ron mỗi lớp và cách liên kết giữa chúng. Hai vấn đề này có thể được thực hiện đồng thời hoặc tách biệt. Nếu các mô hình, hàm chi phí và thuật toán học được lựa chọn một cách thích hợp, thì mạng ANN sẽ cho kết quả có thể vô cùng mạnh mẽ và hiệu quả.



*Hình 2. 2: Quá trình xử lý thông tin của một mạng nơ-ron nhân tạo*

**Inputs:** Mỗi Input tương ứng với 1 đặc trưng của dữ liệu. Ví dụ như trong ứng dụng của ngân hàng xem xét có chấp nhận cho khách hàng vay tiền hay không thì mỗi input là một thuộc tính của khách hàng như thu nhập, nghề nghiệp, tuổi, số con,…

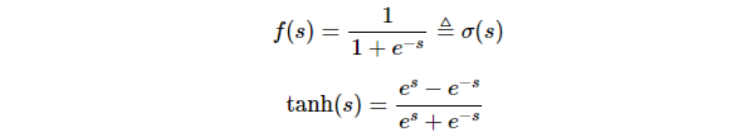
**Output:** Kết quả của một ANN là một giải pháp cho một vấn đề.

**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 chuyển đổi dữ liệu từ Layer này sang layer khác. Quá trình học của ANN thực ra là quá trình điều chỉnh các trọng số Weight của các dữ liệu đầu vào 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. 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 của nơ-ron đó còn gọi là kích hoạt bên trong. Các nơ-ron này có thể sinh ra một output hoặc không trong mạng ANN, nói cách khác rằng có thể output của 1 Nơ-ron có thể được chuyển đến layer tiếp trong mạng Nơ-ron theo hoặc không. Mối quan hệ giữa hàm tổng và kết quả output được thể hiện bằng hàm chuyển đổi.

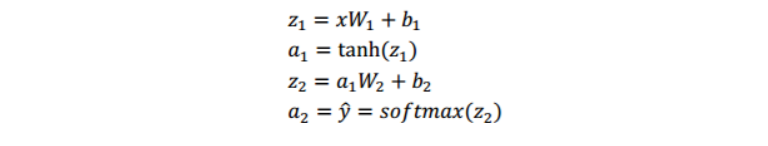
Việc lựa chọn hàm chuyển đổi có tác động lớn đến kết quả đầu ra của mạng ANN. Hàm chuyển đổi phi tuyến được sử dụng phổ biến trong mạng ANN là sigmoid hoặc tanh.



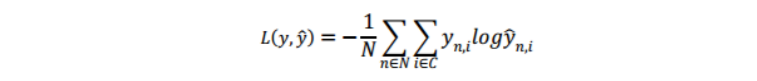
Trong đó, hàm tanh là phiên bản thay đổi tỉ lệ của sigmoid , tức là khoảng giá trị đầu ra của hàm chuyển đổi thuộc khoảng [-1, 1] thay vì [0,1] của Sigmoid nên chúng còn gọi là hàm chuẩn hóa (Normalized Function).

Kết quả xử lý tại các nơ-ron (Output) đôi khi rất lớn, vì vậy hàm chuyển đổi 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 neuron tại một layer nào đó trước khi chuyển các output này đến các Layer tiếp theo. Nếu output của một neuron nào đó nhỏ hơn Threshold thì nó sẻ không được chuyển đến Layer tiếp theo.

Mạng nơ-ron của chúng ta dự đoán dựa trên lan truyền thẳng (forward propagation) là các phép nhân ma trận cùng với activation function để thu được kết quả đầu ra. Nếu input x là vector 2 chiều thì ta có thể tính kết quả dự đoán 𝑦 bằng công thức :



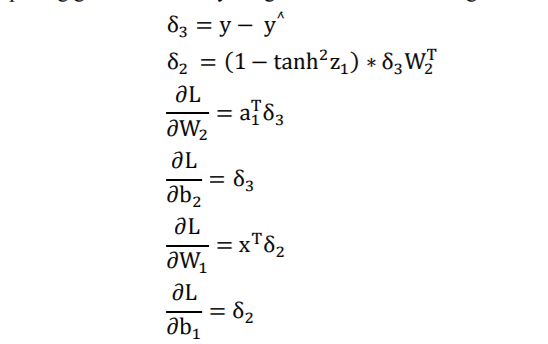
Trong đó, 𝑧𝑖 là input của layer thứ 𝑖, 𝑎𝑖 là output của layer thứ 𝑖 sau khi áp dụng activation function. 𝑊1, 𝑏1, 𝑊2, 𝑏2 là các tham số (parameters) cần tìm của mô hình mạng nơ-ron. Huấn luyện để tìm các tham số cho mô hình tương đương với việc tìm các tham số 𝑊1, 𝑏1, 𝑊2, 𝑏2 sao cho hàm lỗi của mô hình đạt được là thấp nhất. Ta gọi hàm lỗi của mô hình là loss function. Đối với softmax function, ta dùng crossentropy loss (còn gọi là negative log likelihood). Nếu ta có N ví dụ dữ liệu huấn luyện, và C nhóm phân lớp, khi đó hàm lỗi giữa giá trị dự đoán 𝑦 và 𝑦 được tính:



Ý nghĩa công thức trên nghĩa là: lấy tổng trên toàn bộ tập huấn luyện và cộng dồn vào hàm loss nếu kết quả phân lớp sai. Độ dị biệt giữa hai giá trị 𝑦 và 𝑦 càng lớn thì độ lỗi càng cao. Mục tiêu của chúng ta là tối thiểu hóa hàm lỗi này. Ta có thể sử dụng phương pháp gradient descent để tối tiểu hóa hàm lỗi. Có hai loại gradient descent, một loại với fixed learning rate được gọi là batch gradient descent, loại còn lại có learning rate thay đổi theo quá trình huấn luyện được gọi là SGD (stochastic gradient descent) hay minibatch gradient descent

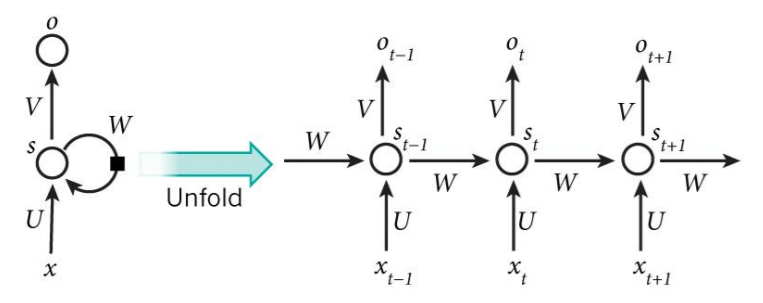
Gradient descent cần các gradient là các vector có được bằng cách lấy đạo hàm của loss function theo từng tham số để tính các gradient này, ta sử dụng thuật toán lan truyền ngược (backpropagation). Đây là cách hiệu quả để tính gradient khởi điểm từ output layer.

Áp dụng giải thuật lan truyền ngược ta có các đại lượng:



## Mạng nơ ron hồi quy RNN

Ý tưởng của RNN đó là thiết kế một Neural Network sao cho có khả năng xử lý được thông tin dạng chuỗi (sequential information), ví dụ một câu là một chuỗi gồm nhiều từ. Recurrent có nghĩa là thực hiện lặp lại cùng một tác vụ cho mỗi thành phần trong chuỗi. Trong đó, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước đó. Nói cách khác, RNN là một mô hình có trí nhớ (memory), có khả năng nhớ được thông tin đã tính toán trước đó. Không như các mô hình Neural Network truyền thống đó là thông tin đầu vào (input) hoàn toàn độc lập với thông tin đầu ra (output). Về lý thuyết, RNN có thể nhớ được thông tin của chuỗi có chiều dài bất kì, nhưng trong thực tế mô hình này chỉ nhớ được thông tin ở vài bước trước đó. Về cơ bản một mạng RNN có dạng:



*Hình 2. 3: Mạng RNN*

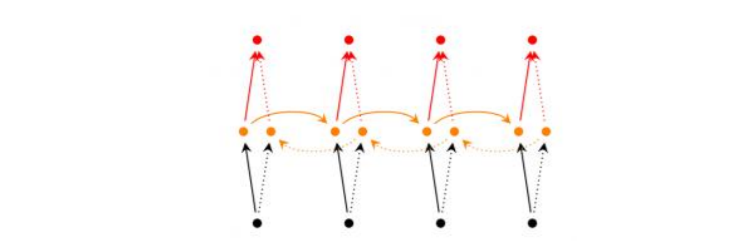
Mô hình trên mô tả phép triển khai nội dung của một RNN. Triển khai ở đây có thể hiểu đơn giản là ta vẽ ra một mạng nơ-ron chuỗi tuần tự. Ví dụ ta có một câu gồm 5 chữ “Thông tin tài khoản”, thì mạng nơ-ron được triển khai sẽ gồm 4 tầng nơ-ron tương ứng với mỗi chữ một tầng. Lúc đó việc tính toán bên trong RNN được thực hiện như sau:

xt : là đầu vào tại bước t. Ví dụ, x1 là một vec-tơ one-hot tương ứng với từ thứ 2 của câu.

st : là trạng thái ẩn tại t. Nó chính là bộ nhớ của mạng. st được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó:st = f(Uxt+Wst−1). Hàm f thường là một hàm phi tuyến tính như tang hyperbolic (tanh) hay ReLu. Để làm phép toán cho phần tử ẩn đầu tiên ta cần khởi tạo thêm s−1, thường giá trị khởi tạo được gắn bằng 0.

ot : là đầu ra tại bước t. Ví dụ, ta muốn dự đoán từ tiếp theo có thể xuất hiện trong câu thì ot chính là một vectơ xác xuất các từ trong danh sách từ vựng của ta: ot = softmax(Vst) Trong vài năm qua, các nhà nghiên cứu đã phát triển nhiều loại mạng RNNs ngày càng tinh vi để giải quyết các mặt hạn chế của RNN. Dưới đây, là một số phiên bản mở rộng của RNN.

Bidirectinal RNN (2 chiều) : dựa trên ý tưởng output tại thời điểm t không chỉ phụ thuộc vào các thành phần trước đó mà còn phụ thuộc vào các thành phần trong tương lai. Ví dụ, để dự đoán một từ bị thiếu (missing word) trong chuỗi, ta cần quan sát các từ bên trái và bên phải xung quanh từ đó. Mô hình này chỉ gồm hai RNNs nạp chồng lên nhau. Trong đó, các hidden state được tính toán dựa trên cả hai thành phần bên trái và bên phải của mạng.



*Hình 2. 4: Mạng RNN 2 chiều*

**Deep RNN** : tương tự như Bidirectional RNN, điểm khác biệt đó là mô hình này gồm nhiều tầng Bidirectional RNN tại mỗi thời điểm. Mô hình này sẽ cho ta khả năng thực hiện các tính toán nâng cao nhưng đòi hỏi tập huấn luyện phải đủ lớn.

*Hình 2. 5: Mạng RNN nhiều tầng*

Long short-term memory network (LSTM) : mô hình này có cấu trúc tương tự như RNNs nhưng có cách tính toán khác đối với các trạng thái ẩn. Memory trong LSTMs được gọi là cells (hạt nhân). Ta có thể xem đây là một hộp đen nhận thông tin đầu vào gồm hidden state và giá trị . Bên trong các hạt nhân này, chúng sẽ quyết định thông tin nào cần lưu lại và thông tin nào cần xóa đi, nhờ vậy mà mô hình này có thể lưu trữ được thông tin dài hạn.

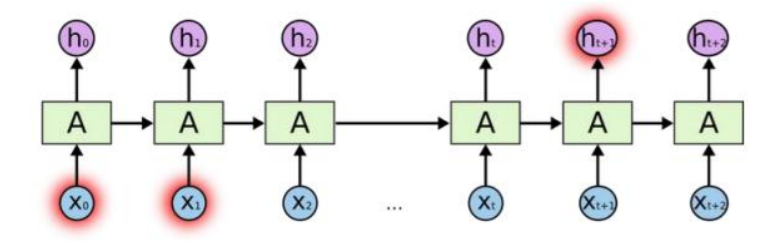
## Mạng Long short Term Memory (LSTM)

### Vấn đề phụ thuộc quá 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.

*Hình 2. 6: RNN phụ thuộc short-term*

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:



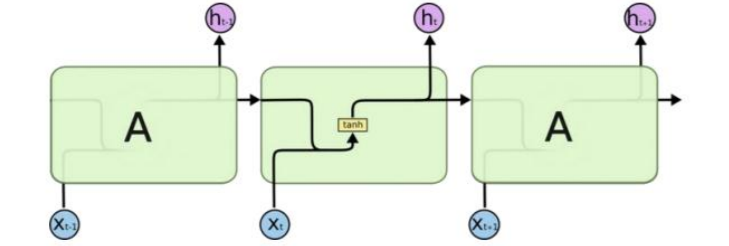
*Hình 2. 7: RNN phụ thuộc long-term*

Về lý thuyết, RNN hoàn toàn có khả năng xử lý “long-term dependencies”, nghĩa là thông tin hiện tại có được là nhờ vào chuỗi thông tin trước đó. Đáng buồn là, trong thực tế, RNN dường như không có khả năng này. Vấn đề này đã được Hochreiter (1991) [German] and Bengio, và công sự đưa ra như một thách thức cho mô hình RNN.

### Kiến trúc mạng LSTM

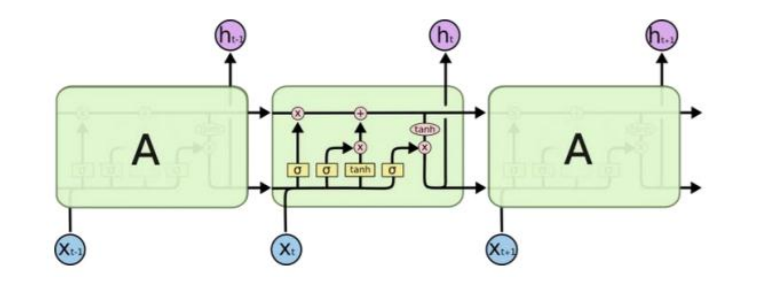
Long Short Term Memory network (LSTM) là trường hợp đặc biệt của RNN, có khả năng học long-term dependencies. Mô hình này được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được cải tiến lại. Sau đó, mô hình này dần trở nên phổ biến nhờ vào các công trình nghiên cứu gần đây. Mô hình này có khả năng tương thích với nhiều bài toán nên được sử dụng rộng rãi ở các ngành liên quan

**LSTM** được thiết kế nhằm loại bỏ vấn đề phụ thuộc quá dài. Ta quan sát lại mô hình RNN bên dưới, các layer đều mắc nối với nhau thành các module neural network. Trong RNN chuẩn, module repeating này có cấu trúc rất đơn giản chỉ gồm một lớp đơn giản tanh layer.



*Hình 2. 8: Các mô-đun lặp của mạng RNN chứa một layer*

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



*Hình 2. 9: Các mô-đun lặp của mạng LSTM chứa bốn layer*

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

Neural Network Layer: là các lớp ẩn của mạng nơ-ron.

Pointwise Operation: toán tử Pointwise, biểu diễn các phép toán như cộng, nhân vector

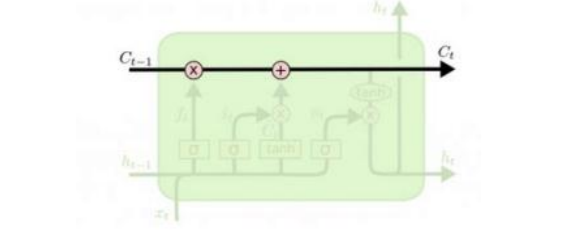
Vector Transfer: vector chỉ đầu vào và đầu ra của một nút

Concatenate: biểu thị phép nối các toán hạng.

Copy: biểu thị cho sự sao chép từ vị trí này sang vị trí khác.

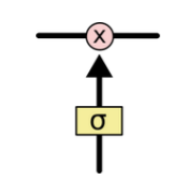
### Phân tích mô hình LSTM

Mấu chốt của LSTM là cell state (tế bào trạng thái), đường kẻ ngang chạy dọc ở trên của sơ đồ hình vẽ. Cell state giống như băng chuyền. Nó chạy xuyên thẳng toàn bộ mắc xích, chỉ một vài tương tác nhỏ tuyến tính (minor linear interaction) được thực hiện. Điều này giúp cho thông tin ít bị thay đổi xuyên suốt quá trình lan truyền.



*Hình 2. 10: Tế bào trạng thái LSTM giống như một băng truyền*

LSTM có khả năng thêm hoặc bớt thông tin vào cell state, được quy định một cách cẩn thận bởi các cấu trúc gọi là cổng (gate). Các cổng này là một cách (tuỳ chọn) để định nghĩa thông tin băng qua. Chúng được tạo bởi hàm sigmoid và một toán tử nhân pointwise.

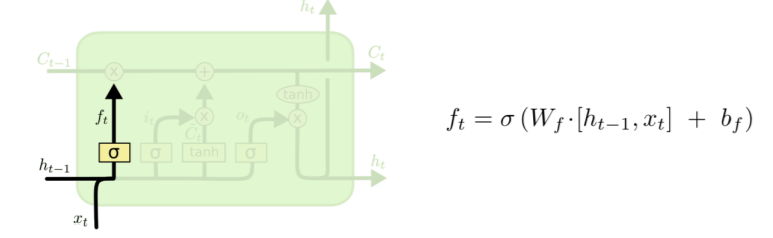


*Hình 2. 11: Cổng trạng thái LSTM*

Hàm kích hoạt Sigmoid có giá trị từ [0 – 1], mô tả độ lớn thông tin được phép truyền qua tại mỗi lớp mạng. Nếu ta thu được 0 điều này có nghĩa là “không cho bất kỳ cái gì đi qua”, ngược lại nếu thu được giá trị là 1 thì có nghĩa là “cho phép mọi thứ đi qua”. Một LSTM có ba cổng như vậy để bảo vệ và điều khiển cell state.

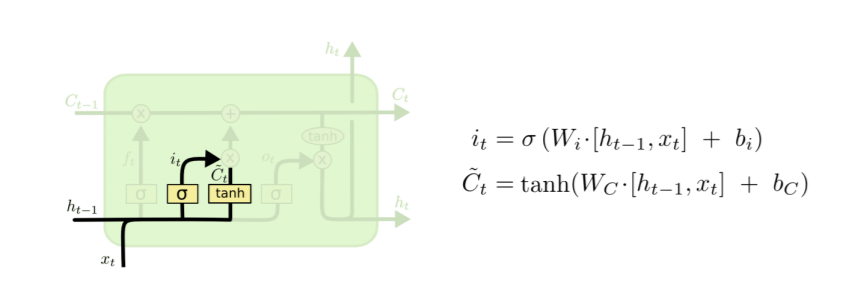
Quá trình hoạt động của LSTM đƣợc thông qua các bƣớc cơ bản sau:

Bước đầu tiên của mô hình LSTM là quyết định xem thông tin nào chúng ta cần loại bỏ khỏi cell state. Tiến trình này được thực hiện thông qua một sigmoid layer gọi là “forget gate layer” – cổng chặn. Đầu vào là ℎ𝑡−1 và 𝑥𝑡, đầu ra là một giá trị nằm trong khoảng [0, 1] cho cell state 𝐶𝑡−1. 1 tương đương với “giữ lại thông tin”, 0 tương đương với “loại bỏ thông tin”



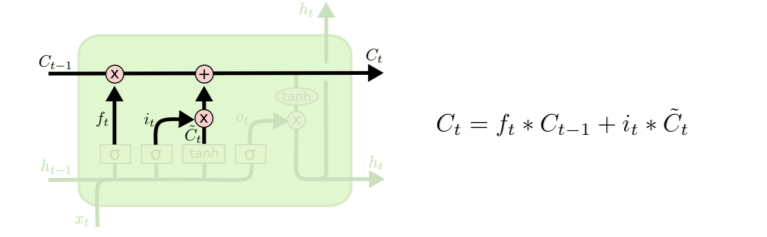
*Hình 2. 12: LSTM focus f*

Bước tiếp theo, ta cần quyết định thông tin nào cần được lưu lại tại cell state. Ta có hai phần. Một, single sigmoid layer được gọi là “input gate layer” quyết định các giá trị chúng ta sẽ cập nhật. Tiếp theo, một 𝑡𝑎𝑛ℎ layer tạo ra một vector ứng viên mới, C t . được thêm vào trong ô trạng thái.



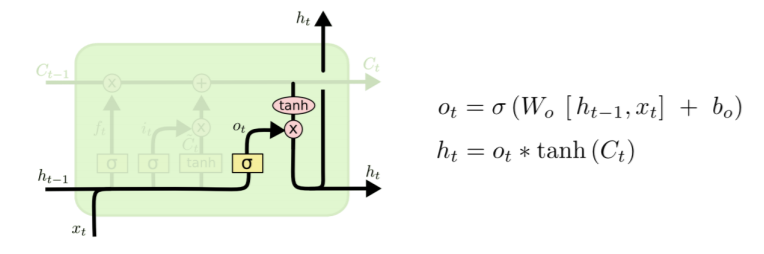
*Hình 2. 13: LSTM focus I*

Ở bước tiếp theo, ta sẽ kết hợp hai thành phần này lại để cập nhật vào cell state. Lúc cập nhật vào cell state cũ, Ct-1 , vào cell state mới Ct . Ta sẽ đưa state cũ hàm ff , để quên đi những gì trước đó. Sau đó, ta sẽ thêm it \* Ct . Đây là giá trị ứng viên mới, co giãn (scale) số lượng giá trị mà ta muốn cập nhật cho mỗi state.



*Hình 2. 14: LSTM focus c*

Cuối cùng, ta cần quyết định xem thông tin output là gì. Output này cần dựa trên cell state của chúng ta, nhưng sẽ được lọc bớt thông tin. Đầu tiên, ta sẽ áp dụng single sigmoid layer để quyết định xem phần nào của cell state chúng ta dự định sẽ output. Sau đó, ta sẽ đẩy cell state qua tanh giá trị khoảng [-1 và 1] và nhân với một output sigmoid gate, để giữ lại những phần ta muốn output ra ngoài.



*Hình 2. 15: LSTM focus o*

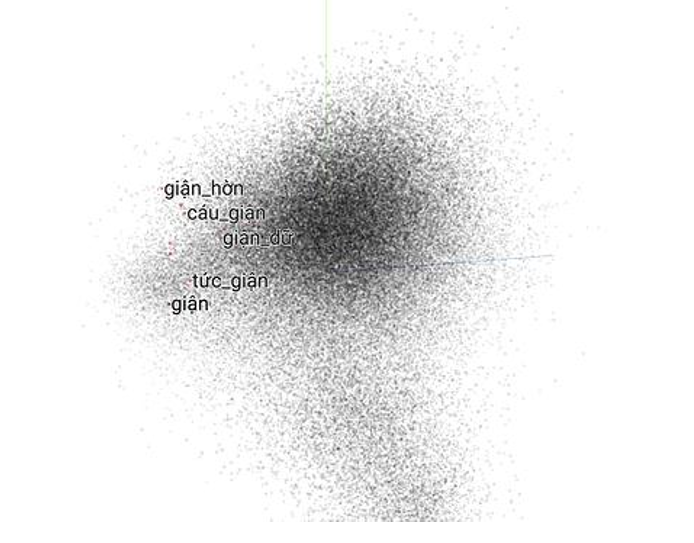
## Word embeddings

Biểu diễn ngôn ngữ hay vector hóa từ là thành phần quan trọng để giúp máy tính có thể hiểu được ngôn ngữ từ dạng văn bản sang dạng số. Tức là đưa văn bản dạng text vào một không gian mới người ta gọi là embbding space( không gian từ nhúng).

Word embeddings (tập nhúng từ) là phương pháp ánh xạ mỗi từ vào một không gian số thực nhiều chiều nhưng có kích thước nhỏ hơn nhiều so với kích thước từ điển. Word embbding có 2 model nổi tiếng là word2vec và Glove.

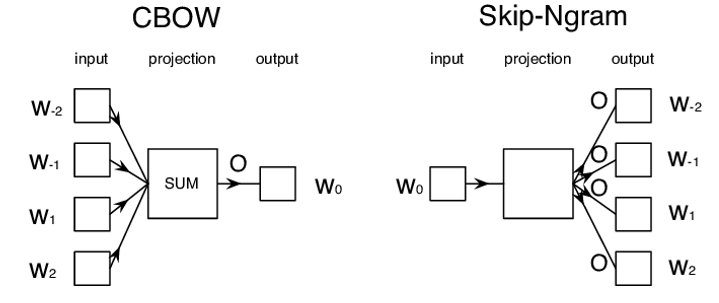
### Word2vec

Word2vec được tạo ra năm 2013 bởi một kỹ sư ở google có tên là Tomas Mikolov. Về mặt toán học, thực chất Word2Vec là việc ánh xạ từ từ một tập các từ vocabulary sang một không gian vector, mỗi vector được biểu diễn bởi n số thực. Mỗi từ ứng với một vector cố định. Sau quá trình huấn luyện mô hình bằng thuật toán backprobagation, trọng số các vector của từng từ được cập nhật liên tục. Từ đó, ta có thể thực hiện tính toán được khoảng cách giữa các từ và những từ càng "gần" nhau thường là các từ hay xuất hiện cùng nhau trong văn cảnh, các từ đồng nghĩa.



*Hình 2. 16: Mô hình từ nhúng*

Word2vec có 2 word vector là skip-gram và Continuous Bag-of-Words (Cbow).



*Hình 2. 17: Mô hình CBOW và Skip-Ngram*

CBoW: Dự đoán từ hiện tại dựa trên ngữ cảnh của các từ trước đó:

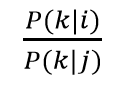
* + Cho các từ ngữ cảnh
  + Đoán xác suất của của một từ đích

Skip-gram: Dự đoán các từ xung quanh khi cho bởi từ hiện tại:

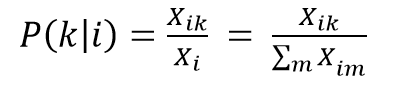
* + Cho từ đích
  + Đoán xác suất cho các từ ngữ cảnh

### Glove

GloVe là một trong những phương pháp mới để xây dựng vectơ từ (giới thiệu vào năm 2014), nó được xây dựng dựa trên ma trận đồng xảy ra (Co-occurrence Matrix). GloVe dựa trên ý tưởng tính tỉ lệ xác xuất:

 (1)

Với P(k|i) là xác suất từ k xuất hiện trong ngữ cảnh của từ i, tương tự vậy với P(k|j). Công thức tính của P(k|i):

 (2)

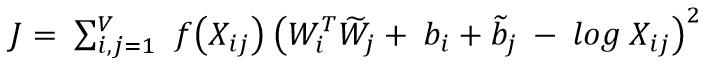
Xik : là số lần xuất hiện của từ k trong ngữ cảnh của từ i (và ngược lại).

Xi : là số lần xuất hiện của từ i trong ngữ cảnh của toàn bộ các từ còn lại ngoại trừ i.

Ý tưởng chính của GloVe được tính dựa trên độ tương tự ngữ nghĩa giữa hai từ i, j được xác định thông qua độ tương tự ngữ nghĩa giữa từ k với mỗi từ i, j, những từ k có tính xác định ngữ nghĩa tốt chính là những từ làm cho giá trị được tính từ công thức (1) nằm trong khoảng [0 ,1]

Ví dụ, nếu từ i là “table”, từ j là “cat” và từ k là “chair” thì công thức (1) sẽ cho giá trị tiệm cận đến 1 do “chair” có nghĩa gần hơn với “table” hơn là “cat”, ở trường hợp khác, nếu ta thay từ k là “ice cream” thì giá trị công thức (1) sẽ xấp xỉ bằng 0 do “ice cream” hầu như chẳng lên quan gì tới “table” và “cat”.

Dựa trên tầm quan trọng của công thức (1) , GloVe khởi đầu bằng việc là nó sẽ tìm một hàm F sao cho nó ánh xạ từ các vec-tơ từ trong vùng không gian V sang một giá trị tỉ lệ tính theo công thức (1) . Việc tìm F không đơn giản, tuy nhiên, sau nhiều bước đơn giản hóa cũng như tối ưu, ta có thể đưa nó về bài toán hồi quy với việc tính hàm chi phí tối thiểu (minimum cost function) sau:

 (3)

Wi , Wjlà các vector từ.

bi,bj là các bias tương ứng (được thêm vào ở các bước đơn giản hóa và tối ưu).

Xij : mục nhập tương ứng với cặp từ i,j trong ma trận đồng xảy ra

Hàm f(x) được gọi là hàm trọng số (weighting function), được thêm vào để giảm bớt sự ảnh hưởng của các cặp từ xuất hiện quá thường xuyên, hàm này thỏa mãn 3 tính chất:

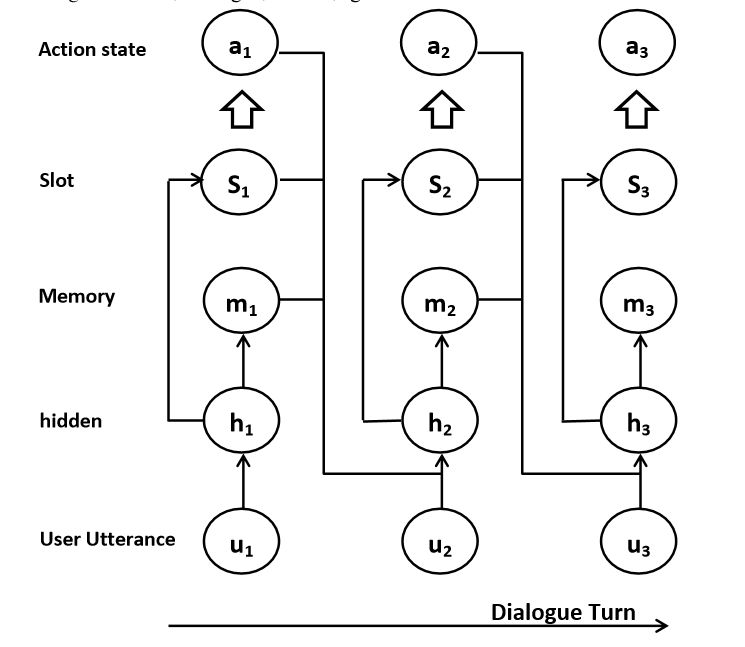
* Có giới hạn tại 0.
* Là hàm không giảm.
* Có giá trị nhỏ khi x rất lớn.

## Ứng dụng RNN vào quản lý hội thoại

Thuật toán RNN áp dụng cho chức năng theo dõi trạng thái hay ngữ cảnh hội thoại Dialogue State Tracking (DTS) trong thành phần quản lý hội thoại.

### Mô hình word-based DST

Thành phần quản lý hội thoại(DM) có nhiệm vụ quan trọng nhất đó là quản lý các trạng thái hay ngữ cảnh của hội thoại để quyết định được hành động tiếp theo. Vậy việc quyết định action tiếp theo thì dựa vào đâu. Đó chính là dữ liệu đầu vào của người dùng, dữ liệu các slot được lưu trong bộ nhớ và trạng thái của các action ở trước. Với khả năng lưu được các thông tin trong việc xử lý các bài toán dạng chuỗi thì mạng RNN được ứng dụng trong việc xác định ngữ cảnh và quyết định action tiếp nhờ vào các thông tin lưu được trong bộ nhớ mạng RNN.



*Hình 2. 18: Mô hình word-based DST với mạng RNN*

User Utterance: là câu dữ liệu đầu vào của người dùng

Hidden: là lớp ẩn được sử dụng trong thành phần NLU để vector hóa ngôn ngữ, phân loại ý định và trích xuất được các thông tin người dùng

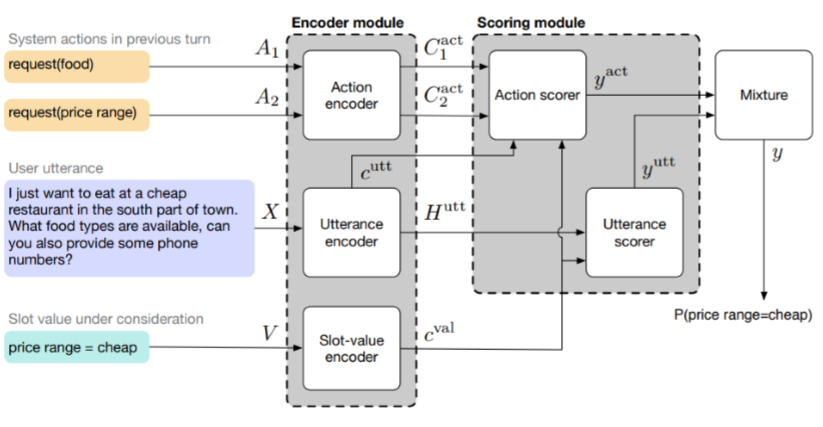
Memory: là bộ nhớ lưu các giá trị vector hóa ngôn ngữ và ngữ cảnh hội thoại bao gồm cả slot

Slot: thông tin được trích xuất được lưu lại trong các câu nói người dùng

Action State: trạng thái action trước. Nó mang tính ngữ cảnh ở trong đoạn hội thoại

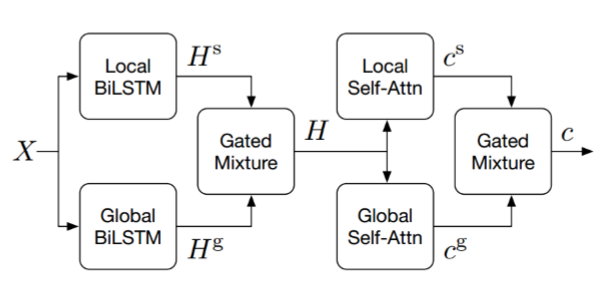
### Mô hình Global-Locally Self-Attentive DST (GLAD)

Với các mô hình theo dõi trạng thái hội thoại khác thường xác định intent người dùng trực tiếp trên danh sách các đoạn hội thoại đã trainning cho bot sẽ gặp vấn đề về nhập nhằng xác định 1 intent trong các đoạn hội thoại có sử dụng cùng 1 intent. Tức là với cùng 1 câu nói người dùng xác định được 1 intent thì intent đó có thể được dùng nhiều trong các đoạn hội thoại. Với tư tưởng xác định cả ngữ cảnh cho từng đoạn hội thoại thì việc xác định intent nằm trong đoạn hội thoại nào sẽ tăng thêm độ chính xác.



*Hình 2. 19: Mô hình Global-Locally Self-Attentive DST (GLAD)*

Các tham số đầu vào cho phần local và global là các giá trị intent người dùng (X), các slot và action state (C)



*Hình 2. 20: Global-locally self-attentive encoder modul*

Việc dùng các slot trong mô hình cần lưu ý là sau kết thúc mỗi hội thoại thì nên xóa các slot không dùng nữa để tránh nhập nhằng xác định ví trí chính xác intent của người dùng trong danh sách các đoạn hội thoại mà bot được đào tạo.

# Chương 3. Phân tích thiết kế hệ thống



## Khảo sát

# Dữ liệu training sẽ được lưu dưới dạng file Json gồm có: tag (nhấn lớp cho nội dung nhập của người dùng), patterns(mẫu câu đầu vào được training để phân lớp), responses(các câu trả lời được mapping cho chatbot để phản hồi những request trước đó.

## Các chức năng của hệ thống

### Yêu cầu chức năng của hệ thống

- Chatbot tương tác nhanh, chính xác, hiệu quả.

- Không xảy ra lỗi trong quá trình thực hiện

### Yêu cầu phi chức năng của hệ thống

Giao diện phải được thiết kế sao cho đơn giản, dễ nhìn nhưng vẫn đảm bảo đẹp mắt và người dùng có thể dễ dàng tương tác với chatbot mà không gặp phải bất kỳ trở ngại nào.

## Phân tích use case

### Biểu đồ use case tổng quát

## 

### Đặc tả use case

#### Mô tả use case “tương tác với chatbot”

1. **Tên use case:** Xem danh sách ca sĩ.
2. **Mô tả vắn tắt:** Use case cho phép người sử dụng, người quản trị, khách hàng có thể tìm kiếm sản phẩm được lưu trong CSDL.
3. Luồng sự kiện:

3.1. Luồng cơ bản:

3.1.1. Use case này bắt đầu khi người dùng nhập chữ vào ô text box trên màn hình hiển thị, hệ thống sẽ kiểm tra xem từ ngữ vừa nhập rồi hiển thị câu trả lời được lấy trong CSDL để tương tác với người dùng.

3.1.2. Khi người dùng không có câu hỏi để hỏi thì sẽ ấn nút thoát và use case kết thúc.

3.2. Luồng rẽ nhánh:

3.2.1. Tại bất cứ thời điểm nào của quá trình thực hiện, nếu không thể kết nối với CSDL thì sẽ hiển thị thông báo lỗi và use case kết thúc.

1. Các yêu cầu đặc biệt:

Không có.

1. Tiền điều kiện:

Không có.

1. Hậu điều kiện:

Không có.

1. Điểm mở rộng:

Không có.

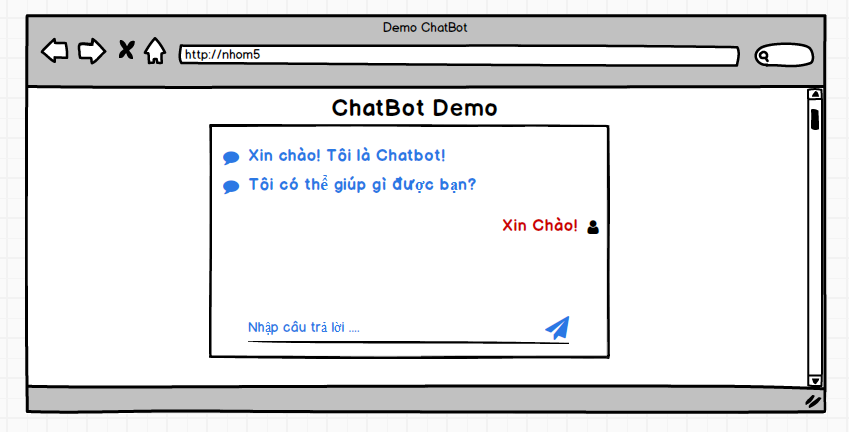
\*Biểu đồ tuần tự



\*Biểu đồ lớp phân tích



\*Hình dung màn hình



# Chương 4. Ứng dụng chatbot



## Ngôn ngữ lập trình Python

### 4.1.1 Python là gì?

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng. Ngôn ngữ lập trình Python được tạo bởi Guido van Rossum và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

**4.1.2 Tính năng chính của Python.**

* ***Ngôn ngữ lập trình đơn giản, dễ học:*** Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp.
* ***Miễn phí, mã nguồn mở:*** Bạn có thể tự do sử dụng và phân phối Python, thậm chí là dùng cho mục đích thương mại. Vì là mã nguồn mở, bạn không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật.
* ***Khả năng di chuyển:*** Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy liền mạch trên hầu hết tất cả các nền tảng như Windows, macOS, Linux.
* ***Khả năng mở rộng và có thể nhúng:*** Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.
* ***Ngôn ngữ thông dịch cấp cao:*** Không giống như C/C++, với Python, bạn không phải lo lắng những nhiệm vụ khó khăn như quản lý bộ nhớ, dọn dẹp những dữ liệu vô nghĩa,... Khi chạy code Python, nó sẽ tự động chuyển đổi code sang ngôn ngữ máy tính có thể hiểu. Bạn không cần lo lắng về bất kỳ hoạt động ở cấp thấp nào.
* ***Thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến:*** Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình của bạn trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code. Ví dụ: Bạn cần kết nối cơ sở dữ liệu MySQL trên Web server? Bạn có thể nhập thư viện MySQLdb và sử dụng nó. Những thư viện này được kiểm tra kỹ lưỡng và được sử dụng bởi hàng trăm người. Vì vậy, bạn có thể chắc chắn rằng nó sẽ không làm hỏng code hay ứng dụng của mình.
* ***Hướng đối tượng:*** Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, bạn có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

## Thư viện sử dụng

### 4.2.1 Chatterbot

ChatterBot là một thư viện của Python giúp chúng ta dễ dàng tạo các phản hồi tự động cho đầu vào của người dùng. ChatterBot sử dụng các thuật toán **machine learning** để xử lý dữ liệu với nhiều ngữ cảnh khác nhau. Điều này giúp các nhà phát triển dễ dàng tạo ra các bot trò chuyện và tự động hóa các cuộc hội thoại với người dùng. Để biết thêm chi tiết mọi người có thể đọc thêm tài liệu về Chatterbot ở trên trang chủ của nó.

Thiết kế Độc lập Ngôn ngữ của ChatterBot cho phép nó được Huấn luyện để có thể nói bất cứ ngôn ngữ nào. Ngoài ra bản chất Machinelearing của ChatterBot cho phép một Instance là tác nhân cải thiện kiến thức của chính nó về các phản ứng có thể có khi nó tương tác với con người và từ các nguồn dữ liệu khác.

ChatterBot là một thư viện Python được thiết kế để giúp dễ dàng tạo một phần mềm mà ở đó nó có thể tham gia vào việc giao tiếp hay còn gọi là chatbot.

Một instance của ChatterBot chưa được train, start mà chưa có kiến thức về cách giao tiếp. Mỗi khi người dùng nhập một statement, thư viện sẽ lưu văn bản mà họ đã nhập và văn bản mà bot phản hồi lại. Khi số lượng input mà ChatterBot nhận được tăng thì số lượng các response mà nó có thể trả lời cũng như độ chính xác của từng response có liên quan đến input statement đó cũng tăng theo.

Chương trình này lựa chọn response phù hợp nhất bằng cách tìm kiếm statement đã biết gần nghĩa nhất với input statement, sau đó nó chọn một response từ các response đã biết của statement đó.

Tạo mới một chat bot:



Sau khi tạo mới một instance Chatbot, bạn đã có thể train cho nó. Training là một cách tốt để đảm bảo rằng con bot của bạn khởi đầu với một lượng kiến thức về các response mà nó có thể sử dụng. Phương thức training hiện tại là nhận một list các statement (câu nói) trong cuộc nói chuyện thông thường.

Chatter Bot này đã có một số bộ ngôn ngữ nên chúng ta có thể sử dụng luôn mà không cần phải training lại. Nhưng với tiến Việt thì phải training.

Một ví dụ nho nhỏ cho việc training:



Để lấy ra response trả về. Ở đây tham số đầu vào là một input của người dùng và respone là câu phản hồi của bot:



### 4.2.2 Flask

Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép bạn xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các api nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại. Flask cung cấp cho bạn công cụ, các thư viện và các công nghệ hỗ trợ bạn làm những công việc trên.

Flask là một micro-framework. Điều này có nghĩa Flask là một môi trường độc lập, ít sử dụng các thư viện khác bên ngoài. Do vậy, Flask có ưu điểm là nhẹ, có rất ít lỗi do ít bị phụ thuộc cũng như dễ dàng phát hiện và xử lý các lỗi bảo mật.

Nếu bạn đã cài Python và thiết lập môi trường thành công. Bạn có thể sử dụng command sau đây để cài Flask:

*pip install Flask*

Một ứng dụng đơn giản nhất sử dụng Flask framework có thể viết bằng vài dòng code như sau:



Sau khi chạy, console của bạn sẽ có 1 dòng log giống như thế này.

*\* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)*

Bạn có thể copy và dán đường dẫn này lên trình duyệt để xem kết quả. Bạn sẽ thấy dòng chữ *Hello, World!* được in ra trên trình duyệt.

### NLTK

NLTK hay Natural Language Toolkit - Bộ công cụ ngôn ngữ tự nhiên, là một thư viện được viết bằng Python hỗ trợ xử lý ngôn ngữ tự nhiên. Bằng cách cung cấp các cơ chế và kỹ thuật xử lý ngôn ngữ phổ biến, nó giúp cho việc xử lý ngôn ngữ tự nhiên trở lên dễ dàng và nhanh chóng hơn. Được viết bởi Steven Bird và Edward Loper, làm việc tại Khoa Máy Tính, Đại Học Pennsylvania, Hoa Kỳ và năm 2001. Ngoài việc hỗ trợ xử lý ngôn ngữ, NLTK còn có các mô phỏng đồ hoạ và dữ liệu mẫu hữu ích.

NLTK cung cấp các xử lý như classification, tokenization, stemming, tagging, parsing, và semantic reasoning... Những ứng dụng này chúng ta sẽ dần được tìm hiểu ở những bài viết sau. Ngoài việc phục vụ xử lý ngôn ngữ tự nhiên, NLTK còn được sử dụng trong Machine Learning với tác dụng làm sạch dữ liệu, xử lý dữ liệu đầu vào cho các thuật toán Machine Learning.

Nếu bạn đang sử dụng Windows hoặc Linux hoặc Mac, bạn có thể cài đặt NLTK [bằng pip](https://likegeeks.com/import-create-install-reload-alias-python-modules/#Install-Python-Modules-Using-pip) :  *# pip install nltk.*

Khi bạn đã cài đặt NLTK, bạn nên cài đặt các gói NLTK bằng cách chạy mã sau:

*Import nltk*

*nltk.download()*

Bạn có thể cài đặt tất cả các gói vì tất cả chúng đều có kích thước nhỏ mà không có vấn đề gì.

### Tensorflow

Sự phát triển của trí tuệ nhân tạo dẫn đến việc tìm hiểu về machine learning và deep learning đã trở thành xu thế hiện nay. Việc sử dụng các thư viện có sẵn để tính toán đã giúp việc tiếp cận các bài toán trở nên đơn giản hơn. Tensorflow là một thư viện phần mềm mã nguồn mở hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning.

* **Cách thức hoạt động của Tensorflow**

**TensorFlow** cho phép các nhà phát triển tạo các biểu đồ dataflow – các cấu trúc mô tả cách thức dữ liệu di chuyển qua biểu đồ hoặc một loạt các node xử lý. Mỗi node trong biểu đồ đại diện cho một hoạt động toán học và mỗi kết nối giữa các node là một mảng dữ liệu đa chiều hoặc tenxor.

TensorFlow cung cấp tất cả những điều này cho lập trình viên bằng ngôn ngữ Python. Python rất dễ học và làm việc, nó cung cấp các cách thuận tiện để diễn tả mức độ trừu tượng cao được ghép với nhau. Các node và tenxor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow chính là các ứng dụng Python.

Các hoạt động toán học thực tế, tuy nhiên chúng không được thực hiện trong Python. Các thư viện biến đổi có sẵn thông qua **TensorFlow** được viết dưới dạng nhị phân C ++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các tóm tắt lập trình cấp cao để nối chúng lại với nhau.

Các ứng dụng **TensorFlow** có thể chạy trên hầu hết mọi mục tiêu thuận tiện: một máy cục bộ, một cụm trong cloud, thiết bị iOS và Android, CPU hoặc GPU. Nếu bạn sử dụng cloud của Google, bạn có thể chạy **TensorFlow** trên silicon của đơn vị xử lý TensorFlow (TPU) tùy chỉnh của Google để tăng tốc hơn nữa. Tuy nhiên, các mô hình kết quả được tạo bởi **TensorFlow**, có thể được triển khai trên hầu hết mọi thiết bị nơi chúng được sử dụng để phục vụ những dự đoán.

**TensorFlow 2.0**, được phát hành vào tháng 10 năm 2019, đã cải thiện framework theo nhiều cách dựa trên phản hồi của người dùng, để làm việc dễ dàng hơn (ví dụ: bằng cách sử dụng Keras API tương đối đơn giản để đào tạo mô hình) và hiệu quả hơn. Đào tạo phân tán dễ chạy hơn nhờ API mới và hỗ trợ cho **TensorFlow Lite** cho phép triển khai các mô hình trên nhiều nền tảng khác nhau. Tuy nhiên, code được viết cho các phiên bản trước của **TensorFlow** đôi khi chỉ được viết lại một chút, đôi khi đáng kể, để tận dụng tối đa các tính năng mới của **TensorFlow 2.0.**

* **Lợi ích của Tensorflow**

Lợi ích lớn nhất mà cũng là duy nhất **TensorFlow** cung cấp cho sự phát triển của **machine learning** là không tưởng. Thay vì xử lý các chi tiết khó hiểu khi thực hiện các **thuật toán** tìm ra cách thích hợp để output của một function sang input của một function khác, nhà phát triển có thể tập trung vào logic tổng thể của ứng dụng.

TensorFlow cung cấp các tiện ích bổ sung cho các nhà phát triển, những người cần debug và hướng vào các ứng dụng TensorFlow. Chế độ **Eager execution**cho phép bạn đánh giá và sửa đổi từng hoạt động của biểu đồ một cách riêng biệt và minh bạch, thay vì xây dựng toàn bộ biểu đồ dưới dạng một đối tượng mờ đục và đánh giá tất cả cùng một lúc. Bộ phần mềm trực quan **TensorBoard** cho phép bạn kiểm tra và lập hồ sơ cách thức biểu đồ chạy bằng bảng điều khiển dựa trên web tương tác.

Một cảnh báo: Một số chi tiết về việc triển khai **TensorFlow**, dẫn đến việc khó có được kết quả hoàn toàn xác định cho một số công việc đào tạo. Đôi khi một mô hình được đào tạo trên một hệ thống sẽ thay đổi một chút so với mô hình được đào tạo trên một hệ thống khác, ngay cả khi chúng được cung cấp cùng một dữ liệu. Điều đó có nghĩa rằng, có thể giải quyết các vấn đề đó và TensorFlow đang xem xét nhiều kiểm soát hơn để ảnh hưởng đến tính quyết định trong quy trình làm việc.

## Chương trình ứng dụng

### Chương trình code



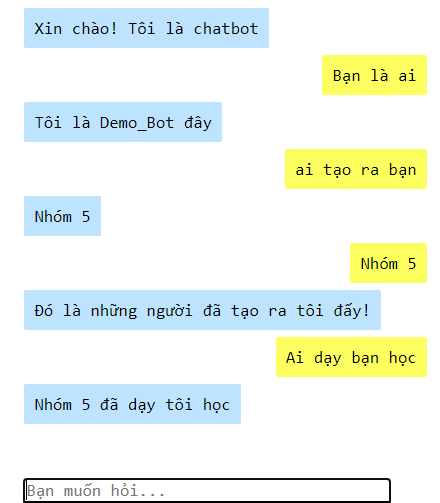
*Hình 3.1: Ảnh code quá trình huấn luyện chatbot*



*Hình 3.2: Code khởi chạy*

### Demo sản phẩm

Chatbot sẽ tự động trả lời câu hỏi dựa trên một số mẫu câu hỏi mà Chatbot đã được người lập trình huấn luyện sẵn



### *Giao diện chính của chương trình*

# Kết luận

* Kết quả đạt được:
* Về lý thuyết:

- Nắm được quy trình xây dựng một ứng dụng python nói chung và Chatbot nói riêng.

- Nắm được những lý thuyết về Chatbot, NLTK, Flask cũng như cách xử lý dữ liệu và nguyên lý làm việc của các thư viện

* Về thực hành:

- Xây dựng và huấn luyện thành công Chatbot với chức năng tương tác với một số câu hỏi của người sử dụng

- Xây dựng được giao diện website tương tác với Chatbot đẹp mắt và dễ dàng tương tác

* Hạn chế :
* Chưa đưa được ứng dụng Chatbot lên tương tác trực tuyến
* Cơ sở dữ liệu chưa đủ lớn và chưa tối ưu

# Tài liệu tham khảo

[1] <https://stackoverflow.com/>

[2] <https://www.w3schools.com/>

[3] <https://www.codecademy.com/>