

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ MÔN
NHẬP MÔN TÍNH TOÁN ĐA PHƯƠNG TIỆN

**TÌM HIỂU VÀ XÂY DỰNG HỆ THỐNG
TRẢ LỜI TỰ ĐỘNG ĐA CHỦ ĐỀ**

Người hướng dẫn: ThS VŨ ĐÌNH HỒNG
TS LÊ ANH CƯỜNG
TS PHẠM VĂN HUY

Người thực hiện: NGUYỄN MINH THÀNH – 51800727
NGUYỄN ĐỒNG HUY – 51800682
NGUYỄN ĐÌNH LUÂN - 51800994

Lớp: 18050402, 18050401

Khóa: 22

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO CUỐI KỲ MÔN
NHẬP MÔN TÍNH TOÁN ĐA PHƯƠNG TIỆN

**TÌM HIỂU VÀ XÂY DỰNG HỆ THỐNG
TRẢ LỜI TỰ ĐỘNG ĐA CHỦ ĐỀ**

Người hướng dẫn: ThS VŨ ĐÌNH HỒNG

TS LÊ ANH CƯỜNG

TS PHẠM VĂN HUY

Người thực hiện: NGUYỄN MINH THÀNH – 51800727

NGUYỄN ĐỒNG HUY – 51800682

NGUYỄN ĐÌNH LUÂN - 51800994

Lớp: 18050402, 18050401

Khóa: 22

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2021

LỜI CẢM ƠN

Trong suốt học kỳ I năm học 2021 – 2022, chúng em đã nhận được sự hướng dẫn, quan tâm rất tận tình và giúp đỡ của các quý thầy/cô và bạn bè. Với lòng biết ơn sâu sắc và chân thành nhất, em xin gửi lời cảm ơn đến quý thầy/cô ở khoa Công Nghệ Thông Tin – Trường Đại học Tôn Đức Thắng đã giúp đỡ em rất nhiều trong việc tiếp thu đầy đủ các kiến thức chuyên ngành. Trong học kỳ này, khoa công nghệ thông tin đã mở cho chúng em lớp về “Nhập môn tính toán đa phương tiện” để chúng em có cơ hội được tiếp cận với môn học mà theo em nghĩ là rất hữu ích vì tính ứng dụng vào thực tế. Với báo cáo giữa kỳ của môn “Nhập môn tính toán đa phương tiện” đã giúp chúng em hiểu rõ hơn về các khái niệm cơ bản về các khái niệm tính toán đa phương tiện trong công nghệ thông tin và các ứng dụng của nó trong thực tế. Báo cáo giữa kỳ này cũng đã giúp chúng em củng cố lại tất cả các kiến thức về lập trình web và xử lý ngôn ngữ tự nhiên mà bản thân đã tích lũy được trong suốt một nửa học kỳ I học tập trực tuyến vừa qua. Qua lời cảm ơn này, chúng em đặc biệt gửi lời cảm ơn đến thầy Vũ Đình Hồng – phụ trách giảng dạy phần lập trình web với Django và thầy Lê Anh Cường – phụ trách giảng dạy phần xử lý ngôn ngữ tự nhiên trong môn “Nhập môn tính toán đa phương tiện”. Các thầy đã tận tâm hướng dẫn chúng em qua từng buổi học lý thuyết và cũng như là thực hành trên lớp. Cùng với việc đưa ra các bài thực hành song song với bài giảng trên lớp và cùng các ví dụ minh họa dễ hiểu, dễ hình dung và giúp chúng em ghi nhớ được lâu hơn các bài học. Cùng với tri thức và sự tâm huyết của các thầy đã truyền đạt cho chúng em những vốn kiến thức quý báu về an toàn mạng không dây và di động trong suốt thời gian học trực tuyến tại nhà. Với vốn kiến thức tiếp thu được, em sẽ ứng dụng nó vào các ứng dụng và công việc thực tế sau này.

Chúng em xin bày tỏ lòng biết ơn đến các ban lãnh đạo của trường Đại học Tôn Đức Thắng và các khoa phòng ban chức năng, các bạn học đã trực tiếp và gián tiếp giúp đỡ chúng em trong quá trình học tập và hoàn thành bài báo cáo cuối kỳ của môn.

Với điều kiện về thời gian, bài báo cáo này sẽ không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của thầy để có cơ hội củng cố lại những lỗ hỏng kiến thức cũng như là giúp bản thân chúng em nâng cao được tri thức của mình để phục vụ tốt hơn cho các công tác thực tế sau này.

BÁO CÁO CUỐI KỲ ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm tiêu luận của chúng tôi và được sự hướng dẫn của thầy Vũ Đình Hồng, Lê Anh Cường và thầy Phạm Văn Huy. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất cứ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

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

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

TP. Hồ Chí Minh, ngày tháng năm
Tác giả
(ký tên và ghi rõ họ tên)

Nguyễn Minh Thành
Nguyễn Đồng Huy
Nguyễn Đình Luân

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

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

Phần đánh giá của GV chấm bài

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

TÓM TẮT

Trong thời đại công nghệ 4.0 hiện đại ngày nay, con người ngày càng tiếp thu và ứng dụng các thành tựu mới của khoa học công nghệ. Cùng với sự phát triển đó, nhiều ngành khoa học và công nghệ đặc biệt là trong lĩnh vực công nghệ thông tin cũng phát triển theo xu hướng đó. Trong đó, tính toán đa phương tiện trong công nghệ thông tin cũng là một yếu tố góp phần không nhỏ vào sự phát triển đó. Trong lĩnh vực đó, ta có thể thấy nổi bật nhất là việc ứng dụng website trong nhiều các lĩnh vực khác nhau như thương mại, quản lý, ... và do đó nó đang rất phát triển. Bên cạnh đó, các phần nhỏ trong lĩnh vực tính toán đa phương tiện như xử lý ngôn ngữ tự nhiên hay đặc biệt là xử lý ảnh số cũng góp phần vào sự phát triển của nó.

Ở bài báo cáo giữa kỳ này, chúng ta sẽ tiến hành tìm hiểu về hai khái niệm đầu tiên trong môn học là lập trình website và xử lý ngôn ngữ tự nhiên trong môn “Nhập môn tính toán đa phương tiện”. Với bài báo cáo giữa kỳ của môn “Nhập môn tính toán đa phương tiện” sẽ giúp cho chúng ta hiểu sâu hơn về các khái niệm cơ bản của ngôn ngữ lập trình python với việc ứng dụng nó trong lập trình website và xử lý ngôn ngữ tự nhiên và cũng như là phân tích và đánh giá kết quả thu được sau khi việc triển khai đã hoàn thành. Ngoài ra, mục tiêu chính của bài báo cáo là tìm hiểu và triển khai xây dựng hệ thống trả lời tự động - Chatbot, đây sẽ là tiền đề để chúng em có thể tìm hiểu sâu hơn về lĩnh vực xử lý ngôn ngữ tự nhiên nói riêng và ứng dụng nó vào trong website. Tóm lại, báo cáo giữa kỳ của môn “Nhập môn tính toán đa phương tiện” bao gồm 5 chương:

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT – TỔNG QUAN VỀ NGÔN NGỮ TỰ NHIÊN VÀ CÁC KHÁI NIỆM CƠ BẢN CỦA NÓ.

CHƯƠNG 2: HỆ THỐNG TRẢ LỜI TỰ ĐỘNG – TÌM HIỂU CÁC KHÁI NIỆM CƠ BẢN VỀ HỆ THỐNG TRẢ LỜI TỰ ĐỘNG.

CHƯƠNG 3: ỨNG DỤNG NEURAL NETWORK CHO BÀI TOÁN CHATBOT - TÌM HIỂU CÁC KHÁI NIỆM LIÊN QUAN ĐẾN MẠNG NEURAL VÀ NEURAL HỒI QUY VÀ MÔ HÌNH SEQ2SEQ.

CHƯƠNG 4: TRIỂN KHAI HỆ THỐNG TRẢ LỜI TỰ ĐỘNG – GIẢI THÍCH CHI TIẾT VỀ MÃ CHƯƠNG TRÌNH VÀ HƯỚNG DẪN TRIỂN KHAI HỆ THỐNG.

CHƯƠNG 5: TỔNG KẾT– TỔNG KẾT VÀ KẾT LUẬN NHỮNG KẾT QUẢ ĐẠT ĐƯỢC CỦA BÀI BÁO CÁO.

MỤC LỤC

LỜI CẢM ƠN.....	3
PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN.....	5
TÓM TẮT.....	6
DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT.....	11
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ.....	12
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	16
1.1 Tổng quan về ngôn ngữ tự nhiên.....	16
1.1.1 Ý tưởng.....	16
1.1.2 Lịch sử hình thành	17
1.2 Các khái niệm cơ bản về xử lý ngôn ngữ tự nhiên.....	19
1.2.1 Khái niệm trí tuệ nhân tạo	19
1.2.2 Khái niệm ngôn ngữ tự nhiên	22
1.2.3 Khái niệm xử lý ngôn ngữ tự nhiên.....	22
1.2.4 Nhập nhằng.....	24
1.2.5 Dịch máy	25
1.2.6 Xác suất thống kê	27
1.3 Lý thuyết thông tin	29
1.3.1 Khái niệm	29
1.3.2 Entropy	29
1.3.3 Perplexity - Cross Entropy	30
1.3.3.1 Sự liên quan của Entropy đối với ngôn ngữ.....	30
1.3.3.2 Perplexity.....	30
1.3.3.3 Entropy rate	31
1.3.3.4 Cross Entropy	31
1.4 Quy trình xử lý ngôn ngữ tự nhiên	32
1.5 Một số các mô hình trong ngôn ngữ tự nhiên	34
1.5.1 Neural Network	34
1.5.2 Support Vector Machines	35
1.5.3 Naïve Bayes	36
1.6 Các ứng dụng của xử lý ngôn ngữ tự nhiên	38
1.5 Tổng kết chương 1	43

CHƯƠNG 2: TỔNG QUAN VỀ HỆ THỐNG TRẢ LỜI TỰ ĐỘNG CHATBOT	44
2.1 Tổng quan về Chatbot	44
2.1.1 Khái niệm	44
2.1.2 Lịch sử	45
2.1.3 Cấu tạo của hệ thống Chatbot.....	48
2.2 Hiểu ngôn ngữ tự nhiên	50
2.2.1 Phân loại ý định	51
2.2.2 Trích xuất thông tin	53
2.3 Quản lý hội thoại	54
2.3.1 Mô hình FSA	55
2.3.2 Mô hình Frame-Based	56
2.4 Mô hình sinh ngôn ngữ.....	57
2.4.1 Template-Based.....	57
2.4.2 Plan-Based.....	58
2.4.3 Class-Based	58
2.5 Tổng kết chương 2.....	59
CHƯƠNG 3: ỨNG DỤNG RECURRENT NEURAL NETWORK CHO BÀI TOÁN CHATBOT.	60
3.1 Tổng quan về mạng Neural	60
3.1.1 Mạng Neural sinh học	60
3.1.2 Quá trình học của bộ não.....	61
3.2 Neural nhân tạo.....	61
3.2.1 Khái niệm	61
3.2.2 Mô hình Neural.....	62
3.2.2.1 Neural một đầu vào	62
3.2.2.2 Neural nhiều đầu vào.....	64
3.3 Mạng Neural nhân tạo	65
3.3.1 Định nghĩa	65
3.3.2 Lịch sử phát triển	66
3.3.3 Kiến trúc mạng Neural nhân tạo.....	68
3.4 Mạng neural hồi quy.....	71

3.4.1 Khái niệm	71
3.4.2 Phân loại	72
3.5 Long Short-Term Memory	74
3.5.1 Khái niệm	74
3.5.2 Phân tích mô hình LSTM	76
3.5.3 Nguyên lý hoạt động của LSTM	77
3.5.4 Mô hình Seq2Seq	79
3.5.4.1 Giới thiệu	79
3.5.4.2 Lịch sử	79
3.5.4.3 Khái niệm	80
3.5.4.4 Cấu trúc mô hình Seq2Seq	80
3.6 Tổng kết chương 3	82
CHƯƠNG 4: TRIỀN KHAI HỆ THỐNG TRẢ LỜI TỰ ĐỘNG	83
4.1 Tổng quan về Django Framework.....	83
4.1.1 Khái niệm	83
4.1.2 Mô hình MTV.....	84
4.1.3 Đặc điểm của Django	85
4.1.4 Triển khai Django.....	86
4.1.5 Tổng kết.....	94
4.2 Sơ đồ chương trình	94
4.3 Mã chương trình	96
4.3.1 Xây dựng mô hình Chatbot	96
4.3.2 Triển khai Chatbot với Django	107
4.4 Thực thi chương trình	117
4.4.1 Khởi chạy	117
4.4.2 Hệ thống Chatbot.....	117
4.5 Tổng kết chương 4	123
CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ	124
5.1 Tổng kết.....	124
5.2 Đánh giá và phân tích	124
5.2.1 Phân tích	124
5.2.2 Đánh giá.....	125

TÀI LIỆU THAM KHẢO	126
Tài liệu tham khảo chính	126
Tài liệu tham khảo phụ	126
BẢNG PHÂN CÔNG CÔNG VIỆC	128

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

CÁC KÝ HIỆU CÁC CHỮ VIẾT TẮT

Thuật ngữ	Tên đầy đủ
NLP	Natural Language Processing
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
ML	Machine Learning
AI	Artificial Intelligence
DL	Deep Learning
Seq2Seq	Sequence to Sequence
SVM	Support Vector Machine
NLU	Natural Language Understading
CSDL	Cơ sở dữ liệu

DANH MỤC CÁC BẢNG BIẾU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

Hình 1.1: Mô hình giọng nói.	17
Hình 1.2: Ứng dụng của AI lĩnh vực robot.	19
Hình 1.3: Ứng dụng công nghệ AI trong lái xe tự động của Tesla.	21
Hình 1.4: Công nghệ AI đối với thiết bị bay không người lái trong lĩnh vực y tế.	21
Hình 1.5: Vị trí của xử lý ngôn ngữ tự nhiên trong trí tuệ nhân tạo.	23
Hình 1.6: Một số các bài toán được ứng dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên.	23
Hình 1.7: Các yếu tố trong khái niệm nhập nhằng.	24
Hình 1.8: Các bước trong quy trình xử lý của ngôn ngữ tự nhiên.	32
Hình 1.9: Phân tích cú pháp dưới dạng cây.	33
Hình 1.10: Mô hình của mạng Neural Network.	35
Hình 1.11: Mô tả thuật toán SVM.	36
Hình 1.12: Ví dụ minh họa cho ứng dụng Chatbot trong NLP.	39
Hình 1.13: Ví dụ về ứng dụng giám sát mạng xã hội.	39
Hình 1.14: Ví dụ về Google Translate được ứng dụng trong dịch máy.	40
Hình 1.15: Ví dụ về ứng dụng nhận dạng giọng nói trong NLP.	40
Hình 1.16: Hình ảnh minh họa về phân tích cảm xúc trong NLP.	41
Hình 1.17: Ví dụ về việc tóm tắt văn bản tự động trong NLP.	41
Hình 1.18: Ví dụ về ứng dụng kiểm tra chính tả của NLP.	42
Hình 1.19: Ví dụ về mô hình phân loại thư rác.	42
Hình 1.20: Ví dụ về mô hình trích xuất thông tin trong NLP.	43
Hình 2.1: Ứng dụng trả lời tự động chatbot.	44
Hình 2.2: Các thành phần cấu tạo nên hệ thống Chatbot.	48
Hình 2.3: Quy trình xử lý của các thành phần trong hệ thống Chatbot.	49
Hình 2.4: Các bước xử lý chính trong thành phần NLU.	50
Hình 2.5: Các bước xử lý chính trong NLU.	51
Hình 2.6: Mô hình các bước của việc xác định ý định.	52
Hình 2.7: Mô hình quản lý trạng thái và đưa ra quyết định trong hội thoại.	55
Hình 2.8: Quản lý hội thoại theo mô hình FSA.	56
Hình 2.9: Phương pháp sinh ngôn ngữ Plan-Based.	58
Hình 2.10: Phương pháp sinh ngôn ngữ Class-Based.	59

Hình 3.1: Mô hình tế bào thần kinh của người.....	60
Hình 3.2: Mô hình Neural chỉ gồm một đầu vào.....	63
Hình 3.3: Mô hình mạng Neural nhiều đầu vào.....	64
Hình 3.4: Mô hình neural nhiều đầu vào rút gọn.....	65
Hình 3.5: Mô hình của mạng neural nhân tạo ANN.....	66
Hình 3.6: Cấu trúc của mạng neural nhân tạo.....	68
Hình 3.7: Quá trình xử lý thông tin của một neural nhân tạo.....	69
Hình 3.8: Mô hình mạng RNN.....	72
Hình 3.9: Mô hình LSTM.....	73
Hình 3.10: Mô hình của Bidirectional RNN.....	73
Hình 3.11: Mô hình Deep RNN.....	74
Hình 3.12: Các module lặp của mạng RNN chuẩn. [15].....	75
Hình 3.13: Cấu trúc lặp của LSTM với 4 tầng. [15]	75
Hình 3.14: Cell State trong mô hình LSTM. [15]	76
Hình 3.15: Cổng trạng thái của mô hình LSTM. [15]	77
Hình 3.16: Bước đầu tiên trong mô hình LSTM. [15]	77
Hình 3.17: Bước thứ hai trong mô hình LSTM. [15]	78
Hình 3.18: Bước thứ ba trong mô hình LSTM. [15]	78
Hình 3.19: Bước cuối cùng trong mô hình LSTM. [15]	79
Hình 3.20: Một số mô hình Seq2Seq.....	80
Hình 3.21: Mô hình Seq2Seq sử dụng hai bộ Encoder và Decoder.....	80
Hình 3.22: Mô hình Seq2Seq trong việc sinh chuỗi.....	81
 Hình 4.1: Logo Django Framework. [17].....	83
Hình 4.2: Mô hình MVT trong Django Framework. [17]	84
Hình 4.3: Kiểm tra phiên bản Python cài đặt trên máy.	87
Hình 4.4: Câu lệnh cài đặt Django.	87
Hình 4.5: Câu lệnh tạo project mới trong Django.	88
Hình 4.6: Cấu trúc thư mục project sau khi tạo.....	88
Hình 4.7: Câu lệnh khởi chạy server cho project.	89
Hình 4.8: Giao diện sau khi khởi chạy server thành công.....	90
Hình 4.9: Câu lệnh khởi tạo web app cho project Django.	90
Hình 4.10: Thư mục web app sau khi khởi tạo thành công.....	91
Hình 4.11: Khởi tạo bảng trong CSDL thông qua Model.	92
Hình 4.12: Tạo tài khoản Admin cho trang quản trị.	93
Hình 4.13: Giao diện đăng nhập vào trang Admin.....	93

Hình 4.14: Giao diện quản lý của trang Admin.	94
Hình 4.15: Sơ đồ hoạt động của hệ thống Chatbot.	95
Hình 4.16: Các thư viện cần thiết trong việc xây dựng mô hình Seq2Seq.	96
Hình 4.17: Dữ liệu Chatbot. [19].	98
Hình 4.18: Mảng lưu giá trị đường dẫn đến các tệp dữ liệu Chatbot.	98
Hình 4.19: Thực hiện xử lý lấy ra tập câu hỏi và câu trả lời.	99
Hình 4.20: Hàm xử lý câu và thực hiện tách từ trong mỗi câu.	100
Hình 4.21: Xử lý xóa các câu trả lời rỗng.	100
Hình 4.22: Thực hiện tách các từ trong câu thành các từ.	101
Hình 4.23: Thực hiện đếm tần suất xuất hiện của các từ.	101
Hình 4.24: Thực hiện thêm các tag cho các câu trả lời.	101
Hình 4.25: Thực hiện chuyển các từ thành các số.	102
Hình 4.26: Thêm các tag vào biến từ điển.	102
Hình 4.27: Thực hiện chuyển đổi các câu hỏi và trả lời thành dạng vector.	103
Hình 4.28: Thực hiện padding giá trị của các vector.	104
Hình 4.29: Khai báo các tham số cần thiết cho mô hình.	104
Hình 4.30: Khởi tạo lớp Embedding và Encoder LSTM.	105
Hình 4.31: Khởi tạo lớp Decoder LSTM.	105
Hình 4.32: Khởi tạo lớp Dense đầu ra và xây dựng mô hình.	106
Hình 4.33: Tóm tắt mô hình sau khi khởi tạo.	106
Hình 4.34: Training dữ liệu cho model.	106
Hình 4.35: Lưu mô hình đã training.	107
Hình 4.36: Thực hiện lưu giá trị của từ điển word2index và index2word để tái sử dụng.	107
Hình 4.37: Các thư viện cần thiết.	108
Hình 4.38: Hàm xử lý giao diện chính, đăng nhập và cập nhật thông tin.	108
Hình 4.39: Hàm đăng ký tài khoản, đăng xuất và hiển thị lỗi.	109
Hình 4.40: Hàm cập nhật thông tin người dùng.	110
Hình 4.41: Tái sử dụng lại mô hình và các biến từ điển.	111
Hình 4.42: Khai báo các tham số cho mô hình.	111
Hình 4.43: Tóm tắt mô hình Seq2Seq sau khi được tái sử dụng.	112
Hình 4.44: Lấy ra các lớp từ mô hình được sử dụng lại.	112
Hình 4.45: Xây dựng mô hình Encoder cho chuỗi đầu vào.	113
Hình 4.46: Xây dựng Inference Model để dự đoán.	113
Hình 4.47: Xử lý các tin nhắn đầu vào cho chương trình.	114
Hình 4.48: Truyền câu hỏi vào Encoder LSTM.	114

Hình 4.49: Vòng lặp thực hiện dự đoán từng với decoder model.....	115
Hình 4.50: Danh sách các địa chỉ url trong web app.....	116
Hình 4.51: Danh sách địa chỉ url chính của project.	116
Hình 4.52: Khởi tạo Server cho Django.....	117
Hình 4.53: Giao diện đăng nhập chương trình. [24]	118
Hình 4.54: Giao diện đăng ký tài khoản. [25]	118
Hình 4.55: Giao diện cập nhật thông tin cá nhân. [26]	119
Hình 4.56: Giao diện trang web báo lỗi. [27].....	119
Hình 4.57: Giao diện chính của chương trình Chatbot. [23].....	120
Hình 4.58: Các tên hoặc liên kết đến các MXH sử dụng Template Tag.....	120
Hình 4.59: Quá trình trò chuyện và giao tiếp giữa người dùng và hệ thống.....	121
Hình 4.60: Hàm thêm tin nhắn khi người dùng nhấn vào nút gửi.....	122
Hình 4.61: Xử lý bắt sự kiện và gửi dữ liệu đến máy chủ.....	122

DANH MỤC BẢNG

Bảng 2.1: Bảng danh sách các Frame cho mô hình Frame-Based.	56
Bảng 2.2: Bảng danh sách các câu hỏi dựa trên phương pháp Template-Based.....	57
Bảng 3.1: Bảng các hàm kích hoạt và công thức tương ứng.....	64

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 Tổng quan về ngôn ngữ tự nhiên

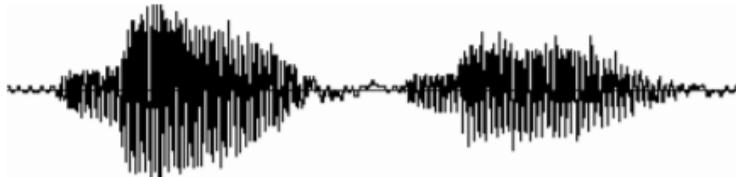
Một trong những mong muốn mãnh liệt, xuất hiện từ rất sớm của các nhà khoa học máy tính (Computer Science) nói chung và trí tuệ nhân tạo (Artificial Intelligence) nói riêng là xây dựng thành công các hệ thống, chương trình máy tính có khả năng giao tiếp với con người thông qua ngôn ngữ tự nhiên (Natural Language), tức thứ ngôn ngữ con người sử dụng hàng ngày thay vì các ngôn ngữ lập trình (Programming Language) hay ngôn ngữ máy (Computer Language) bậc thấp. Xử lý ngôn ngữ tự nhiên (Natural Language Processing), một nhánh nghiên cứu của trí tuệ nhân tạo, trong đó phát triển các thuật toán, xây dựng các chương trình máy tính có khả năng phân tích, xử lý, và hiểu ngôn ngữ của con người. Để máy tính có thể hiểu và thực thi một chương trình được viết bằng ngôn ngữ cấp cao, ta cần phải có một trình biên dịch thực hiện việc chuyển đổi chương trình đó sang chương trình ở dạng ngôn ngữ đích.

1.1.1 Ý tưởng

Xử lý ngôn ngữ chính là xử lý thông tin khi đầu vào là “dữ liệu ngôn ngữ” (dữ liệu cần biến đổi), tức dữ liệu “văn bản” hay “tiếng nói”. Các dữ liệu liên quan đến ngôn ngữ viết (văn bản) và nói (tiếng nói) đang dần trở thành kiểu dữ liệu chính và lưu trữ dưới dạng điện tử. Đặc điểm chính của các kiểu dữ liệu này là không có cấu trúc hoặc nửa cấu trúc và chúng không thể lưu trữ trong các khuôn dạng cố định như các bảng biểu. Theo đánh giá của công ty Oracle, hiện có đến 80% dữ liệu không cấu trúc trong lượng dữ liệu của loài người đang có (Oracle Text). Với sự ra đời và phổ biến của Internet, sách báo điện tử, máy tính cá nhân, viễn thông, thiết bị âm thanh, ... mọi người ai cũng có thể tạo ra dữ liệu văn bản hay tiếng nói. Vấn đề là làm sao ta có thể xử lý chúng, chuyển từ các dạng ta chưa hiểu được thành các dạng ta có thể hiểu và giải thích được, tức là ta có thể tìm ra thông tin, tri thức hữu ích cho mình.

Giả sử ta có các từ trong tiếng nước ngoài như: Hello, Bonjour, Ciao. Nếu có ai đó dịch, hoặc có một chương trình máy tính dịch (biến đổi) chúng ra tiếng Việt, ta sẽ hiểu nghĩa các từ trên đều là: “xin chào”. Nếu các từ này được lưu trữ như các tệp tiếng Anh, Pháp, Ý và Việt như ta nhìn thấy ở trên, ta có các dữ liệu “văn bản”. Nếu ai đó đọc các từ này, ghi âm lại, ta có thể chuyển chúng vào máy tính dưới dạng các tệp các tín hiệu

(signal) “tiếng nói”. Tín hiệu sóng âm của hai âm tiết tiếng Việt có thể nhìn thấy như sau:



Hình 1.1: Mô hình giọng nói.

Tuy nhiên, một văn bản thật sự (ví dụ một bài báo khoa học) có thể có đến hàng nghìn câu và hàng triệu văn bản. Web là một nguồn dữ liệu văn bản khổng lồ, cùng với các thư viện điện tử – khi trong một tương lai gần, các sách báo xưa nay và các nguồn âm thanh được chuyển hết vào máy tính (chẳng hạn bằng các chương trình nhận dạng chữ, thu nhập âm thanh, hoặc gõ thẳng vào máy) – sẽ sớm chứa hầu như toàn bộ kiến thức của nhân loại. Vấn đề là làm sao “xử lý” (chuyển đổi) được khỏi dữ liệu văn bản và tiếng nói khổng lồ này qua dạng khác để mỗi người có được thông tin và tri thức cần thiết từ chúng.

Xử lý ngôn ngữ tự nhiên đã được ứng dụng trong thực tế để giải quyết các bài toán như: nhận dạng chữ viết, nhận dạng tiếng nói, tổng hợp tiếng nói, dịch tự động, tìm kiếm thông tin, tóm tắt văn bản, khai phá dữ liệu và phát hiện tri thức.

1.1.2 Lịch sử hình thành

Đề tài về dịch máy được nhắc đến vào thế kỷ XVII, khi đó các triết gia như Leibniz và Descartes đưa ra đề xuất một số sự liên quan từ ngữ giữa các ngôn ngữ. Tất cả các đề xuất này vẫn nằm trên trang giấy, chưa được phát triển trên máy tính. Giữa năm 1930, Georges Artsrouni đề xuất bộ từ điển song ngữ tự động bằng cách sử dụng *Thí nghiệm Turing*. Song song đó, Peter Troyanskii đề xuất thêm một phương pháp mới để áp dụng với các loại ngôn ngữ dựa trên Esperanto. Phép thử Turing là một cách để trả lời câu hỏi “máy tính có biết nghĩa không?”, được phát biểu dưới dạng một trò chơi. Hình dung có ba người tham gia trò chơi, một người đàn ông (A), một người đàn bà (B) và một người chơi (C). Người chơi ngồi ở một phòng tách biệt với A và B, không biết gì về A và B (như hai đối tượng ăn X và Y) và chỉ đặt các câu hỏi cũng như nhận trả lời từ A và B qua một màn hình máy tính. Người chơi kết luận trong X và Y ai là đàn ông ai là đàn bà. Trong phép thử này, A luôn tìm cách làm cho C bị nhầm lẫn và B luôn tìm cách giúp

C tìm được câu trả lời đúng. Ý nghĩa rất lớn của nó nằm ở chỗ đã nhấn mạnh rằng khả năng giao tiếp thành công của máy với con người trong một cuộc đối thoại tự do và không hạn chế biểu hiện chính yếu của trí thông minh nhân tạo.

Các nghiên cứu về xử lý ngôn ngữ tự nhiên được diễn ra trong nhiều thập kỷ, bắt đầu trở lại vào năm 1940. Dịch máy - Machine Translation (MT) là máy tính đầu tiên dựa trên ứng dụng liên quan đến ngôn ngữ tự nhiên. Trong khi Weaver và Booth bắt đầu một dự án MT vào năm 1946 trong lĩnh vực dịch thuật dựa trên chuyên môn nhưng lại bị ngưng lại bởi thế chiến thứ II. Năm 1949, bản ghi nhớ của Weaver truyền cảm hứng cho người dự án. Ông đề nghị sử dụng ý tưởng từ mật mã học và lý thuyết thông tin cho ngôn ngữ dịch. Nghiên cứu đã bắt đầu tại các tổ chức nghiên cứu khác nhau ở Hoa Kỳ trong nhiều năm. Trong giai đoạn này, những ứng dụng xử lý ngôn ngữ tự nhiên khác bắt đầu nổi lên, chẳng hạn như nhận dạng tiếng nói. Cộng đồng xử lý ngôn ngữ và cộng đồng tiếng nói sau đó được chia làm 2 phe. Cộng đồng xử lý ngôn ngữ bị chi phối bởi các quan điểm về lý thuyết ngữ pháp thông dụng và cộng đồng tiếng nói bị chi phối bởi lý thuyết thông tin.

Trong cuối năm 1970, có một sự thay đổi lớn về vấn đề ngữ nghĩa, đưa ra mục tiêu giao tiếp và lên kế hoạch. Grosz đã phân tích công việc theo định hướng đối thoại và đề xuất một lý thuyết phân vùng giữa các đơn vị dựa trên việc tìm kiếm thông tin liên quan về cấu trúc của một tác vụ và cấu trúc của một hội thoại hướng nhiệm vụ. Mann và Thompson đã phát triển lý thuyết cấu trúc Rhetorical, phân cấp cấu trúc. Những nhà nghiên cứu khác cũng đã góp phần quan trọng bao gồm Hobbs và Rosenschein, Polanyi và Scha và Reichman. Giai đoạn này chứng kiến được khối lượng công việc đáng kể trên thế hệ ngôn ngữ tự nhiên. TEXT của McKeown's đưa ra việc lập kế hoạch và McDonald trình bày MUMMBEL sử dụng để mô tả các dạng văn bản ngắn, thông thường là đoạn văn. Khả năng của TEXT là tạo ra phản ứng mạch lạc trực tuyến với một chuyên gia của ngành.

Trong những năm 1980, sự thúc đẩy của hệ thống tính toán xử lý vượt bậc, đã thay thế các vấn đề hạn chế của xử lý ngôn ngữ tự nhiên và hướng ứng dụng làm việc với ngôn ngữ được mở rộng. Các nhà nghiên cứu kiểm tra các phương pháp tiếp cận thực tế. Vào cuối năm 1980, phương pháp tiếp cận mang tính tượng trưng đã giải quyết được nhiều vấn đề quan trọng trong xử lý ngôn ngữ tự nhiên và phương pháp tiếp cận thống kê được bổ sung ở nhiều khía cạnh. Sự ra đời của phương pháp thống kê đã thành công

trong việc đối phó với nhiều vấn đề chung trong tính toán ngôn ngữ học như là xác định giọng nói, từ định hướng, ... đã trở thành tiêu chuẩn trong xử lý ngôn ngữ tự nhiên. Các nhà nghiên cứu bây giờ đang phát triển tiếp hệ thống xử lý ngôn ngữ tự nhiên thế hệ mới phù hợp hơn với văn bản chung và những sự mơ hồ của ngôn ngữ.

1.2 Các khái niệm cơ bản về xử lý ngôn ngữ tự nhiên

1.2.1 Khái niệm trí tuệ nhân tạo

Đầu tiên, để tìm hiểu các khái niệm cơ bản về lĩnh vực xử lý ngôn ngữ tự nhiên thì ta điểm qua một chút về các khái niệm liên quan đến trí tuệ nhân tạo. Trí tuệ nhân tạo chính là nền tảng và tiền đề cho việc phát triển của ngôn ngữ tự nhiên.

Trí tuệ nhân tạo hay AI (được biết với tên đầy đủ là Artificial Intelligence), thường được gọi là trí thông minh nhân tạo. Hay nói một cách dễ hiểu thì nó là trí thông minh được thể hiện bằng máy móc, trái ngược với trí thông minh tự nhiên trong bộ não của con người. Thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy móc hay máy tính có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với tâm trí, như "giải quyết vấn đề" hay "học tập". Nó được con người lập trình nên với mục tiêu giúp máy tính có thể tự động hóa các hành vi của con người.



Hình 1.2: Ứng dụng của AI lĩnh vực robot.

Trí tuệ nhân tạo khác với lập trình logic trong các ngôn ngữ lập trình ở chỗ nó sử dụng các hệ thống máy học để mô phỏng trí thông minh của con người trong các quy trình mà con người có thể làm tốt hơn máy tính.

Đặc biệt, trí tuệ nhân tạo giúp máy tính tiếp thu các trí tuệ của con người như: tư duy và suy nghĩ để giải quyết vấn đề, giao tiếp thông qua hiểu ngôn ngữ và ngôn ngữ, học tập

và tự thích ứng ... Mặc dù trí tuệ nhân tạo có rất nhiều trong các tác phẩm khoa học viễn tưởng mang nội hàm của trí thông minh là một trong những ngành quan trọng nhất của khoa học máy tính. Trí tuệ nhân tạo liên quan đến hành vi, học máy và trí tuệ thích ứng.

Công nghệ AI được chia thành 4 loại chính:

- ❖ **Công nghệ AI phản ứng:** là công nghệ có khả năng phân tích những hành động, động thái khả thi nhất của chính mình và của đối thủ và từ đó tính toán đưa ra được giải pháp tốt nhất.
- ❖ **Công nghệ AI với bộ nhớ hạn chế:** là công nghệ có đặc điểm sử dụng những kinh nghiệm trong quá khứ để đưa ra các quyết định trong tương lai. Thường được kết hợp với cảm biến môi trường xung quanh nhằm mục đích dự đoán những trường hợp có thể xảy ra và đưa ra quyết định tốt nhất cho thiết bị.
- ❖ **Lý thuyết với trí tuệ nhân tạo:** là công nghệ được sử dụng để tự học hỏi và suy nghĩ và áp dụng cho những gì học được để thực hiện một việc cụ thể. Tuy nhiên vẫn chưa phải là một phương án khả thi.
- ❖ **Loại tự nhận thức:** là công nghệ có khả năng tự nhận thức bản thân, có cảm xúc, ý thức và hành xử như con người. Đây được xem là bước phát triển cao nhất của công nghệ AI và cho đến thời điểm hiện tại thì công nghệ này vẫn chưa thực sự khả thi.

Công nghệ trí tuệ nhân tạo AI hiện nay được ứng dụng nhiều trong các lĩnh vực khác nhau của đời sống con người, ta có thể kể đến một vài các ví dụ điển hình như:

- **Ngành vận tải:** được ứng dụng trên nhiều các phương tiện tự vận hành mà điển hình là xe ô tô. Hiện nay, các xe ô tô tự lái đang được phát triển và sẽ là xu hướng của tương lai khi nó góp phần mang lại lợi ích kinh tế cao hơn nhờ khả năng cắt giảm chi phí cũng như hạn chế những tai nạn nguy hiểm đến tính mạng con người.



Hình 1.3: Ứng dụng công nghệ AI trong lái xe tự động của Tesla.

- **Sản xuất:** được ứng dụng để xây dựng những quy trình sản xuất ngày càng tối ưu hơn do nó có khả năng phân tích và đưa ra các cơ sở định hướng cho việc ra quyết định trong sản xuất.
- **Y tế:** được ứng dụng nổi bật trong việc các thiết bị bay không người lái trong các trường hợp khẩn cấp.



Hình 1.4: Công nghệ AI đối với thiết bị bay không người lái trong lĩnh vực y tế.

- **Giáo dục:** được ứng dụng trong việc chấm điểm hay dạy kèm học sinh nhờ vào sự tự động hóa của nó. Nó có thể chỉ ra các vấn đề liên quan đến việc đánh giá việc học tập nhờ vào khả năng phân tích và đánh giá cao của nó.
- **Truyền thông:** được ứng dụng trong việc tiếp cận đối với khách hàng mục tiêu. Nhờ vào khả năng này, các công ty quảng cáo và truyền thông có thể cung cấp

quảng cáo vào đúng thời điểm, đúng khách hàng tiềm năng do dựa trên việc phân tích các đặc điểm về thói quen, tìm kiếm hay nhân khẩu học, ...

- **Dịch vụ:** giúp cho ngành dịch vụ hoạt động tốt hơn và tối ưu hơn, góp phần mang đến những trải nghiệm mới mẻ và tốt hơn cho khách hàng.

1.2.2 Khái niệm ngôn ngữ tự nhiên

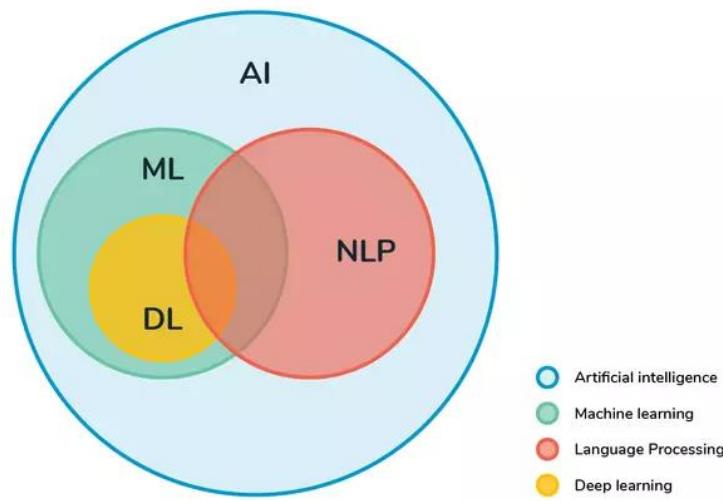
Ngôn ngữ tự nhiên là một thành phần trong lĩnh vực ngôn ngữ học rộng lớn. Một ngôn ngữ tự nhiên (Natural Language) có thể được hiểu là bất cứ ngôn ngữ nào phát sinh hay được tạo ra mà ngôn ngữ đó không trải qua bất cứ một giai đoạn suy nghĩ trước nào đó trong não bộ của con người. Ngôn ngữ tự nhiên tồn tại dưới nhiều trạng thái cuộc sống của chúng ta. Và chính việc không tuân thủ theo bất cứ một sự định hướng cũng như hướng dẫn chỉ định từ đầu đã tạo ra những nét đặc trưng riêng biệt khiến cho ngôn ngữ tự nhiên khác với những ngôn ngữ thường.

Một ví dụ điển hình đó là một số ngôn ngữ mà con người được sử dụng để giao tiếp với nhau, dù là ngôn ngữ nói, ngôn ngữ ký hiệu, ký hiệu xúc giác hay chữ viết. Những ngôn ngữ này khác với ngôn ngữ được xây dựng và ngôn ngữ hình thức chẳng hạn như ngôn ngữ lập trình hoặc nghiên cứu logic.

Ngôn ngữ tự nhiên có một số các đặc điểm mà nổi bật chính là sự đa dạng, đây là một đặc điểm dễ nhận thấy nhất của loại ngôn ngữ này. Nó có thể được tạo ra mỗi ngày khi mà con người sử dụng ngôn ngữ để giao tiếp bằng lời nói, cử chỉ, ký hiệu, cảm xúc hoặc dưới dạng chữ viết, văn bản, ... Ngôn ngữ tự nhiên còn mang những đặc điểm khác biệt so với các loại ngôn ngữ được xây dựng dựa trên những mục đích cụ thể của con người như các ngôn ngữ lập trình, ngôn ngữ nghiên cứu theo logic, ... Hiểu một cách đơn giản nhất, con người chúng ta có thể hiểu rằng ngôn ngữ tự nhiên là ngôn ngữ trái ngược với ngôn ngữ nhân tạo và ngôn ngữ được xây dựng.

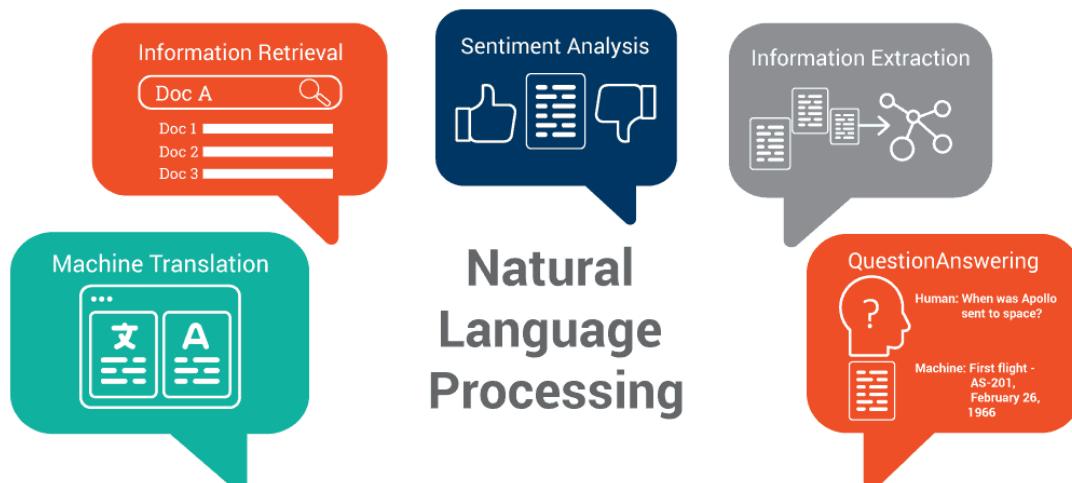
1.2.3 Khái niệm xử lý ngôn ngữ tự nhiên

Xử lý ngôn ngữ tự nhiên hay được biết với tên tiếng Anh là Natural Language Processing – viết tắt là NLP, là một nhánh thuộc trong lĩnh vực khoa học máy tính, kỹ thuật thông tin và trí tuệ nhân tạo tập trung vào các ứng dụng trên ngôn ngữ của con người. Trong đó, NLP tập trung nghiên cứu về khả năng tương tác ngôn ngữ giữa máy tính với con người sao cho máy tính có thể hiểu và xử lý được ngôn ngữ của con người. Trong trí tuệ nhân tạo thì xử lý ngôn ngữ tự nhiên là một trong những phần khó nhất vì nó liên quan đến việc phải hiểu ý nghĩa ngôn ngữ bởi sự đa dạng về ngữ nghĩa.



Hình 1.5: Vị trí của xử lý ngôn ngữ tự nhiên trong trí tuệ nhân tạo.

Ta cùng xem xét hình 1.4, ta thấy rằng lĩnh vực AI bao trùm cả các lĩnh vực như ML – Machine Learning, DL – Deep Learning hay NLP – Natural Language Processing (Xử lý ngôn ngữ tự nhiên). Nên ta có thể nói NLP chính là một phần trong lĩnh vực trí tuệ nhân tạo và có liên quan mật thiết với học máy và học sâu. NLP sở dĩ là một nhánh trong lĩnh vực trí tuệ nhân tạo là vì mục đích của nó là tập trung vào việc nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người. Tóm lại, mục tiêu chính của NLP là giúp cho máy tính hiểu và thực hiện hiệu quả những nhiệm vụ liên quan đến ngôn ngữ của con người như là tăng tương tác giữa người và máy, cải thiện hiệu suất hay cũng như là nâng cao hiệu quả xử lý văn bản.



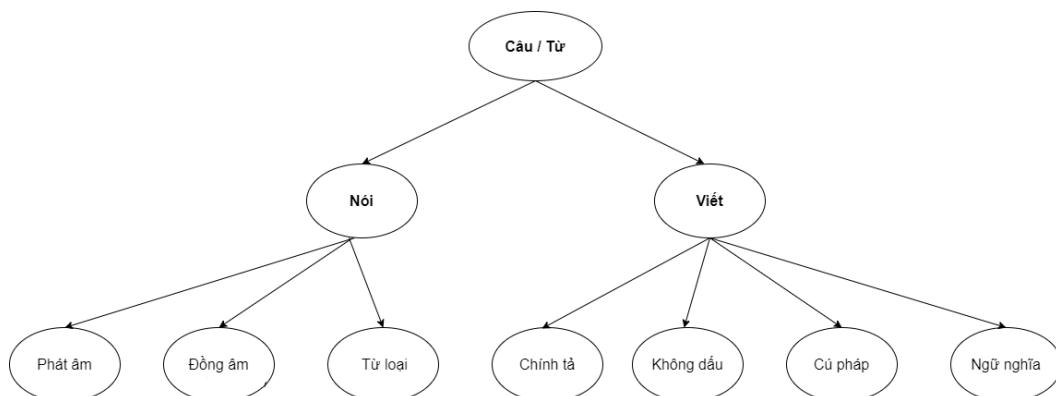
Hình 1.6: Một số các bài toán được ứng dụng trong lĩnh vực xử lý ngôn ngữ tự nhiên.

Xử lý ngôn ngữ tự nhiên có thể được chia ra thành hai nhánh lớn, không hoàn toàn độc lập, bao gồm:

- **Xử lý tiếng nói (Speech Processing):** xử lý tiếng nói tập trung nghiên cứu, phát triển các thuật toán, chương trình máy tính xử lý ngôn ngữ của con người ở dạng tiếng nói hay dữ liệu ở dạng âm thanh. Các ứng dụng quan trọng của xử lý tiếng nói bao gồm nhận dạng tiếng nói và tổng hợp tiếng nói. Nếu như nhận dạng tiếng nói là chuyển ngôn ngữ từ dạng tiếng nói sang dạng văn bản thì ngược lại, tổng hợp tiếng nói chuyển ngôn ngữ từ dạng văn bản thành tiếng nói.
- **Xử lý văn bản (Text Processing):** xử lý văn bản tập trung vào phân tích dữ liệu văn bản. Các ứng dụng quan trọng của xử lý văn bản bao gồm tìm kiếm và truy xuất thông tin, dịch máy, tóm tắt văn bản tự động, hay kiểm lỗi chính tả tự động. Xử lý văn bản đôi khi được chia tiếp thành hai nhánh nhỏ hơn bao gồm hiểu văn bản và sinh văn bản. Nếu như hiểu liên quan tới các bài toán phân tích văn bản thì sinh liên quan tới nhiệm vụ tạo ra văn bản mới như trong các ứng dụng về dịch máy hoặc tóm tắt văn bản tự động.

1.2.4 Nhập nhằng

Trong lĩnh vực ngôn ngữ học, nhập nhằng là hiện tượng thường gặp nhất là trong giao tiếp hằng ngày của con người. Trong đó, tiếng Việt là một thứ tiếng đa nghĩa, một từ có thể mang nhiều ý nghĩa khác nhau do đó thường xảy ra các hiện tượng nhập nhằng trong khi nói hay viết, ... Chúng ta thường ít để ý đến hiện tượng này bởi vì thường xử lý tốt cho trường hợp này. Nhưng đối với các ứng dụng trong xử lý ngôn ngữ tự nhiên, khi chúng thao tác với ý nghĩa từ vựng mà điển hình là bài toán dịch văn bản (Machine Translation) thì nhập nhằng trở thành một vấn đề nghiêm trọng.



Hình 1.7: Các yếu tố trong khái niệm nhập nhằng.

Chẳng hạn như chúng ta cùng xem xét sự xuất hiện của từ “đường”, với câu “ra ngoài siêu thị mua cho mẹ một gói đường” và vấn đề phát sinh ra là từ “đường” này cần dịch là sugar hay là road. Đối với con người, chúng ta có thể dễ dàng xác định từ “đường” trong câu trên ứng với từ “sugar” do căn cứ dựa vào ngữ cảnh và các dấu hiệu nhận biết khác nhau. Nhưng đối với máy thì việc xác định ngữ cảnh hay các dấu hiệu là điều không thể. Một số hiện tượng nhập hằng có thể kể đến như:

- **Nhập hằng ranh giới từ:** do tiếng Việt là ngôn ngữ đơn lập nên từ vựng chủ yếu là các từ ghép vì thế khoảng trắng giữa các từ không phải luôn luôn là ranh giới chính xác. Ta cùng xem xét hai câu sau:
 - “He is a student”: ở câu này ta dễ dàng xác định được ranh giới dễ dàng giữa các từ He/ is/ a/ student.
 - “Anh ấy là học sinh”: nhưng đối với việc xác định ranh giới từ của câu tiếng Việt thì phải là Anh ấy/ là/ giáo viên thì mới đúng chính xác về ngữ nghĩa.
- **Nhập hằng từ đa nghĩa:** bất cứ ngôn ngữ nào cũng có từ đa nghĩa (từ mà có 2 nghĩa trở lên) và đây là vấn đề cản trở khá lớn trong lĩnh vực xử lý ngôn ngữ tự nhiên nói chung và dịch máy nói riêng. Chẳng hạn như từ “ăn” trong “ăn uống” hay “ăn cướp” vừa có nét giống và khác nhau về nghĩa và theo từ điển tiếng Việt thì từ “ăn” có đến 12 nghĩa.
- **Nhập hằng từ đồng âm:** là những từ có âm phát ra giống nhau nhưng lại mang nghĩa khác nhau hoặc những từ giống nhau về mặt ký tự nhưng nghĩa khác nhau. Ở tiếng Việt, những từ đồng âm cũng là những từ đồng tự và ở các ngôn ngữ khác thì hiện tượng này không trùng khớp nhau.
- **Nhập hằng từ loại:** từ loại là một yếu tố quan trọng trong việc xác định chính xác nghĩa của từ và sắp xếp các từ thành câu hoàn chỉnh trong dịch tự động.

1.2.5 Dịch máy

Khi nhắc đến lĩnh vực xử lý ngôn ngữ tự nhiên thì ta không thể không nhắc đến khái niệm dịch máy. Dịch máy hay còn được biết với tên tiếng Anh là Machine Translation và thường được gọi là dịch tự động, là khái niệm đã được nhiều tác giả trong lĩnh vực xử lý ngôn ngữ tự nhiên. Tuy có sự khác biệt giữa các định nghĩa đó, nhưng nhìn chung thì hầu hết đều tương đương với các định nghĩa sau:

- Hệ thống dịch máy (Machine Translation System) là một hệ thống sử dụng máy tính để chuyển đổi các câu văn bản đầu vào và ứng dụng việc xử lý trong ngôn ngữ tự nhiên để sinh ra bản dịch tương ứng với đầu vào.
- Ngôn ngữ của văn bản cần dịch được gọi là ngôn ngữ nguồn và ngôn ngữ của bản dịch được gọi là ngôn ngữ đích.
- Đầu vào của hệ thống dịch máy là một văn bản nguồn, đầu ra là văn bản đích. Kết hợp của văn bản dịch đầu ra có thể hiệu đính để bản dịch được tốt hơn và gần với ngôn ngữ tự nhiên của con người hơn.

Khái niệm dịch máy bao gồm các phương pháp sau:

- ❖ Phương pháp dịch máy dựa trên cơ sở luật (Rule Based Machine Translation - RBMT):
 - Là các hệ thống dựa trên luật do sử dụng tri thức ngôn ngữ như thông tin cú pháp, ngữ nghĩa nên việc dịch văn bản khá hiệu quả. Tuy nhiên rằng, máy tính vẫn khó có thể phân tích các cú pháp cho những câu có nghĩa quá phức tạp, và việc xây dựng ra một tập quy tắc về cú pháp và các quy tắc chuyển đổi để có thể bao quát được mọi trường hợp rất khó khăn đòi hỏi người thực hiện phải có kiến thức sâu về ngôn ngữ.
 - Quá trình dịch dựa trên cơ sở luật thực hiện phân tích cú pháp câu được nhập vào và sau đó áp dụng những luật ngôn ngữ và từ vựng (luật chuyển đổi) để ánh xạ các thông tin văn bản từ ngôn ngữ này sang ngôn ngữ khác. Do đó, ta không thể giải quyết được các trường hợp nhập nhằng ngữ nghĩa của câu có cùng cấu trúc nhưng lại mang ý nghĩa khác nhau.
 - Kết hợp giữa mức độ phân tích cú pháp và phân tích ngữ nghĩa. Phương pháp này hệ thống chủ yếu dựa vào việc phân tích cú pháp và phân giải ngữ nghĩa ở mức cần thiết để loại bỏ các trường hợp nhập nhằng về ngữ nghĩa.
- ❖ Phương pháp dịch máy dựa trên cơ sở ví dụ (Rule Based Machine Translation - RBMT):
 - Phương pháp dịch máy dựa trên mẫu ví dụ được giới thiệu lần đầu bởi Nagao vào năm 1948 trong việc nỗ lực xây dựng hệ thống dịch tự động từ tiếng Anh sang tiếng Nhật. Ý tưởng tiếp cận của phương pháp dịch này

khá đơn giản, để dịch một câu chúng ta có thể sử dụng kết quả dịch của một câu khác gần giống như vậy và thực hiện chỉnh sửa và sửa đổi đôi chút.

- Với việc tiếp cận như vậy, nó có nhiều các ưu điểm như:
 - Các ngôn ngữ nguồn và đích không cần phải được khảo sát trước về mặt ngữ pháp và từ vựng.
 - Có thể áp dụng cho bất kỳ ngôn ngữ nào với điều kiện tập dữ liệu ví dụ phải đủ lớn.
 - Với tập dữ liệu ví dụ càng lớn thì chất lượng của dịch vụ càng cao và đây là ưu thế so với một số phương pháp khác.
 - Tuy nhiên rằng, phương pháp này cũng có nhược điểm đó là phụ thuộc vào chất lượng của các cặp ví dụ được sử dụng để làm mẫu và thuật toán đổi chiều mẫu còn thực hiện khá chậm so với một số các phương pháp tiếp cận khác.
- ❖ Phương pháp dịch máy thống kê (Statistical Machine Translation):
- Là phương pháp tiếp cận dựa trên việc thống kê, được ra đời vào cuối những năm 1980 và được đề xuất bởi trung tâm nghiên cứu IBM TJ Watson với việc dịch máy từ ngôn ngữ Anh sang Pháp. Ý tưởng của phương pháp dựa trên việc ứng dụng cơ sở toán học, thay vì sử dụng các ví dụ mẫu đã có sẵn hay tự xây dựng một từ điển, ... thì phương pháp dịch máy này sẽ tự động xây dựng ra các từ điển, các quy luật dựa trên thống kê. Với cách tiếp cận này không đòi hỏi sự phân tích quá chuyên sâu về ngôn ngữ mà chúng thực hiện hoàn toàn tự động các quá trình chuyển đổi và phân tích, ...

1.2.6 Xác suất thống kê

Để có thể hiểu hơn về các khái niệm nâng cao ở các chương phía sau, ở phần này chúng ta hãy cùng ôn tập và xem xét lại các kiến thức cơ bản về xác suất thống kê để giúp ta có một cái nhìn tổng quan hơn về các chương phía sau:

❖ **Không gian mẫu và thực nghiệm:**

Trong thực tế ngày nay, ta thường gặp nhiều các thí nghiệm, quan sát mà các kết quả của nó không thể dự báo trước được. Ta gọi chúng là các phép thử ngẫu nhiên.

Phép thử ngẫu nhiên có nghĩa là chúng ta có thể liệt kê được hoặc biểu diễn tất cả các kết quả của phép thử. Mỗi kết quả của phép thử được gọi là một biến cố sơ cấp. Tập hợp tất cả các biến cố sơ cấp của phép thử thì chúng được gọi là không gian mẫu.

Ví dụ:

- Phép thử tung đồng xu có không gian mẫu là: $\Omega = \{S, N\}$
- Phép thử tung một con xúc xắc với không gian mẫu là: $\Omega = \{1,2,3,4,5,6\}$

❖ **Sự kiện (Events):**

Sự kiện A là một tập các mẫu $A \in \Omega$, và tập tất cả A là 2^Ω . Ω là sự kiện chắc chắn, \emptyset là sự kiện không xảy ra. Ví dụ: Tung đồng xu 3 lần

$$\diamond \quad \Omega = \{\text{HHH}, \text{HHT}, \text{HTH}, \text{HTT}, \text{THH}, \text{THT}, \text{TTH}, \text{TTT}\}$$

Tính các trường hợp có đúng 2 lần xuất hiện Tail. A = {HTT, THT, TTH}. Tất cả Head: A = {HHH}

❖ **Xác suất (Probability):**

Thực hiện một thực nghiệm nhiều lần: có bao nhiêu lần sự kiện A xảy ra (“count” c1). Mỗi lần thực nghiệm này gọi là dãy (hoặc bộ). Thực hiện các dãy này nhiều lần, ghi nhớ lại con số ci. Nếu thực hiện thật sự thực nghiệm nhiều lần, tỉ số ci/Ti (Ti là tổng số lần thực nghiệm trong dãy thứ i) dần tới một hằng số chưa biết. Gọi giá trị này Xác suất của A.

Kí hiệu: $p(A)$

❖ **Kỳ vọng (Expectation):** là tổng trọng số của giá trị của X, hay là giá trị trung bình của biến ngẫu nhiên. Kỳ vọng được tính bằng công thức:

$$E(X) = \sum_x p(x)$$

❖ **Phương sai (Variance):** là trung bình bình phương của độ lệch hay nói một cách đơn giản thì nó là độ lệch của biến X so với trung bình của nó. Phương sai được tính bằng công thức:

$$\text{Var}(X) = \sum_x p(x)(x - E(x))^2$$

1.3 Lý thuyết thông tin

1.3.1 Khái niệm

Để hiểu rõ hơn về các khái niệm trong lĩnh vực xử lý ngôn ngữ tự nhiên thì chúng ta sẽ tìm hiểu về các khái niệm liên quan đến lĩnh vực thông tin để có một cái nhìn tổng quan hơn về các khái niệm liên quan đến các mô hình học máy.

Lý thuyết thông tin được xây dựng và khởi xướng bởi Claude E. Shannon vào năm 1948 trong bài báo khoa học “A Mathematical Theory of Communication” với mục đích để xác định các giới hạn cơ bản trong các hoạt động xử lý tín hiệu chẳng hạn như nén dữ liệu hay lưu trữ và truyền dẫn dữ liệu. Ngày nay, phần lớn các ứng dụng của lý thuyết thông tin thường liên quan đến việc nén dữ liệu (ZIP, MP3, JPEG, ...) và ứng dụng trong mã hóa kênh (truyền số liệu qua đường dây điện thoại). Lý thuyết thông tin là một nhánh của toán học ứng dụng và kỹ thuật điện nghiên cứu về đo đạc lượng thông tin.

Entropy được sử dụng như là độ đo thông tin hay nói một cách đơn giản thì nó là số lượng bit trung bình cần thiết để cho việc lưu trữ hay truyền dữ liệu. Ngay từ những ngày đầu, lĩnh vực thông tin đã mở rộng phạm vi ứng dụng ra nhiều lĩnh vực khác, bao gồm suy luận thống kê, mật mã học, các mạng lưới bên cạnh mạng lưới viễn thông, phát hiện sao chép và các hình thức phân tích dữ liệu khác và đặc biệt là trong lĩnh vực xử lý ngôn ngữ tự nhiên.

1.3.2 Entropy

Entropy là khái niệm được dùng để đại diện cho độ đo của thông tin và nó thường được biểu diễn dưới dạng số lượng bit cần thiết trung bình để lưu trữ hoặc dẫn truyền. Entropy lượng hóa sự không chắc chắn trong việc dự đoán giá trị của một biến ngẫu nhiên. Chẳng hạn như việc xác định kết quả của một lần tung đồng xu đồng chất với hai kết quả có khả năng như nhau thì sẽ cho ít thông tin hơn (Entropy nhỏ hơn) là việc xác định kết quả của một lần tung xúc sắc với sáu kết quả có khả năng như nhau.

Nếu X là một tập hợp tất cả các thông điệp và được biểu diễn dưới dạng $\{x_1, x_2, \dots, x_n\}$ mà X có thể nhận giá trị, và $p(x)$ là xác suất X khi nhận giá trị $x \in X$ tương ứng, thì entropy của X được định nghĩa tính bằng công thức như sau:

$$H(x) = E_x[I(x)] = - \sum_{x \in X} p(x) \log p(x)$$

Với công thức tính giá trị entropy ở trên, ta thấy rằng entropy được tính hoàn toàn dựa vào xác suất. Chẳng hạn như ta có một phân phối xác suất về thời tiết bao gồm hai biến cố “nắng” và “mưa”, $P = \{p_1, p_2\}$. Người ta nhận ra Entropy(P) đạt giá trị cực đại khi $p_1 = p_2$, tức là khi p_1 và p_2 càng lệch nhau thì Entropy(P) sẽ càng giảm. Nếu $p_1 = p_2 = 0.5$, thì giá trị entropy đạt cực đại thì lúc đó ta rất khó để dự đoán thời tiết ngày mai sẽ là nắng hay mưa. Nếu $p_1 = 0.1$ và $p_2 = 0.9$, ta sẽ dễ dàng dự đoán được thời tiết ngày mai sẽ là nắng, lúc này thì giá trị của entropy sẽ có giá trị thấp.

Entropy càng cao thì điều đó đồng nghĩa với việc khó đoán trước được sự kiện sắp xảy ra. Và sự khó đoán trước đó được gọi là sự không ổn định, bất ổn và nó chính là khái niệm chính của entropy – thước đo sự “khó đoán” của thông tin.

1.3.3 Perplexity - Cross Entropy

1.3.3.1 Sự liên quan của Entropy đối với ngôn ngữ

Khái niệm Entropy thường liên quan đến sự không chính xác, chẳng hạn ta có một vấn đề mà trong đó càng có nhiều thông tin thì Entropy càng thấp và ngược lại nếu có trong đó có ít thông tin thì Entropy càng cao.

Chẳng hạn ta cũng xem xét một ví dụ, giả sử ta có một mô hình mã hóa ký tự với trung bình 2.5 bít được sử dụng trên mỗi ký tự. Đây được gọi là mô hình ngôn ngữ 0-gram, nếu ta đặt trong sự liên kết của các âm tiết (cho entropy 1.22 bít trên một ký tự) thì chúng ta có thể sinh được mô hình tốt hơn so với mô hình trước.

1.3.3.2 Perplexity

Giả sử ta có một Entropy của một phân bố $p(X)$ là: $H(p)$ thì nếu ta có giá trị $2H$ thì nó sẽ được gọi là Perplexity. Khái niệm Perplexity có thể được hiểu một cách đơn giản đó là số lượng mẫu trung bình mà một biến phải lựa chọn. Khi giá trị của Perplexity càng nhỏ đồng nghĩa với giá trị Entropy càng nhỏ thì mô hình của chúng ta càng tốt do đó số lượng bit dùng để mã hóa thông tin ngày càng bé. Để hiểu rõ hơn về Perplexity, chúng ta cùng xem xét một ví dụ sau:

Chẳng hạn chúng ta có 8 con ngựa thì xác suất để lựa chọn ra từng con ngựa như sau:

- Ngựa 1: 1/2 ngựa 2: 1/4 ngựa 3: 1/8 ngựa 4: 1/16
- Ngựa 5: 1/64 ngựa 2: 1/64 ngựa 3: 1/64 ngựa 4: 1/64

1.3.3.3 Entropy rate

Để tính giá trị Entropy của một dãy các từ trong một loại ngôn ngữ L nào đó, ta áp dụng công thức tính:

$$H(w_1, \dots, w_n) = - \sum_{W \in L} p(W) \log(p(W))$$

Entropy rate được xem như là per-word entropy. Vì nó coi một ngôn ngữ như một quá trình sản xuất ngẫu nhiên một dãy các từ. Entropy rate thường quan tâm đến một dãy vô hạn của các từ. Để tính được Entropy rate và nó được ký hiệu là $H(L)$ thì chúng ta sẽ định nghĩa nó theo công thức dưới đây:

$$H(L) = \lim_{n \rightarrow \infty} \frac{1}{n} H(w_1, \dots, w_n) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{W \in L} p(W) \log(p(w_1, \dots, w_n))$$

1.3.3.4 Cross Entropy

Như chúng ta đã tìm hiểu các khái niệm liên quan đến entropy ở phần trên, entropy là kích thước mã hóa trung bình tối thiểu theo lý thuyết cho các sự kiện tuân thủ theo một phân phối xác suất cụ thể. Tuy nhiên rằng, không phải lúc nào chúng ta cũng có thể biết trước được phân phối đó. Chẳng hạn như ví dụ về việc dự đoán thời tiết nắng hoặc mưa, phân phối mà chúng ta áp dụng thực chất chỉ là phân phối được thu về từ việc thống kê các chuỗi sự kiện đã xảy ra trong quá khứ. Và điều này sẽ không đảm bảo rằng thời tiết trong nhiều năm sắp tới cũng giống tương tự như thời tiết của năm nay. Do đó, khái niệm Cross Entropy đã được ra đời và nó được tính theo công thức sau:

$$\text{CrossEntropy}(P, Q) = \sum_i p_i * \log_2^{q_i}$$

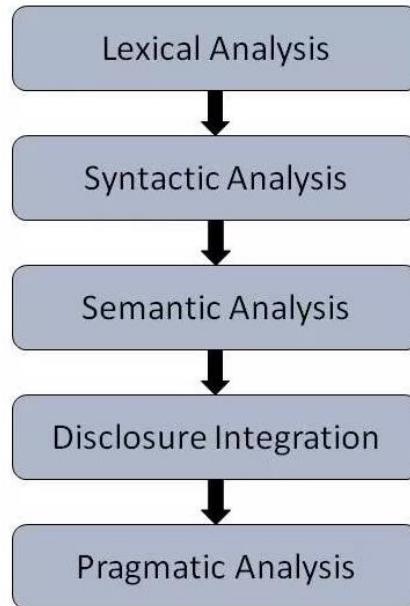
Cross Entropy có một số các tính chất đặc biệt sau:

- Đạt giá trị nhỏ nhất nếu $P = Q$.
- Rất nhạy cảm với sự sai khác giữa p_i và q_i khi p_i và q_i càng khác nhau thì giá trị của Cross Entropy tăng càng nhanh.

Do các bài toán về Machine Learning thường được quy về bài toán xây dựng mô hình (model) sao cho giá trị đầu ra của nó càng gần với target (mục tiêu đầu ra) càng tốt và thường output và target thường ở dạng phân phối xác suất. Dựa vào tính chất nhạy cảm này của Cross Entropy thì nó được sử dụng trong việc tối ưu hóa các mô hình trong học máy.

1.4 Quy trình xử lý ngôn ngữ tự nhiên

Bất cứ một việc xử lý nào cũng có một quy trình các bước được đặt ra trước để thực hiện, và trong lĩnh vực xử lý ngôn ngữ tự nhiên cũng tương tự như thế. Một quy trình xử lý ngôn ngữ tự nhiên bao gồm các bước sau:



Hình 1.8: Các bước trong quy trình xử lý của ngôn ngữ tự nhiên.

- **Lexical Analysis (Phân tích từ vựng):** là quá trình cố gắng để hiểu ý nghĩa của các từ, tìm hiểu ngữ cảnh của chúng và ghi nhận mối quan hệ của một từ với những từ khác. Nó thường là một tập hợp các từ và cụm từ trong ngôn ngữ. Việc phân tích từ vựng này bao gồm việc xác định và phân tích cấu trúc của từ và chia bộ văn bản thành các đoạn văn, câu và các từ.

Các nhiệm vụ chính của quá trình phân tích từ vựng bao gồm:

- Đọc ký tự đầu vào.
- Phát sinh ra các chuỗi dấu hiệu đầu ra.
- Bỏ khoảng trắng, cách dòng hoặc tab.
- Ghi lại vị trí các dấu hiệu được dùng cho bước xử lý tiếp theo.

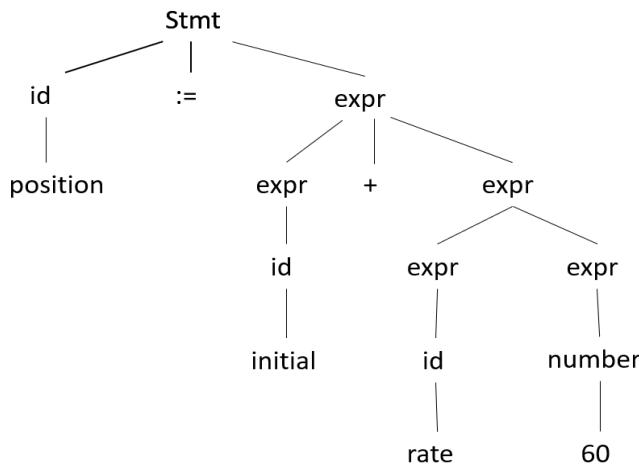
- **Syntactic Analysis (Phân tích cú pháp):** là quá trình phân tích các từ trong ngữ pháp câu và sắp xếp các từ theo cách thể hiện mối quan hệ giữa các từ. Các từ thường được chấp nhận là đơn vị cú pháp nhỏ nhất. Cú pháp đề cập đến các nguyên tắc và quy tắc chi phối cấu trúc của câu trong bất kỳ ngôn ngữ riêng lẻ nào. Và nó tập trung vào thứ tự thích hợp của các từ có thể ảnh hưởng đến ý nghĩa.

của nó. Điều này liên quan đến việc phân tích các từ trong một câu bằng cách tuân theo cấu trúc ngữ pháp. Các từ được chuyển đổi cấu trúc để chỉ ra các từ có liên quan với nhau. Chúng ta cùng xét ví dụ sau:

$\text{Stmt} \rightarrow \text{id} := \text{expr}$

$\text{expr} \rightarrow \text{expr} + \text{expr} \mid \text{expr} * \text{expr} \mid \text{id} \mid \text{number}$

Với đầu vào là: $\text{position} := \text{initial} + \text{rate} * 60$, cây phân tích cú pháp được xây dựng như sau:



Hình 1.9: Phân tích cú pháp dưới dạng cây.

Bên cạnh đó, phương pháp phân tích cú pháp bao gồm một số các phương pháp có thể kể đến như là:

- Top-Down
- Bottom-up
- CYK (Cocke Younger Kasami)
- **Semantic Analysis (Phân tích ngữ nghĩa):** là quá trình phân tích liên quan đến cấu trúc ngữ nghĩa, từ cấp độ cụm từ, mệnh đề, câu và đoạn đến cấp độ toàn bài viết với ý nghĩa độc lập của chúng. Hay nói một cách dễ hiểu thì phân tích ngữ nghĩa là quá trình dựa vào nghĩa chính xác hoặc nghĩa từ điển của văn bản. Văn bản được kiểm tra ý nghĩa và điều này được thực hiện bằng cách ánh xạ các cấu trúc cú pháp và các đối tượng trong các tác vụ miền. Chẳng hạn như cụm từ “kem nóng” thì trình phân tích ngữ nghĩa sẽ thực hiện từ chối.

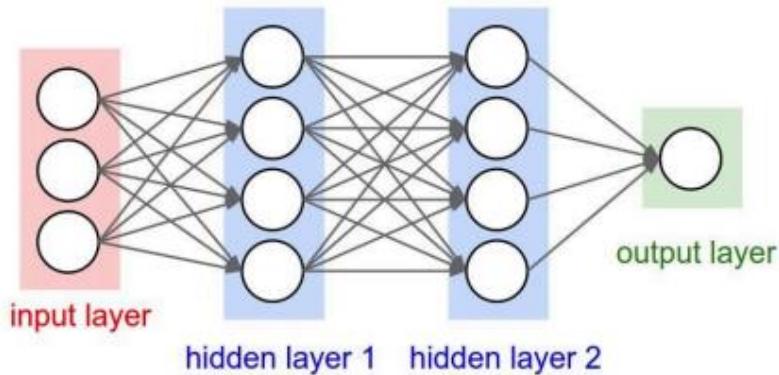
Ví dụ: Trong biểu thức position: = initial + rate * 60 thì các tên biến được khai báo là real và 60 là số nguyên vì thế nên trình biên dịch chuyển đổi số 60 ở kiểu số nguyên này thành kiểu số thực là 60.0.

- **Discourse Integration (Tích hợp bài giảng):** ý nghĩa của bất kỳ câu nào cũng phụ thuộc vào ý nghĩa của câu ngay trước nó để xem xét ý nghĩa của câu. Ngoài ra, nó kéo theo cảm giác câu trực tiếp sau. Chẳng hạn như “He want that” thì từ that sẽ phụ thuộc vào ngữ cảnh diễn ngôn trong câu trước đó.
- **Pragmatic Analysis (Phân tích thực dụng):** là quá trình đề cập đến nội dung giao tiếp và nội dung xã hội tổng thể và nó ảnh hưởng đến việc diễn giải. Nó có nghĩa là trừu tượng hóa hoặc suy ra việc sử dụng ngôn ngữ có ý nghĩa trong các tình huống. Trong quá trình phân tích này, trọng tâm chính luôn là những gì đã nói khi được giải thích lại về ý nghĩa của nó. Phân tích thực dụng giúp cho người dùng phát hiện ra hiệu ứng dự kiến này bằng cách áp dụng một bộ quy tắc đặc trưng cho các cuộc đối thoại. Chẳng hạn như câu “close the window?” nên được hiểu là một yêu cầu thay vì một mệnh lệnh.

1.5 Một số các mô hình trong ngôn ngữ tự nhiên

1.5.1 Neural Network

Neural Network là một hệ thống tính toán được lấy cảm hứng từ sự hoạt động của các nơ ron trong hệ thần kinh. Mạng Neural nhân tạo (ANN) thường được gọi đơn giản là mạng Neural (NN) bao gồm một lớp nút, chứa một lớp đầu vào, một hoặc nhiều lớp ẩn và một lớp đầu ra. Mỗi nút, hoặc nơ-ron nhân tạo, kết nối với một nút khác và có trọng số và ngưỡng liên quan. Nếu đầu ra của bất kỳ nút riêng lẻ nào vượt quá giá trị ngưỡng được chỉ định, nút đó sẽ được kích hoạt, gửi dữ liệu đến lớp tiếp theo của mạng. Nếu không, không có dữ liệu nào được chuyển đến lớp tiếp theo của mạng.



Hình 1.10: Mô hình của mạng Neural Network.

Neural nhân tạo là một đơn vị tính toán có nhiều đầu vào và một đầu ra, mỗi đầu vào đến từ một liên kết. Đặc trưng của Neural là một hàm kích hoạt phi tuyến chuyển đổi tổ hợp tuyến tính của tất cả các tín hiệu đầu vào thành tín hiệu đầu ra. Hàm kích hoạt này đảm bảo tính chất phi tuyến cho tính toán của mạng Neural.

1.5.2 Support Vector Machines

❖ Khái niệm:

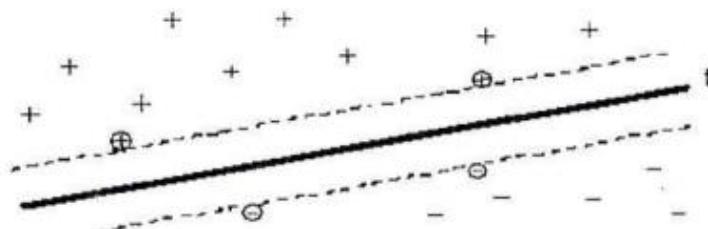
Mô hình Support Vector Machines hay còn được gọi tắt là SVM rất hiệu quả để giải quyết các bài toán với dữ liệu có số chiều lớn như các vector biểu diễn văn bản. Thuật toán SVM ban đầu nó chỉ được thiết kế để giải quyết bài toán phân lớp nhị phân tức là số lớp hạn chế là hai lớp. Hiện nay, SVM được đánh giá là bộ phân lớp chính xác nhất cho bài toán phân lớp văn bản, bởi vì đó là bộ phân lớp với tốc độ rất nhanh và hiệu quả đối với bài toán phân lớp văn bản.

Phương pháp SVM được coi là phương pháp hiệu quả để giải quyết bài toán phân lớp với dữ liệu có số chiều lớn như các vector biểu diễn văn bản. Về mặt lý thuyết, thuật toán phân lớp nhị phân này cũng có thể sử dụng cho bài toán phân lớp đa lớp bằng cách chuyển bài toán đa lớp thành bài toán nhị phân. Tuy nhiên, đối với bài toán phân lớp văn bản sử dụng phương pháp SVM thì việc lựa chọn thuộc tính cho từng phân lớp lại là vấn đề cực kỳ quan trọng, nó quyết định đến hiệu quả của phân lớp.

❖ Ý tưởng:

Một tập huấn luyện được cho trước và nó được biểu diễn trong không gian vector, trong đó mỗi tài liệu là một điểm, phương pháp này tìm ra một siêu phẳng quyết định

tốt nhất có thể chia các điểm trên không gian này thành hai lớp riêng biệt tương ứng là lớp + và lớp -. Chất lượng của siêu phẳng này được quyết định bởi khoảng cách (được gọi là biên) của điểm dữ liệu gần nhất của mỗi lớp đến mặt phẳng này. Lúc đó, khoảng cách biên càng lớn thì mặt phẳng quyết định càng tốt, đồng thời việc phân loại càng chính xác.



Hình 1.11: Mô tả thuật toán SVM.

❖ Công thức:

Bộ phân loại SVM được định nghĩa như sau:

$$f(x) = \text{sign}(C + \sum w_i x_i)$$

Trong đó:

- $\text{sign}(z) = +1$ nếu $z \geq 0$
- $\text{sign}(z) = -1$ nếu $z < 0$

Nếu $f(x) = +1$ thì x thuộc về lớp dương (lĩnh vực được quan tâm), và ngược lại, nếu $f(x) = -1$ thì x thuộc về lớp âm (các lĩnh vực khác).

1.5.3 Naïve Bayes

❖ Định nghĩa

Thuật toán Naive Bayes Classification hay còn được gọi tắt là NBC, là một thuật toán dựa trên định lý Bayes về lý thuyết xác suất để đưa ra các phán đoán và cũng như là sự phân loại dữ liệu dựa trên các dữ liệu được thống kê và quan sát. Đây là một trong những thuật toán được ứng dụng rất nhiều trong các lĩnh vực của Machine Learning và do nó khá dễ hiểu và có độ chính xác khá cao nên có thể được dùng để đưa các dự đoán chính xác nhất dựa trên một tập dữ liệu đã được thu thập từ trước đó.

❖ Định lý Bayes

Do thuật toán phân lớp Naive Bayes Classification là một giải thuật thuộc lớp giải thuật thống kê, nên nó có thể dự đoán xác suất của một phần tử dữ liệu thuộc vào một lớp là bao nhiêu. Và cơ sở của thuật toán này được dựa trên định lý Bayes.

Định lý Bayes cho phép ta tính được xác suất xảy ra của một sự kiện ngẫu nhiên A nào đó khi mà ta đã biết được sự kiện đã xảy ra B. Xác suất này được ký hiệu là $P(A|B)$ – được đọc là “xác suất của A trong điều kiện B” và ta có thể nói $P(A|B)$ là một xác suất có điều kiện.

Công thức của định lý Bayes được phát biểu như sau:

$$P(B|A) = \frac{P(AB)}{P(A)} = \frac{P(A|B)P(B)}{P(A)}$$

Trong đó:

- $P(A|B)$: là xác suất xảy ra của một biến cố (sự kiện) ngẫu nhiên A khi đã biết trước sự kiện B đã xảy ra.
- $P(B|A)$: là xác suất xảy ra B khi biết A đã xảy ra.
- $P(A)$: là xác suất xảy ra của sự kiện A
- $P(B)$: là xác suất xảy ra của sự kiện B

❖ Naive Bayes Classification

Thuật toán phân lớp Naive Bayes Classification hoạt động bao gồm các bước sau:

Bước 1: Gọi D là một tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n các giá trị thuộc tính $A_1, A_2, A_3, \dots, A_n = \{x_1, x_2, x_3, \dots, x_n\}$.

Bước 2: Giả sử ta có m lớp C_1, C_2, \dots, C_m , ta cho một phần tử dữ liệu X bộ phân lớp sẽ gán nhãn cho X là lớp có xác suất hậu nghiệm lớn nhất. Nghĩa là thuật toán phân lớp Naive Bayes Classification sẽ dự đoán X thuộc vào lớp C_i khi và chỉ khi:

$$P(C_i|X) > P(C_j|X) \quad (1 \leq i, j \leq m, i \neq j)$$

Bước 3: Để tìm được xác suất lớn nhất, ta nhận thấy các giá trị $P(X)$ là giống nhau với mọi lớp nên không cần tính. Do đó, ta cần tìm giá trị lớn nhất của $P(X|C_i) * P(C_i)$. Trong đó, $P(C_i)$ được ước lượng bằng $|D_i|/|D|$, trong đó D_i là tập các phần tử dữ liệu thuộc lớp C_i . Nếu xác suất tiền nghiệm $P(C_i)$ cũng không xác định được thì ta coi chúng bằng nhau $P(C_1) = P(C_2) = \dots = P(C_m)$, khi đó ta chỉ cần tìm giá trị $P(X|C_i)$ lớn nhất.

Bước 4: Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toán $P(X|C_i)$ là rất lớn, do đó ta có thể giảm độ phức tạp của thuật toán Naive Bayes và giả thiết các thuộc tính là độc lập nhau. Khi đó ta có thể tính bằng công thức:

$$P(X|C_i) = P(X_1|C_i) \dots P(X_n|C_i)$$

❖ Các bước thực hiện

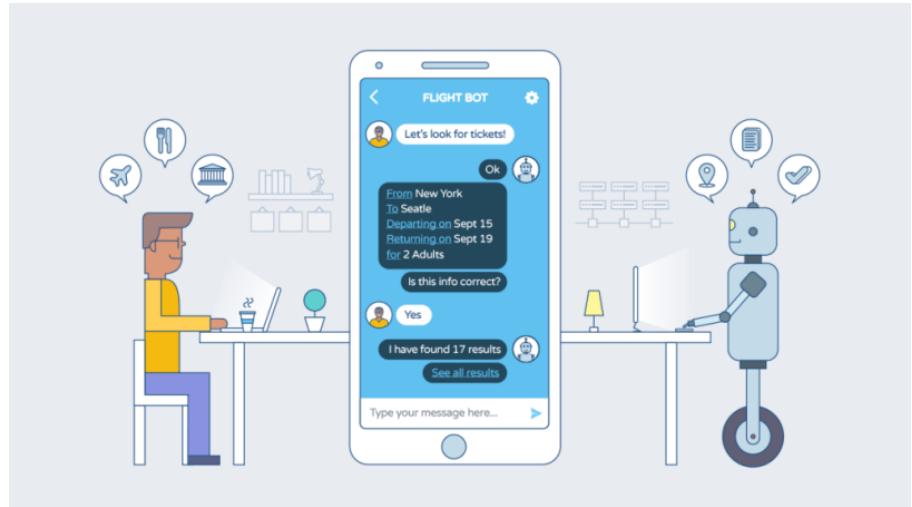
Quá trình thực hiện phương pháp Naive Bayes nhìn chung bao gồm các bước sau:

- Xử lý dữ liệu đầu vào
- Tóm tắt dữ liệu.
- Dự đoán kết quả.
- Đánh giá độ chính xác.

1.6 Các ứng dụng của xử lý ngôn ngữ tự nhiên

Hiện nay, có rất nhiều các ứng dụng trong thực tế triển khai việc xử lý ngôn ngữ tự nhiên để phục vụ cho nhiều mục đích khác nhau. Ta có thể kể đến một số các ứng dụng nổi bật và thành công của xử lý ngôn ngữ tự nhiên như sau:

- **Ứng dụng trong hệ thống trả lời tự động (Chatbots):** đây là một trong những ứng dụng nổi bật của NLP. Chatbots có hiệu quả cao ở khía cạnh kinh doanh và người tiêu dùng giúp cho việc trả lời các câu hỏi truy vấn khác nhau của người dùng khi cần thiết. Tuy nhiên, các doanh nghiệp ngày nay cần phải đẩy mạnh hơn việc phát triển hệ thống chatbot để có thể giao tiếp ở cấp độ con người với độ phức tạp cao hơn. Chatbots hiện tại rất hữu ích cho doanh nghiệp khi nói đến việc cải thiện sự trải nghiệm và hài lòng của khách hàng.



Hình 1.12: Ví dụ minh họa cho ứng dụng Chatbot trong NLP.

- **Ứng dụng trong giám sát mạng xã hội:** đây là một ứng dụng thành công và đang được ứng dụng nhiều trong thực tế. Nhiệm vụ của việc giám sát là tiến hành tìm hiểu và phân tích ý kiến của người dùng về bất cứ sản phẩm hoặc dịch vụ trên các trang mạng xã hội phổ biến như Facebook, ... Từ đó, NLP có thể nắm bắt được sự hài lòng hoặc không hài lòng của người dùng từ đó giúp cho nhà phát triển có hướng xử lý tốt hơn.



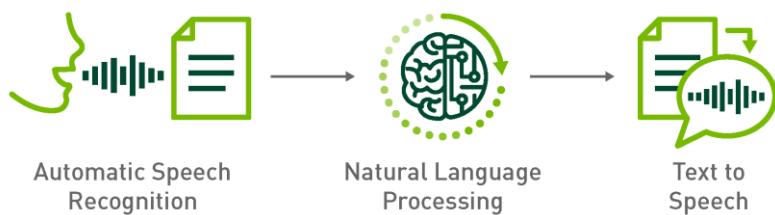
Hình 1.13: Ví dụ về ứng dụng giám sát mạng xã hội.

- **Dịch máy (Machine Translation):** đây là một ứng dụng rộng rãi trong lĩnh vực NLP. Nhờ vào khả năng dịch tự động được phát triển trong các thuật toán nên không cần con người tham gia vào quá trình này. Một số các ứng dụng nổi tiếng để dịch máy hiện nay là Google Dịch và Amazon Dịch.



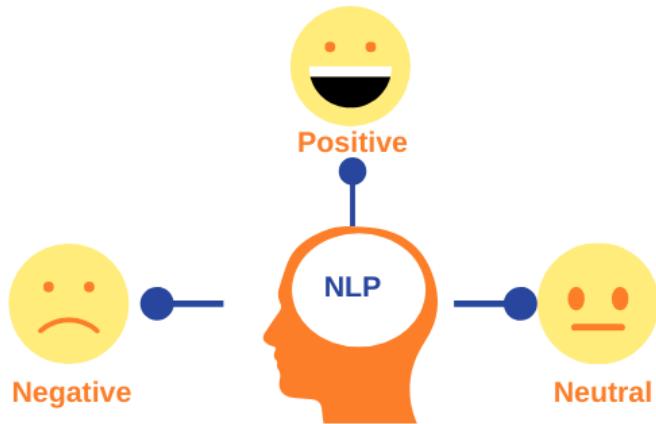
Hình 1.14: Ví dụ về Google Translate được ứng dụng trong dịch máy.

- **Nhận dạng giọng nói (Speech Recognition):** đây là ứng dụng đã xuất hiện rất lâu từ vài thập kỷ gần đây, được sử dụng phổ biến trong việc giải mã giọng nói của các thiết bị di động như điện thoại thông minh, tự động hóa trong gia đình, ... Các công cụ thực hiện nhận dạng giọng nói được phát triển với sự hỗ trợ của NLP đã được ứng dụng rộng rãi trong các công ty khi chúng tạo ra các giao diện điều khiển giọng nói thông minh cho các hệ thống trong từng lĩnh vực kinh doanh của họ.



Hình 1.15: Ví dụ về ứng dụng nhận dạng giọng nói trong NLP.

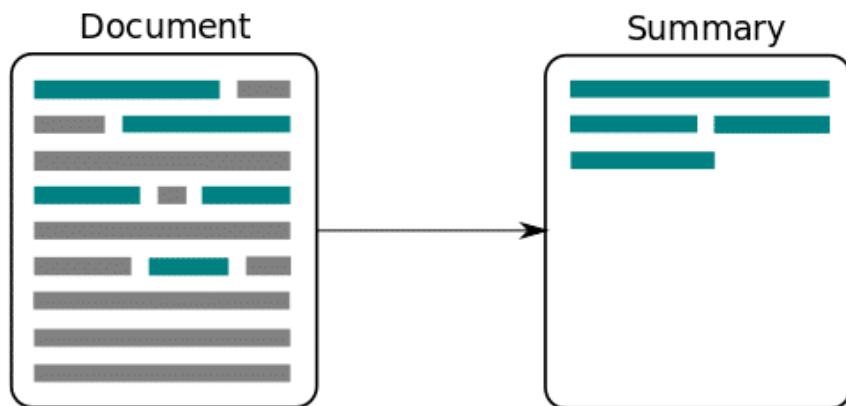
- **Phân tích cảm xúc (Sentiment Analysis):** nó giải quyết khuynh hướng của mọi người về các chủ đề hoặc dịch vụ nhất định. Nó chịu trách nhiệm kiểm tra xem sự hài lòng của khách hàng về dịch vụ hoặc sản phẩm. Giúp cho công ty có được kiến thức về cách khách hàng cảm nhận đối với họ và giúp cải thiện chất lượng giúp nâng cao chất lượng hài lòng.



Sentiment Analysis

Hình 1.16: Hình ảnh minh họa về phân tích cảm xúc trong NLP.

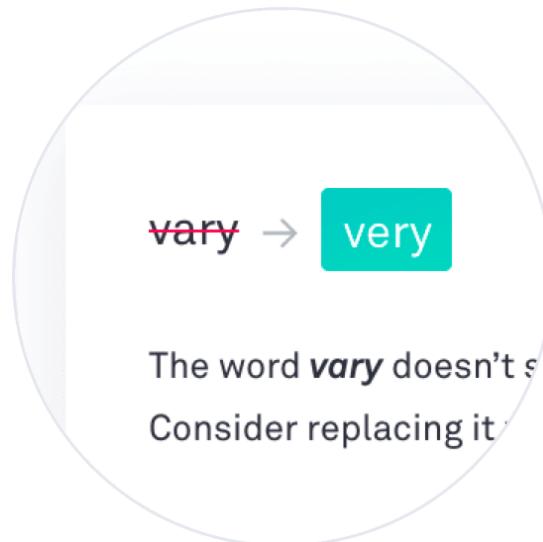
- **Tóm tắt văn bản (Text Summarization):** với lượng thông tin mà chúng ta cần xử lý hằng ngày thì việc quá tải thông tin có thể là vấn đề lớn trong xã hội ngày nay. Ứng dụng tóm tắt văn bản tự động được tạo ra nhằm mục đích tạo ra một bản tóm tắt ngắn gọn, chính xác và trôi chảy từ một văn bản dài đầu vào. Nhờ vào tóm tắt tự văn bản tự động, nó giúp cho chúng ta giảm thời gian đọc và ngoài ra kết quả của nó có thể làm đầu vào cho nhiều ứng dụng hoặc nhiệm vụ khác trong NLP.



Hình 1.17: Ví dụ về việc tóm tắt văn bản tự động trong NLP.

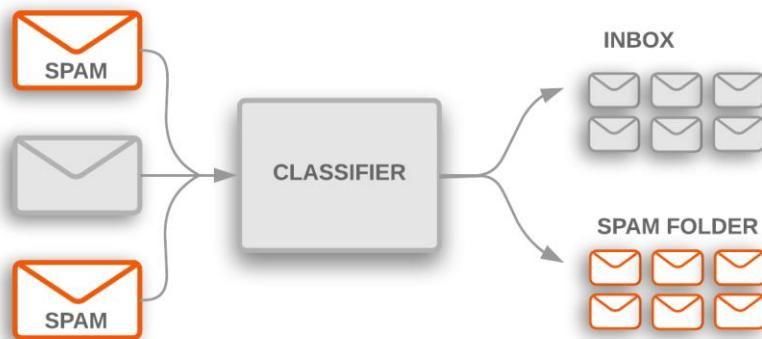
- **Kiểm tra chính tả (Spell Checking):** đây là ứng dụng giúp cho việc tìm tự động tìm ra lỗi sai và sửa các lỗi chính tả đó. Thông thường, người viết ra văn bản rất khó để có thể tìm ra lỗi sai của mình đặc biệt là những văn bản với kích thước

dài. Với ứng dụng kiểm tra chính tả của NLP sẽ giúp cho chúng ta thực hiện việc sửa các lỗi chính tả đó một cách tự động.



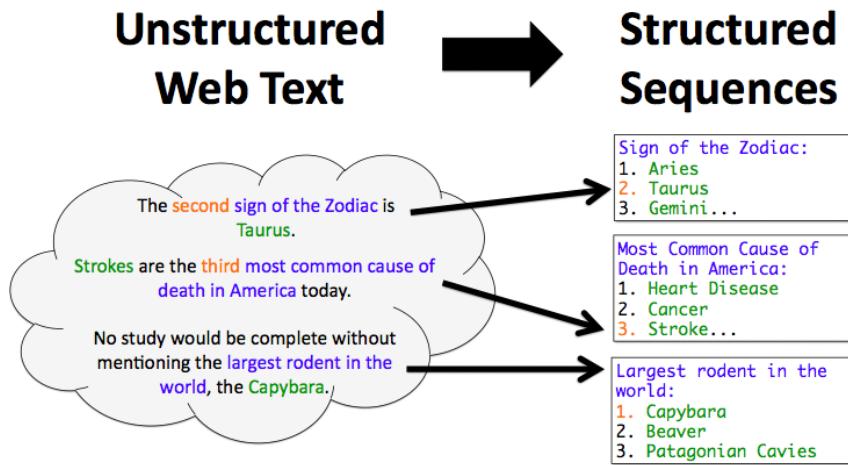
Hình 1.18: Ví dụ về ứng dụng kiểm tra chính tả của NLP.

- **Bộ lọc thư rác:** đây cũng là một trường hợp sử dụng phổ biến của ngôn ngữ tự nhiên, ứng dụng này chặn các email không mong muốn và lọc ra những email không phải là spam bằng cách trích xuất ý nghĩa và tần suất của một số các từ nhất định được phát hiện trong nội dung mail.



Hình 1.19: Ví dụ về mô hình phân loại thư rác.

- **Trích xuất thông tin:** ứng dụng này cho phép thu thập và trích xuất thông tin liên quan đã được đổi chiều một cách nhanh chóng để có thể sử dụng được hiệu quả trong việc phát triển cũng như là cải tiến các doanh nghiệp khác nhau. Khai thác thông tin sử dụng dữ liệu ở dạng phi cấu trúc và từ dữ liệu được thu thập từ nhiều nguồn khác nhau thì nó chuyển đổi thành dữ liệu có thể truy cập được.



Hình 1.20: Ví dụ về mô hình trích xuất thông tin trong NLP.

1.5 Tổng kết chương 1

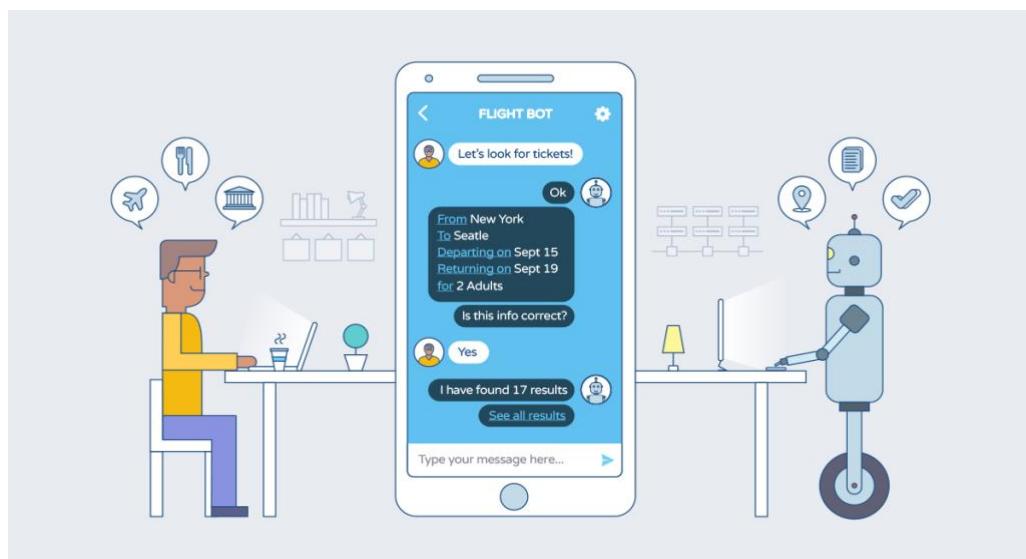
Qua chương 1 của bài báo cáo, ta đã tìm hiểu các khái niệm cơ bản liên quan đến đến lĩnh vực xử lý ngôn ngữ tự nhiên như khái niệm về ngôn ngữ tự nhiên, lịch sử hình thành của nó hay một số các khái niệm liên quan đến xử lý ngôn ngữ tự nhiên, các mô hình nổi bật thường được sử dụng cho đến các ứng dụng của nó, ... Đây chỉ mới là chương khởi đầu, giúp nhắc lại cho chúng ta các khái niệm cơ bản nhất để phục vụ việc xử lý các thông tin bên dưới các chương. Kế tiếp, chúng ta sẽ tiếp tục tìm hiểu đến các khái niệm về hệ thống trả lời tự động (Chatbots) nói chung cũng như là các khái niệm sâu hơn về mạng neural nhân tạo ứng dụng trong bài toán trả lời tự động đó.

CHƯƠNG 2: TỔNG QUAN VỀ HỆ THỐNG TRẢ LỜI TỰ ĐỘNG CHATBOT

2.1 Tổng quan về Chatbot

2.1.1 Khái niệm

Hệ thống trả lời tự động hay còn được biết với cái tên là Chatbot, là một hình thức thô sơ của trí tuệ nhân tạo, là một chương trình được tạo ra từ máy tính và thực hiện các cuộc trò chuyện với con người thông qua các phương pháp như nhập văn bản, âm thanh, cảm ứng có thể trả lời các câu hỏi và xử lý các tình huống, nó là một công cụ có thể giao tiếp, tương tác với con người thông qua một trí tuệ nhân tạo đã được lập trình sẵn. Hay nói một cách dễ hiểu thì Chatbot là một chương trình máy tính có khả năng giao tiếp tự động với con người bằng cách trả lời các câu hỏi hoặc xử lý các tình huống khác nhau. Khả năng thông minh của Chatbot được xác định phụ thuộc vào thuật toán mà người tạo nên chúng sử dụng. Chatbot hiện nay được ứng dụng trong rất nhiều các lĩnh vực khác nhau của cuộc sống đặc biệt là thương mại điện tử, chăm sóc dịch vụ khách hàng, tài chính ngân hàng, giải trí, giáo dục, ...



Hình 2.1: Ứng dụng trả lời tự động chatbot.

Trong hầu hết các trường hợp thì Chatbot được ứng dụng để sử dụng trong việc nhắn tin để giao tiếp và trò chuyện với con người. Nó có khả năng trả lời những câu hỏi mà người dùng đặt ra và việc trả lời các câu hỏi của con người thì chúng thường dựa vào những từ khóa trong câu hỏi và dần dần nó sẽ học hỏi được thêm từ trải nghiệm với

người dùng và làm cho các cuộc trò chuyện với con người trong tương lai trở nên cá nhân hơn, hay có thể làm cho chúng ta cảm nhận đang giao tiếp với người thật hơn.

Chatbot có thể được phân chia thành hai loại cơ bản:

- **Miền mở (Open Domain):** là mô hình cho phép người dùng có thể tham gia trò chuyện với một đề tài hay chủ đề bất kỳ mà mình mong muốn và không bắt buộc phải có một mục tiêu rõ ràng hay một ý định cụ thể nào. Chẳng hạn như các cuộc trò chuyện trên các trang mạng xã hội như Facebook, Zalo thường là miền mở và chúng có thể đi thực hiện trả lời bất kỳ chủ đề nào. Với mô hình miền mở, nó cung cấp số lượng các chủ đề thảo luận được đề cập đến là không giới hạn. Vì thế, các tri thức yêu cầu được tạo ra để trả lời cho các câu đố thoại thuộc miền mở trở nên khó hơn do bao quát nhiều lĩnh vực và nhiều trường hợp hơn. Tuy nhiên rằng, nếu sử dụng mô hình miền mở thì việc thu thập hay trích rút dữ liệu từ mô hình này sẽ khá phong phú và đơn giản.
- **Miền đóng (Close Domain):** là một mô hình thường tập trung vào trả lời các câu hỏi đố thoại liên quan đến một chủ đề hay lĩnh vực nhất định nào đó. Chẳng hạn như chủ đề liên quan đến mua sắm, giáo dục, du lịch, tình yêu, ... Trong mô hình miền đóng này thì không gian các mẫu câu hỏi đầu vào và các câu trả lời đầu ra là có sự giới hạn. Sự giới hạn đó là bởi vì các hệ thống này chủ yếu được xây dựng để cố gắng để đạt được một mục tiêu rất cụ thể và chỉ quan tâm đến lĩnh vực đó. Hiện nay, chúng ta có thể kể đến một số hệ thống Chatbot thuộc miền đóng như hệ thống hỗ trợ kỹ thuật hay tư vấn và hỗ trợ mua hàng trên các nền tảng bán hàng trực tuyến như Shopee, Lazada, Tiki, ... Do đặc điểm là miền đóng và giới hạn chỉ có một chủ đề nên các hệ thống này không thể đố thoại liên quan đến các lĩnh vực khác giống như mô hình miền mở mà việc mô hình này cần thực hiện đó là đảm bảo sao cho việc trả lời các câu hỏi của người dùng đạt hiệu quả nhất có thể. Chắc chắn, người dùng vẫn có thể hỏi đáp bất cứ điều gì, nhưng hệ thống sẽ không yêu cầu phải xử lý những trường hợp ngoại lệ này.

2.1.2 Lịch sử

Phép thử Turing được ra đời năm 1950 bởi Alan Turing. Phép thử Turing là một bài kiểm tra khả năng trí tuệ của máy tính. Mô hình chuẩn của phép thử Turing, trong đó người chơi C, đóng vai trò là người chất vấn, có nhiệm vụ xác định người chơi A và

B, bên nào là máy tính, bên nào là con người bằng cách đặt các câu hỏi và nhận câu trả lời từ A và B. Phép thử Turing dựa trên giả thiết rằng người ta có thể đánh giá tính "thông minh" của máy tính bằng cách so sánh hành vi của nó với hành vi của con người.

Năm 1956, một hội thảo nghiên cứu mang tên Dartmouth về Trí tuệ nhân tạo được tổ chức, thiết lập AI là một lĩnh vực nghiên cứu. Hội thảo được tổ chức bởi Trợ lý Giáo sư Toán học John McCarthy và được kéo dài khoảng 6 đến 8 tuần.

Chatbot đầu tiên ra đời năm 1960, tên là Eliza, và là một chương trình máy tính của Joseph Weizenbaum (Viện Công nghệ Massachusetts, Mỹ). Chương trình được thiết kế theo cách bắt chước cuộc trò chuyện của con người. Chatbot Eliza hoạt động bằng cách chuyển các từ mà người dùng đã nhập vào máy tính và sau đó ghép nối chúng vào danh sách các câu trả lời có kịch bản. Nó sử dụng một kịch bản mô phỏng một nhà tâm lý trị liệu.

Parry được xây dựng bởi bác sĩ tâm thần người Mỹ Kenneth Colby vào năm 1972. Chương trình bắt chước một bệnh nhân tâm thần phân liệt. Nó cố gắng để mô phỏng bệnh và là một chương trình ngôn ngữ tự nhiên tương tự như suy nghĩ của một cá nhân. Parry hoạt động thông qua một hệ thống phức tạp các giả định, phân bổ và “phản ứng cảm xúc” được kích hoạt bằng cách thay đổi trọng số được gán cho các đầu vào bằng lời nói.

Chinese Room (còn được hiểu là căn phòng tiếng Trung Quốc) là một thí nghiệm tưởng tượng của nhà triết học người Mỹ John Searle. Thí nghiệm này được đề xuất vào năm 1980 nhằm thách thức cái gọi là trí thông minh nhân tạo. Searle tưởng tượng rằng mình ở trong phòng gồm những hộp đựng chữ Trung Quốc. Ông hoàn toàn không biết nghĩa của những chữ này, nhưng bên cạnh chúng là một cuốn sách hướng dẫn về tiếng Trung Quốc. Nếu như có một người nào đó nói tiếng Trung Quốc nói chuyện với ông qua cửa căn phòng trên thì ông có thể dựa vào sự hướng dẫn của cuốn sách đó để trò chuyện lại với người đó cũng qua cửa của căn phòng.

Dr. Sbaits là một chatbot được tạo ra bởi Creative Labs cho MS-Dos vào năm 1992. Đây là một trong những nỗ lực sớm nhất của việc kết hợp AI vào một chatbot và được công nhận cho chương trình trò chuyện có lời thoại đầy đủ. Chương trình “trò chuyện” với người dùng như một nhà tâm lý học, hầu hết các câu hỏi thường là “Bạn cảm thấy như thế nào?” chứ không phải là những tương tác phức tạp. Khi gặp phải những câu thoại phức tạp, Dr. Sbaits thường trả lời “Không phải là vấn đề của tôi”.

Alice là một chatbot xử lý ngôn ngữ phổ thông sử dụng mẫu tương tác heuristic để thực hiện các cuộc hội thoại. Trong đó, Heuristic là các thuật toán dựa trên kinh nghiệm để giải quyết vấn đề, học hỏi hay khám phá nhằm đưa ra một giải pháp ở mức nhận thức thông thường. Năm 1995, Richard Wallace đi tiên phong trong việc xây dựng Alice.

Jabberwacky được tạo ra bởi lập trình viên người Anh Rollo Carpenter năm 1997 là một trong những hình thức sớm nhất của AI dựa trên cuộc trò chuyện của con người. Được xây dựng chủ yếu như một hình thức giải trí, Carpenter cũng dự định Jabberwacky có khả năng vượt qua bài kiểm tra Turing.

SmarterChild, bot thông minh được phát triển bởi ActiveBuddy năm 2001, được phát hành trên các nền tảng tin nhắn, SMS và nhanh chóng trở nên phổ biến. SmarterChild được đánh giá cao qua việc đứng đầu danh sách bạn bè AIM của hàng triệu trẻ em và người lớn trên khắp thế giới cho đến khi công nghệ được xếp sau khi Microsoft mua lại công ty.

Googly Minotaur, một bot AOL Instant Messenger, được phát triển bởi ActiveBuddy vào năm 2001 để quảng cáo cho album thứ năm của Radiohead, Amnesiac. Bản phát hành của nó đánh dấu một trong những phiên bản đầu tiên của chương trình được sử dụng cho các phương tiện thương mại.

Siêu máy tính của IBM Watson, được đặt tên theo CEO đầu tiên của công ty, được phát triển vào năm 2006 với khả năng trả lời các câu hỏi được đặt ra bằng ngôn ngữ tự nhiên.

Siri, trợ lý thoại cá nhân thông minh của Apple, được phát triển bởi Siri vào năm 2010 và được phát hành dưới dạng ứng dụng độc lập. Sau khi được Apple mua lại trong năm đó, chương trình đã được tích hợp vào iOS, với khả năng tương tác với một số ứng dụng mặc định của Apple.

Google Now (2012), trợ lý cá nhân thông minh của Google, được phát hành cho Android. Sử dụng giao diện người dùng ngôn ngữ tự nhiên, bot có thể trả lời các câu hỏi, đưa ra các đề xuất và thực hiện các hành động trên các dịch vụ web khác nhau.

Alexa là một trợ lý cá nhân thông minh được phát triển bởi Amazon. Nó được giới thiệu vào năm 2014 và hiện được tích hợp vào các thiết bị như Amazon Echo, Echo Dot, Echo Show và nhiều hơn nữa.

Tương tự như với Siri của Apple, Microsoft phát hành trợ lý cá nhân thông minh của riêng họ, Cortana. Cortana có sẵn trong nhiều ngôn ngữ, Cortana phục vụ như là một thành phần quan trọng của hệ điều hành của Microsoft "makeover".

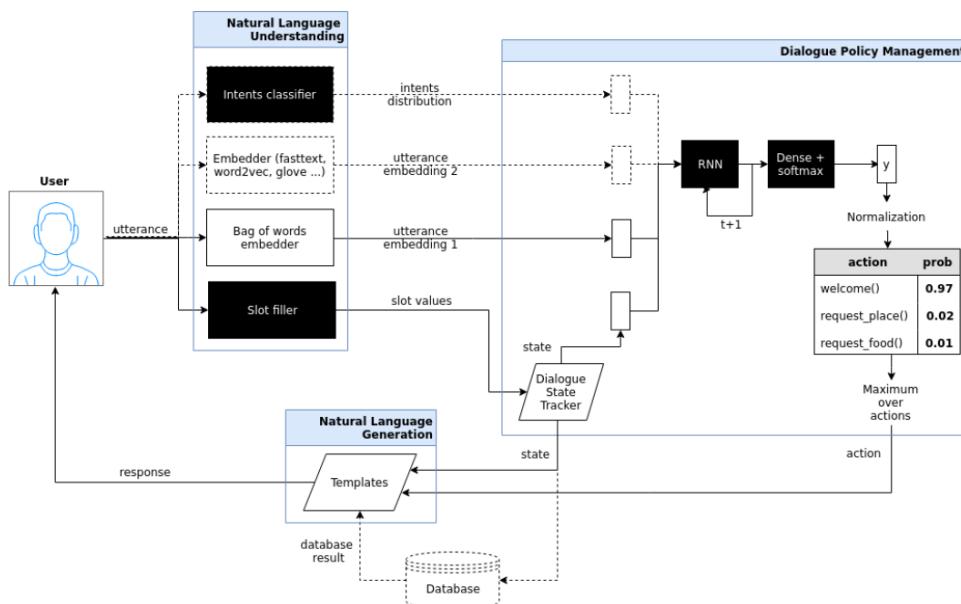
Microsoft phát hành một chatbot thông minh, có tên là Tay (2016), trên Twitter dưới sự quản lý @TayandYou. Được thiết kế để bắt chước các mẫu ngôn ngữ của một cô gái mười chín tuổi và học hỏi từ việc tương tác với người dùng Twitter, Tay sớm được biết đến với tên gọi "The AI with zero Chill" khi nó bắt đầu thể hiện hành vi xúc phạm.

Bots ở khắp mọi nơi. Facebook thông báo một nền tảng để xây dựng bot cho Messenger, và hàng chục ngàn chương trình được tạo ra trong vòng vài tháng. Các dịch vụ nhắn tin khác, như Slack, Telegram và Kik cũng làm như vậy. Sau đó trong năm, Apple mở iMessage cho các nhà phát triển bên thứ ba. Bots đã chính thức được tiếp cận gần hơn.

Roko Labs (2017) khởi chạy Instabot, một nền tảng để tạo chatbot của riêng bạn mà bạn có thể khởi chạy trên các ứng dụng di động, trang web và email hiện có.

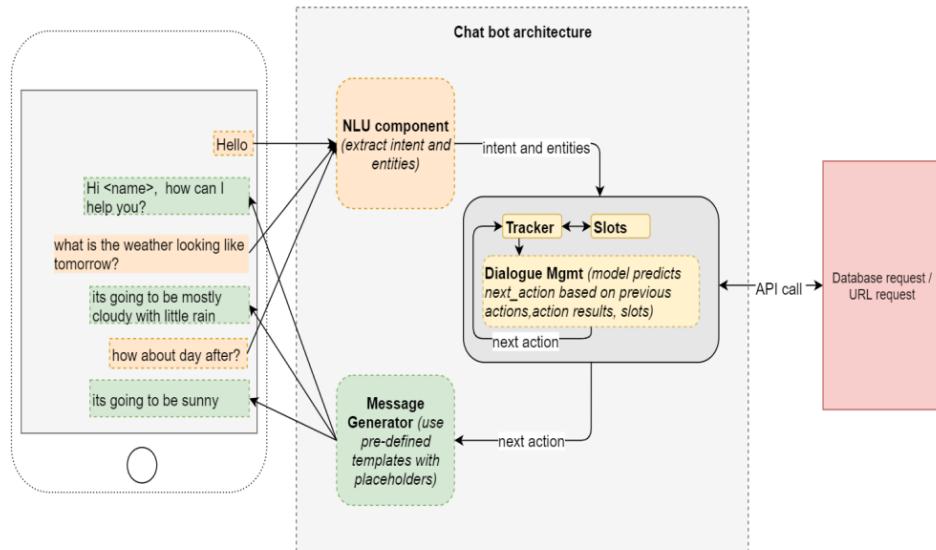
2.1.3 Cấu tạo của hệ thống Chatbot

Một hệ thống Chatbot thông thường gồm có các thành phần kết hợp lại với nhau để tạo nên một hệ thống hoàn chỉnh và tương ứng với mỗi thành phần đó thì sẽ đảm nhận một nhiệm vụ với vai trò khác nhau. Chúng ta cùng xem xét hình 2.2 bên dưới để phân tích và đánh giá nhiệm vụ cho từng thành phần trong hệ thống.



Hình 2.2: Các thành phần cấu tạo nên hệ thống Chatbot.

- **Thành phần hiểu ngôn ngữ tự nhiên (Natural Language Understanding):** bao gồm những tác vụ liên quan đến việc xử lý ngôn ngữ tự nhiên có nhiệm vụ xác định được ý định câu hỏi (Intent Classification), chuyển đổi câu hỏi đầu vào thông qua các lớp Embedder hay sử dụng Bag of words cho đầu vào và trích chọn thông tin (Slot Filter).
- **Thành phần quản lý hội thoại (Dialogue Manager):** là thành phần có nhiệm vụ xác định được hành động tiếp theo dựa vào trạng thái hành động trước đó hoặc dựa vào ngữ cảnh phía trước để đưa ra câu trả lời cho câu hỏi tiếp theo. Các ngữ cảnh này thường được đào tạo và huấn luyện sẵn trong các kịch bản.
- **Thành phần sinh ngôn ngữ (Natural Language Generation):** là thành phần chịu trách nhiệm sinh ngôn ngữ dựa vào chính sách và hành động được xác định trong thành phần quản lý hội thoại thông qua các tập hội thoại. Thành phần này có thể được sinh ra câu trả lời dựa vào tập mẫu câu trả lời đã được xây dựng sẵn dựa trên các mô hình đã đào tạo cho chương trình Chatbot.



Hình 2.3: Quy trình xử lý của các thành phần trong hệ thống Chatbot.

Với hình 2.3, ta thấy rằng khi người dùng nhập vào một câu hỏi thì hệ thống sẽ chuyển câu hỏi của người dùng đến thành phần NLU – thành phần hiểu ngôn ngữ tự nhiên để xác định được ý định và thực thể trong câu hỏi. Kế tiếp, hệ thống sẽ chuyển thông điệp được xử lý ở thành phần NLU đến thành phần điều khiển hội thoại DM. Trong thành phần DM, nó sẽ sử dụng mô hình để dự đoán các hành động tiếp theo và thực hiện gọi đến cơ sở dữ liệu để lấy ra dữ liệu ứng với kết quả được dự đoán. Và cuối

cùng, từ thành phần điều khiển hội thoại thì sẽ chuyển đến thành phần sinh ngôn ngữ - NLG bằng cách sử dụng các mẫu đã được định nghĩa trước để thay thế cho các dự đoán từ thành phần DM.

2.2 Hiểu ngôn ngữ tự nhiên

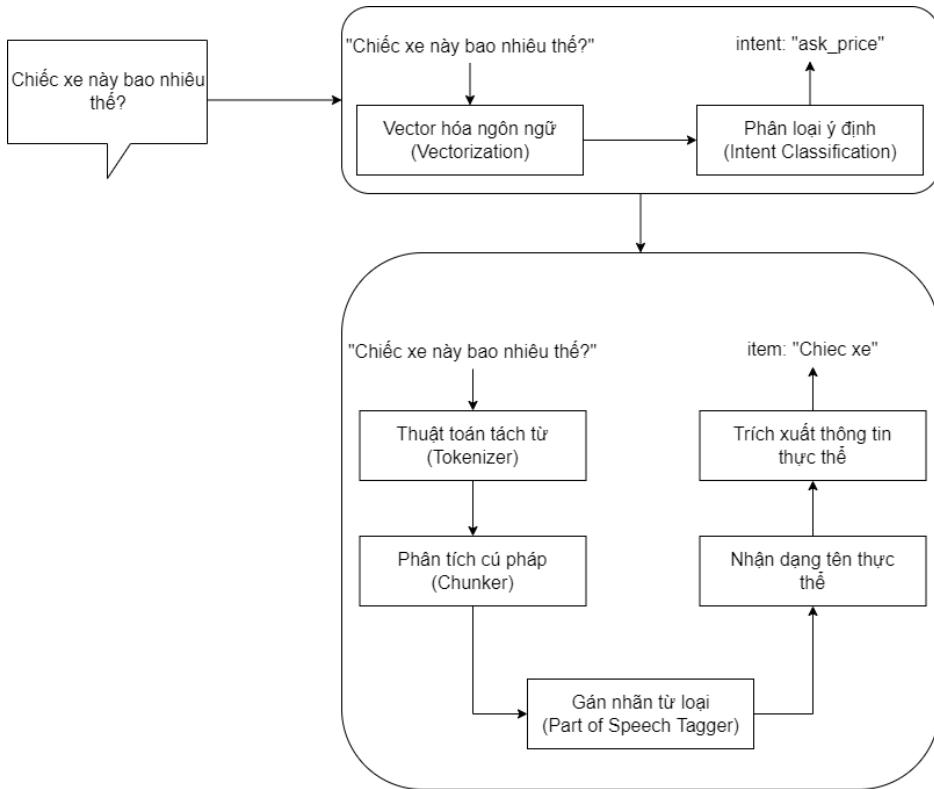
Đối với một hệ thống Chatbot, thành phần hiểu ngôn ngữ tự nhiên có thể nói là thành phần quan trọng nhất của hệ thống. Việc xác định rằng một hệ thống có thực sự thông minh hay không chính là dựa vào thành phần quyết định này. Nhiệm vụ của thành phần này là thực hiện việc phân tích và chuyển đổi tin nhắn hay thông điệp đầu vào theo nguyên tắc của ngôn ngữ tự nhiên. Và ở đây, nó sẽ đảm nhận việc trích xuất ra hai thành phần thông tin từ người dùng:

- **Phân loại ý định (Intent Classification):** là việc phân loại xem câu hỏi của người dùng thuộc về lĩnh vực hay phạm vi nào. Chẳng hạn như hỏi về dịch vụ giao hàng, giá cả hàng hóa, khuyến mãi, ...
- **Trích xuất thông tin (Slot Filter hay Entity Extraction):** là việc xác định xem thông tin nào liên quan đến ý định đã được xác định ở phía trên. Chẳng hạn, ở trên chúng ta đã xác định được ý định của người dùng là về việc hỏi giá cả. Nhưng chúng ta chưa biết giá cả đó là thuộc về loại hàng hóa nào. Ở bước trích xuất thông tin này, cho ta biết được thông tin về thành phần mà người dùng có ý định hỏi. Ví dụ như câu “Chiếc xe này bao nhiêu tiền?”, thì việc trích xuất thông tin chính là xác định “Chiếc xe”. Nên từ đó, hệ thống mới có cơ sở để trả lời cho người dùng.



Hình 2.4: Các bước xử lý chính trong thành phần NLU.

Trong các bước xử lý chính của NLU, chúng ta có thể thay đổi và tùy chỉnh các thành phần và các bước tiền xử lý dữ liệu, các thuật toán hay phương pháp để tách từ và trích xuất các thực thể khác nhau. Tùy thuộc vào nhu cầu và mục đích của người sử dụng.



Hình 2.5: Các bước xử lý lý chính trong NLU.

Với hình 2.5, ta thấy rằng khi một câu được đưa vào thành phần hiểu ngôn ngữ tự nhiên thì nó sẽ chuyển đổi thông điệp đầu vào thành dạng vector và phân loại được ý định của người dùng. Sau đó sẽ chuyển đến bước kế tiếp là trích xuất thông tin thực thể, ở bước này thì nó sẽ thực hiện tách các từ ra trong câu đầu vào và tiến hành phân tích ngữ nghĩa các từ đó. Sau khi phân tích xong, nó tiến hành gán nhãn cho từ loại (POS Tag) và tiến hành nhận dạng tên của thực thể và cuối cùng là trích xuất ra được thông tin của thực thể.

2.2.1 Phân loại ý định

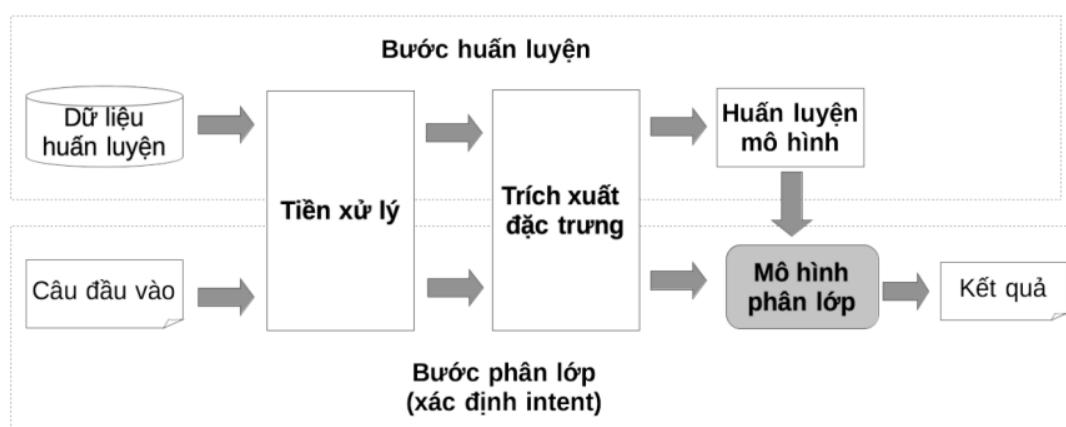
Để thực hiện phân loại ý định của một câu hỏi đầu vào của người dùng, chúng ta cần mô hình hóa ngôn ngữ bằng cách biểu diễn ngôn ngữ đầu vào đó dưới dạng một vector số học để máy tính hiểu và quá trình này được gọi là vector hóa. Phương pháp vector hóa phổ biến và thông dụng nhất hiện nay là Word Embedding được hiểu là nhúng từ. Word Embedding là tên chung cho một tập hợp các mô hình và phương pháp ngôn ngữ dành riêng cho xử lý ngôn ngữ tự nhiên, trong đó các từ hoặc cụm từ vựng được ánh xạ tới các vector số thực. Về mặt khái niệm, Word Embedding liên quan đến việc nhúng toán học từ một không gian có một chiều cho mỗi từ vào không gian vectơ liên

tục với các kích thước thấp hơn nhiều. Ta có thể kể đến một số các phương thức đại diện phổ biến như Word2Vec, FastText hay GloVe, ...

Sau khi đã thực hiện việc mô hình hóa ngôn ngữ như việc bao gồm dữ liệu đầu vào việc huấn luyện cho bot thì việc xác định mô hình mô hình dùng để huấn luyện cho bot cũng ảnh hưởng và quan trọng không kém đến kết quả đầu ra. Ở hệ thống Chatbot này, chúng ta có thể sử dụng một số các mô hình như:

- Convolution Neural Network (CNN)
- Recurrent Neural Network (RNN).
- Long Short-Term Memory (LSTM, Bi-LSTM).
- Naive Bayes.
- Vector Support Machine (SVM).
- Decision Tree (Random Forest).

Gần như hầu hết các hệ thống Chatbot ở thời điểm hiện tại đều sử dụng mô hình Deep Learning như mạng nơ ron hồi quy như RNN và LSTM để sử dụng cho việc phân loại ý định người dùng. Thủ thách lớn nhất cho các hệ thống Chatbot ở bước này là xác định nhiều ý định trong một câu nói người dùng. Chẳng hạn như ta có một câu hỏi “Xin chào, giá xe này bao nhiêu vậy ạ?” thì bot phải xác định được hai ý định trong câu hỏi của người dùng là “chào hỏi” và “giá xe”. Nếu việc bot có xác định và trả lời được câu hỏi với nhiều ý định thì sẽ giúp cho việc tương tác giữa người dùng và bot trở nên tự nhiên hơn.



Hình 2.6: Mô hình các bước của việc xác định ý định.

Bước đầu tiên trong việc xác định ý tưởng là tiến hành việc tiền xử lý dữ liệu để loại bỏ các thông tin dư thừa, chuẩn hoá các từ viết sai chính tả thành đúng chính tả theo

chuẩn tiếng Việt, chuẩn hóa dữ liệu, chuẩn hoá các từ viết tắt hay thực hiện tách các từ trong câu, ... Với bước tiền xử lý dữ liệu này, nó đóng một vai trò vô cùng quan trọng trong hệ thống Chatbot bởi nó ảnh hưởng đến độ chính xác cũng như sự thông minh cho chương trình Chatbot của mình.

Bước kế tiếp là việc trích xuất ra các đặc trưng (được gọi là Feature Extraction) từ những dữ liệu đã được chuẩn hoá ở bước đầu tiên. Trong các mô hình học máy truyền thống thì bước trích xuất đặc trưng này có sự ảnh hưởng lớn đến độ chính xác của mô hình phân lớp. Để có thể trích xuất ra được những đặc trưng tốt nhất, chúng ta cần phân tích dữ liệu một cách chuyên sâu để giúp phân tích được dữ liệu một cách chính xác nhất.

Ở bước tiếp theo, đây là bước huấn luyện cho các mô hình học máy với các input đầu vào là các đặc trưng quan trọng đã được trích xuất ở bước thứ hai. Sau đó tiến hành áp dụng các thuật toán học máy để tạo ra một mô hình phân lớp. Các mô hình phân lớp ở đây có thể là các vector trọng số tương ứng với các đặc trưng được trích xuất (như trong các mô hình mạng Neron, mô hình mạng SVM hoặc mô hình hồi quy Logistic).

Sau khi đã xây dựng được một mô hình phân lớp để xác định ý định, chúng ta có thể sử dụng mô hình đó để tiến hành phân lớp một câu hỏi hoặc yêu cầu mới từ người dùng. Câu hỏi đầu vào mới này cũng phải trải qua các bước tiền xử lý dữ liệu và trích xuất đặc trưng, sau đó mô hình phân lớp đã được xây dựng trước đó sẽ tính toán các giá trị để cho ra giá trị xác suất cho từng ý định trong tập các ý định và ý định nào có giá trị cao nhất thì sẽ được hiển thị. Việc xác định ý định của người dùng sẽ quyết định các câu hỏi thoại tiếp theo sau sẽ được diễn ra như thế nào. Vì thế, nếu xác định sai ý định của người dùng thì Chatbot sẽ đưa ra những câu trả lời sai và không phù hợp với ngữ cảnh của hội thoại.

2.2.2 Trích xuất thông tin

Và bước cuối cùng trong thành phần NLU, đó là việc trích xuất ra được thông tin trong các câu hỏi của người dùng. Đối với các thông tin cần trích xuất thì nó thường được biểu diễn dưới dạng chuỗi, số hoặc thời gian và chúng phải được huấn luyện từ trước để mới có thể trích xuất thông tin.

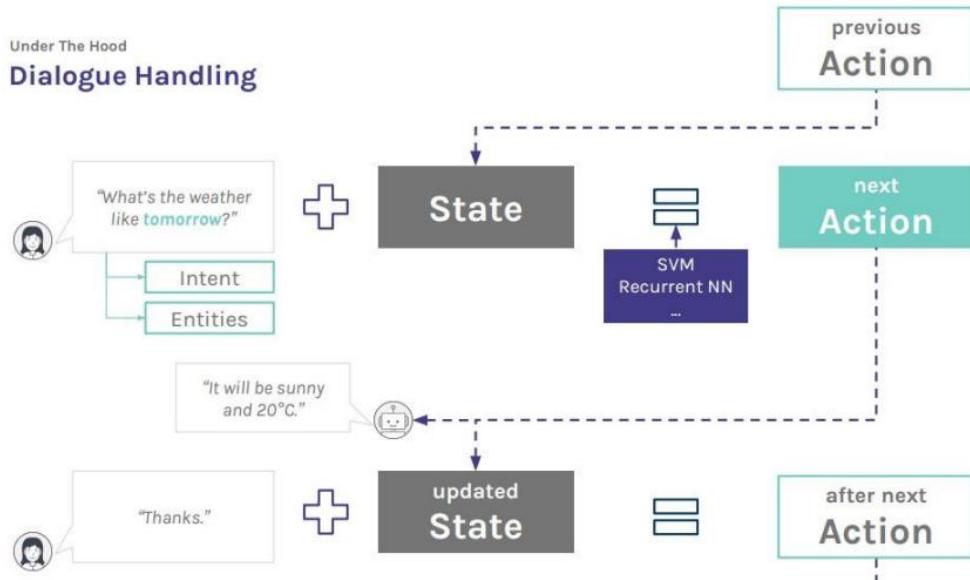
Trong quá trình trích xuất thông tin, ta nó sẽ thực hiện việc phân tách thông điệp đầu vào thành các từ (được gọi là Tokenization hay Word Segmentation). Việc tách từ là cần thiết bởi vì nó là một quá trình xử lý nhằm mục đích xác định ranh giới của các

từ trong câu văn hay cũng có thể hiểu một cách đơn giản rằng việc tách từ là quá trình xác định trong câu có các từ nào là từ đơn hay từ ghép, Việc xác định và phân chia các từ trong câu là cần thiết để có thể xác định cấu trúc ngữ pháp của một câu hay dùng để xác định từ loại trong câu đó. Đối với con người, thì việc xác định và phân tách các từ tưởng chừng như là rất đơn giản nhưng đối với máy tính thì lại khác, đây là bài toán rất khó để giải quyết đặc biệt là với ngôn ngữ phong phú và đa dạng như Tiếng Việt. Thông thường thì các ngôn ngữ như Tiếng Anh thường phân tách các từ bởi khoảng trắng trong câu nhưng đối với ngôn ngữ Tiếng Việt do nó có rất nhiều từ ghép và cụm từ khác nhau nên việc xác định nó khá khó khăn với máy tính. Chẳng hạn như câu “Có gắng học tập” thì khi thực hiện việc tách từ thì “Có gắng” và “học tập” sẽ là 2 từ được phân tách dựa vào nghĩa của nó thay vì tách từng từ theo khoảng trắng.

Để giải quyết vấn đề tách từ đó, hiện nay có một số thuật toán hỗ trợ giải quyết bài toán này như mô hình so khớp từ dài nhất (Longest Matching), Markov ẩn (Hidden Markov Models - HMM), so khớp cực đại (Maximum Matching), hay mô hình CRF (Conditional Random Field). Ở bài tiểu luận giữa kỳ này, chúng em sẽ sử dụng thư viện Underthesea của tác giả Vũ Anh cho phép hỗ trợ tách từ Tiếng Việt với tỷ lệ chính xác cao để triển khai chương trình Chatbot của nhóm.

2.3 Quản lý hội thoại

Trong các cuộc hội thoại trò chuyện dài giữa người dùng và Chatbot, điều cần thiết là làm thế nào để Chatbot có thể ghi nhớ được những thông tin về ngữ cảnh trong cuộc hội thoại hoặc quản lý các trạng thái hội thoại. Việc quản lý hội thoại khi đó đóng vai trò khá quan trọng trong việc đảm bảo việc trao đổi giữa người và hệ thống Chatbot là thông suốt. Quá trình hoạt động của thành phần quản lý hội thoại là nhận giá trị đầu vào từ thành phần hiểu ngôn ngữ tự nhiên NLU đã được xử lý và tiến hành quản lý các trạng thái cuộc hội thoại hay ngữ cảnh trong cuộc hội thoại đó. Đây chính là kết quả đầu ra cũng như là kết quả đầu vào quan trọng cho thành phần sinh ngôn ngữ (Natural Language Generation - NLG).

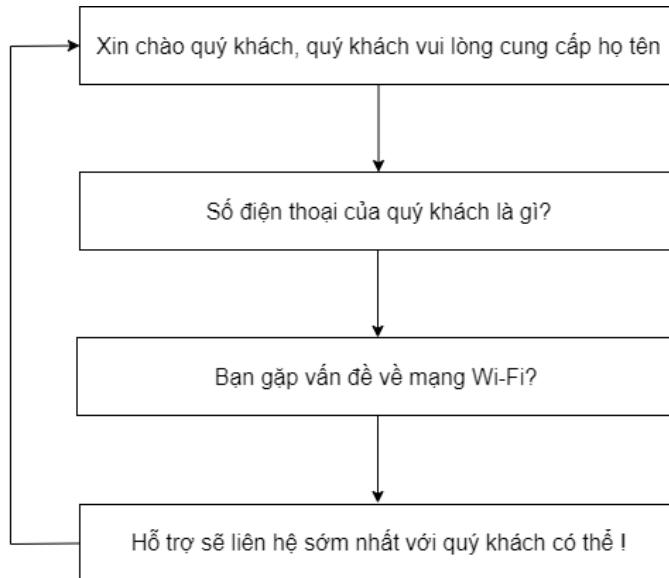


Hình 2.7: Mô hình quản lý trạng thái và đưa ra quyết định trong hội thoại.

Hiện nay, có nhiều các mô hình quản lý hội thoại và các mô hình Chatbot thường ứng dụng có thể kể đến như là mô hình máy trạng thái hữu hạn (Finite State Automata – FSA), mô hình Frame-based (Slot Filling), hoặc mô hình kết hợp giữa hai mô hình này. Một số hướng nghiên cứu mới hiện nay có áp dụng mô hình neural ANN vào việc quản lý hội thoại giúp hệ thống Chatbot thông minh hơn.

2.3.1 Mô hình FSA

Mô hình FSA hay còn được gọi là mô hình máy trạng thái sở hữu là mô hình quản lý hội thoại đơn giản và cơ bản nhất trong các mô hình. Ví dụ như các hệ thống hỏi đáp tự động trong lĩnh vực về chăm sóc khách hàng của một công ty nào đó thì quá trình hội thoại được thiết lập sẵn tuần tự theo từng câu hỏi như hỏi tên khách hàng, số điện thoại, vấn đề cần hỗ trợ, ... Trong mô hình FSA, Chatbot đóng vai trò định hướng người dùng sử dụng trong cuộc hội thoại. Các phản hồi này phụ thuộc vào phản hồi của người ứng với các câu hỏi.



Hình 2.8: Quản lý hội thoại theo mô hình FSA.

Ưu điểm nổi bật của mô hình quản lý FSA này đó là sự đơn giản do định hướng trước những câu hỏi mà người dùng muốn hỏi từ phía người dùng. Tuy nhiên, nhược điểm của mô hình quản lý FSA này thực sự không phù hợp với các hệ thống Chatbot đòi hỏi yêu cầu phức tạp khi người dùng muốn đưa ra nhiều thông tin khác trong hội thoại. Không những không giải quyết được vấn đề mà đôi khi còn làm người dùng cảm thấy khó chịu khi lặp lại các câu hỏi một cách tuần tự.

2.3.2 Mô hình Frame-Based

Mô hình Frame-Based là mô hình được sinh ra để khắc phục những nhược điểm của mô hình FSA. Mô hình này dựa trên các khung được xác định sẵn để định hướng cho cuộc hội thoại. Ứng với mỗi frame thì sẽ bao gồm các thông tin cần phải trả lời và các câu hỏi tương ứng mà Chatbot hỏi người dùng. Mô hình này cho phép người dùng điền thông tin vào nhiều các vị trí khác nhau trong khung (frame).

Slot	Câu hỏi
Họ tên	Xin chào quý khách, quý khách vui lòng cung cấp họ tên.
Số điện thoại	Số điện thoại của quý khách là gì?
Sự cố	Bạn gặp vấn đề về mạng WiFi?

Bảng 2.1: Bảng danh sách các Frame cho mô hình Frame-Based.

Thành phần quản lý hội thoại dựa vào mô hình Frame-based sẽ đưa ra các câu hỏi cho khách hàng và tự động thực hiện điền thông tin vào các slot dựa trên thông tin khách hàng cung cấp ứng với các câu hỏi thoại cho đến khi có đầy đủ các thông tin. Nếu trường

hợp người dùng trả lời nhiều câu hỏi vào cùng một thời điểm thì hệ thống sẽ phải điền thông tin trả lời của người dùng vào các slot tương ứng. Sau khi thực hiện lưu trữ thì hệ thống sẽ thực hiện ghi nhớ các thông tin trả lời đó để không phải đưa ra những câu hỏi trùng lặp với những câu hỏi mà người dùng đã trả lời.

Một vấn đề được đặt ra ở đây cho người phát triển hệ thống đó là khi đó làm sao để biết khi nào cần chuyển đổi giữa các khung đối với một hệ thống Chatbot phức tạp và có rất nhiều các khung khác nhau. Cách để giải quyết thường được sử dụng để quản lý việc chuyển đổi điều khiển giữa các khung đó là việc định nghĩa ra các luật. Các luật này dựa trên một số các câu hỏi thoại hoặc câu hỏi hoặc yêu cầu gần nhất mà người dùng đưa ra.

2.4 Mô hình sinh ngôn ngữ

Mô hình sinh ngôn ngữ Natural Language Generation là mô hình được dùng để sinh câu trả lời cho Chatbot. Mô hình này phụ thuộc vào việc ánh xạ các hành động của quản lý hội thoại vào ngôn ngữ tự nhiên để trả lời cho người dùng. Có bốn phương pháp ánh xạ thường được sử dụng trong mô hình sinh ngôn ngữ là: RNN-Based, Template-Based, Plan-Based, Class-Based. Ở phần này, chúng ta chỉ tìm hiểu sơ qua ba mô hình sinh ngôn ngữ là Template-Based, Plan-Based, Class-Based còn mô hình RNN-Based sẽ được tìm hiểu thông qua các lý thuyết về mạng Neural ở phần sau của bài báo cáo.

2.4.1 Template-Based

Là phương pháp ánh xạ câu trả lời dựa vào việc dùng những câu mẫu trả lời của bot đã được định nghĩa trước để sinh câu trả lời cho người dùng.

Semantic Frame	Natural Language
confirm()	“Bạn có chắc muốn đặt đặt hàng không?”
hello(user=\$V)	“Xin chào \$V, rất vui được gặp bạn!”
bye(user=\$V)	“Hẹn gặp lại \$V vào lần sau!”

Bảng 2.2: Bảng danh sách các câu hỏi dựa trên phương pháp Template-Based.

Phương pháp Template-Based nhìn chung cũng có ưu và khuyết điểm:

❖ Ưu điểm:

- Thực hiện đơn giản.
- Có thể kiểm soát dễ dàng.

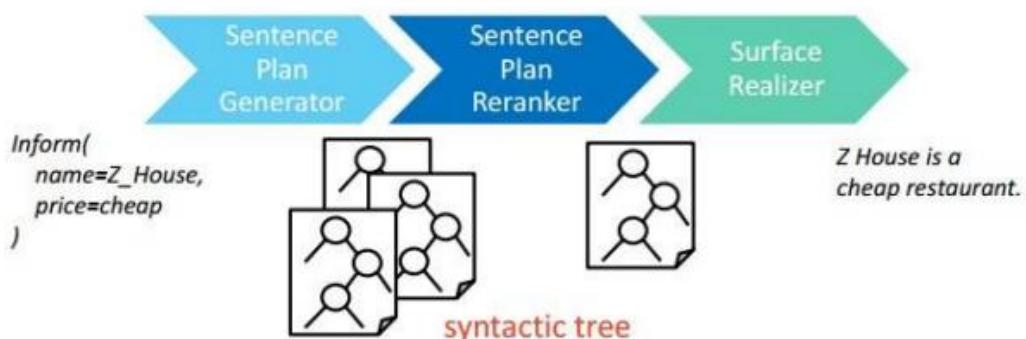
- Phù hợp với các bài toán miền đóng (Close Domain).

❖ **Nhược điểm:**

- Do được xây dựng sẵn các mẫu nên nó không mang tính chất tự nhiên cho các câu trả lời.
- Tốn khá nhiều thời gian để định nghĩa các luật.
- Khó kiểm soát các luật đối với các hệ thống lớn dẫn đến khó phát triển và duy trì.

2.4.2 Plan-Based

Là phương pháp sinh câu trả lời dựa vào kế hoạch đã được đặt ra bao gồm nhiều kế hoạch khác nhau như: kế hoạch sinh câu, kế hoạch xếp hạng lại câu, tạo đầu ra.



Hình 2.9: Phương pháp sinh ngôn ngữ Plan-Based.

Cũng giống với Template-Based, Plan-Based cũng có những ưu điểm và nhược điểm của nó.

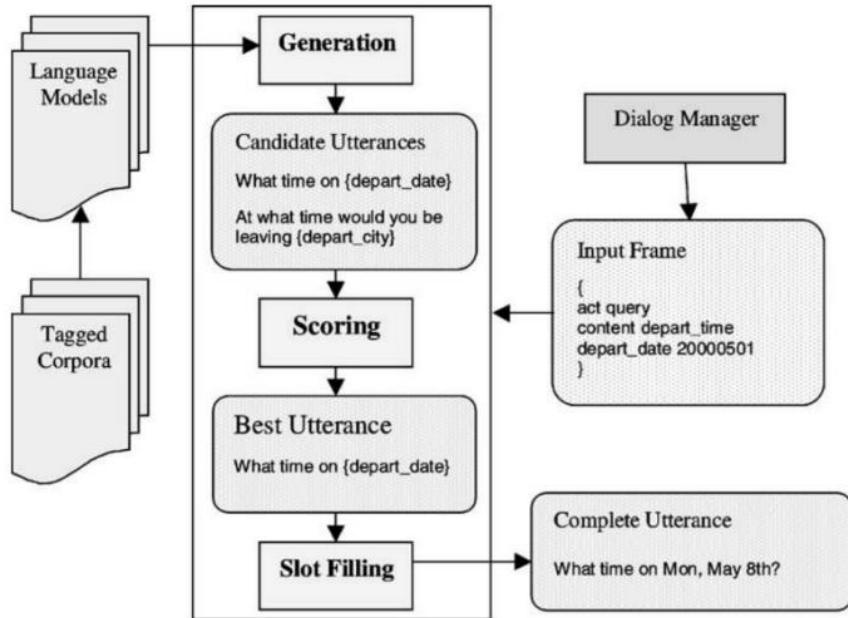
❖ **Ưu điểm:** Có thể thực hiện mô hình hóa cấu trúc ngữ pháp phức tạp.

❖ **Nhược điểm:**

- Thiết kế phức tạp, nặng nề.
- Đòi hỏi phải rõ miền kiến thức.

2.4.3 Class-Based

Là phương pháp dựa trên việc huấn luyện cho bot học những câu trả lời với các câu hỏi đầu vào đã được gán nhãn. Ứng với các hành động và thông tin từ thành phần quản lý hội thoại (DM) thì bot sẽ đưa ra câu trả lời gần nhất dựa trên tập dữ liệu trả lời được đào tạo từ trước đó.



Hình 2.10: Phương pháp sinh ngôn ngữ Class-Based.

Phương pháp Class-Based có các ưu và nhược điểm như sau:

- ❖ **Ưu điểm:** Dễ dàng thực hiện và triển khai.
- ❖ **Nhược điểm:**
 - Phụ thuộc nhiều vào dữ liệu đào tạo được gán nhãn từ trước.
 - Việc tính toán các giá trị đánh giá không hiệu quả dẫn đến việc sinh câu trả lời sai.

2.5 Tổng kết chương 2

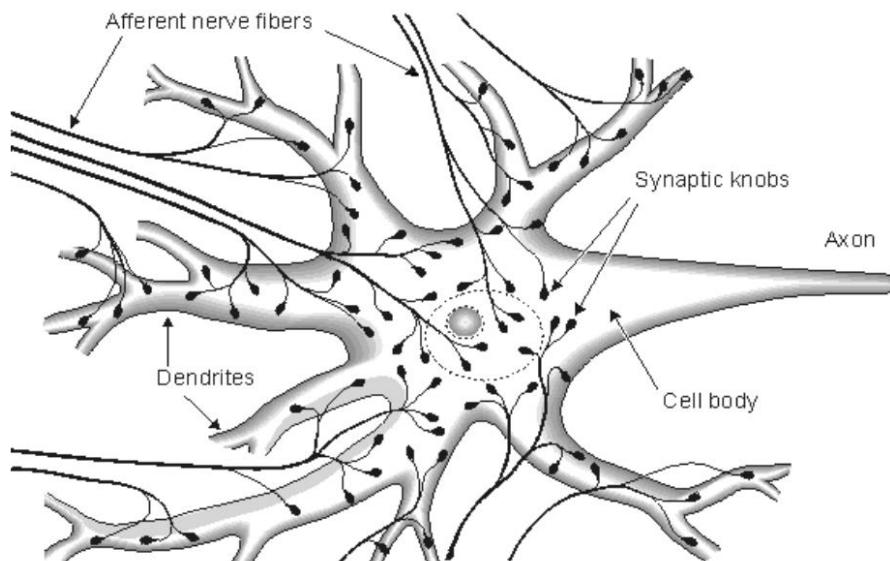
Như vậy là ta đã đi qua chương hai của bài báo cáo giữa kỳ. Qua chương hai này, chúng ta đã tìm hiểu về khái niệm hệ thống trả lời tự động hay còn được gọi là Chatbot, lịch sử hình thành của nó, các khái niệm liên quan như phân loại các loại Chatbot. Cùng với đó là việc phân tích và tìm hiểu từng thành phần trong hệ thống Chatbot bằng cách phân tích từng thành phần ứng với các phương pháp khác nhau trong nó. Ngoài ra, chúng ta còn đưa ra các nhận xét và đánh giá về các ưu điểm và nhược điểm của từng phương pháp trong các thành phần chính của hệ thống Chatbot. Để tiếp tục tìm hiểu rõ hơn về các phương pháp triển khai hệ thống Chatbot, chúng ta sẽ tìm hiểu phương pháp mạng Recurrent Neural Networks - RNN ứng dụng cho bài toán Chatbot ở chương tiếp theo.

CHƯƠNG 3: ÚNG DỤNG RECURRENT NEURAL NETWORK CHO BÀI TOÁN CHATBOT.

3.1 Tổng quan về mạng Neural

3.1.1 Mạng Neural sinh học

Đầu tiên, để tìm hiểu về các khái niệm liên quan đến mạng neural nhân tạo thì chúng ta sẽ tìm hiểu về khái niệm và cấu trúc của mạng neural sinh học. Trong bộ não của con người chúng ta, tồn tại một mạng lưới gồm khoảng 10^{11} tế bào thần kinh được gọi là các nơ-ron và chúng được liên kết phức tạp với nhau. Mỗi tế bào thần kinh gồm 3 thành phần chính: thân tế bào thần kinh (cell body còn được gọi là soma), hệ thống các dây thần kinh tiếp nhận (dendrites) và một sợi trực thần kinh (axon).



Hình 3.1: Mô hình tế bào thần kinh của người.

Trong hệ thống dây thần kinh tiếp nhận là một mạng lưới dày đặc các dây thần kinh ở dạng cây bao bọc xung quanh thân tế bào và các dây thần kinh này sẽ dẫn các tín hiệu đến phần thân của tế bào. Khi mà thân tế bào nhận được các tín hiệu này, nó sẽ tổng hợp các tín hiệu đó và làm thay đổi điện thế của nó cho đến khi vượt qua một mức ngưỡng nhất định nào đó thì nó sẽ xuất ra một xung điện trên sợi trực thần kinh (Axon). Các dây thần kinh ra (Axon) rẽ ra nhiều nhánh khác nhau để kết nối đến các dây thần kinh vào hoặc cũng có thể kết nối trực tiếp với phần thân của tế bào thần kinh khác thông qua các khớp thần kinh.

Khi một tế bào thần kinh hoạt động, các tế bào đó sẽ được kích thích tạo ra một tín hiệu điện và tín hiệu đó chạy dọc theo sợi thần kinh ra (Axon) dẫn đến các khớp thần kinh. Ở khớp thần kinh, nó được chia ra thành hai loại:

- Khớp nối kích thích (Excitatory)
- Khớp nối ức chế (Inhibitory)

Tại hai khớp thần kinh trên, chúng xảy ra quá trình phản ứng và giải phóng các chất hữu cơ tạo nên tín hiệu điện kích thích cho tế bào thần kinh. Các nghiên cứu về quá trình hoạt động của hệ thần kinh đã chỉ ra rằng, quá trình mà bộ não của chúng ta tiếp thu chính là việc hình thành hoặc thay đổi mức độ liên kết của các khớp nối.

Với khả năng của mạng Neural sinh học trong bộ não của con người, bộ nhớ của chúng có các đặc điểm nổi bật sau:

- Được tổ chức theo dạng các bó thông tin và truy nhập theo nội dung.
- Có khả năng truy xuất các tri thức hay các mối liên hệ chung giữa các đối tượng trong một khái niệm nào đó.
- Khả năng tổng quát hóa.
- Đặc biệt là bộ não có khả năng học.

3.1.2 Quá trình học của bộ não

Quá trình học của bộ não diễn ra khi các xung tín hiệu từ các dây thần kinh axon tới các khớp nối và từ khớp nối sẽ truyền tín hiệu đi qua một neural tiếp theo. Do đó, quá trình học hình thành một con đường truyền xung nhất định.

Việc học sẽ khiến cho các đường truyền xung được lặp đi lặp lại nhiều lần do đó sức cản của các khớp nối sẽ được nhỏ dần và tạo điều kiện cho những lần lặp lại dễ dàng hơn. Tất cả các kiến thức, kinh nghiệm của một người nào đó được tích lũy và lưu giữ trong bộ não của họ chính là sức cản của các khớp nối.

3.2 Neural nhân tạo

3.2.1 Khái niệm

Mạng Neural nhân tạo (Artificial Neural Networks được viết tắt là ANN) là sự mô phỏng xử lý thông tin hay thường được gọi là mạng neural. Mỗi neural nhân tạo - ANN thực hiện hai chức năng bao gồm chức năng tổng hợp đầu vào và chức năng tạo đầu ra. Mỗi một ANN có một số đầu vào và một đầu ra ứng với mỗi giá trị đầu vào được gắn

với một hệ số nhân gọi là trọng số (weight). Trọng số có thể là giá trị âm hoặc dương và nó có ý nghĩa như mức độ liên kết tại khớp nối trong mạng neural sinh học gồm các khớp nối kích thích và khớp nối ức chế.

Mỗi neural nhân tạo có một giá trị ngưỡng nhất định. Với chức năng đầu vào chính là một tổng có trọng số các tín hiệu vào kết hợp với ngưỡng để tạo ra tín hiệu đầu vào được gọi là *net input*. Sự kết hợp này được thực hiện bằng một tổng hay bằng hàm PSP (Post Synaptic Potential) theo một số tài liệu. Kế đến là chức năng tạo đầu ra của mạng ANN, nó được thể hiện bằng hàm chuyển đổi (Transfer Function). Hàm chuyển đổi này sẽ nhận tín hiệu đầu vào là giá trị net input ở chức năng đầu vào và tạo tín hiệu đầu ra cho neural.

3.2.2 Mô hình Neural

Một mạng neural nhân tạo bao gồm hai thành phần:

- Các nút hay còn được gọi là các neural hoặc đơn vị xử lý
- Các liên kết giữa các nút đó và được gán một trọng số nào đó đặc trưng cho cường độ liên kết.

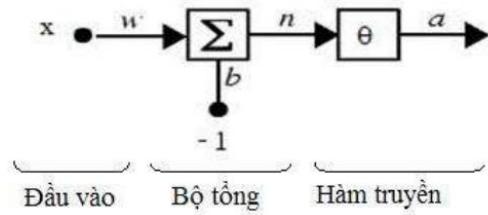
Chúng ta xác định các ký hiệu trong mạng neural như sau:

- P_i là tín hiệu đầu vào.
- X_i là tín hiệu đầu ra của neural thứ i.

Với trạng thái đầu vào của neural thứ i thì nó được xác định bởi tổng tuyến tính của các tín hiệu vào có trọng số từ các neural thứ j.

3.2.2.1 Neural một đầu vào

Đúng với tên gọi của nó là neural một đầu vào thì nó có cấu tạo khá đơn giản chỉ bao gồm một đầu vào như hình 3.2. Với đầu vào vô hướng p được nhân với một trọng số vô hướng là w và thu được kết quả là w_p . Hai tham số đầu vào đó chính là một trong hai số hạng được đưa vào bộ tổng. Bên cạnh đó, một đầu vào khác là 1 được nhân với một hệ số có tên là bias (ký hiệu là b). Sau đó các giá trị sẽ được đưa vào bộ tổng để tính toán như trong hình. Sau khi hoàn tất quá trình tính toán trong bộ tổng, nó cho ra kết quả n, và giá trị n này thường được gọi là tín hiệu đầu vào net input. Cuối cùng, giá trị n được cho qua một hàm kích hoạt (Activation Function) và thu được kết quả được đầu ra a của neural.



Hình 3.2: Mô hình Neural chỉ gồm một đầu vào.

Để dễ dàng hình dung, chúng ta tiến hành liên hệ mô hình đơn giản này với một neural sinh học. Với giá trị trọng số w thì nó sẽ tương ứng với độ mạnh liên kết của các khớp nối và đầu vào p được đại diện cho các dây thần kinh tiếp nhận. Đối với phần thân của neural (gọi là Cell Body) thì nó được được mô hình hóa bởi bộ tổng và hàm kích hoạt. Cuối cùng đầu ra a của neural diễn tả tín hiệu ra trên sợi trực neural sinh học (axon). Nhìn chung, ta có được một công thức tổng quát để tính giá trị đầu ra a cho neural chỉ có một đầu vào là:

$$a = f(wp + b)$$

Chẳng hạn, ta có $w = 4$, $p = 3$ và $b = -2.5$ thì $a = (4 \times 3 + (-2.5)) = f(9.5)$. Ta có thể thấy rằng, đầu ra a của neural phụ thuộc vào hàm kích hoạt được chọn là hàm nào trong từng trường hợp cụ thể. Còn đối với hệ số bias, nó như là một trọng số với đầu vào luôn có giá trị là 1 và giá trị bias này có thể có hoặc không có tùy vào mạng neural.

Đối với hàm kích hoạt f , nó có thể là hàm tuyến tính hoặc phi tuyến tính đối với giá trị n được tính từ bộ tổng. Có rất nhiều dạng hàm kích hoạt được sử dụng trong một neural.

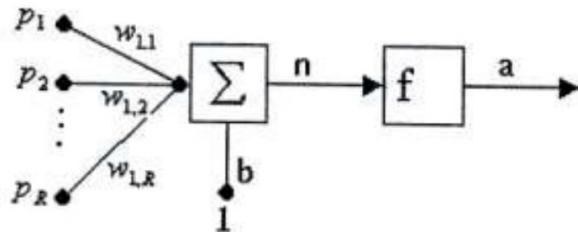
Tên hàm	Công thức
hardlim	$a = 0$ với $n < 0$ $a = 1$ với $n \geq 0$
hardlims	$a = -1$ với $n < 0$ $a = 1$ với $n \geq 0$
purelin	$a = n$
satlin	$a = 0$ với $n < 0$ $a = n$ với $0 \leq n \leq 1$ $a = 1$ với $n > 1$
satlins	$a = -1$ với $n < 0$ $a = n$ với $0 \leq n \leq 1$ $a = 1$ với $n > 1$

tansig	$a = \frac{e^n - e^{-n}}{1 + e^{-n}}$
poslin	$a = 0$ với $n < 0$ $a = n$ với $n \geq 0$
compet	$a = 1$ với neural có n lớn nhất $a = 0$ với neural còn lại
logsig	$a = \frac{1}{1 + e^{-n}}$

Bảng 3.1: Bảng các hàm kích hoạt và công thức tương ứng.

3.2.2.2 Neural nhiều đầu vào

Khác với neural một đầu vào, neural nhiều đầu vào bao gồm nhiều đầu vào vô hướng riêng biệt nhau.

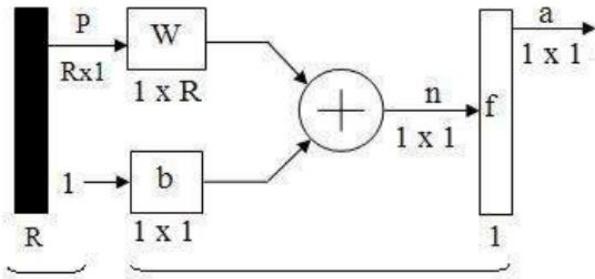


Hình 3.3: Mô hình mạng Neural nhiều đầu vào.

Úng với mỗi đầu vào riêng biệt thứ p_n như p_1, p_2, \dots, p_n thì sẽ có một trọng số tương ứng là w_1, w_2, \dots, w_n trong ma trận trọng số W . Lúc này, ta có giá trị sau khi đi qua bộ tổng với một giá trị bias được tính theo công thức là:

$$n = w_1.p_1 + w_2.p_2 + \dots + w_n.p_n + b$$

Sau đó, giá trị n sẽ đi qua một hàm kích hoạt f được tính theo công thức $a = f(wp + b)$. Tương ứng với mỗi phần tử của ma trận W thì ta quy ước rằng w_{ij} đại diện cho trọng số đầu vào thứ j ứng với neural thứ i . Để có một cái nhìn đơn giản và trực quan hóa về mô hình neural nhiều đầu vào, ta có thể biểu diễn như sau:



Hình 3.4: Mô hình neural nhiều đầu vào rút gọn.

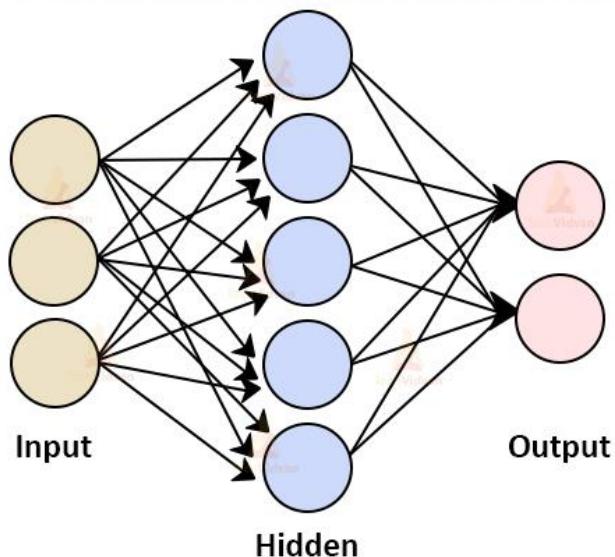
Ở hình 3.4, ta thấy rằng vector đầu vào P gồm R phần tử và ma trận trọng số W cũng tương ứng có R phần tử. Hằng số đầu vào 1 sẽ được nhân với hệ số bias cùng với wp cho đi qua một bộ tính tổng để tạo ra tín hiệu đầu vào cho hàm kích hoạt là số vô hướng n . Hàm kích hoạt f sẽ thực hiện biến đổi n thành đầu ra a dựa vào các thuật toán kích hoạt được chọn.

3.3 Mạng Neural nhân tạo

3.3.1 Định nghĩa

Mạng neural nhân tạo là sự kết hợp giữa các neural nhân tạo với nhau. Mạng neural nhận tạo được nghiên cứu và xây dựng dựa trên ý tưởng của mạng neural sinh học như bộ não của sinh vật hay con người. Mạng neural nhân tạo bao gồm một số lượng lớn các mối gắn kết để xử lý các yếu tố làm việc trong môi quan hệ giải quyết vấn đề rõ ràng. ANN được giới thiệu lần đầu tiên vào năm 1943 bởi nhà nghiên cứu về thần kinh học Warren McCulloch và nhà logic học Walter Pits. Hoạt động của neural nhân tạo giống như hoạt động bộ não của con người vì nó được học hỏi bởi kinh nghiệm thông qua việc đào tạo và huấn luyện bên cạnh đó thì nó cũng sở hữu khả năng lưu giữ được các tri thức và sử dụng những tri thức đó trong việc phán đoán các dữ liệu mới. Processing Elements của ANN gọi là neural, nó nhận nhiều các dữ liệu vào (inputs) xử lý chúng và cho ra một kết quả (output) duy nhất. Kết quả xử lý của một neuron này có thể được sử dụng để làm input cho các neuron khác.

Architecture of Artificial Neural Network



Hình 3.5: Mô hình của mạng neural nhân tạo ANN.

Mạng neural nhân tạo bao gồm các đặc điểm sau:

- Mạng được xây dựng bằng các neural liên kết lại với nhau.
- Chức năng của mạng được xác định bởi các yếu tố như: cấu trúc, quá trình xử lý bên trong của từng neural và mức độ liên kết giữa các neural đó.
- Mức độ liên kết giữa các neural được xác định thông qua quá trình huấn luyện.
- Nhiệm vụ chính của quá trình huấn luyện là cập nhật các trọng số khi có thông tin về các mẫu học.

3.3.2 Lịch sử phát triển

Vào cuối thế kỷ 19 và khoảng đầu thế kỷ 20, một số nghiên cứu về vật lý, tâm lý và hệ thần kinh của các nhà khoa học bao gồm Herman, Ernst Mach và Ivan Iavalov đã đưa ra các lý thuyết về sự quyết định, quá trình học, ...của hệ thần kinh nhưng chưa có sự mô tả toán học cho hoạt động của mạng neural.

Vào năm 1943, đây là điểm khởi đầu của lĩnh vực mạng neural. Với mô hình đơn giản về mạng neural được thiết kế bằng mạch điện tử lần đầu tiên được đưa ra bởi Warren McCulloch và Walter Pits. Cùng với sự khẳng định mạng neural nhân tạo về nguyên lý có thể thực hiện được trong phạm vi tính toán các hàm số học và logic.

Vào những năm 1949, Donald Hebb đã đưa ra một cơ chế giải thích cho quá trình học diễn ra trong các neural sinh học được viết trong Organization of Behavior.

Vào cuối thập niên 50, Frank Rosenblatt đã đưa ra ứng dụng đầu tiên của mạng neural nhân tạo được ứng dụng vào thực tế. Mạng Perceptron là mạng neural mà ông đưa ra là với sự kết hợp luật học dùng để nhận dạng mẫu. Cùng vào thời điểm đó, Ted Hoff và Bernard Widrow đã giới thiệu một số thuật toán học và ứng dụng nó để huấn luyện các mạng neural tuyến tính.

Vào những năm 1969, hai nhà toán học nổi tiếng là Minsky và Papert đã nêu ra các điểm còn hạn chế của mạng Widrow Hoff và mạng Perceptron của Rosenblatt. Điều này làm cho nhiều người có suy nghĩ rằng việc nghiên cứu về mạng neural sẽ đi vào ngõ cụt và không thể phát triển. Một điều quan trọng đó là vào thời gian này vẫn chưa có những máy tính thực sự mạnh mẽ để có thể thực nghiệm mạng neural nên những nghiên cứu về mạng neural đã bị trì hoãn gần một thập kỷ.

Sau đó 3 năm, Stephen Grossberg đang nghiên cứu một cách tích cực về các mạng tự tổ chức. Cùng với đó thì hai người tên là James Anderson và Teuvo Kohonen đã độc lập phát triển các mạng neural mới có khả năng tự tổ chức và có năng lực nhớ.

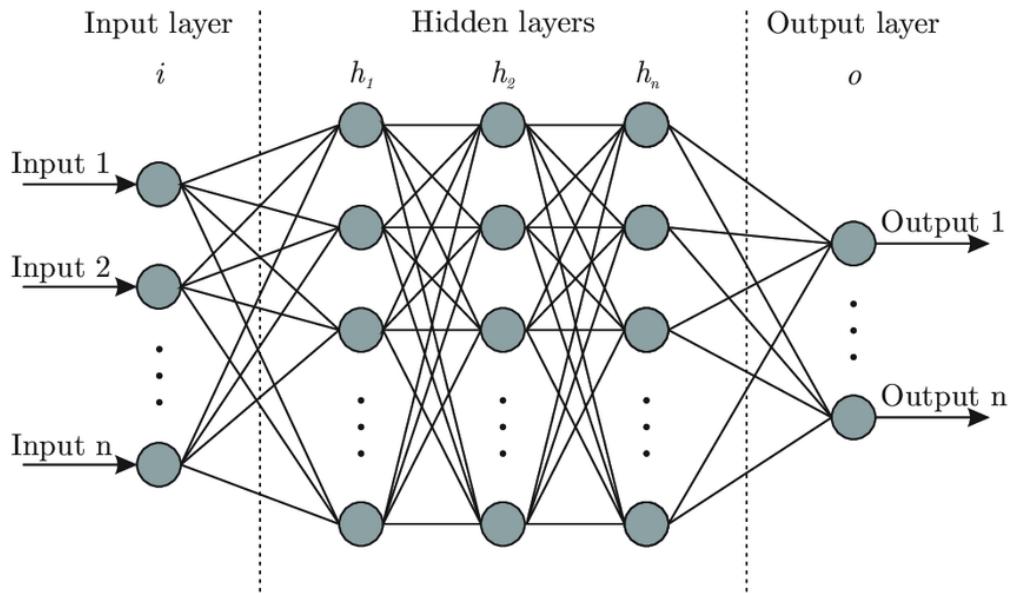
Bước sang thập kỷ 80, với sự phát triển mạnh mẽ của ngành công nghiệp máy tính thì các nghiên cứu về mạng neural đột nhiên tăng lên một cách nhanh chóng và đột ngột. Tại thời điểm này, có hai phát kiến nổi bật và quan trọng nhất là:

- Giải thích hoạt động của mạng hồi quy một lớp (Recurrent Network) – một mạng nhà vật lý John Hopfield mô tả là được dùng như một bộ nhớ kết hợp, bằng việc sử dụng cơ học thống kê.
- Thuật toán lan truyền ngược được James McClelland và David Rumelhart trình bày có ảnh hưởng nhất bằng việc sử dụng thuật toán lan truyền ngược để huấn luyện các mạng perceptron đa lớp.

Cho đến thời điểm hiện tại, lĩnh vực nghiên cứu về mạng neural nhân tạo đang rất được triển trong lĩnh vực trí tuệ nhân tạo nói riêng và trong các lĩnh vực của đời sống nói chung. Nó đang được nghiên cứu và phát triển mạnh mẽ bởi tính ứng dụng vào thực tế là rất cao.

3.3.3 Kiến trúc mạng Neural nhân tạo

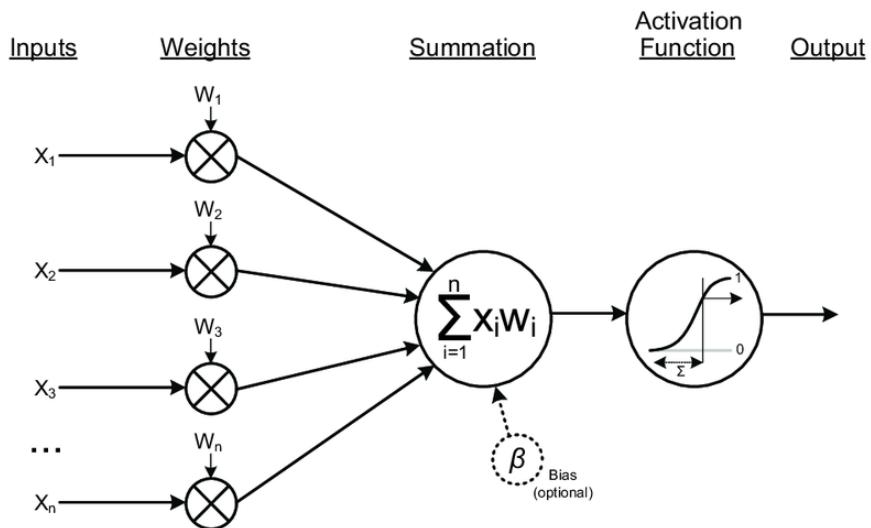
Kiến trúc chung của một mạng neural nhân tạo ANN thường bao gồm 3 thành phần chính đó là: Input Layer, Hidden Layer và Output Layer.



Hình 3.6: Cấu trúc của mạng neural nhân tạo.

Chúng ta cùng xem xét hình 3.2, ta thấy rằng cấu trúc của một mạng neural nhân tạo sẽ bao gồm ba thành phần chính đã được liệt kê ở trên. Ứng với mỗi lớp sẽ có từng đặc điểm riêng của nó. Trong đó, ta thấy rằng lớp Hidden Layer gồm nhiều các neural nhận dữ liệu đầu vào từ lớp Input và chuyển đổi các input đó cho các bước xử lý tiếp theo. Trong một mạng neural nhân tạo, có thể có nhiều lớp ẩn (Hidden Layer). Một ưu điểm nổi bật lớn nhất của các mạng ANN chính là khả năng được sử dụng như một cơ chế xấp xỉ hàm từ các dữ liệu được quan sát. Tuy nhiên, việc sử dụng mạng ANN không đơn giản như bởi nó còn phụ thuộc vào một số các đặc điểm và kinh nghiệm khi thiết kế một mạng neural ANN sao cho phù hợp.

Chẳng hạn đối với mỗi bước huấn luyện mô hình cho Chatbot thì phương pháp sẽ thực hiện việc tính toán tỷ lệ dữ liệu đầu ra từ dữ liệu đầu vào một cách chính xác bằng cách tính toán các trọng số cho mỗi kết nối từ các lần lặp lại trong tập dữ liệu “huấn luyện” cho Chatbot. Mỗi khi thực hiện các bước huấn luyện cho Chatbot, nó sẽ tiến hành việc sửa đổi các trọng số dẫn đến dữ liệu output được trích xuất ra với độ chính xác ngày càng cao hơn.



Hình 3.7: Quá trình xử lý thông tin của một neural nhân tạo.

Với hình 3.7, ta thấy rằng quá trình xử lý thông tin của một mạng neural nhân tạo thường bao gồm các thành phần sau:

- **Lớp Input:** mỗi giá trị input đầu vào ứng với một giá trị thuộc tính của dữ liệu. Chẳng hạn như các thuộc tính để xác nhận cho việc đặt hàng của hệ thống bán hàng như tên người mua, địa chỉ, số điện thoại, ...
- **Lớp Output:** là kết quả sau khi thực hiện qua các bước tính toán bên trong các lớp neural ẩn. Kết quả đầu ra thường là một giải pháp cho một vấn đề, chẳng hạn như bài toán đặt hàng trên hệ thống bán hàng thì kết quả đầu ra sẽ là giao hoặc không giao.
- **Connection Weights:** hay còn được gọi là trọng số liên kết, đây là thành phần rất quan trọng trong một mạng neural nhân tạo. Nó thể hiện mức độ quan trọng hoặc độ mạnh của dữ liệu đầu vào đối với việc xử lý thông tin khi chuyển đổi từ lớp này sang lớp khác. Quá trình học của một mạng ANN có thực chất là quá trình điều chỉnh các trọng số Connection Weight của dữ liệu đầu vào để thu được kết quả mong muốn.
- **Summation Function:** còn được gọi là hàm tổng, nó có nhiệm vụ thực hiện việc tính tổng của các trọng số của tất cả các input được đưa vào mỗi neural. Hàm tính tổng của một neural tương ứng với n giá trị input đầu vào được tính theo công thức sau:

$$Y = \sum_{i=1}^n X_i W_i$$

- **Activation Function:** hàm Summation ở trên của một neural cho biết khả năng kích hoạt (Activation) của nó. Các neural có thể sinh ra một hoặc không trong mạng ANN hay nói một cách dễ hiểu thì các giá trị output của một neural có thể được chuyển đến layer tiếp theo trong mạng ANN hay không. Mỗi liên hệ giữa hàm Summation và kết quả đầu ra được thể hiện bằng hàm kích hoạt. Việc lựa chọn hàm kích hoạt cũng là một việc rất quan trọng và tác động khá lớn đến kết quả đầu ra của mạng ANN. Một số các hàm chuyển đổi phi tuyến thường được sử dụng phổ biến trong mạng neural nhân tạo là hàm sigmoid hoặc hàm tanh. Hàm sigmoid và tanh được tính theo các công thức sau:

$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s)$$

$$\tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}}$$

Trong công thức của hàm tanh, ta thấy rằng khoảng giá trị đầu ra của hàm kích hoạt này nằm trong khoảng $[-1, 1]$ thay vì nằm trong khoảng từ $[0, 1]$ đối với hàm sigmoid nên hai hàm này còn được gọi là hàm chuẩn hóa.

Đôi khi kết quả xử lý output tại các neural là rất lớn, do đó việc sử dụng hàm kích hoạt để giúp cho việc xử lý các kết quả đầu ra này trước khi chuyển đến layer tiếp theo để giảm bớt độ phức tạp của nó. Không phải lúc nào cũng phải bắt buộc sử dụng Activation Function mà đôi khi người ta còn sử dụng giá trị ngưỡng (Threshold value) để điều khiển và kiểm soát các giá trị đầu ra output của các neuron tại một lớp trước khi chuyển các giá trị này đến những lớp kế tiếp trong mạng ANN. Nếu giá trị đầu ra output của một neural lớn hơn hoặc bằng với giá trị ngưỡng Threshold thì nó sẽ được chuyển đến lớp tiếp theo và ngược lại nếu output nhỏ hơn Threshold thì nó sẽ không được chuyển đến lớp tiếp theo.

Tóm lại, mạng neural dự đoán dựa trên lan truyền thẳng là các phép nhân ma trận cùng với một hàm kích hoạt (Activation Function) để thu được kết quả đầu ra. Nếu giá trị đầu là vector 2 chiều thì kết quả dự đoán y được tính bằng công thức:

$$z_1 = \mathbf{x}W_1 + \mathbf{b}_1$$

$$a_1 = \tanh(z_1)$$

$$z_2 = a_1W_2 + \mathbf{b}_2$$

$$a_2 = \hat{y} = \text{softmax}(z_2)$$

Trong đó:

- Z_i : là input đầu vào của lớp thứ i.
- a_i : là output đầu ra của lớp thứ i sau khi áp dụng hàm kích hoạt.
- W_1, W_2 : là các trọng liên kết.
- b_1, b_2 : các giá trị bias.

Việc huấn luyện là quá trình mà tìm ra các giá trị W_1, W_2, b_1, b_2 sao cho giá trị của hàm lỗi là nhỏ nhất. Hàm lỗi ở đây được gọi là loss function và đối với hàm softmax thì ta sẽ dùng crossentropy loss. Ta có công thức tính hàm lỗi giữa giá trị dự đoán \hat{y} và y như sau:

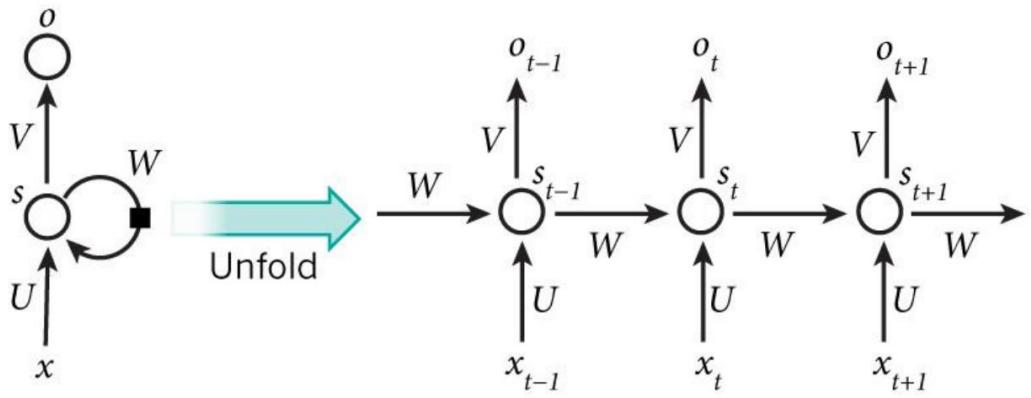
$$L(y, \hat{y}) = -\frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \hat{y}_{n,i}$$

Với công thức trên, nếu ta có N ví dụ dữ liệu huấn luyện và C nhóm phân lớp thì ta sẽ lấy tổng trên toàn bộ tập huấn luyện và cộng dồn vào hàm loss nếu kết quả phân lớp sai. Độ khác biệt giữa hai giá trị \hat{y} và y càng lớn thì độ lỗi càng cao.

3.4 Mạng neural hồi quy

3.4.1 Khái niệm

Mạng neural hồi quy (Recurrent Neural Network - RNN) dựa trên ý là thiết kế một mạng neural mà nó có khả năng xử lý được thông tin dạng chuỗi (Sequential Information). Trong các mạng neural truyền thống thì tất cả các đầu vào và cả đầu ra là độc lập với nhau. Điều đó nghĩa là chúng không liên kết thành chuỗi với nhau. Chẳng hạn ta có một câu là một chuỗi bao gồm nhiều từ. Đối với RNN, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần khác ở những thời điểm trước đó. Nói một cách đơn giản, mạng RNN là mô hình có khả năng ghi nhớ và có khả năng lưu trữ các thông tin đã được tính toán và xử lý trước đó để áp dụng tính toán cho các quá trình ở phía sau. Một mạng RNN về cơ bản có cấu trúc như sau:



Hình 3.8: Mô hình mạng RNN.

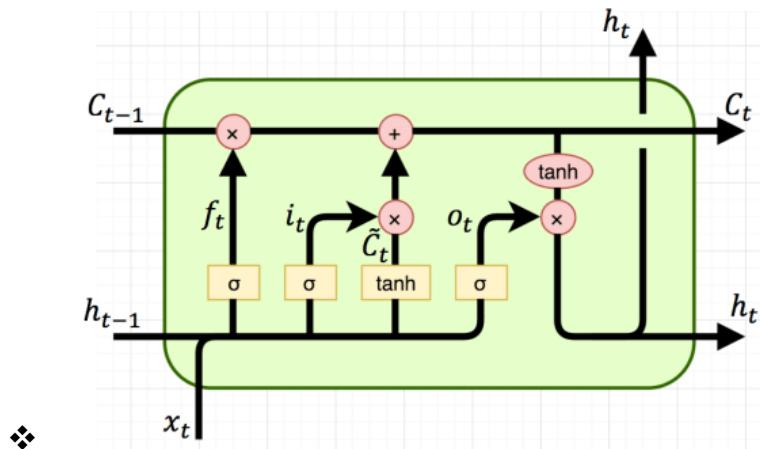
Với mô hình 3.8, giả sử ta có một câu đầu vào gồm 6 từ “Trường Đại học Tôn Đức Thắng” thì lúc này mạng neural được triển khai sẽ bao gồm 4 tầng tương ứng với mỗi chữ cái là một tầng. Lúc này, việc tính toán sẽ được diễn ra tuần tự như trong hình với các ký hiệu sau:

- X_t : là giá trị đầu vào tại bước thứ t .
- S_t : là trạng thái ẩn tại bước t .
- O_t : là đầu ra tương ứng tại bước t .

3.4.2 Phân loại

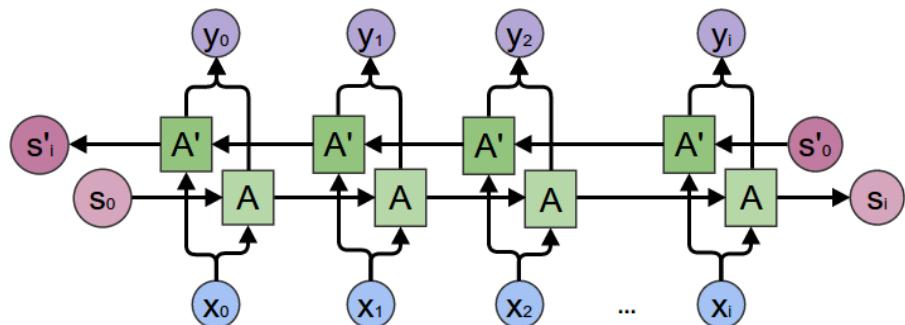
Trong những năm trở lại đây, những nhà nghiên cứu đã phát triển nhiều loại mạng RNNs ngày càng phức tạp và tinh vi để giải quyết các vấn đề còn hạn chế của mạng RNN. Ta có thể kể đến một số các nghiên cứu nổi bật về RNN như sau:

- ❖ **Long Short-Term Memory Network (LSTM):** là một biến thể được cải tiến từ RNN, có cấu trúc tương tự như RNNs nhưng có cách tính toán khác đối với các trạng thái ẩn. Các cells trong LSTM có thể được xem như là một hộp đen nhận thông tin đầu vào gồm các giá trị Hidden State và các giá trị khác. Bên trong các cell đó, chúng sẽ tính toán và đưa ra 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.



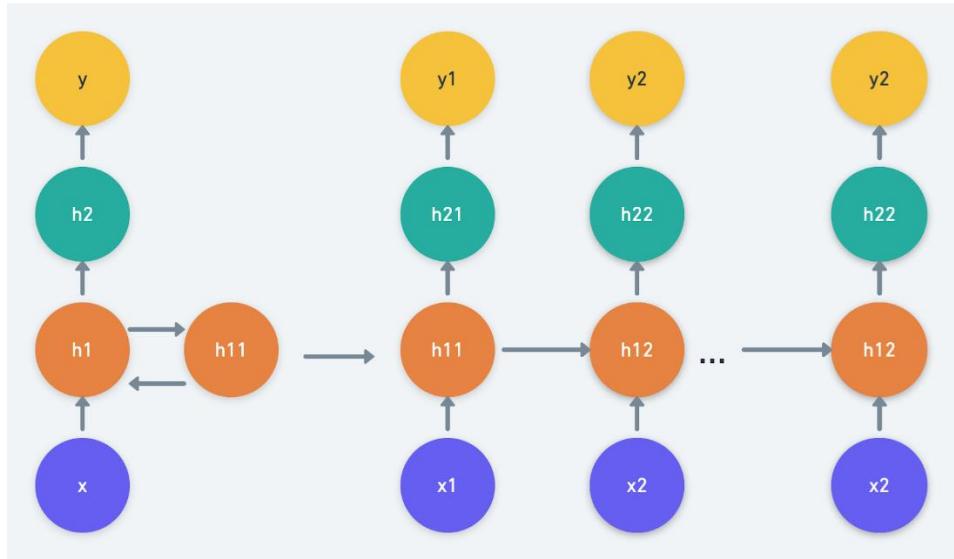
Hình 3.9: Mô hình LSTM.

- ❖ **Bidirectional RNN:** là mạng được dựa trên ý tưởng giá trị đầu ra tại một thời điểm t nào đó không chỉ phụ thuộc vào các thành phần trước đó mà còn phụ thuộc vào các thành phần trong tương lai. Một ví dụ để hiểu rõ hơn về Bidirectional, nó có thể được ứng dụng để dự đoán một từ bị thiếu trong chuỗi. Bằng cách quét cả hai chiều cả từ trái sang phải và từ phải sang trái xung qua của từ đó. Mô hình này gồm hai mạng RNN được chồng lên nhau. Trong Bidirectional, các giá trị trạng thái ẩn Hidden State sẽ được tính toán dựa vào cả hai thành phần bên trái và bên phải của mạng.



Hình 3.10: Mô hình của Bidirectional RNN.

- ❖ **Deep RNN:** cũng giống với Bidirectional RNN ở trên, điểm khác biệt nổi bật của Deep RNN so với Bidirectional RNN đó là việc mô hình này gồm nhiều tầng Bidirectional RNN tại mỗi thời điểm. Mô hình Deep RNN sẽ cung cấp cho ta khả năng thực hiện các tính toán nâng cao nhưng đòi hỏi tập huấn luyện phải đủ lớn.



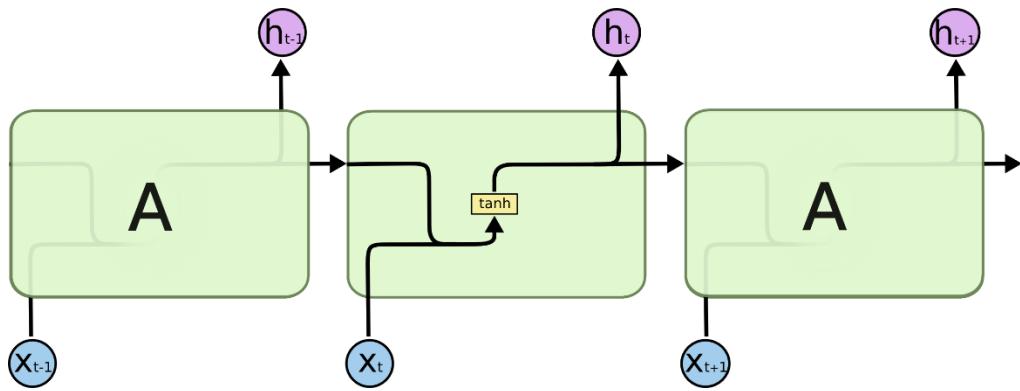
Hình 3.11: Mô hình Deep RNN.

3.5 Long Short-Term Memory

3.5.1 Khái niệm

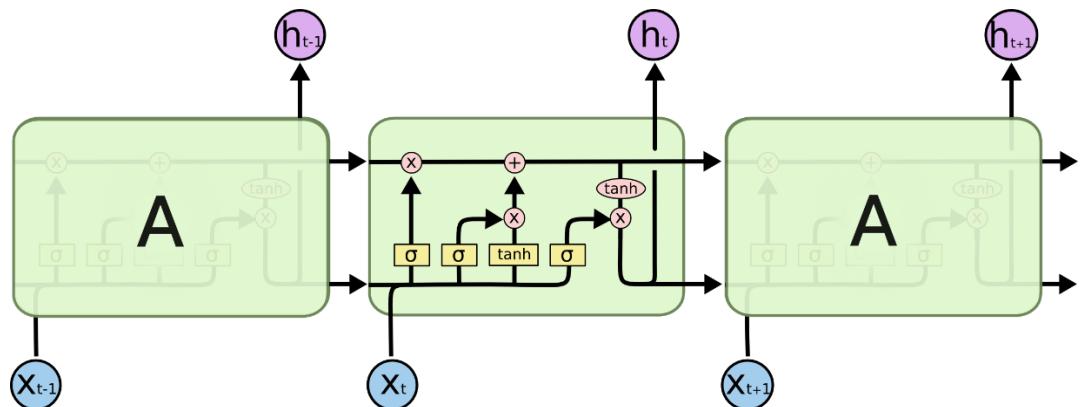
[15] Mạng bộ nhớ dài-ngắn được viết với tên đầy đủ là Long Short-Term Memory và thường được gọi tắt là LSTM, là một trường hợp đặc biệt của mạng neural hồi quy RNN. Mô hình này được giới thiệu bởi nhà nghiên cứu Hochreiter & Schmidhuber vào năm 1997. Sau đó, mô hình LSTM đã được cải tiến và phổ biến lại bởi nhiều người trong lĩnh vực nghiên cứu về RNN. Nhờ vào khả năng học được các phụ thuộc từ xa nên hoạt động rất hiệu quả trong nhiều bài toán khác nhau và dần dần trở nên phổ biến cho đến thời điểm hiện nay.

Mạng LSTM được thiết kế nhằm loại bỏ được các vấn đề liên quan đến phụ thuộc xa. Mọi mô hình mạng neural hồi quy – RNN đều có một dạng là chuỗi các module lặp đi lặp lại của mạng neural. Các lớp được mắc nối tiếp với nhau và tạo thành các module neural network. Trong mạng RNN chuẩn, các module được lặp lại này có cấu trúc khá đơn giản và thông thường chỉ bao gồm một tầng tanh.



Hình 3.12: Các module lặp của mạng RNN chuẩn. [15]

Đối với LSTM, nó cũng có cấu trúc mắt xích tương tự với cấu trúc của RNN chuẩn nhưng điểm khác biệt là cấu trúc của các module trong LSTM lặp có cấu trúc khác hẳn so với RNN. Thay vì chỉ có một tầng ở RNN chuẩn thì với LSTM chúng có tới 4 tầng, và mỗi tầng tương tác với nhau theo một cấu trúc cụ thể.



Hình 3.13: Cấu trúc lặp của LSTM với 4 tầng. [15]

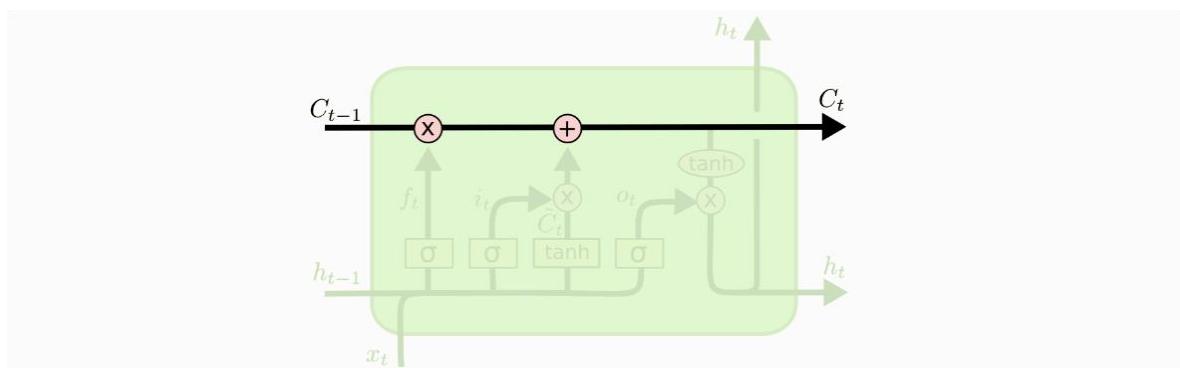
Trong đó, các ký hiệu trong hình 3.13 với các ý nghĩa như sau:

- : là lớp ẩn được học trong từng mạng neural.
- : được gọi là Pointwise, dùng để biểu diễn các phép toán như cộng hoặc nhân vector.
- : được gọi là Vector Transform, dùng để biểu diễn mạng đầu vào và đầu ra của một nút.

-  : được gọi là Concatenate, dùng để biểu thị phép nối các toán hạng.
-  : được gọi là Copy, dùng để biểu thị nội dung sẽ được sao chép và chuyên tới các vị trí khác.

3.5.2 Phân tích mô hình LSTM

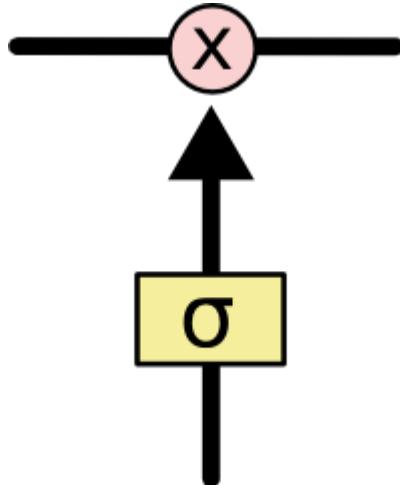
Điểm quan trọng của mô hình LSTM đó là các tế bào trạng thái (Cell State) và nó được ký hiệu là một đường kẻ ngang chạy dọc qua một module của mạng LSTM.



Hình 3.14: Cell State trong mô hình LSTM. [15]

Tế bào trạng thái thực hiện truyền xuyên thảng toàn bộ các mắc xích trong mạng và chỉ thực hiện một vài các tương tác nhỏ tuyến tính. Chính vì điều này, các tế bào trạng thái giúp cho thông tin có thể dễ dàng được truyền đi thông suốt mà không sợ bị thay đổi trong suốt quá trình lan truyền.

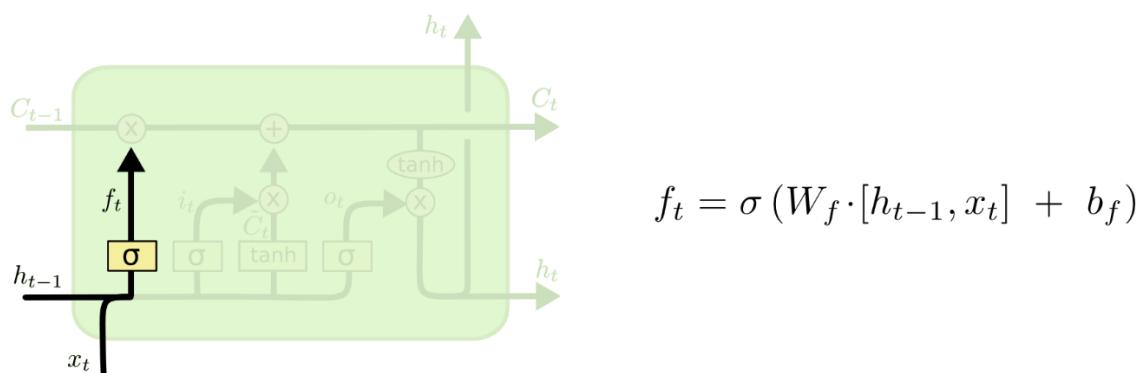
Bên cạnh đó, mô hình LSTM còn có khả năng bỏ đi hoặc thêm các thông tin vào các tế bào trạng thái (Cell State), việc thêm thông tin vào tế bào trạng thái được quy định một cách cẩn thận bởi các nhóm được gọi là cổng (Gate). Các cổng này là nơi thực hiện việc sàng lọc thông tin và thường chúng được kết hợp tạo bởi một hàm sigmoid với một toán tử nhân Pointwise.



Hình 3.15: Cổng trạng thái của mô hình LSTM. [15]

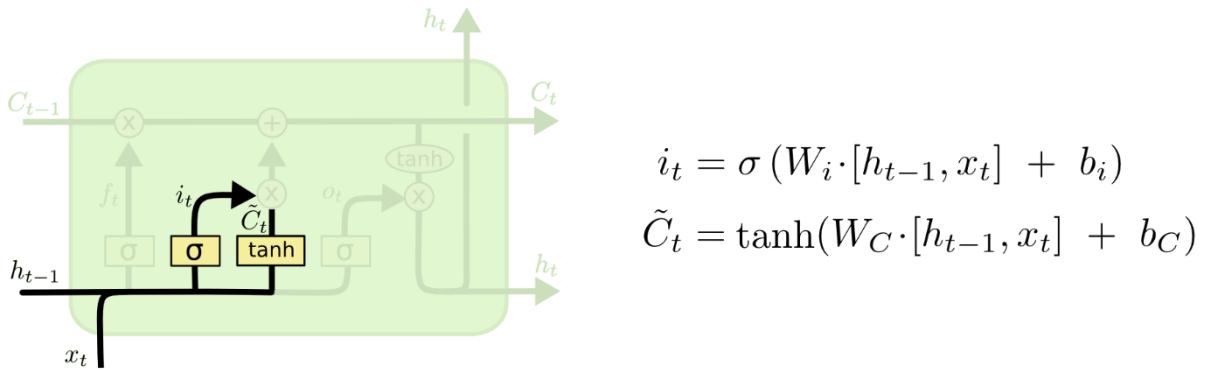
3.5.3 Nguyên lý hoạt động của LSTM

Bước đầu tiên của mô hình LSTM, đó là đưa ra quyết định xem thông tin nào cần loại bỏ khỏi tế bào trạng thái (Cell State). Ở bước này, quá trình được thực hiện thông qua tầng sigmoid được gọi là “Forget Gate Layer”. Nó thực hiện lấy đầu vào là h_{t-1} và x_t rồi đưa kết quả là một số trong khoảng $[0, 1]$ ứng với mỗi số trong tế bào trạng thái C_{t-1} . Nó được biểu diễn dưới dạng 1 và 0, tương ứng với 1 là “giữ lại thông tin” và ứng với giá trị 0 là “loại bỏ thông tin”.



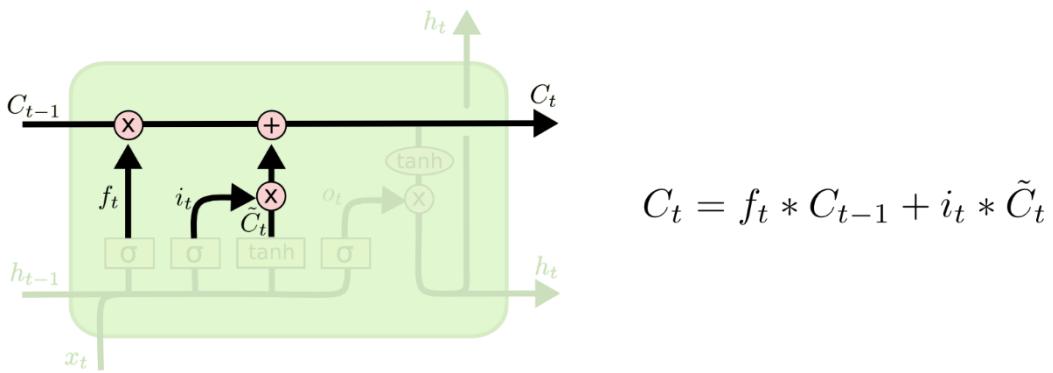
Hình 3.16: Bước đầu tiên trong mô hình LSTM. [15]

Bước kế tiếp, chúng ta cần quyết định rằng những thông tin nào cần được ghi nhớ, lưu trữ lại tại tế bào trạng thái. Ở bước này, ta có hai phần bao gồm tầng single sigmoid được gọi là “Input Gate Layer” thực hiện quyết định các giá trị chúng ta sẽ cập nhật. Phần thứ hai là một *tanh* layer để tạo ra một vector cho giá trị mới \tilde{C}_t nhằm thêm vào cho trạng thái.



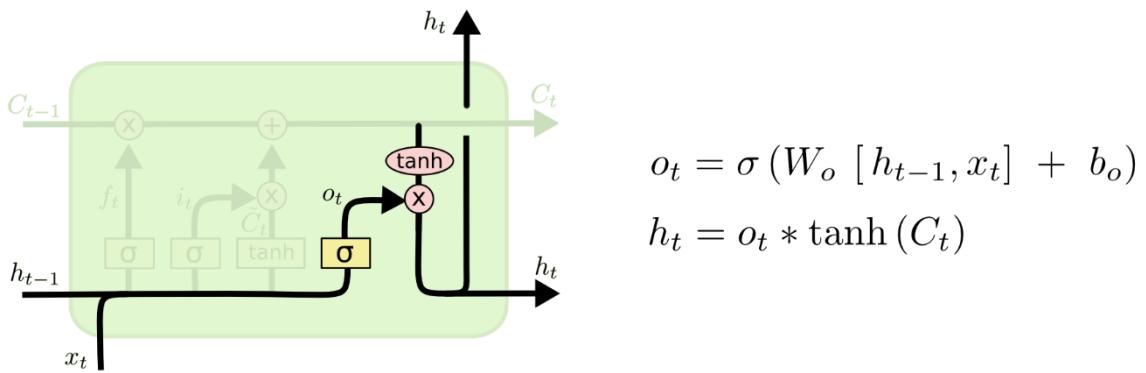
Hình 3.17: Bước thứ hai trong mô hình LSTM. [15]

Ở bước tiếp theo này, chúng ta sẽ thực hiện kết hợp hai thành phần ở trên lại với nhau để truyền các cập nhật vào tế bào trạng thái (Cell State). Việc cập nhật sẽ diễn ra bằng cách thay thế các tế bào trạng thái cũ C_{t-1} thành trạng thái mới C_t . Sau đó, ta sẽ thực hiện nhân trạng thái cũ với f_t để quên đi những gì ở trước đó. Sau đó, chúng ta sẽ thực hiện cộng thêm $i_t * \tilde{C}_t$. Và giá trị trạng thái mới phụ thuộc vào việc mà chúng ta quyết định đưa ra các quyết định cập nhật các trạng thái đó như thế nào.



Hình 3.18: Bước thứ ba trong mô hình LSTM. [15]

Bước cuối cùng, chúng ta cần quyết định xem thông tin đầu ra output là gì. Với giá trị output, nó sẽ dựa vào trạng thái tế bào (Cell State) của chúng ta, nhưng sẽ được lọc bỏ đi các thông tin. Quá trình bắt đầu bằng việc áp dụng một tầng sigmoid để quyết định xem rằng phần nào của cell state của chúng ta dự định sẽ xuất ra. Sau đó, chúng ta sẽ thực hiện đưa cell state qua một hàm tanh để rút gọn giá trị của nó về khoảng [-1, 1]. Cuối cùng, ta nhân nó với đầu ra của cổng sigmoid để giữ lại những phần ta muốn output ra ngoài.



Hình 3.19: Bước cuối cùng trong mô hình LSTM. [15]

3.5.4 Mô hình Seq2Seq

3.5.4.1 Giới thiệu

Trong hệ thống Chatbot của nhóm, chúng em sử dụng mô hình Sequence-to-Sequence dựa trên nền tảng của mạng LSTM để thực hiện việc sinh văn bản cho Chatbot. Mô hình Sequence-to-Sequence được viết tắt là Seq2Seq, được ứng dụng cho nhiều bài toán khác nhau trong lĩnh vực Machine Learning như dịch máy (Machine Translation), nhận diện thực thể có tên (Named Entity Recognition), phân loại cảm xúc (Sentiment Classification) và đặc biệt là trong lĩnh vực trả lời tự động (Chatbot). Kể từ khi các kiến trúc về mạng neural hồi quy và LSTM ra đời thì quá trình dịch thuật đã và đang được phát triển hơn. Hiện nay, có rất nhiều bên cung cấp các dịch vụ dịch như Amazon Translate và không thể không nhắc đến một ông lớn đó là Google với Google Translate. Cũng chính Google cũng đang áp dụng mô hình Seq2Seq với kỹ thuật Attention để áp dụng cho dịch vụ ứng dụng dịch của mình.

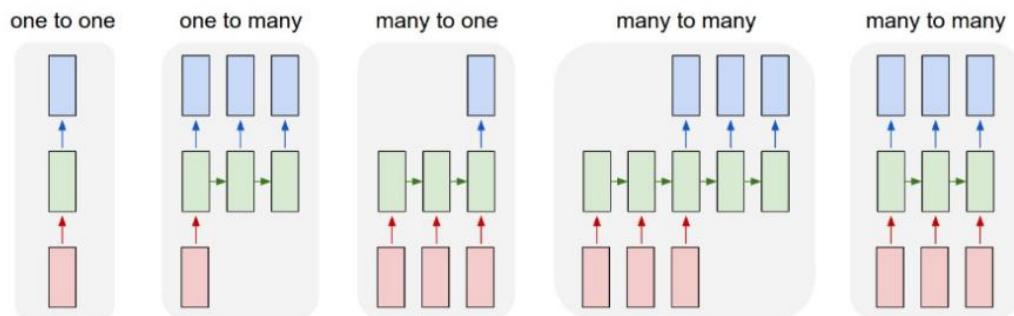
3.5.4.2 Lịch sử

Seq2Seq là một thuật toán được Google phát triển để sử dụng trong việc dịch máy. Vào năm 2019, Facebook đã công bố việc sử dụng mô hình Seq2Seq được tích hợp trong việc giải các phương trình vi phân. Facebook đưa ra tuyên bố rằng với mô hình Seq2Seq thì nó có thể giải các phương trình phức tạp nhanh hơn và với độ chính xác cao hơn các giải pháp thương mại khác như Mathematica, MATLAB và Maple. Vào năm 2020, Google đã cho phát hành Meena là một chatbot dựa trên mô hình Seq2Seq với khoảng 2,6 tỷ tham số được huấn luyện trên tập dữ liệu 341 GB. Google tuyên bố rằng đây là ứng dụng Chatbot có dung lượng mô hình lớn hơn gấp 1,7 lần so với GPT-2 của

OpenAI. Các sự kiện kể trên cũng có thể được xem là các mốc thời gian quan trọng và đặc biệt đối với mô hình Seq2Seq.

3.5.4.3 Khái niệm

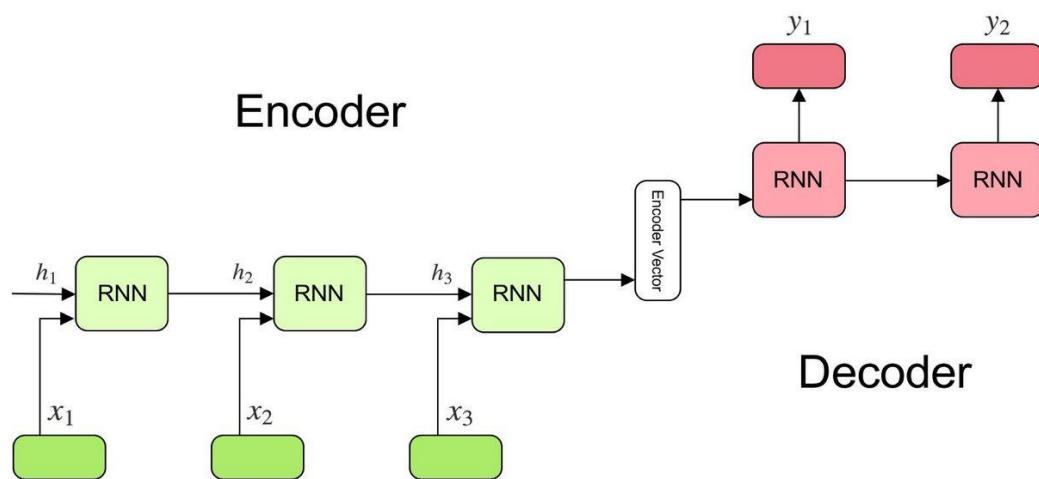
Về mặt khái niệm, Seq2Seq là một mô hình cơ bản trong lĩnh vực học sâu (Deep Learning) và đây là một nhánh của học máy. Với Seq2Seq, nó đã đạt được nhiều thành công với nhiều ứng dụng nổi bật trong lĩnh vực học máy như: dịch máy, chatbot, tóm tắt văn bản, ... Về cơ bản, mô hình Seq2Seq nhận giá trị đầu vào là một chuỗi các từ hoặc ký tự và đầu ra của nó là một chuỗi khác.



Hình 3.20: Một số mô hình Seq2Seq.

3.5.4.4 Cấu trúc mô hình Seq2Seq

Với mô hình Seq2Seq, nó sử dụng kiến trúc Encoder-Decoder bao gồm một bộ mã hóa (Encoder) và một bộ giải mã (Decoder). Hai khối mã hóa và giải mã này được kết nối với nhau thông qua một vector trung gian được gọi là Context Vector.

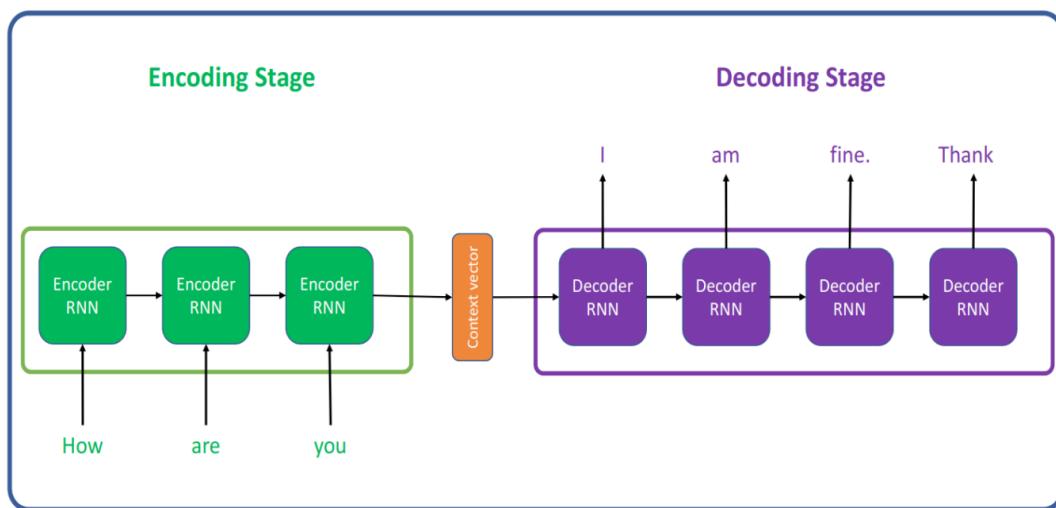


Hình 3.21: Mô hình Seq2Seq sử dụng hai bộ Encoder và Decoder.

- **Bộ mã hóa - Encoder:** một lớp RNN đóng vai trò như một bộ mã hóa và nó xử lý chuỗi đầu vào và trả về trạng thái bên trong của chính nó. Chú ý rằng việc loại

bỏ các đầu ra của bộ mã hóa RNN và chỉ thực hiện khôi phục các trạng thái. Trạng thái đầu ra của bộ mã hóa này sẽ đóng vai trò như là "ngữ cảnh" của bộ giải mã được thực hiện trong bước tiếp theo. Với bộ mã hóa encoder, nó sẽ nhận giá trị đầu vào là một chuỗi

- **Bộ giải mã - Decoder:** một lớp RNN khác hoạt động như một bộ giải mã và nó được sử dụng huấn luyện để dự đoán các ký tự tiếp theo của chuỗi mục tiêu khi ta cho các ký tự. Nó được đào tạo để biến các chuỗi mục tiêu thành các chuỗi giống nhau nhưng bù lại bằng một giá trị timestep trong tương lai. Quá trình đào tạo đó được gọi là "Teacher Forcing" trong trường hợp này. Một điều quan trọng, bộ giải mã – Decoder sử dụng như trạng thái ban đầu của các vectơ trạng thái từ bộ mã hóa. Đó chính là cách mà bộ giải mã thu được thông tin về những gì nó được cho là tạo ra.



Hình 3.22: Mô hình Seq2Seq trong việc sinh chuỗi.

Tuy nhiên rằng, mô hình Seq2Seq cũng có một số các hạn chế của nó do bộ mã hóa chuyển đổi toàn bộ chuỗi đầu vào thành một vector có độ dài cố định và sau đó bộ giải mã dự đoán chuỗi đầu ra nên:

- Kiến trúc này chỉ hoạt động tốt với các chuỗi ngắn.
- Khó để bộ mã hóa ghi nhớ các chuỗi dài thành một vector có độ dài cố định.

Và do đó, một cơ chế mới được phát minh ra nhằm khắc phục những vấn đề đó chính là cơ chế Attention. Nhằm mục đích dự đoán một từ bằng cách xem xét một vài phần cụ thể của chuỗi thay vì toàn bộ chuỗi.

3.6 Tổng kết chương 3

Như vậy là chúng ta đã kết thúc chương thứ ba của bài báo cáo, qua chương này chúng ta đã tìm hiểu các khái niệm, lịch sử hình thành, các đặc điểm về mạng neural sinh học, neural nhân tạo cho đến mạng neural nhân tạo, ... Cùng với đó là việc tìm hiểu sâu hơn về mạng neural hồi quy Recurrent Neural Network – RNN để có một cái nhìn tổng quát và rõ ràng hơn về khái niệm mạng hồi quy này. Bên cạnh đó, chúng ta cũng đã tìm hiểu về các khái niệm của các mô hình liên quan đến RNN như mô hình Long Short-Term Memory với việc tìm hiểu các khái niệm, đặc điểm và nguyên lý hoạt động của nó giúp cho chúng ta hiểu được từng bước tính toán của LSTM nói riêng và RNN nói chung. Cuối cùng, chúng ta đã tìm hiểu sơ qua về mô hình Sequence-to-Sequence với hai bộ mã hóa và giải mã là Encoder và Decoder sẽ được ứng dụng trong hệ thống Chatbot của nhóm. Ké tiếp, chúng ta sẽ thực hiện triển khai chương trình Chatbot và thực hiện phân tích kết quả thu được từ chương trình triển khai. Từ đó, ta có thể đưa ra đánh giá về kết quả thu được.

CHƯƠNG 4: TRIỂN KHAI HỆ THỐNG TRẢ LỜI TỰ ĐỘNG

4.1 Tổng quan về Django Framework

4.1.1 Khái niệm

[17] Django là một loại web framework cao cấp miễn phí được viết hoàn toàn bằng ngôn ngữ lập trình Python. Nó là một framework khá nổi tiếng và là mã nguồn mở (open-source) được thiết kế theo mô hình MTV (Model – Template – Views) và được xây dựng bởi những nhà phát triển có kinh nghiệm và giải quyết nhiều rắc rối trong quá trình phát triển Web với việc hỗ trợ khá đầy đủ các thư viện cùng với module hỗ trợ cho những nhà lập trình viên web. Với việc đơn giản hóa việc tạo ra các website có sử dụng cơ sở dữ liệu thì đây là mục tiêu chính mà Django hướng đến người dùng.

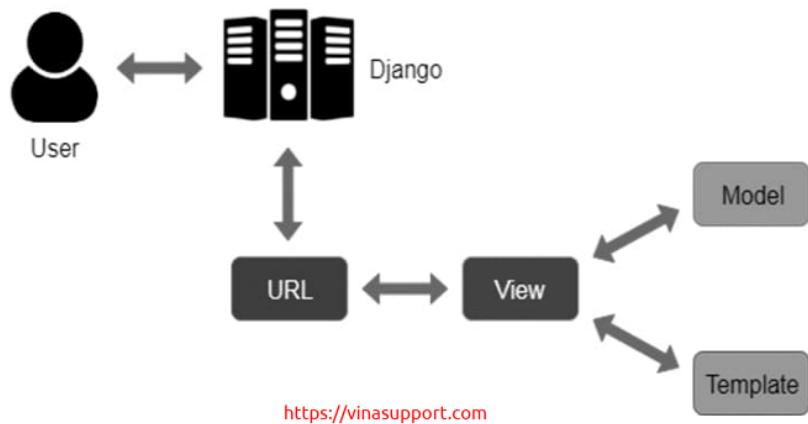


Hình 4.1: Logo Django Framework. [17]

Ngoài ra, các tính năng như việc “có thể tự chạy” và “có thể tái sử dụng” của các component hay tính năng phát triển nhanh, không thực hiện lại những điều đã làm cũng là các tính năng mà Django tập trung hướng đến. Django Framework hiện tại có một cộng đồng người dùng đông đảo và có rất nhiều các tài liệu hỗ trợ liên quan đến nó nên việc sử dụng khá đơn giản và dễ dàng. Một số các website nổi tiếng và phổ biến hiện nay đã sử dụng Django trong việc thiết kế có thể Instagram, Mozilla, Pinterest, ...

4.1.2 Mô hình MTV

Django sử dụng mô hình MTV thay vì mô hình MVC như các framework khác. Tuy nhiên, mô hình MTV là một biến thể của mô hình MVC nên về bản chất cũng gần như tương tự với mô hình MVC cơ bản. MVT là cụm từ viết tắt của Model – View – Template và so với mô hình MVC (Model – View – Controller).



Mô hình MVT của Django

Hình 4.2: Mô hình MVT trong Django Framework. [17]

Django hoàn toàn tuân thủ theo mô hình MVC mặc dù rằng tên của nó khác với mô hình MVC. Bởi vì đối với bản thân framework này, người lập trình phần lớn chỉ tương tác với Model, Template và view nên Django thường được hiểu là sử dụng mô hình MVT. Ứng với mỗi thành phần trong MVT thì nó đảm nhận một chức năng riêng biệt như sau:

Thành phần	Mô tả
Model	Là lớp truy cập dữ liệu hay là nơi để thiết kế ra các table cho database. Nơi này sẽ đảm nhận nhiệm vụ cung cấp các phương thức xử lý, nghiệp vụ lên cơ sở dữ liệu như validate dữ liệu, các phương thức và hành vi của dữ liệu, mối quan hệ của dữ liệu.
View	Là nơi chứa các logic để thực hiện truy cập dữ liệu qua Model và truyền nó ra ngoài cho Template tương ứng. Hay nói một cách đơn giản đây là nơi chứa các hàm function hoặc class để xử lý các request

	tù người dùng. Views được coi như là một cầu nối giữa Model và Template.
Template	Là lớp dùng để hiển thị chứa những gì liên quan đến việc hiển thị dữ liệu cho người dùng. Hay nói một cách đơn giản thì Template là nơi để thiết kế ra và xử lý kết quả output ra mã HTML hoặc CSS cho trang web.

Bảng 4.1: Bảng các chức năng trong mô hình MVT.

4.1.3 Đặc điểm của Django

Django là một framework được sử dụng phổ biến bởi nó có các đặc điểm nổi bật như sau:

- **Tính linh hoạt:** Django có thể được sử dụng để xây dựng nhiều loại trang web với nhiều các lĩnh vực khác nhau như từ bán hàng cho đến các mạng xã hội hay các trang báo tin tức, ... Nó có thể hoạt động cùng với các framework bên ngoài và cũng
- **Độ hoàn thiện cao:** Django cung cấp hầu hết những thứ mà các lập trình viên phát triển web cần bao gồm các thư viện, các module, ... Tất cả các thành phần trong framework hoạt động liền mạch nhau và tuân theo một nguyên tắc thiết kế nhất quán, ... Do đó, chúng ta có thể thiết kế và phát triển web theo phong cách riêng và tiết kiệm được khá nhiều thời gian.
- **Tính bảo mật:** Django rất chú trọng về việc bảo mật giúp cho những nhà phát triển tránh được nhiều lỗi bảo mật cơ bản và phổ biến như: SQL Injection, Cross-site Scripting, Cross-Site Request Forgery, ...

Ví dụ: Django cung cấp một danh sách các giải pháp an toàn trong việc tạo tài khoản người dùng bằng việc quy định cách đặt mật khẩu của người dùng phải tuân thủ theo các tiêu chuẩn bảo mật và an toàn.

- **Khả năng mở rộng:** do Django sử dụng kiến trúc thành phần riêng nên chúng ta có thể mở rộng quy mô của nó bằng cách thêm phần cứng như bộ nhớ đệm, máy chủ cơ sở dữ liệu hay máy chủ ứng dụng, ...
- **Khả năng duy trì:** với việc các đoạn mã code trong Django được viết theo cách sử dụng các nguyên tắc và khuyến khích việc thiết kế tạo ra các mã để có thể dễ

dàng bảo trì và tái sử dụng. Từ đó, giúp cho các website sử dụng Django có khả năng duy trì cao hơn.

Nhìn chung Django là một framework có nhiều các ưu điểm nổi bật và hữu ích. Tuy nhiên nó vẫn có một số nhược điểm như sau:

❖ **Ưu điểm:**

- Có tính bảo mật cao.
- Thực hiện đơn giản và nhanh chóng.
- Phù hợp với mọi các dự án website.
- Có cộng đồng người dùng lớn và tài liệu hỗ trợ phong phú.

❖ **Nhược điểm:**

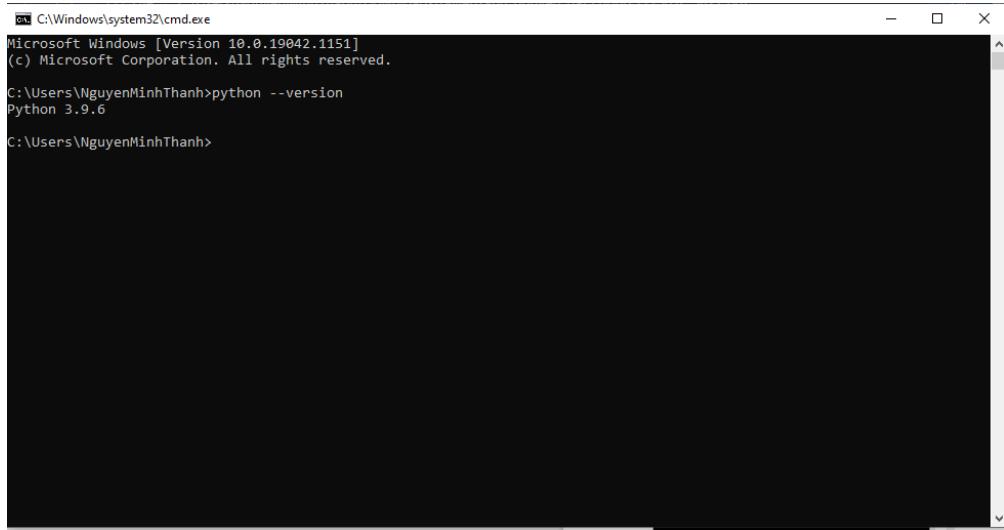
- Phải tuân theo quy tắc và khuôn khổ nên chúng ta phải tuân thủ theo nó.
- Đối với các lỗi trên template, chúng ta sẽ phải tự tìm ra vấn đề nằm ở đâu mà không hiển thị thông báo lỗi. Do đó, dẫn đến mất nhiều thời gian để giải quyết vấn đề.
- Chỉ định URL bằng các quy tắc biểu thức gây khó khăn cho những người mới bắt đầu.

4.1.4 Triển khai Django

Trong phần này, chúng ta sẽ đi tìm hiểu cách cài đặt và xây dựng một dự án Django đơn giản để hiểu rõ hơn về framework này. Việc cài đặt Django sẽ trải qua từng bước như sau:

✚ **Bước 1: Cài đặt môi trường Python**

Để cài đặt Django, trước hết ta phải cài đặt môi trường Python trước khi thực hiện cài đặt framework. Sau khi cài đặt xong môi trường Python cho máy, chúng ta kiểm tra xem rằng quá trình cài đặt đã thực sự thành công hay không bằng cách kiểm tra phiên bản hiện tại có nó bằng câu lệnh “**python –version**”.



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19042.1151]
(c) Microsoft Corporation. All rights reserved.

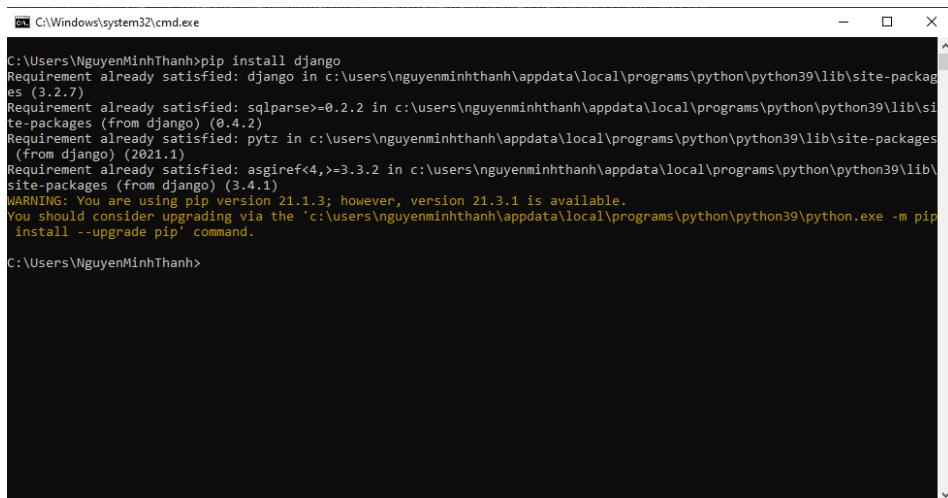
C:\Users\NguyenMinhThanh>python --version
Python 3.9.6

C:\Users\NguyenMinhThanh>
```

Hình 4.3: Kiểm tra phiên bản Python cài đặt trên máy.

⊕ **Bước 2: Cài đặt Django**

Sau khi cài đặt môi trường Python cho máy, ta tiến hành cài đặt framework Django bằng câu lệnh “**pip install django**” trong cửa sổ CMD.



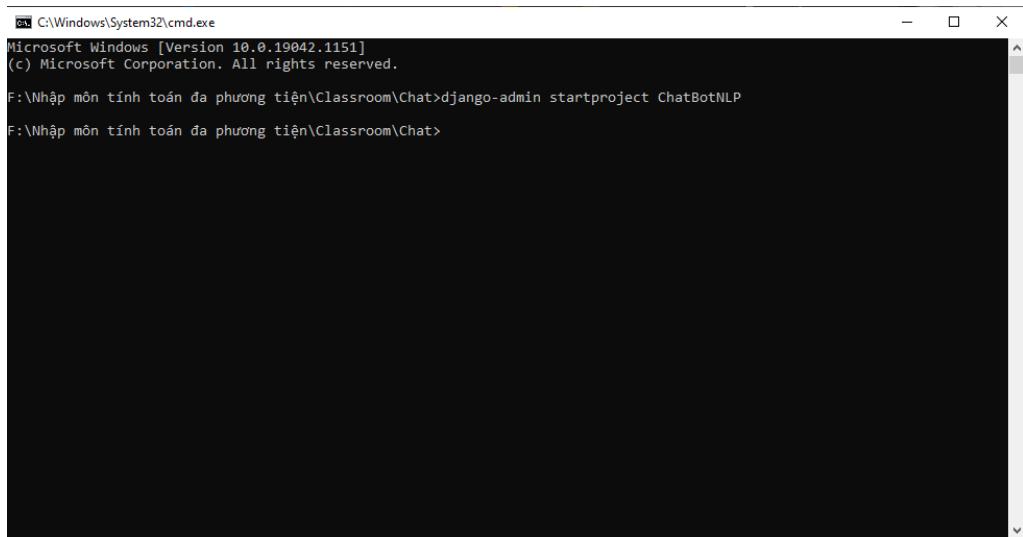
```
C:\Windows\system32\cmd.exe
C:\Users\NguyenMinhThanh>pip install django
Requirement already satisfied: django in c:\users\nguyenminhthan\appdata\local\programs\python\python39\lib\site-packages (3.2.7)
Requirement already satisfied: sqlparse>=0.2.2 in c:\users\nguyenminhthan\appdata\local\programs\python\python39\lib\site-packages (from django) (0.4.2)
Requirement already satisfied: pytz in c:\users\nguyenminhthan\appdata\local\programs\python\python39\lib\site-packages (from django) (2021.1)
Requirement already satisfied: asgiref<4,>=3.3.2 in c:\users\nguyenminhthan\appdata\local\programs\python\python39\lib\site-packages (from django) (3.4.1)
WARNING: You are using pip version 21.1.3; however, version 21.3.1 is available.
You should consider upgrading via the 'c:\users\nguyenminhthan\appdata\local\programs\python\python39\python.exe -m pip
install --upgrade pip' command.

C:\Users\NguyenMinhThanh>
```

Hình 4.4: Câu lệnh cài đặt Django.

⊕ **Bước 3: Tạo project**

Sau khi thực hiện cài đặt xong môi trường và framework Django, ta tiến hành tạo ra một dự án mới bằng cách gõ câu lệnh “**django-admin startproject ChatBotNLP**” vào cửa sổ CMD. Lưu ý rằng, nếu ta muốn tạo một project mới ở vị trí nào thì việc đầu tiên ta phải chuyển đường dẫn hiện tại ở cửa sổ CMD đến vị trí mà ta muốn lưu project bằng câu lệnh “**cd**”.

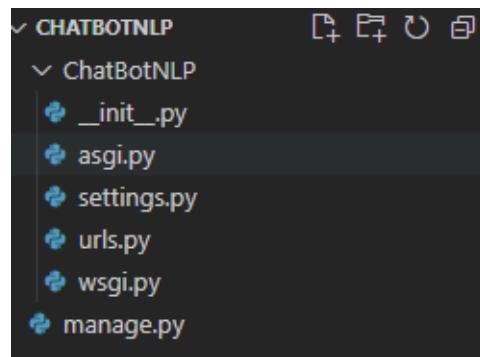


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1151]
(c) Microsoft Corporation. All rights reserved.

F:\Nhập môn tính toán đa phương tiện\Classroom\Chat>django-admin startproject ChatBotNLP
F:\Nhập môn tính toán đa phương tiện\Classroom\Chat>
```

Hình 4.5: Câu lệnh tạo project mới trong Django.

Sau khi thực hiện lệnh “**django-admin startproject ChatBotNLP**” để tạo project mới thì trong thư mục hiện tại sẽ xuất hiện ra một thư mục con có tên là ChatBotNLP. Với cấu trúc bên trong thư mục ChatBotNLP sẽ bao gồm các file và mỗi file sẽ đảm nhận một nhiệm vụ khác nhau.



Hình 4.6: Cấu trúc thư mục project sau khi tạo.

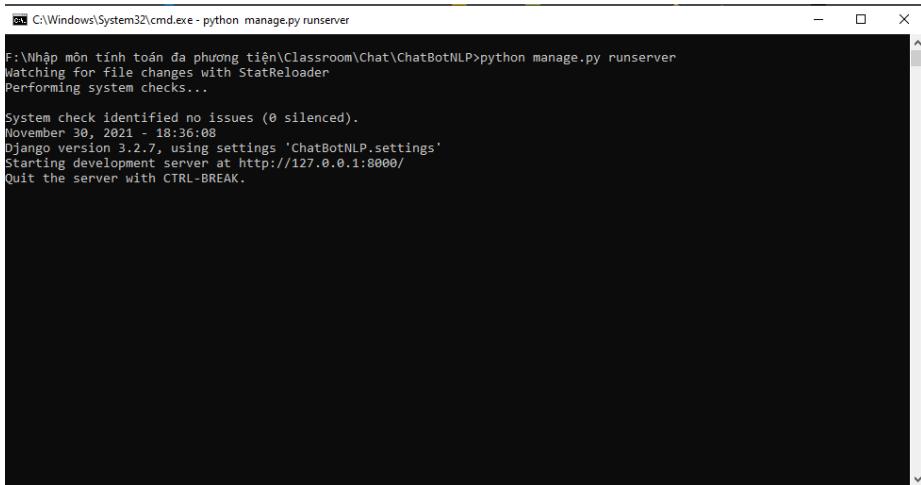
Trong đó:

- `__init__.py`: là một file rỗng được dùng để định nghĩa với trình thông dịch python rằng thư mục này là một python package.
- `asgi.py`: là tệp cho phép các máy chủ web tương thích ASGI phục vụ dự án của bạn (thông thường chỉ Django 3).
- `settings.py`: là file chứa các cài đặt của project như DEBUG, ALLOW_HOSTS, ...
- `urls.py`: là tệp dùng để khai báo các URL của project và nó được liên kết đến các views để xử lý request của người dùng.

- wsgi.py: là file dùng để cấu hình deploy project lên server.
- manage.py: là file cho phép thực hiện tương tác hoặc quản lý với project bằng dòng lệnh như tạo app, tạo superuser, migrate, ...

Bước 4: Chạy Server

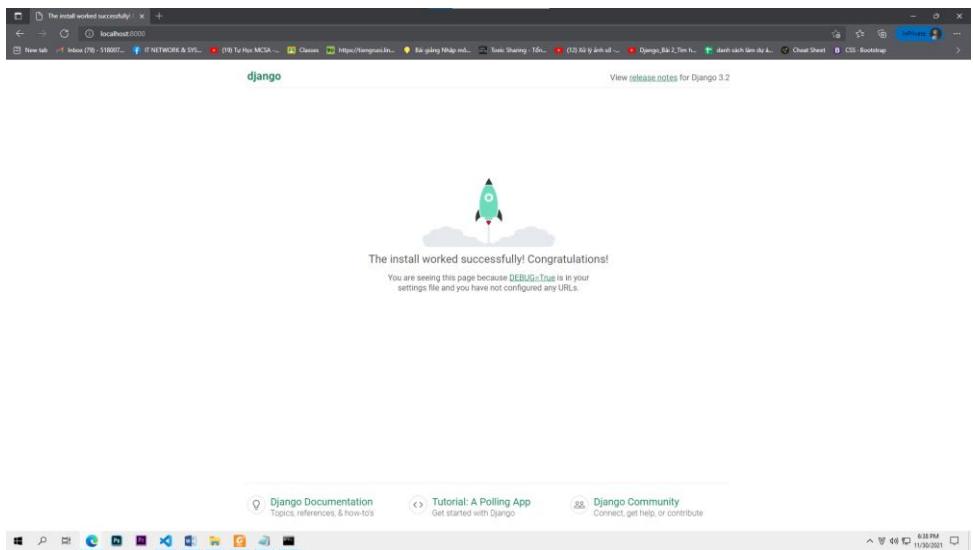
Để kiểm tra xem rằng quá trình tạo project mới đã thành công hay chưa, ta tiến hành khởi chạy server để tiến hành kiểm tra. Ta sử dụng lệnh “**python manage.py runserver 8888**”. Ở đây, ta có thể sử dụng câu lệnh “**python manage.py runserver**” mà không cần có số theo sau. Các số theo sau câu lệnh trên chính là số hiệu cổng (port) mà ta muốn server chạy trên cổng được chỉ định đó. Theo mặc định, nếu người dùng không nhập số hiệu cổng cho câu lệnh khởi chạy server thì hệ thống sẽ mặc định khởi chạy trên cổng 8000.



```
C:\Windows\System32\cmd.exe - python manage.py runserver
F:\NLP> Nhập môn tính toán và phương tiện\Classroom\Chat\ChatBotNLP>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...
System check identified no issues (0 silenced).
November 30, 2021 - 18:36:08
Django version 3.2.7, using settings 'ChatBotNLP.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Hình 4.7: Câu lệnh khởi chạy server cho project.

Sau khi khởi chạy server thành công và không có lỗi xuất hiện, lúc này ta tiến hành mở trình duyệt và nhập địa chỉ tương ứng với số hiệu cổng mặc định mà Django tự khởi tạo là 8000 với địa chỉ “**http://localhost:8000**” hoặc địa chỉ “**http://127.0.0.1:8000**” vào thanh địa chỉ trên trình duyệt.



Hình 4.8: Giao diện sau khi khởi chạy server thành công.

⊕ **Bước 5: Tạo ứng dụng cho project**

Một project Django có thể bao gồm nhiều app khác nhau, trong đó mỗi app thực hiện một công việc khác nhau và riêng biệt. Để tạo một web app ta sử dụng câu lệnh “**python manage.py startapp chatbotapp**” vào cửa sổ CMD. Trong đó, “**python manage.py startapp**” là lệnh để khởi tạo web app và “**chatbotapp**” là tên của web app.

A screenshot of a Windows Command Prompt window titled 'C:\Windows\System32\cmd.exe'. The window shows the following text:

```
Microsoft Windows [Version 10.0.19042.1151]
(c) Microsoft Corporation. All rights reserved.

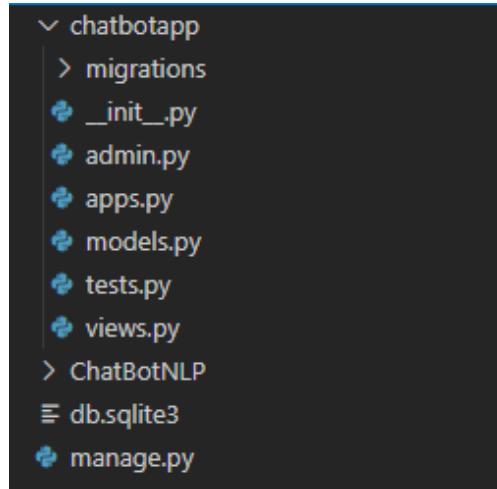
F:\Nhập môn tính toán đa phương tiện\Classroom\Chat\ChatBotNLP>python manage.py startapp chatbotapp

F:\Nhập môn tính toán đa phương tiện\Classroom\Chat\ChatBotNLP>
```

The command 'python manage.py startapp chatbotapp' was entered and executed successfully.

Hình 4.9: Câu lệnh khởi tạo web app cho project Django.

Sau khi thực thi câu lệnh, một thư mục mới sẽ được xuất hiện trong thư mục của project ChatBotNLP như sau:



Hình 4.10: Thư mục web app sau khi khởi tạo thành công.

Ta thấy rằng các sau khi tạo web app, nó cũng bao gồm các file giống với các file ở trong thư mục project lớn của nó như `_init__.py`. Và bao gồm một số các file mới, trong đó:

- **admin.py**: là nơi chứa các thông tin liên quan đến việc triển khai các module admin cho trang web.
- **apps.py**: là nơi chứa các cài đặt của app.
- **models.py**: là nơi để tương tác, giao tiếp với cơ sở dữ liệu của app.
- **tests.py**: là tệp được sử dụng cho việc kiểm thử và thường được dùng để thực hiện unit-test.
- **views.py**: là nơi chứa các logic để tương tác với dữ liệu thông qua model hoặc truyền ra ngoài thông qua các Templates tương ứng.

Bước 5: Khởi tạo Model

Model là thành phần để ánh xạ giữa cơ sở dữ liệu Database và Django, để Python có thể tương tác với Database và lấy được nó thông qua các model. Chúng ta sẽ tiến hành đi xây dựng một cơ sở dữ liệu đơn giản để hiểu rõ hơn về việc tương tác với model.

```

from typing import ChainMap
from django.db import models
from django.utils.timezone import now
from django.contrib.auth.models import User

# Create your models here.
class MyClass(models.Model):
    class Meta:
        verbose_name = 'Lớp'
        verbose_name_plural = 'Danh sách lớp'
    malop = models.CharField(max_length=50,verbose_name='Mã lớp',primary_key=True)
    tenlop = models.CharField(max_length=50,verbose_name='Tên lớp')
    soluong = models.IntegerField(verbose_name='Số lượng')
    gvcn = models.CharField(max_length=50,verbose_name='GVCN')
    owner = models.ForeignKey(User,verbose_name="Người tạo",blank=True,null=True,on_delete=models.SET_NULL)

    def __str__(self):
        return f"{self.malop}"

```

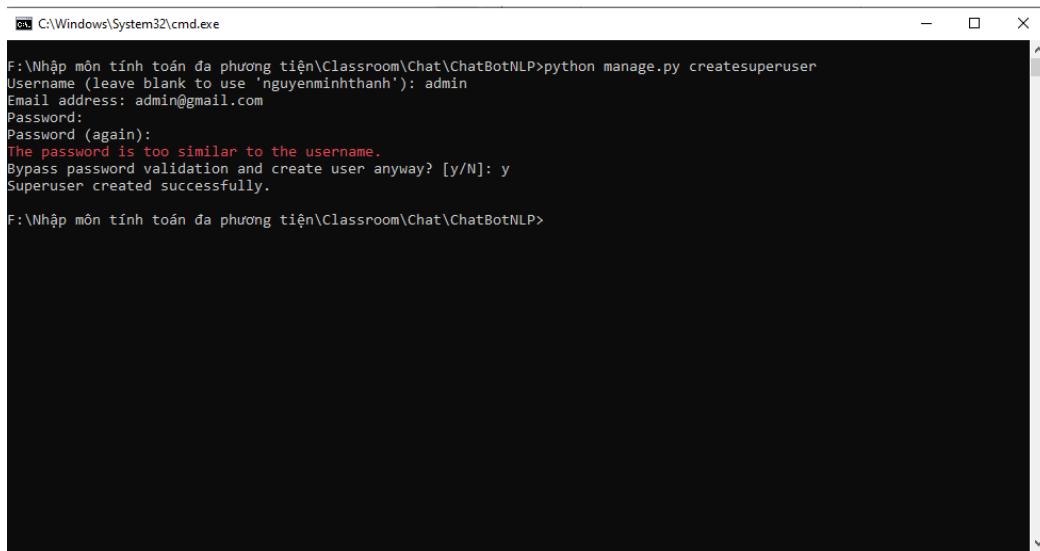
Hình 4.11: Khởi tạo bảng trong CSDL thông qua Model.

Ở bước khởi tạo này, hai từ khóa “verbose_name” và “verbose_name_plural” là hai từ khóa dành cho việc hiển thị tên trên trang quản trị Admin. Ứng với mỗi trường của bảng, ta sẽ có một kiểu dữ liệu khác nhau và phù hợp với nó như CharField, IntegerField, ForeignKey, ... Với các tham số như max_length nghĩa là chiều dài tối đa, primary_key được dùng để xác định một trường là khóa chính hay các từ khóa như blank, null, on_delete, ...

Bước 6: Hệ thống Admin

Khi ta xây dựng một trang web nào đó chẳng hạn như một trang web bán hàng thì chúng ta cần phải xây dựng nên một trang web quản trị để quản lý mọi thứ như là có thêm, xóa, sửa các mặt hàng và hiển thị các thông tin lên giao diện. Nhưng đối với Django, nó hỗ trợ khá mạnh cho người dùng về mảng Backend nên nó đã cung cấp cho người dùng sẵn một giao diện quản lý được xây dựng sẵn và gần như người dùng chỉ việc sử dụng nó mà không cần làm thêm bất cứ trang quản trị nào khác.

Ban đầu, sau khi mới khởi tạo project thì để ta có thể vào được giao diện hệ thống của Admin thì chúng ta phải tiến hành khởi tạo một tài khoản Admin với quyền cao nhất bằng cách gõ câu lệnh “**python manage.py createsuperuser**” vào cửa sổ CMD.

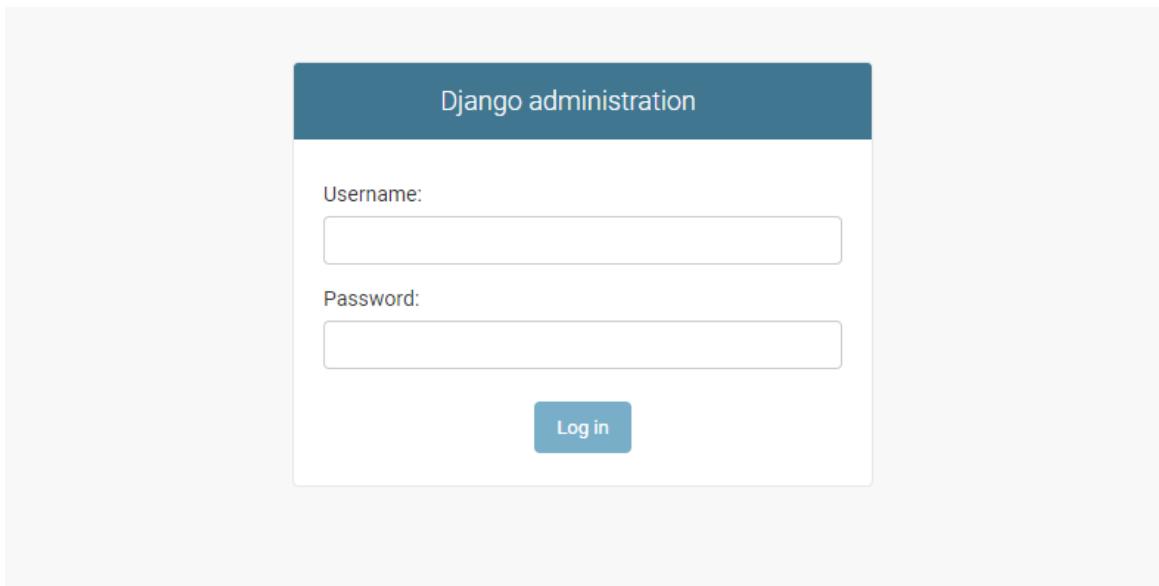


```
C:\Windows\System32\cmd.exe
F:\Nhập môn tính toán đa phương tiện\Classroom\Chat\ChatBotNLP>python manage.py createsuperuser
Username (leave blank to use 'nguyenminhthanh'): admin
Email address: admin@gmail.com
Password:
Password (again):
The password is too similar to the username.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

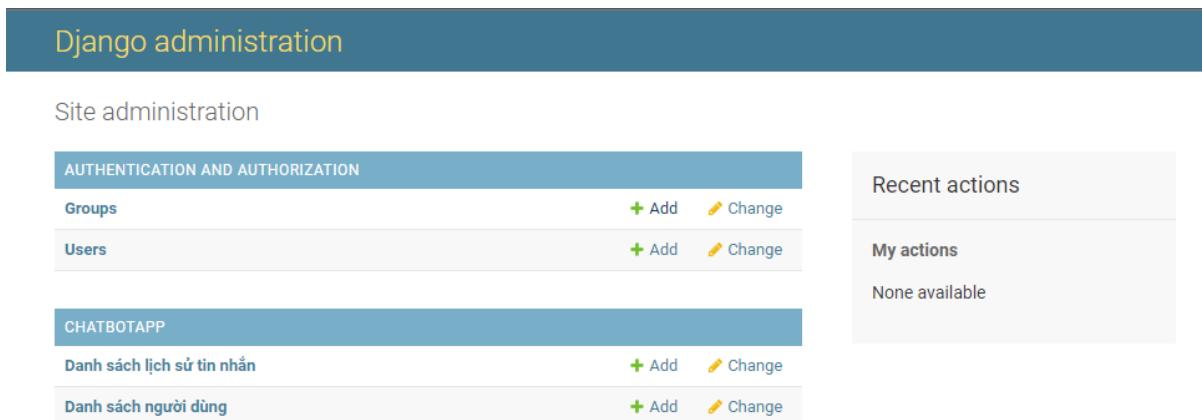
F:\Nhập môn tính toán đa phương tiện\Classroom\Chat\ChatBotNLP>
```

Hình 4.12: Tạo tài khoản Admin cho trang quản trị.

Sau khi nhập câu lệnh, hệ thống sẽ hiển thị ra mục để cho người dùng nhập thông tin vào như username, email, password. Do được thiết kế với độ bảo mật cao nên khi ta tạo tài khoản Admin với những tên thường được sử dụng hay mật khẩu quá đơn giản thì chương trình hệ thống sẽ hiển thị các cảnh báo đối với người dùng. Sau khi đã tạo xong tài khoản, ta tiến hành đăng nhập vào giao diện của trang quản trị Admin bằng địa chỉ “<http://localhost:8000/admin>”.



Hình 4.13: Giao diện đăng nhập vào trang Admin.



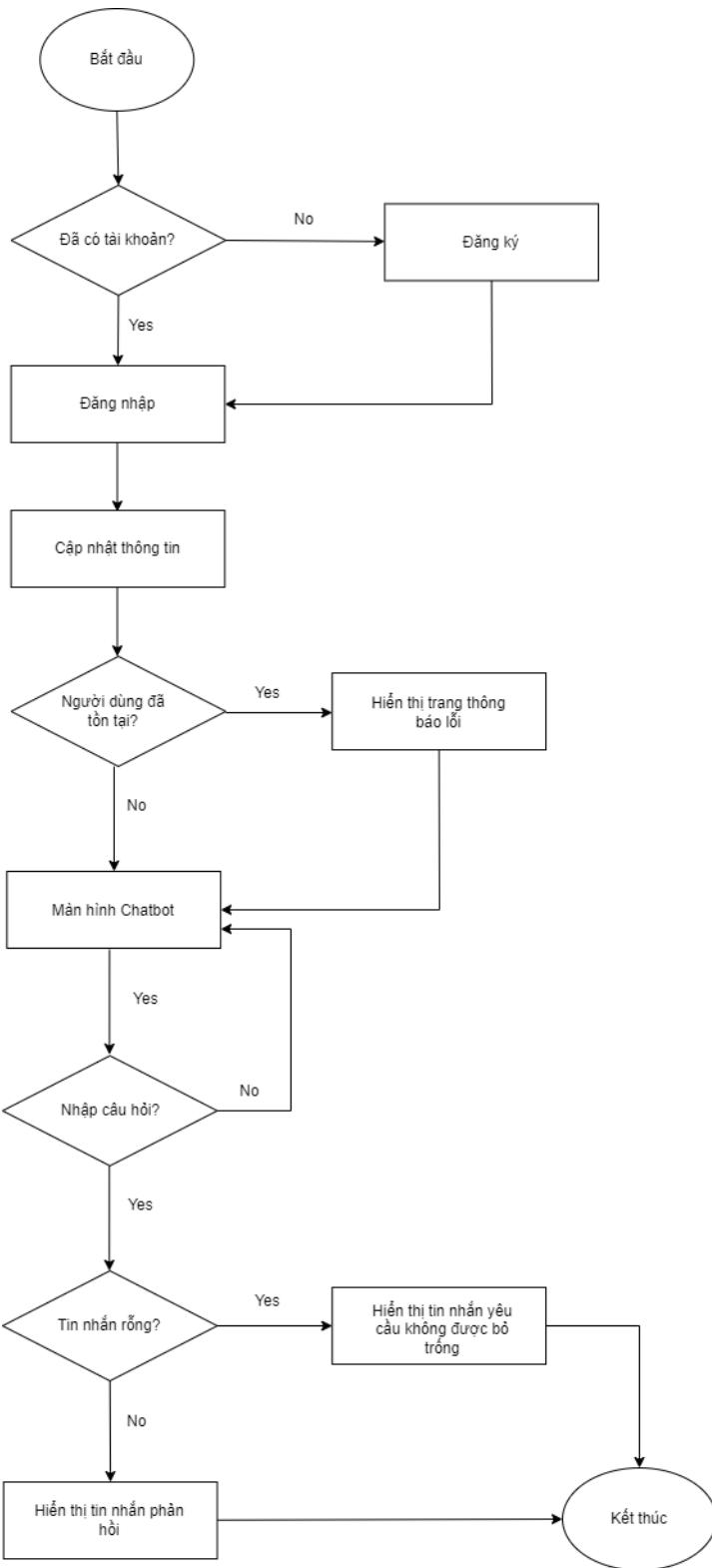
Hình 4.14: Giao diện quản lý của trang Admin.

4.1.5 Tổng kết

Như vậy, chúng ta đã tìm hiểu sơ qua về Django Framework từ các khái niệm cơ bản cho đến các đặc điểm của nó. Bên cạnh đó, chúng ta cũng đã tìm hiểu về mô hình MVT và so sánh nó với mô hình truyền thống MVC để thấy được những sự khác biệt cơ bản. Cuối cùng là việc triển khai cài đặt các Django cơ bản theo các bước sẽ giúp cho chúng ta dễ dàng hình dung hơn và hiểu rõ hơn về việc triển khai Django. Đây sẽ là tiền đề cho việc triển khai hệ thống trả lời tự động bằng việc ứng dụng các mô hình trong lĩnh vực xử lý ngôn ngữ tự nhiên ngay dưới đây.

4.2 Sơ đồ chương trình

Đầu tiên, để triển khai chương trình thì ta phải xem xét qua sơ đồ hoạt động tổng quát của chương trình để hiểu rõ hơn về các bước hoạt động của nó.



Hình 4.15: Sơ đồ hoạt động của hệ thống Chatbot.

Ta tiến hành xem xét hình 4.1, khi bắt đầu khởi chạy chương trình nếu người dùng đã có tài khoản từ trước đó thì một màn hình đăng nhập được hiển thị để người dùng có thể đăng nhập vào và sử dụng tính năng của chương trình. Nếu người dùng chưa đăng ký tài khoản trong cơ sở dữ liệu thì phải tiến hành đăng ký. Việc đăng ký bằng cách

người dùng nhấn vào liên kết đăng ký trên màn hình đăng nhập. Bước tiếp theo, sau khi đăng nhập hoàn tất thì chương trình sẽ hiển thị một bản mẫu để người dùng cập nhập các thông tin cá nhân của mình và mục đích của việc cập nhập là tương tác với cơ sở dữ liệu. Nếu người dùng nhập username đã tồn tại nghĩa là tài khoản đã được đăng ký trước đó thì chương trình sẽ hiển thị ra một trang web thông báo lỗi cho người dùng rằng tên tài khoản đã tồn tại và chuyển hướng người dùng đến trang giao diện chính của chương trình Chatbot. Nếu người dùng nhập username chưa tồn tại vào biểu mẫu cập nhật thông tin thì thông tin mà người dùng nhập vào sẽ được lưu vào cơ sở dữ liệu và tự động được chuyển sang màn hình chính của chương trình. Sau khi ở tại màn hình chính, chúng ta sẽ tiến hành nhập tin nhắn vào khung chat để giao tiếp chương trình. Ở bước này, nếu chúng ta không nhập dữ liệu thì chương trình sẽ sinh ra một tin nhắn yêu cầu chúng ta không được bỏ trống và nếu chúng ta có nhập chữ vào thanh hội thoại thì chương trình sẽ gọi đến mô hình mà chúng ta đã thực hiện training trước đó và sinh ra câu trả lời cho chúng ta. Từ đó, hình thành sự tương tác giữa con người và chương trình Chatbot.

4.3 Mã chương trình

4.3.1 Xây dựng mô hình Chatbot

Thu viện cần thiết

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from underthesea import word_tokenize
from keras.layers import Embedding, Dense, LSTM
from keras.models import Model
from keras.layers import Input
import tensorflow as tf
import warnings
import string
import pickle
from keras.models import load_model
warnings.filterwarnings('ignore')
```

Hình 4.16: Các thư viện cần thiết trong việc xây dựng mô hình Seq2Seq.

Việc triển khai chương trình sẽ bao gồm các thư viện và các module sau:

- **Thư viện underthesea:** để thuận tiện hơn trong việc triển khai và hỗ trợ việc tách các từ trong tập dữ liệu hoặc tin nhắn đầu vào thành các từ với ngôn ngữ Tiếng Việt thì ta sử dụng thư viện underthesea. Với thư viện underthesea, nó cho phép chúng ta tách các từ với độ chính xác cao nên hỗ trợ khá nhiều cho việc cải thiện độ chính xác của chương trình.

- **Thư viện tensorflow:** là một thư viện mã nguồn mở, được sử dụng để cung cấp khả năng tính toán số học dựa trên các biểu đồ mô tả sự thay đổi của dữ liệu. Nó được sử dụng khá phổ biến do được tích hợp sẵn các thư viện trong lĩnh vực học máy và có khả năng tương thích, mở rộng tốt.
- **Thư viện numpy:** là thư viện toán học được sử dụng để tương tác hiệu quả với ma trận và mảng, đặc biệt là các dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần.
- **Thư viện pickle:** là một phần của thư viện Python theo mặc định và cung cấp khả năng lưu trữ các đối tượng Python dưới dạng tệp và có thể tái sử dụng nó cho những lần sau. Ở chương trình Chatbot này, chúng ta sử dụng thư viện pickle để lưu trữ các giá trị từ điển để có thể tái sử dụng.
- **Thư viện warning:** là thư viện xử lý các thông báo về cảnh báo. Ở chương trình, thư viện được sử dụng để loại bỏ một số cảnh báo không cần thiết.
- **Thư viện keras:** là một thư viện open-source dành cho việc nghiên cứu neural network được viết bằng ngôn ngữ Python. Keras là một API bậc cao có thể sử dụng cùng với các thư viện về Deep Learning nổi tiếng như Tensorflow, CNTK hay theano. Trong chương trình này, keras sẽ thực hiện việc tạo ra model, sử dụng các lớp trong mạng neural và thực hiện load lại model.
- **Module preprocessing:** là một module trong thư viện tensorflow, ở chương trình này chúng ta sử dụng hàm pad_sequence để thực hiện đồng nhất độ dài của các đầu vào cho mô hình.
- **Module utils:** là một module trong thư viện tensorflow, nổi bật nhất là hàm to_categorical được sử dụng để chuyển đổi một vector thành một ma trận nhị phân.

Thu thập dữ liệu

Bước kế tiếp của quá trình xây dựng một hệ thống Chatbot và quyết định xem nội dung của nó có thật sự phong phú hay không chính là dựa vào bước thu thập dữ liệu cho mô hình. Ở đây, chúng em sử dụng một tập các danh sách dữ liệu được tham khảo trên mạng Internet gồm những tập tin text có dạng txt như sau:

Name	Date modified	Type	Size
bạn bè.txt	6/17/2021 10:33 AM	Text Document	73 KB
các câu hỏi phức tạp.txt	6/17/2021 10:33 AM	Text Document	17 KB
đất nước.txt	6/17/2021 10:33 AM	Text Document	22 KB
địa chỉ.txt	6/17/2021 10:33 AM	Text Document	47 KB
du lịch.txt	6/17/2021 10:33 AM	Text Document	24 KB
gia đình.txt	6/17/2021 10:33 AM	Text Document	45 KB
giải trí.txt	6/17/2021 10:33 AM	Text Document	29 KB
học tập.txt	6/17/2021 10:33 AM	Text Document	24 KB
nghề nghiệp.txt	6/17/2021 10:33 AM	Text Document	41 KB
nghi lễ.txt	6/17/2021 10:33 AM	Text Document	20 KB
người yêu.txt	6/17/2021 10:33 AM	Text Document	40 KB
robot.txt	6/17/2021 10:33 AM	Text Document	24 KB
shoping.txt	6/17/2021 10:33 AM	Text Document	58 KB
sở thích.txt	6/17/2021 10:33 AM	Text Document	22 KB
tán gẫu.txt	6/17/2021 10:33 AM	Text Document	17 KB
tdtu.txt	6/17/2021 10:33 AM	Text Document	44 KB
thông tin cá nhân.txt	6/17/2021 10:33 AM	Text Document	66 KB
trò chuyện về đi ăn.txt	6/17/2021 10:33 AM	Text Document	23 KB

Hình 4.17: Dữ liệu Chatbot. [19]

Sau khi tìm kiếm và thu thập được dữ liệu, ta tiến hành tạo một mảng để lưu các đường dẫn đến các file dữ liệu.

```
all_dataset = ['./dataset/bạn bè.txt',
  './dataset/các câu hỏi phức tạp.txt',
  './dataset/đất nước.txt',
  './dataset/tdtu.txt',
  './dataset/thông tin cá nhân.txt',
  './dataset/trò chuyện về đi ăn.txt',
  './dataset/địa chỉ.txt',
  './dataset/du lịch.txt',
  './dataset/gia đình.txt',
  './dataset/giải trí.txt',
  './dataset/học tập.txt',
  './dataset/nghề nghiệp.txt',
  './dataset/nghi lễ.txt',
  './dataset/người yêu.txt',
  './dataset/robot.txt',
  './dataset/shoping.txt',
  './dataset/tán gẫu.txt',]
```

Hình 4.18: Mảng lưu giá trị đường dẫn đến các tệp dữ liệu Chatbot.

Tiếp đến, ta tiến hành tạo ra hai mảng để lưu các câu hỏi và câu trả lời của tập dữ liệu. Sau đó, ta tiến hành thực hiện một vòng lặp để duyệt qua tất cả các đường dẫn trong mảng dataset.

```

# Tạo mảng để lưu danh sách câu hỏi và câu trả lời từ tập dataset
questions_chatbot = []
answers_chatbot = []

# Duyệt qua tất cả các tệp trong mảng all_dataset
total_dataset = len(all_dataset)
for i in range(total_dataset):
    with open(all_dataset[i], encoding='UTF-8') as txt:
        train_lines = txt.readlines() # Đọc từng dòng trong file text
    for line in train_lines:
        tmp = line.split("__eou__")
        question = tmp[0].strip()
        answer = tmp[1].strip()
        questions_chatbot.append(question)
        answers_chatbot.append(answer)

```

Hình 4.19: Thực hiện xử lý lấy ra tập câu hỏi và câu trả lời.

Úng với mỗi tệp dữ liệu Chatbot được duyệt qua, chương trình sẽ thực hiện mở nó và đọc từng dòng dữ liệu trong tệp bằng hàm `readlines()`. Sau đó, chương trình tiếp tục dòng vòng lặp `for` và duyệt qua từng dòng đã được đọc ở trên. Trong vòng lặp, nó sẽ cắt mỗi dòng bao gồm câu hỏi và câu trả lời thành câu hỏi và câu trả lời bởi cụm từ “`__eou__`” và tiếp hành hóa xóa đi các khoảng trắng trước và sau của mỗi câu hỏi và trả lời bằng hàm `strip()`. Sau khi xử lý xong, chương trình sẽ thực hiện thêm dữ liệu câu hỏi vào mảng `question_chatbot` và dữ liệu câu trả lời sẽ được thêm vào mảng `answers_chatbot`.

Tiền xử lý dữ liệu (Preprocessing)

Đây là một trong những bước quan trọng và quyết định đến độ chính xác của mô hình Chatbot nên ta cần phải xử lý một cách cẩn thận và đảm bảo không có lỗi trong quá trình xử lý. Để tiện lợi trong việc xử lý các câu hỏi và câu trả lời và có thể sử dụng lại trong những lần tiếp theo thì chúng ta sẽ tạo ra một hàm “`text_process()`” để chuyển tất cả các dữ liệu đầu vào về dạng chữ thường, xóa tất cả các ký tự dấu câu đã được định nghĩa, thay thế các câu có hai khoảng trắng liên tiếp trở lên, xóa các khoảng trắng ở đầu và cuối của mỗi câu và cuối cùng thì thực hiện tách từ trong câu đã xử lý bằng hàm `word_tokenize` của thư viện `underthesea`.

```

def text_process(sent):
    punctuation = {'.', ',', '...', '—', '“‘’, ‘’‘’, ‘;’, ‘(‘ ‘)’, ‘“‘’, ‘!’, ‘&’, ‘;’, ‘?’, ‘*’, ‘]’, ‘>’, ‘_’, ‘‘‘’, ‘“‘’, ‘”‘’’}
    # Chuyển về chữ thường để đồng nhất tất cả
    sent = sent.lower()

    # Lược bỏ các dấu câu
    sent = [char for char in sent if char not in punctuation]
    sent = ''.join(sent)

    # Thay thế các khoảng trắng
    sent = sent.replace("  ", " ")
    sent = sent.replace(" ", " ")
    sent = sent.strip()

    # Tách từ trong từng câu
    sent = word_tokenize(sent)

    return sent

```

Hình 4.20: Hàm xử lý câu và thực hiện tách từ trong mỗi câu.

Sau khi tạo xong hàm tách từ text_process() thì để áp dụng hàm này cho các câu hỏi và trả lời của tập dữ liệu Chatbot thì chúng ta phải xử lý các câu trả lời bị rỗng để tránh cho việc các câu hỏi có mà câu trả lời lại không từ đó làm cho mô hình kém chính xác.

```

# Xóa question và answer khi answer empty string
list_index_of_empty_answer = []
total_answer = len(answers_chatbot)
for i in range(total_answer):
    if(answers_chatbot[i] == ""):
        list_index_of_empty_answer.append(i)

def delete_empty_answer(list_index):
    total_index = len(list_index)
    for i in range(total_index):
        del questions_chatbot[list_index[i] - i]
        del answers_chatbot[list_index[i] - i]
delete_empty_answer(list_index_of_empty_answer)

```

Hình 4.21: Xử lý xóa các câu trả lời rỗng.

Để thực hiện xóa các câu rỗng, ta tiến hành tạo ra một mảng để lưu các giá trị vị trí của các câu trả lời rỗng trong mảng câu trả lời đã được tách ra ở trên. Ta tiến hành thực hiện một vòng lặp for từ 0 đến tổng số các phần tử trong mảng câu hỏi. Sau đó, tiến hành kiểm tra từng phần tử trong mảng câu trả lời bằng cách tham chiếu đến giá trị của mảng tại vị trí thứ i tương ứng với mỗi lần lặp của vòng lặp. Nếu giá trị tại vị trí thứ i là rỗng hoặc không có giá trị thì tiếp hành thêm vị trí thứ i đó vào mảng lưu các giá trị vị trí “list_index_of_empty_answer”. Tiếp tục, chương trình sẽ tạo ra hàm delete_empty_answer với đầu vào là tập danh sách các vị trí rỗng. Ở hàm này, nó cũng duyệt qua tất cả các phần tử lưu vị trí rỗng và tiến hành xóa bằng hàm del.

Sau khi thực hiện việc xóa đi các câu trả lời rỗng và câu hỏi tương ứng với câu trả lời đó thì chúng ta sẽ tiến hành tách các từ trong câu ra thành những từ có ý nghĩa dựa vào hàm text_process() được tạo ra ở trên. Sau khi tách từ, mỗi câu sẽ được tách và lưu

trong một mảng và các mảng này được lưu trữ trong mảng lớn hơn đó là mảng câu hỏi question_chatbot và mảng câu trả lời answer_chatbot.

```
# Xử lý tách từ
total_question = len(questions_chatbot)
for i in range(total_question):
    questions_chatbot[i] = text_process(questions_chatbot[i])
    answers_chatbot[i] = text_process(answers_chatbot[i])
```

Hình 4.22: Thực hiện tách các từ trong câu thành các từ.

Sau đó, ta tiến hành tạo một biến word2count dạng Dictionary và thực hiện đếm số lần xuất hiện của các từ trong mảng câu hỏi và câu trả lời đã được xử lý tách từ. Bằng việc sử dụng vòng lặp duyệt qua từng phần tử trong mảng câu hỏi và câu trả lời, nếu từ chưa xuất hiện trong biến word2count thì gán bằng 1 và nếu có tồn tại thì mỗi lần duyệt qua sẽ tăng thêm 1 vào giá trị value tương ứng với từ đó. Chúng ta cũng thực hiện tương tự đối với mảng câu trả lời.

```
word2count = {}

for line in data:
    for word in line:
        if word not in word2count:
            word2count[word] = 1
        else:
            word2count[word] += 1

for line in data_answer:
    for word in line:
        if word not in word2count:
            word2count[word] = 1
        else:
            word2count[word] += 1

print(word2count)
```

Hình 4.23: Thực hiện đếm tần suất xuất hiện của các từ.

Bước tiếp theo, chúng ta tiến hành gắn của thẻ tag <SOS> (tương ứng với cụm từ viết tắt của Start of Sentences) và thẻ <EOS> (tương ứng với cụm từ viết tắt của End of Sentences) vào mảng các lưu các câu trả lời.

```
temp1 = []
for i in data_answer:
    i.insert(0, "<SOS>")
    i.insert(len(i), "<EOS>")
    temp1.append(i)
```

Hình 4.24: Thực hiện thêm các tag cho các câu trả lời.

Sau khi thực hiện đếm tần suất xuất hiện của các từ, ta tiếp tục tạo ra một biến Dictionary khác là word2index. Biến word2index được tạo để biểu diễn một số đại diện cho một từ. Mục đích của việc tạo biến word2index là để chuyển các câu đầu vào dưới dạng một chuỗi thành một vector ứng với mỗi từ trong câu là giá trị của nó trong biến word2index.

```
# Chuyển từ thành số
word2index = {}
word_number = 0

# Duyệt qua từng từ và gán thứ tự cho mỗi từ
for word, count in word2count.items():
    word2index[word] = word_number
    word_number += 1
```

Hình 4.25: Thực hiện chuyển các từ thành các số.

Sau khi ta thêm thủ công các tag <EOS>, <SOS> thì chúng ta sẽ thêm nó vào một mảng tokens bao gồm ba tag <EOS>, <SOS>, <OUT> và tiến hành thêm vào biến từ điển word2index bằng cách dùng vòng lặp for duyệt qua từng tag và lần lượt thêm vào vị trí cuối của mảng. Sau khi thực hiện thêm 3 tag mới vào từ điển, ta tiến hành chuyển đổi các giá trị tương ứng với từng từ vào biến index2word.

```
# Thêm 3 tag <EOS>, <SOS>, <OUT> vào từ điển
tokens = ['<EOS>', '<SOS>', '<OUT>']

VOCAB_SIZE = len(word2index)

for token in tokens:
    word2index[token] = VOCAB_SIZE
    VOCAB_SIZE += 1

# Chuyển đổi giá trị thành từ
index2word = {w:v for v, w in word2index.items()}
```

Hình 4.26: Thêm các tag vào biến từ điển.

Tiếp đến, ta sẽ tiến hành chuyển đổi các câu trong tập câu hỏi question và tập câu trả lời answer thành vector và đưa vào lần lượt vào mảng encoder_input_data và decoder_input_data. Việc chuyển đổi thành vector được thực hiện bằng cách sử dụng vòng lặp for duyệt qua từng câu và ứng với từng câu thì ta tiếp tục dùng vòng lặp for để duyệt qua từng từ trong câu đó. Nếu từ hiện tại đang xét không có trong biến từ điển word2index thì thêm vào mảng tmp tag <OUT> ứng với từ đó. Ngược lại, nếu từ đó có trong trong từ điển word2index thì tiến hành thêm vào mảng tmp giá trị value ứng với

key là giá trị của từ đó trong biến từ điển. Cuối cùng thêm câu vừa thực hiện chuyển đổi thành dạng vector vào biến encoder_input_data. Tương tự thực hiện cho việc chuyển tập dữ liệu câu trả lời.

```
# Chuyển các câu trong tập question thành vector
encoder_input_data = []
for line in data:
    tmp = []
    for word in line:
        if word not in word2index:
            tmp.append(word2index['<OUT>'])
        else:
            tmp.append(word2index[word])

    encoder_input_data.append(tmp)

# Chuyển các câu trong tập answer thành vector
decoder_input_data = []
for line in temp1:
    tmp = []
    for word in line:
        if word not in word2index:
            tmp.append(word2index['<OUT>'])
        else:
            tmp.append(word2index[word])
    decoder_input_data.append(tmp)
```

Hình 4.27: Thực hiện chuyển đổi các câu hỏi và trả lời thành dạng vector.

Sau khi chuyển đổi các câu trong tập câu hỏi và trả lời thành các vector, ta tiến hành thực hiện việc đồng bộ hóa độ dài của các vector trong hai tập câu hỏi và trả lời. Đầu tiên, chúng ta xác định giá trị tối đa cho một câu là MAX_LEN = 20. Sau đó, ta tiến hành thực hiện hàm pad_sequences() cho giá trị encoder_input_data với giá trị đầu vào là các câu đã được vector hóa trong mảng encoder_input_data. Cùng với đó là giá trị MAX_LEN để xác định độ dài tối đa, padding = ‘post’ và trucating = ‘post’ lần lượt là thêm vào phía sau câu khi độ dài hiện tại của câu nhỏ hơn giá trị MAX_LEN và xóa các thành phần phía sau khi tin nhắn đầu vào có số từ lớn hơn giá trị MAX_LEN. Tiếp tục quá trình xử lý padding cho decoder_target_data, ta tiến hành bỏ đi tag <SOS> ở đầu mỗi câu và thêm vào mảng decoder_target_data và thực hiện padding giá trị decoder_target_data giống với hai biến ở trên. Cuối cùng, ta sử dụng hàm to_categorical để chuyển giá trị của vector decoder_target_data thành ma trận nhị phân để trích xuất các đặc trưng cho mô hình LSTM bao gồm sample, timestep và feature.

```

# Xác định chiều dài tối đa của câu
MAX_LEN = 20

# Padding độ dài của câu hỏi và trả lời đã được vector hóa sao cho độ dài bằng nhau
encoder_input_data = pad_sequences(encoder_input_data, MAX_LEN, padding='post', truncating='post')
decoder_input_data = pad_sequences(decoder_input_data, MAX_LEN, padding='post', truncating='post')

# Kết quả đầu ra
decoder_target_data = []
for input1 in decoder_input_data:
    decoder_target_data.append(input1[1:]) # Loại bỏ Tag đầu tiên là <SOS>

# Padding vector đầu ra sao cho bằng MAX_LEN
decoder_target_data = pad_sequences(decoder_target_data, MAX_LEN, padding='post', truncating='post')

# Trích xuất ra các đặc trưng (Sample, Timestep, Feature) cho LSTM Model
decoder_target_data = to_categorical(decoder_target_data, len(word2index))

```

Hình 4.28: Thực hiện padding giá trị của các vector.

Xây dựng mô hình

Để xây dựng mô hình, đầu tiên chúng ta phải khai báo trước các tham số cho mô hình của chúng ta. Các tham số bao gồm:

- VOCAB_SIZE: tương ứng với tổng số từ trong từ điển.
- HIDDEN_DIM: là số chiều ẩn trong LSTM.
- INPUT_DIM: là số chiều đầu vào cho lớp Embedding.
- Embedding_dimention: là số chiều đầu ra cho lớp Embedding.

```

# Khai báo các giá trị đầu vào cho các Layer
VOCAB_SIZE = len(word2index)
HIDDEN_DIM = 300
INPUT_DIM = VOCAB_SIZE + 1
embedding_dimention = 100

```

Hình 4.29: Khai báo các tham số cần thiết cho mô hình.

Đầu tiên, để xây dựng mô hình ta khởi tạo một lớp Embedding để giảm chiều dữ liệu đầu vào là các vector số nguyên với các thông số như số chiều đầu vào input_dim, số chiều đầu ra output_dim, chiều dài đầu vào input_length và trainable. Sau đó, chúng ta tiến hành định nghĩa một lớp Encoder đầu vào để xử lý chuỗi đầu vào và trả về trạng thái state bên trong nó. Ở bước xây dựng lớp Encoder, ta sẽ tạo ra một lớp Input đầu vào với độ dài tối đa là MAX_LEN và thực hiện đưa đầu vào của lớp Input đi qua lớp Embedding đã được tạo ra trước đó. Ta tiếp tục tạo ra một lớp LSTM với các tham số đầu vào lần lượt là số chiều ẩn bên trong LSTM, return_sequences và return_state. Đầu ra của lớp Embedding phía sau là đầu vào cho lớp LSTM vừa được khởi tạo bằng cách truyền vào LSTM giá trị của lớp Embedding. Sau khi áp dụng công thức chúng ta sẽ thu

được ba giá trị bao gồm encoder_outputs, state_h và state_c. Và ta chỉ giữ lại giá trị của state_h và state_c bằng cách cho nó vào một mảng, hai giá trị đó tương ứng với Hidden State và Cell State trong mô hình LSTM đã tìm hiểu ở trên. Hai giá trị này sẽ là đầu vào cho lớp Decoder trong bước tiếp theo.

```
# Định nghĩa Lớp Embedding
embed = Embedding(input_dim = INPUT_DIM, output_dim=embedding_dimention, input_length=MAX_LEN, trainable=True)

# Định nghĩa Lớp Encoder của LSTM
# Dùng để xử lý chuỗi đầu vào và trả về state của nó
encoder_inputs = Input(shape=(MAX_LEN, ))
encoder_embed = embed(encoder_inputs)
encoder_lstm = LSTM(HIDDEN_DIM, return_sequences=True, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(encoder_embed)
# Giữ lại giá trị State_h và State_c đóng vai trò là đầu vào cho bước tiếp theo
encoder_states = [state_h, state_c]
```

Hình 4.30: Khởi tạo lớp Embedding và Encoder LSTM.

Bước kế tiếp, ta tiến hành xây dựng một lớp Decoder, cũng giống với Encoder thì chúng ta cũng tạo ra một lớp Input và cho nó đi qua lớp Embedding. Việc tiếp theo là tạo ra một lớp LSTM với tên là decoder_lstm với các tham số lần lượt là HIDDEN_DIM, return_sequences, return_state. Các giá trị đầu ra của lớp Embedding sẽ được sử dụng để làm đầu vào cho lớp Decoder và giá trị intial_state là trạng thái của bộ Encoder được sử dụng như trạng thái ban đầu. Ở bộ Decoder, nó cũng trả về các chuỗi đầu ra đầy đủ và các state bên trong nó. Nhưng ta sẽ thực hiện loại bỏ các trạng thái trả về mà chỉ giữ lại đầu ra đầy đủ cho lớp Decoder.

```
# Định nghĩa Lớp Decoder
# Được train để predict các từ tiếp theo cho các từ trước đó của decoder_target_data
decoder_inputs = Input(shape=(MAX_LEN, ))
decoder_embed = embed(decoder_inputs)
decoder_lstm = LSTM(HIDDEN_DIM, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_embed, initial_state=encoder_states)
```

Hình 4.31: Khởi tạo lớp Decoder LSTM.

Bước cuối cùng của việc khởi tạo mô hình, ta sẽ tiến hành tạo ra lớp Dense với số chiều đầu ra tương ứng với số từ trong từ điển word2index với hàm kích hoạt (Activation Function) là softmax. Sau khi khởi tạo lớp Dense, chương trình sẽ cho kết quả đầu ra của bộ Decoder qua lớp Dense này để có đầu ra thích hợp. Khi hoàn thành xong các bước, ta tiến hành định nghĩa Model với đầu vào là các câu đã được vector hóa bao gồm encoder_input_data, decoder_input_data thành decoder_target_data. Cuối cùng, chương trình tiến hành biên dịch mô hình với hàm loss là “categorical_crossentropy”, hàm tối ưu là “adam” và giá trị metric là “acc”.

```

# Định nghĩa Lớp Dense đầu ra ứng với tổng số từ trong từ điển
dense = Dense(VOCAB_SIZE, activation='softmax')

# Lấy ra giá trị từ Lớp Decoder LSTM
output = dense(decoder_outputs)

# Định nghĩa Model chung
model = Model([encoder_inputs, decoder_inputs], output)

# Compile Model
model.compile(loss='categorical_crossentropy', metrics=['acc'], optimizer='adam')

# Load Model đã Train để tiết kiệm thời gian
model.summary()

```

Hình 4.32: Khởi tạo lớp Dense đầu ra và xây dựng mô hình.

Sau khi biên dịch xong mô hình, chúng ta sử dụng hàm “model.summary” để xem tóm tắt tổng quan về mô hình nói chung và các lớp đã khởi tạo nói riêng.

```

Model: "model"
-----  

Layer (type)          Output Shape         Param #   Connected to  

-----  

input_2 (InputLayer)    [(None, 20)]        0          []  

input_1 (InputLayer)    [(None, 20)]        0          []  

embedding (Embedding)  (None, 20, 100)      472100    ['input_1[0][0]',  

                                         'input_2[0][0]']  

lstm (LSTM)            [(None, 20, 300),  

                         (None, 300),  

                         (None, 300)]    481200    ['embedding[0][0]']  

lstm_1 (LSTM)          [(None, 20, 300),  

                         (None, 300),  

                         (None, 300)]    481200    ['embedding[1][0]',  

                                         'lstm[0][1]',  

                                         'lstm[0][2]']  

dense (Dense)          (None, 20, 4720)     1420720   ['lstm_1[0][0]']  

-----  

Total params: 2,855,220  

Trainable params: 2,855,220  

Non-trainable params: 0

```

Hình 4.33: Tóm tắt mô hình sau khi khởi tạo.

Sau cùng, ta tiến hành huấn luyện (training) cho model với giá trị EPOCHS là 120 và BATCH_SIZE là 32.

```

In [33]: BATCH_SIZE = 32
EPOCHS = 120

model.fit([encoder_input_data, decoder_input_data],decoder_target_data,epochs=EPOCHS,batch_size=BATCH_SIZE)
Epoch 120/120
174/174 [=====] - 57s 325ms/step - loss: 0.0446 - acc: 0.9902
Epoch 121/120
174/174 [=====] - 57s 328ms/step - loss: 0.0336 - acc: 0.9917
Epoch 122/120
174/174 [=====] - 56s 322ms/step - loss: 0.0319 - acc: 0.9917
Epoch 123/120
174/174 [=====] - 56s 321ms/step - loss: 0.0293 - acc: 0.9920
Epoch 124/120
174/174 [=====] - 56s 323ms/step - loss: 0.0286 - acc: 0.9920
Epoch 125/120
174/174 [=====] - 56s 322ms/step - loss: 0.0278 - acc: 0.9920
Epoch 126/120
174/174 [=====] - 56s 321ms/step - loss: 0.0273 - acc: 0.9920
Epoch 127/120
174/174 [=====] - 56s 324ms/step - loss: 0.0274 - acc: 0.9921
Epoch 128/120
174/174 [=====] - 59s 340ms/step - loss: 0.0276 - acc: 0.9918

```

Hình 4.34: Training dữ liệu cho model.

Quá trình huấn luyện nhanh hoặc chậm tùy thuộc vào mô hình mà chúng ta xây dựng. Ở mô hình này, chúng ta mất khoảng tầm 1 phút để hoàn thành cho một phần train. Sau khi hoàn tất quá trình training, ta thấy rằng giá trị độ chính xác accuracy là 0.9918 và giá trị loss 0.0276. Đây là các giá trị khá cao dành cho mô hình Seq2Seq.

⊕ Lưu mô hình

Sau khi đã training dữ liệu cho mô hình thì chúng ta sẽ thực hiện lưu mô hình đã được training. Mục đích của việc lưu mô hình đã huấn luyện đó là tiết kiệm được thời gian training bởi vì nó vốn chiếm khá nhiều thời gian.

```
model.save('LSTM_ChatBot_Final.h5')
```

Hình 4.35: Lưu mô hình đã training.

Bên cạnh đó, ta cũng thực hiện lưu các giá trị của từ điển bao gồm hai biến word2index và index2word.

```
# Lưu từ điển
with open('word2index.pkl', 'wb') as f:
    pickle.dump(word2index, f)

with open('index2word.pkl', 'wb') as f:
    pickle.dump(index2word, f)
```

Hình 4.36: Thực hiện lưu giá trị của từ điển word2index và index2word để tái sử dụng.

4.3.2 Triển khai Chatbot với Django

Ở phần triển khai giao diện cho hệ thống Chatbot, chúng ta sẽ sử dụng Django Framework trên nền ngôn ngữ Python. Ở phần này, chúng ta sẽ đi phân tích và giải thích các đoạn mã trong từng tệp module trong framework này.

⊕ views.py

Việc đầu tiên cần thực hiện đó là thêm các thư viện cần thiết cho việc xử lý trong file views. Đây là file thực hiện các việc xử lý logic và tương tác với Template với người dùng thông qua các URL.

```
from django.shortcuts import render
from django.views.generic import ListView
from . import models
from django.contrib.auth.decorators import login_required

from django.contrib.auth import authenticate, login, logout
from django.shortcuts import redirect
from django.contrib.auth.forms import UserCreationForm
from .form import CreateUserForm
```

Hình 4.37: Các thư viện cần thiết.

Sau đây, chúng ta tiến hành tìm hiểu một số hàm trong file views.py của web app trong dự án Django lần này. Ở phần xử lý trong views.py, để đơn giản hóa trong việc triển khai thì chúng em đã sử dụng gần như tất cả là Function View.

```
def chatbotvietnamese(request):
    if not request.user.is_authenticated:
        return redirect('/login/user/')
    else:
        return render(request, "chatbot/chatbotvietnamese.html")

@login_required(login_url="/login/user")
def update_form(request):
    return render(request, "chatbot/updateprofile.html")

def login_func(request):
    us = request.POST.get('name', '')
    ps = request.POST.get('pass', '')
    context = {'name':us, 'pass':ps}
    if us and ps:
        user = authenticate(username = us, password = ps)
        if user:
            login(request, user)
            url = request.GET.get('next', '/updateform')
            return redirect(url)
        context['msg']="Sai tên đăng nhập hoặc mật khẩu"
    return render(request, "chatbot/login1.html", context)
```

Hình 4.38: Hàm xử lý giao diện chính, đăng nhập và cập nhật thông tin.

Ta cùng xem xét hình 4.38 ở bên trên, ta thấy rằng có các hàm:

- **chatbotvietnamese()**: là hàm để hiển thị giao diện Chatbot, đầu tiên để vào được giao diện Chatbot nó sẽ thực hiện kiểm tra xem người dùng hiện tại đã đăng nhập hay chưa. Nếu chưa đăng nhập thì sẽ chuyển hướng đến url của form đăng nhập còn nếu đã đăng nhập thì nó sẽ tiến hành render ra giao diện Chatbot.
- **updateform()**: dùng để hiển thị ra form cập nhật thông tin của người dùng. Để vào được trang cập nhật thông tin này thì chúng ta phải tiến hành đăng nhập thông qua một decorator login_required.

- **login_func()**: là hàm thực hiện việc đăng nhập cho chương trình. Nó lấy ra các giá trị name và pass thông qua phương thức POST được gửi giá trị từ trình duyệt và tiến hành kiểm tra username và password mà người dùng nhập vào đã có trong cơ sở dữ liệu hay không. Nếu có tồn tại user thì chương trình sẽ tiến hành đăng nhập vào giao diện của chương trình và chuyển hướng đến trang updateform để người dùng cập nhật các thông tin của họ vào cơ sở dữ liệu. Ngược lại, nếu người dùng không có trong hệ thống thì sẽ gán giá trị và hiển thị ra màn hình đăng nhập thông điệp “Sai tên đăng nhập hoặc mật khẩu.”.

Chúng ta cùng tiếp tục xem qua hình 4.39 ở bên dưới, ta thấy rằng có các hàm:

```

def signup_func(request):
    form = CreateUserForm()
    if request.method == "POST":
        form = CreateUserForm(request.POST)
        if form.is_valid():
            form.save()
            url = request.GET.get('next', '/')
            return redirect(url)
    context = {'form':form}
    return render(request, "chatbot/signup.html", context)

def logout_func(request):
    logout(request)
    url = request.GET.get('next', '/')
    return redirect(url)

def error(request):
    return render(request, "chatbot/error.html")

```

Hình 4.39: Hàm đăng ký tài khoản, đăng xuất và hiển thị lỗi.

- **signup_func()**: là hàm đảm nhận việc đăng ký tài khoản cho người dùng. Nó thực hiện việc đăng ký nhờ vào sử dụng CreateUserForm được xây dựng sẵn để tương tác với CSDL User trong hệ thống Admin. Nếu method được yêu cầu đến đường dẫn là POST thì nó sẽ kiểm tra tính hợp lệ của các giá trị trong form mà người dùng gửi đến bằng hàm is_valid() và nếu hợp lệ thì nó sẽ thực hiện tạo và lưu thông tin của người dùng vào CSDL. Sau khi đăng ký xong, nó sẽ chuyển hướng đến trang chủ của chương trình Chatbot và tại bước này, chúng ta sẽ tiến hành đăng nhập tài khoản vừa được tạo là có thể đăng nhập thành công vào hệ thống.
- **logout_func()**: là hàm đảm nhận nhiệm vụ đăng xuất tài khoản ra khỏi hệ thống. Việc đăng xuất được thực hiện qua hàm logout() với tham số truyền vào là request

từ người dùng hiện tại. Sau khi nhấn vào liên kết “Đăng xuất” thì chương trình sẽ trở về giao diện đăng nhập.

- **error()**: là hàm dùng để hiển thị ra trang lỗi khi mà có một yêu cầu lỗi nào đó được thực hiện.

Hàm cuối cùng trong views.py chính là hàm update_info(). Đây là hàm được sử dụng để cập nhật thông tin của người dùng vào CSDL. Việc cập nhật được bắt đầu với việc kiểm tra xem phương thức khi người dùng nhấn vào button submit có phải là POST hay không. Nếu là POST thì chương trình sẽ tiến hành lấy ra các giá trị tương ứng với từng tên trường mà chúng ta đã điền vào ở phía giao diện người dùng. Sau đó, ta thực hiện việc kiểm tra xem tên tài khoản username hiện tại đã có người sử dụng hay chưa bằng cách sử dụng hàm filter để kiểm tra sự tồn tại của username trong CSDL mà người dùng gửi đến. Nếu có tồn tại thì tiến hành gọi đến trang hiển thị lỗi và ngược lại nếu chưa tồn tại thì chương trình sẽ tiến hành tạo ra các giá trị vào CSDL và lưu nó. Sau khi thêm mới dữ liệu, chương trình sẽ tự động chuyển hướng cho chúng ta về giao diện chính của chương trình.

```
def update_info(request):
    if request.method == "POST":
        username = request.POST['username']
        hoilot = request.POST['holot']
        ten = request.POST['ten']
        email = request.POST['email']
        gioitinh = request.POST['gioitinh']
        diachi = request.POST['diachi']

        if models.Register.objects.filter(username=username).exists():
            return render(request, "chatbot/error.html")
        info = models.Register.objects.create(username = username,
                                              hoilot = hoilot,
                                              ten = ten,
                                              email = email,
                                              gioitinh = gioitinh,
                                              diachi = diachi,
                                              )
        info.save()
        return redirect('/chatbot/')
    else:
        return render(request, "chatbot/error.html")
```

Hình 4.40: Hàm cập nhật thông tin người dùng.

ChatBotVietNamese.py

Đây là hàm xử lý chính của chương trình, ở hàm này nó thực hiện việc xử lý các câu hỏi, thông điệp đầu vào và sử dụng mô hình Seq2Seq đã được xây dựng từ trước để thực hiện lấy ra các câu trả lời và hiển thị trả lại cho bên phía người dùng.

Do nếu ta thực hiện lại việc tiền xử lý dữ liệu hay huấn luyện cho mô hình đặc biệt ở đây là mô hình Seq2Seq LSTM thì nó sẽ làm rất nhiều thời gian. Nên ở đây, chúng em đã sử dụng lại mô hình đã được huấn luyện từ trước để nhanh chóng hơn trong việc triển khai. Bằng cách sử dụng hàm load_model trong thư viện keras và gán nó vào biến model1. Sau đó, ta cũng tương tự thực hiện việc tái sử dụng lại hai tệp word2index và index2word để tiết kiệm thời gian xử lý và làm cho chương trình nhanh hơn.

```
# Load Model đã Train để tiết kiệm thời gian
# model1 = load_model('./LSTM_Model.h5')
model1 = load_model('./LSTM_ChatBot_Final.h5')

model1.summary()

# Load lại hai biến Dict là word2index và index2word để phục vụ cho việc chuyển đổi
with open('word2index.pkl', 'rb') as f:
    word2index = pickle.load(f)

with open('index2word.pkl', 'rb') as f:
    index2word = pickle.load(f)
```

Hình 4.41: Tái sử dụng lại mô hình và các biến từ điển.

Tiếp theo, ta tiến hành khai báo các tham số cần thiết cho mô hình bao gồm số chiều ẩn cho LSTM và MAX_LEN là độ dài tối đa cho các chuỗi đầu vào.

```
# Khai báo số chiều và độ dài tối đa cho mỗi message
HIDDEN_DIM = 300
MAX_LEN = 20
```

Hình 4.42: Khai báo các tham số cho mô hình.

Sau khi thực hiện load mô hình để tái sử dụng, chúng ta phải lấy ra các lớp trong mô hình đó để sử dụng. Để thuận tiện và dễ dàng hơn trong việc lấy ra các lớp dựa vào vị trí thì chúng ta sử dụng câu lệnh “model1.summray()” để hệ thống sẽ hiển thị thông tin tổng quan về mô hình được load lại bao gồm cấu tạo của mô hình được xây dựng từ những lớp nào. Từ đó, ta có thể dễ dàng lấy ra được từng lớp một cách chính xác và nhanh chóng.

Model: "model"				
Layer (type)	Output Shape	Param #	Connected to	
input_2 (InputLayer)	[(None, 20)]	0	[]	
input_1 (InputLayer)	[(None, 20)]	0	[]	
embedding (Embedding)	(None, 20, 100)	472100	['input_1[0][0]', 'input_2[0][0]']	
lstm (LSTM)	[(None, 20, 300), (None, 300), (None, 300)]	481200	['embedding[0][0]']	
lstm_1 (LSTM)	[(None, 20, 300), (None, 300), (None, 300)]	481200	['embedding[1][0]', 'lstm[0][1]', 'lstm[0][2]']	
dense (Dense)	(None, 20, 4720)	1420720	['lstm_1[0][0]']	

Total params: 2,855,220
Trainable params: 2,855,220
Non-trainable params: 0

Hình 4.43: Tóm tắt mô hình Seq2Seq sau khi được tái sử dụng.

Ta tiếp tục tiến hành lấy ra từng lớp trong các mô hình, các lớp được lấy ra bao gồm hai lớp Input đầu vào cho bộ mã hóa (Encoder) và bộ giải mã (Decoder), lớp Embedding, lớp LSTM cho Encoder, lớp LSTM cho Decoder và lớp đầu ra Dense.

```
# Load lớp Input cho Encoder
input_encoder = model1.input[0]

# Load lớp Input cho bộ Decoder
input_decoder = model1.input[1]

# Load lớp Embedding
embed = model1.layers[2]

# Load lớp LSTM cho Encoder
output_encoder, state_h, state_c = model1.layers[3].output

# Load lớp LSTM cho Decoder
decoder_lstm = model1.layers[4]

# Load lớp Dense
dense = model1.layers[5]
```

Hình 4.44: Lấy ra các lớp từ mô hình được sử dụng lại.

Sau khi load lớp LSTM cho Encoder, ta chỉ giữ lại các trạng thái bao gồm state_h và state_c tương ứng với Hidden State và Cell State và lưu vào mảng encoder_states. Sau đó, tiến hành xây dựng mô hình Encoder là encoder_model với đầu vào là input_encoder ứng với vector của câu hỏi đầu vào và trả về các giá trị final state vector của bộ Encoder LSTM.

```

#Load Encoder Model
# Mảng lưu hai trạng thái state_h và state_c của Encoder
encoder_states = [state_h, state_c]

# Model cho đầu vào của Message và trả về các final state vector của bộ Encoder LSTM
encoder_model = Model([input_encoder], encoder_states)

```

Hình 4.45: Xây dựng mô hình Encoder cho chuỗi đầu vào.

Tiếp tục, ta thực hiện triển khai mô hình Inference (dành cho quá trình giải mã câu). Ta bắt đầu khởi tạo hai đầu vào tương ứng cho hai state đầu ra của bộ Encoder và đưa hai giá trị state vào mảng decoder_states_input. Sau đó, tiến hành gọi lớp decoder_lstm với đầu vào là Input của nó được đi qua lớp Embedding và giá trị initial_state là hai giá trị state đầu vào. Sau đó, ta chỉ giữ lại hai state đầu ra từ bộ decoder_lstm và gán cho biến decoder_states. Giá trị decoder_states và decoder_output chính là những từ tiếp theo được dự đoán. Sau cùng, chúng ta thực hiện tạo model cho lớp decoder với một lớp đầu vào cộng với hai giá trị trạng thái của lớp trước đó cùng với decoder_outputs và decoder_states chính là những từ tiếp theo được dự đoán.

```

# Decoder Model dùng để dự đoán suy luận từ

decoder_state_input_h = Input(shape=(HIDDEN_DIM,))
decoder_state_input_c = Input(shape=(HIDDEN_DIM,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]

# Lấy ra các state từ Encoder
decoder_outputs, state_h_dec, state_c_dec = decoder_lstm(embed(input_decoder), initial_state=decoder_states_inputs)
decoder_states = [state_h_dec, state_c_dec]

decoder_model = Model([input_decoder] + decoder_states_inputs, [decoder_outputs] + decoder_states)

```

Hình 4.46: Xây dựng Inference Model để dự đoán.

[21] Sau khi chúng ta đã xây dựng hoàn tất xong các model, chúng ta tiến hành đi xây dựng hàm Chat() để xử lý các câu hỏi đầu vào và tham chiếu đến mô hình. Ở hàm này, nó sẽ xử lý tin nhắn đầu vào bằng cách chuyển tất cả về chữ thường và kiểm tra các điều kiện. Nếu tin nhắn đến có giá trị là “quit” hoặc “stop” thì trả về tin nhắn chào người dùng. Và ngược lại, tin nhắn đầu vào sẽ được đưa vào hàm text_process để thực hiện xử lý thông điệp đầu vào dưới dạng mà model có thể hiểu được và thực hiện tách các từ. Sau đó, tin nhắn đã được xử lý sẽ được gán vào mảng question_input_list và tiến hành thực hiện chuyển đổi tin nhắn đã được xử lý tách từ trở về dạng vector. Để chuyển về dạng vector, chương trình sẽ thực hiện một vòng lặp for duyệt qua mảng câu hỏi đã xử lý qua hàm text_process và tạo ra một mảng temp rỗng. Sau đó, hàm sẽ thực hiện tiếp tục một vòng lặp khác để duyệt qua từng từ trong câu đó và thực hiện thêm vào mảng

temp rỗng được tạo ở trên các giá trị của mỗi từ trong biến word2index mà chúng ta đã thực hiện load ở bước đầu tiên. Nếu từ không có trong biến word2index thì tiến hành gán giá trị cho từ đó tương ứng với giá trị value của tag <OUT> trong word2index. Sau khi quá trình duyệt qua các từ hoàn tất, hàm sẽ thực hiện thêm các giá trị số trong mảng temp vào mảng question_vector. Sau khi chuyển đổi tin nhắn đầu vào về dạng vector, chúng ta tiến hành padding độ dài của vector sao cho phù hợp với đầu vào của model là MAX_LEN ứng với 20 bằng hàm pad_sequences.

```
def chat(message):
    question_input = message
    if question_input.lower() == "quit" or question_input.lower() == "stop": # Check điều kiện chào tạm biệt
        end_question = "Xin chào bạn hẹn bạn gặp lại vào lần sau"
        return end_question
    else:
        question_input = text_process(question_input)
        question_input_list = [question_input]
        question_vector = []

        # Chuyển đổi message đầu vào thành dạng vector
        for x in question_input_list: # Duyệt qua các câu trong mảng câu hỏi đầu vào
            temp = [] # Tạo mảng để lưu trữ các giá trị được chuyển đổi về dạng vector
            for y in x: # Duyệt qua từng từ
                try:
                    temp.append(word2index[y]) # Thêm vào mảng temp các giá trị value tương ứng với key đầu vào là từ của message trong biến Dict word2index
                except:
                    temp.append(word2index['<OUT>']) # Nếu từ không có trong từ điển thì vào temp value của tag <OUT> trong Dict word2index
            question_vector.append(temp) # Thêm vào mảng question_vector

        # Padding độ dài vector của tin nhắn đầu vào sao cho đồng nhất với mô hình (MAX_LEN = 20)
        question_vector = pad_sequences(question_vector, MAX_LEN, padding='post')
```

Hình 4.47: Xử lý các tin nhắn đầu vào cho chương trình.

Vector tin nhắn đầu vào sau khi được padding với độ dài MAX_LEN sẽ được truyền vào mô hình của bộ Encoder để lấy ra các state của nó. Và sau đó, tạo ra một chuỗi rỗng có độ dài là 1 và gán giá trị đó là giá trị của tag <SOS> trong biến word2index.

```
# Truyền câu hỏi vào cho bộ Encoder LSTM để trả về các state của bộ Encoder LSTM
stat = encoder_model.predict(question_vector)

# Tạo một chuỗi mục tiêu có độ dài 1 và gán cho nó là giá trị index của <SOS> trong Dict word2index
empty_target_seq = np.zeros( ( 1 , 1 ) )
empty_target_seq[0, 0] = word2index['<SOS>']
```

Hình 4.48: Truyền câu hỏi vào Encoder LSTM.

Tiếp tục, ta tạo điều kiện dừng và khởi tạo một biến rỗng để lưu trữ giá trị được trả về từ bộ decoder. Hàm sử dụng một vòng lặp while với điều kiện dừng là stop_condition, và nó sử dụng decoder_model để dự đoán giá trị từ với giá trị đầu vào là giá trị của tag <SOS> được lưu trong mảng empty_target_seq cộng với câu đã được chuẩn hóa vector là stat. Ta sẽ thu được ba giá trị là dec_outputs, h và c. Từ giá trị dec_outputs ta cho nó đi qua lớp Dense đầu ra và thu được giá trị là mảng các danh sách giá trị xác suất có kích thước một hàng, một cột và 4720 giá trị tương ứng với tổng số từ trong từ điển.

Bước kế tiếp, hàm sẽ thực hiện lấy ra giá trị lớn nhất trong biến decoder_concat_input bằng cách lấy tất cả các phần tử và cho qua hàm np.argmax và trả về giá trị lớn nhất tương ứng với giá trị của từ đầu ra.

Tiếp tục, chương trình tạo ra một biến sampled_word và tiến hành mapping giá trị số của từng từ với biến Dictionary index2word. Nếu giá trị của biến sampled_word khác không bằng giá trị là “<EOS>” tức là dấu hiệu của kết thúc một câu thì ta tiếp tục giá trị hiện tại của biến sampled_word vào biến decoded_translation và cứ tiếp tục như vậy cho đến khi vòng lặp kết thúc. Nếu giá trị của sampled_word là <EOS> hoặc độ dài hiện tại của biến decoded_translation lớn hơn MAX_LEN thì tiến hành gán giá trị stop_condition là True và kết thúc vòng lặp. Bên dưới, hàm sẽ thực hiện khởi tạo lại biến empty_target_seq và gán giá trị của từ vừa được xử lý cùng với việc cập nhật lại biến stat gồm giá trị state_h và state_c để làm đầu vào cho lần dự đoán của từ kế tiếp. Cuối cùng, chương trình sẽ trả về biến decoded_translation để hiển thị ra thông điệp sau khi hàm xử lý xong.

```
# Điều kiện dừng cho bộ Decoder
stop_condition = False
decoded_translation = ''

while not stop_condition:
    # Lấy ra prediction từ state của encoder và index trước đó
    dec_outputs, h, c = decoder_model.predict([empty_target_seq] + stat)
    decoder_concat_input = dense(dec_outputs)

    # Từ giá trị prediction ở trên, ta lấy ra giá trị xác suất lớn nhất
    sampled_word_index = np.argmax(decoder_concat_input[0, -1, :])

    # Thực hiện nối từ tương ứng với giá trị index trong biến Dict index2word vào biến sampled_word
    sampled_word = index2word[sampled_word_index] + ' '

    # Nếu word không phải EOS thì nối vào answer
    if sampled_word != '<EOS>':
        decoded_translation += sampled_word

    # Nếu giá trị của biến word là EOS hoặc độ dài của vượt quá 20 thì dừng việc dự đoán
    if sampled_word == '<EOS>' or len(decoded_translation.split()) > MAX_LEN+1:
        stop_condition = True

    # Khởi tạo lại empty_target_seq và đặt token_word_index thành đầu ra của current time step.
    # Sau đó nó sẽ chuyển làm đầu vào của next time step.
    empty_target_seq = np.zeros((1, 1))
    empty_target_seq[0, 0] = sampled_word_index

    # Cập nhật State
    stat = [h, c]
```

Hình 4.49: Vòng lặp thực hiện dự đoán từng với decoder model.

"urls.py

Là nơi lưu các giá trị đường dẫn để thực hiện việc xử lý đến các views. Việc lựa chọn views nào tùy thuộc vào việc thiết lập giá trị URL của chúng ta. Đối với các

function view thì chúng ta chỉ cần gọi đến tệp views nơi chứa các hàm xử lý, nhưng đối với class view thì ta phải thêm thuộc tính `as_view()` để mới có thể sử dụng. Khi gõ địa chỉ trên thanh tìm kiếm của trình duyệt, Django sẽ tiến hành đọc biến urlpatterns và đọc theo thứ tự từ trên xuống dưới cho đến khi trùng khớp với url mà chúng ta nhập vào. Mỗi url được truyền vào với ba tham số bao gồm:

- **Phần địa chỉ:** bao gồm các đường dẫn ứng với mỗi chức năng mà người dùng muốn ánh xạ nó.
- **Phần views:** tương ứng với mỗi giá trị url địa chỉ thì chúng ta sẽ cho nó chạy từng hàm trong tệp views.py theo ý muốn.
- **Phần tên:** mỗi url sẽ có một giá trị name khác nhau để có thể tham chiếu đến chúng trong các Template.

```
from django.conf.urls import url
from chatbotapp import ChatBotVietNamese
from . import views
urlpatterns = [
    url('^/$', views.chatbotvietnamese, name="chatbot"),
    url('^login/user/?$', views.login_func, name="login_func"),
    url('^logout/?$', views.logout_func, name="logout_func"),
    url('^signup/?$', views.signup_func, name="signup_func"),
    url('^updateform/?$', views.update_form, name="updateform"),
    url('^api/vietnamese/?(\?pk>[\w|\W]+)?$', ChatBotVietNamese.APIChatVietNamese.as_view(), name="chatvietnamese"),
    url('^chatbot/?$', views.chatbotvietnamese, name="chatbot"),
    url('^error/?$', views.error, name="error"),
    url('^updateInfo/?$', views.update_info, name="update"),
]
handler404 = "chatbotapp.views.handle_not_found"
```

Hình 4.50: Danh sách các địa chỉ url trong web app.

Và để có thể sử dụng ánh xạ các địa chỉ từ trong web app thì chúng ta phải thực hiện khai báo một url và tiến hành thêm danh sách các địa chỉ urls trong web app vào bằng hàm include như bên dưới.

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import handler404, url, include
from chatbotapp import views

urlpatterns = [
    path('admin/', admin.site.urls),
    url(r'^', include('chatbotapp.urls')),
]
```

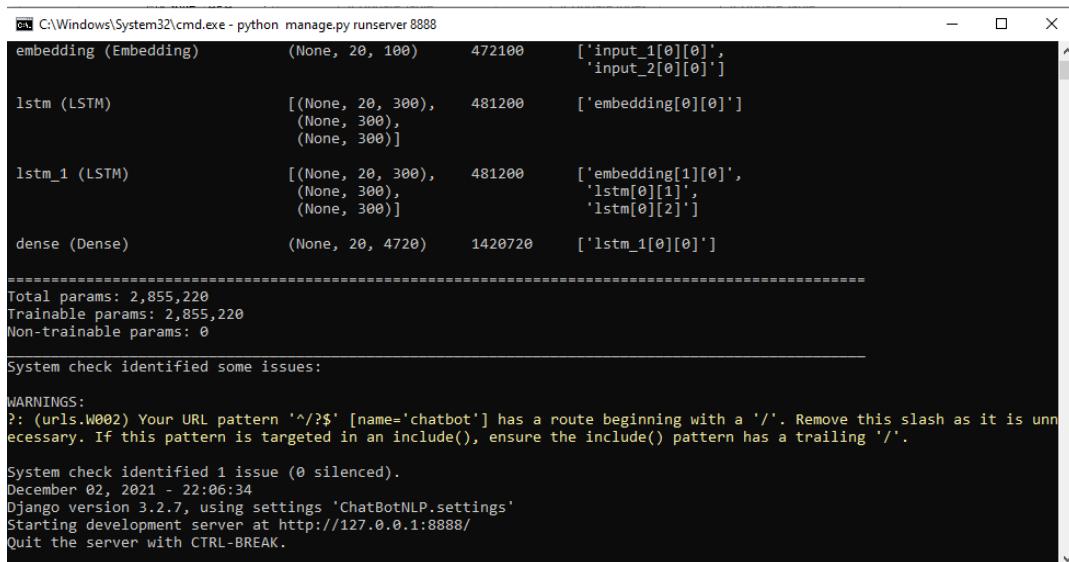
Hình 4.51: Danh sách địa chỉ url chính của project.

4.4 Thực thi chương trình

4.4.1 Khởi chạy

Do được xây dựng dựa trên Django Framework nên chúng ta phải tiến hành khởi chạy giao diện web theo một quy tắc nhất định. Để khởi chạy giao diện, ta tiến hành vào thư mục chứa project và tiến hành mở cửa sổ CMD. Ở cửa sổ CMD, ta nhập câu lệnh “**python manage.py runserver 8888**” để khởi chạy giao diện web.

Chúng ta phải tiến hành đợi một khoảng thời gian để chương trình khởi chạy một số thứ cần thiết cho hệ thống Chatbot của chúng ta. Sau khi khởi chạy hoàn tất, ta sẽ có được một giao diện CMD như sau.



```
C:\Windows\System32\cmd.exe - python manage.py runserver 8888
embedding (Embedding)      (None, 20, 100)    472100    ['input_1[0][0]', 'input_2[0][0]']
lstm (LSTM)                [(None, 20, 300), (None, 300), (None, 300)]   481200    ['embedding[0][0]']
lstm_1 (LSTM)              [(None, 20, 300), (None, 300), (None, 300)]  481200    ['embedding[1][0]', 'lstm[0][1]', 'lstm[0][2]']
dense (Dense)              (None, 20, 4720)   1420720   ['lstm_1[0][0]']

=====
Total params: 2,855,220
Trainable params: 2,855,220
Non-trainable params: 0

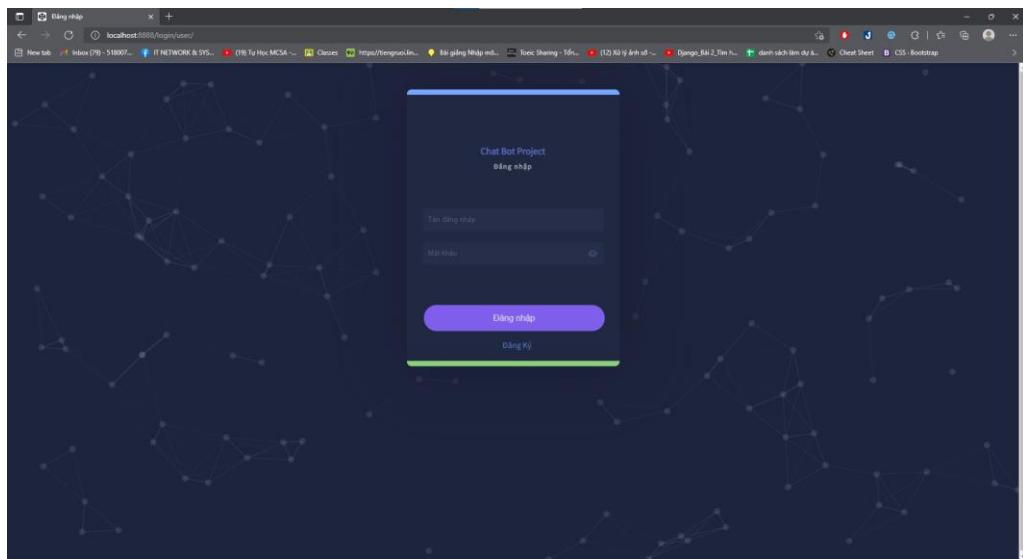
System check identified some issues:
WARNINGs:
?: (urls.W002) Your URL pattern '^/?$' [name='chatbot'] has a route beginning with a '/'. Remove this slash as it is unnecessary. If this pattern is targeted in an include(), ensure the include() pattern has a trailing '/'.

System check identified 1 issue (0 silenced).
December 02, 2021 - 22:06:34
Django version 3.2.7, using settings 'ChatBotNLP.settings'
Starting development server at http://127.0.0.1:8888/
Quit the server with CTRL-BREAK.
```

Hình 4.52: Khởi tạo Server cho Django.

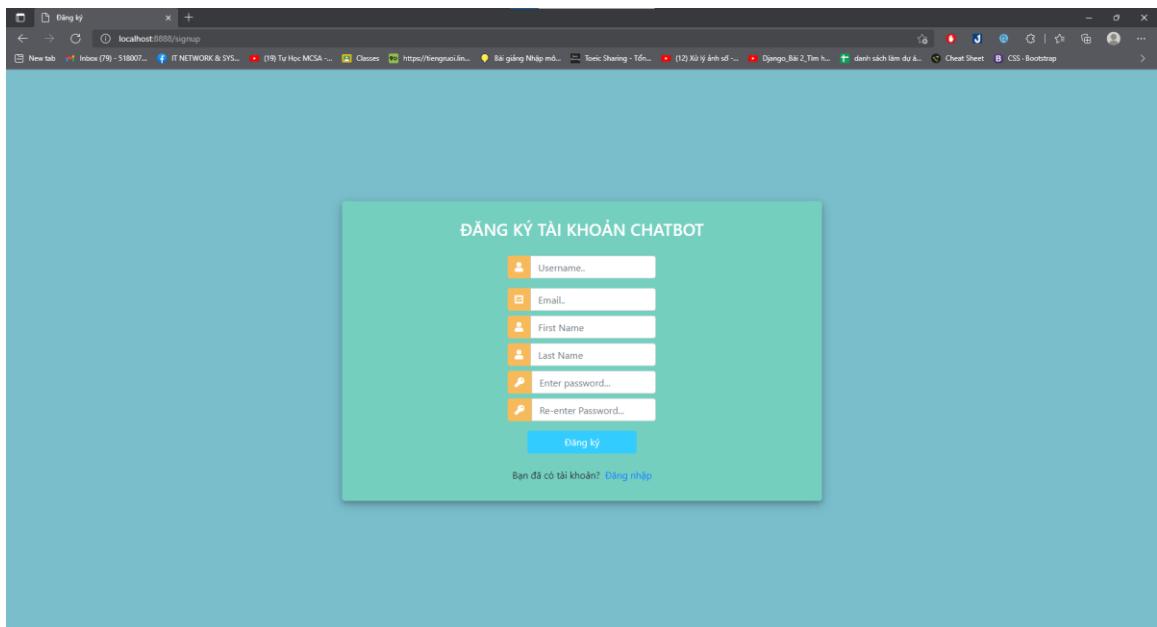
4.4.2 Hệ thống Chatbot

Sau khi khởi động xong server để thực hiện chạy hệ thống Chatbot ta tiến hành mở trình duyệt và nhập địa chỉ <http://localhost:8888> vào thanh tìm kiếm của trình duyệt.



Hình 4.53: Giao diện đăng nhập chương trình. [24]

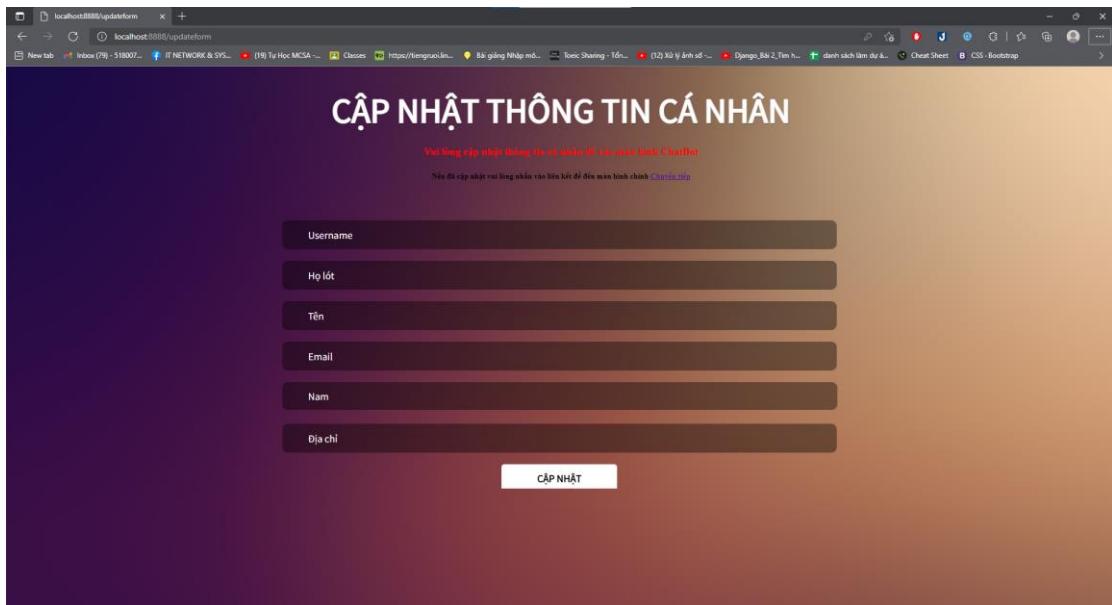
Sau khi nhấn truy cập, thì một giao diện đăng nhập sẽ được hiển thị để người dùng đăng nhập vào hệ thống. Nếu người dùng đã có tài khoản thì chúng ta có thể nhập username và password và tiến hành đăng nhập. Còn đối với những người chưa có tài khoản thì ta sẽ nhấn vào liên kết đăng ký bên dưới.



Hình 4.54: Giao diện đăng ký tài khoản. [25]

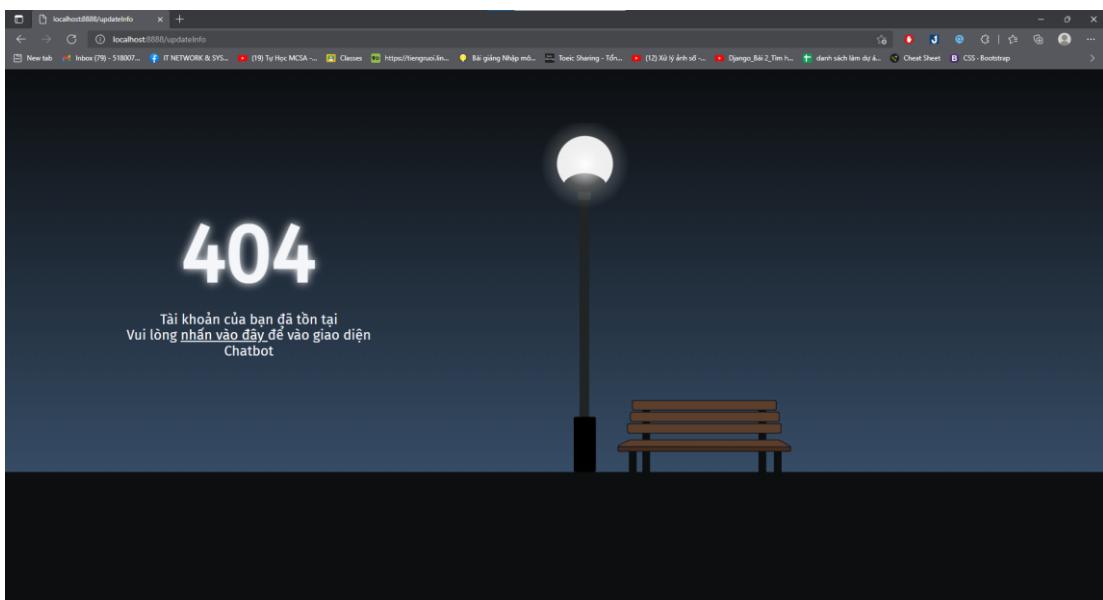
Sau khi nhấn vào liên kết đăng ký, một giao diện đăng ký tài khoản được hiển thị ra cho người dùng đăng ký tài khoản. Những thông tin cần thiết như username, email, firstname, lastname, password và re-password. Sau khi nhấn vào nút Đăng ký thì chương trình sẽ chuyển chúng ta về trang đăng nhập và chúng ta có thể tiến hành đăng nhập vào

hệ thống. Sau khi đã đăng nhập thành công, một giao diện cập nhật thông tin cá nhân được xuất hiện để người dùng thêm dữ liệu vào cơ sở dữ liệu.



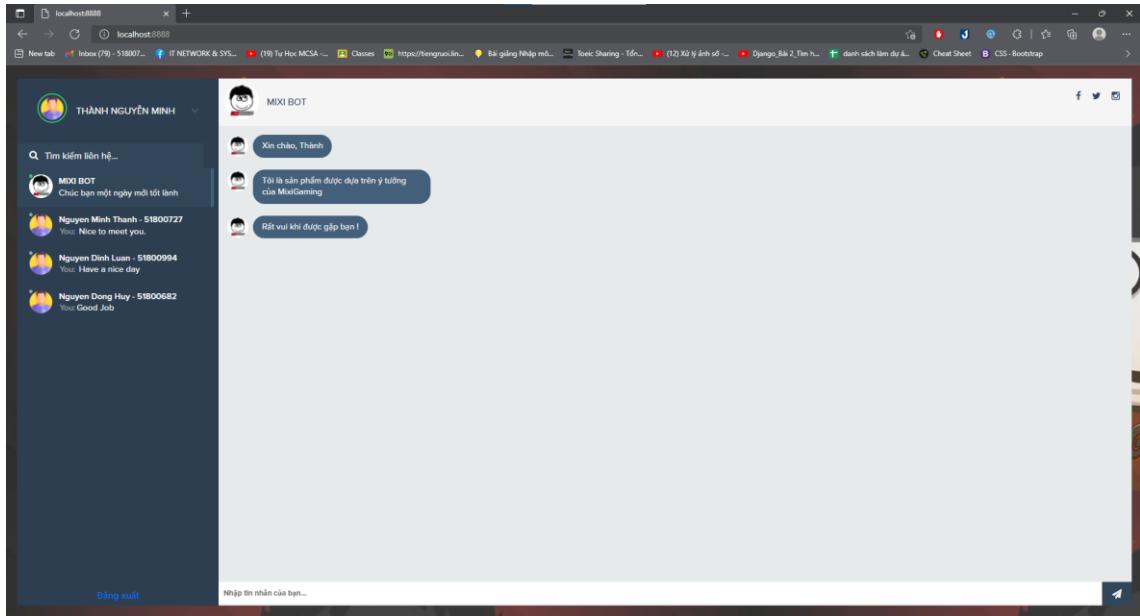
Hình 4.55: Giao diện cập nhật thông tin cá nhân. [26]

Nếu người dùng nhấp cập nhật tài khoản với username đã tồn tại trong hệ thống thì hệ thống sẽ hiển thị một trang báo lỗi và thông báo với người dùng rằng tên tài khoản đã tồn tại và người dùng có thể nhấn vào liên kết trên trang để vào giao diện chính của chương trình.



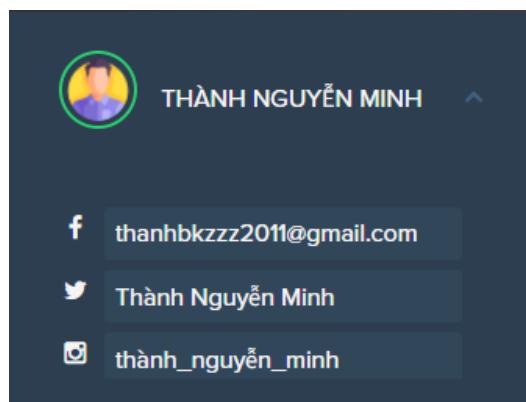
Hình 4.56: Giao diện trang web báo lỗi. [27]

Sau khi cập nhật xong các thông tin cá nhân hoặc nhấn vào liên kết từ giao diện báo lỗi với tài khoản đã tồn tại thì chúng ta sẽ được chuyển đến giao diện chính của chương trình Chatbot.



Hình 4.57: Giao diện chính của chương trình Chatbot. [23]

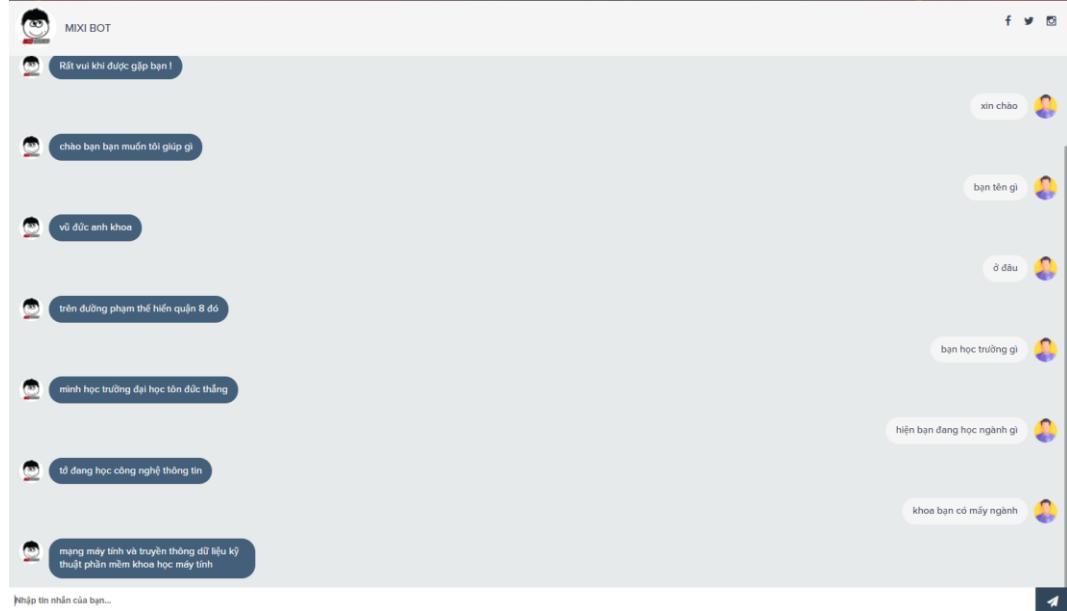
Với giao diện Chatbot ở hình 4.57, bố cục được chia thành hai phần chính với thành phần bên trái để hiển thị các thông tin như tên của người dùng được lấy một cách tự động từ tên mà người dùng đã đăng ký. Bên cạnh đó, bên trái của trang giao diện có một mục để hiển thị các liên kết đến các trang mạng xã hội. Các tên tài khoản này được tạo một cách tự động và sử dụng Template Tag để xử lý.



Hình 4.58: Các tên hoặc liên kết đến các MXH sử dụng Template Tag.

Tiếp đến, chúng ta cùng phân tích đến phần xử lý chính của chương trình. Đầu tiên, để có thể giao tiếp với hệ thống ở phía sau giao diện chính này chúng ta phải nhập dữ liệu vào ô input ở bên dưới với thông điệp “Nhập tin nhắn của bạn ...”. Sau khi

nhập tin nhắn, người dùng có thể gửi bằng cách nhấn phím Enter hoặc nhấn vào button gửi ngay bên cạnh trường nhập tin nhắn. Sau khi thực hiện gửi tin nhắn, giao diện sẽ hiển thị tin nhắn vừa gửi ở bên phải và tin nhắn phản hồi sẽ được hiển thị bên trái của khung trò chuyện.



Hình 4.59: Quá trình trò chuyện và giao tiếp giữa người dùng và hệ thống.

Khi người dùng nhập tin nhắn và nhấn vào nút gửi hoặc phím Enter thì tin nhắn sẽ được gửi đến hàm xử lý tin nhắn bằng phương thức POST với Django Rest Framework được cài đặt trong tệp ChatBotVietNamese. Nếu người dùng không nhập ký tự hoặc gửi các khoảng trắng thì chương trình sẽ hiển thị ra các thông báo lỗi yêu cầu người dùng nhập tin nhắn. Để hiểu rõ hơn về quá trình xử lý chương trình Chatbot bên phía người dùng thì chúng ta cùng xem xét quá trình xử lý của nó để có một cái nhìn tổng quan hơn về cách hoạt động của chương trình.

Để có thể thuận tiện hơn trong việc thêm và hiển thị tin nhắn vừa gửi của người dùng vào bên trái của giao diện mỗi khi nhấn gửi tin nhắn thì chúng ta sẽ thực hiện tạo ra một hàm mới có tên là newMessage(). Trong hàm newMessage(), việc đầu tiên là lấy ra dữ liệu văn bản từ trường nhập tin nhắn. Nếu người dùng nhập vào giá trị rỗng thì sẽ không thực hiện thêm và hiển thị tin nhắn. Nếu tin nhắn nhập vào không rỗng thì thực hiện thêm các tag vào tag trong màn hình Chatbot. Sau đó, hàm sẽ thực hiện xóa giá trị tại trường nhập tin nhắn bằng hàm val(null). Sau đó, chương trình cũng thực hiện cập nhật tin nhắn vừa gửi vào bên trái nơi hiển thị các mục lịch sử tin nhắn người dùng và cuối cùng mỗi khi thêm tin nhắn vào giao diện thì sẽ có một animation

là scrollTop với tốc độ chậm (slow) và khoảng cách mỗi lần thêm sẽ cộng thêm 3000 để đẩy tin nhắn khi thêm vào luôn luôn ở bên dưới.

```
// Hàm hiển thị tin nhắn bên người gửi
function newMessage() {
    message = $(".message-input input").val();
    if($.trim(message) == '') {
        return false;
    }
    $('<li class="sent"><p>' + message + '</p></li>').appendTo($('.messages ul'));
    $('.message-input input').val(null);
    $('.contact.active .preview').html('<span>You: </span>' + message);
    $('.messages').animate({ scrollTop: $(document).height() + 3000 }, "slow");
}
```

Hình 4.60: Hàm thêm tin nhắn khi người dùng nhấn vào nút gửi.

Tiếp đến, ta đến với hàm xử lý chính để gửi dữ liệu lên phía máy chủ thông qua AJAX. Đầu tiên là việc bắt sự kiện khi người dùng nhấn vào phím Enter để gửi tin nhắn. Khi nhấn vào phím Enter, một biến message sẽ được tạo để lưu giá trị mà người dùng đã nhập vào trường nhập tin nhắn. Sau đó, chương trình sẽ thực hiện khởi tạo một phương thức AJAX để gửi dữ liệu đến máy chủ để thực hiện quá trình truy xuất mô hình đã được huấn luyện. Với các giá trị khởi tạo bao gồm:

- url:** là địa chỉ truy xuất đến lớp rest framework trong Django.
- method:** là phương thức truyền tải mà ở đây là POST.
- datatype:** là kiểu dữ liệu trả về mà ở đây là JSON.
- data:** là nơi khởi tạo các csrf token cho việc giao tiếp với Django.
- success:** là nơi thực hiện các việc khi quá trình nhận dữ liệu từ phía máy chủ thành công.
- error:** là nơi thực hiện các hàm khi quá trình nhận dữ liệu không thành công.

```
// Hàm bắt sự kiện gửi tin nhắn của chương trình
$(window).on('keydown', function(e) {
    if (e.which == 13) { //Nếu nhấn vào Enter
        message = $(".message-input input").val(); // Lấy ra giá trị trong trường tin nhắn
        console.log(message) //Hiển thị ra màn hình console
        $.ajax({ //Định nghĩa AJAX để gửi dữ liệu đến phía máy chủ
            url: '/api/' + message, // Định nghĩa đường dẫn
            method: 'post', // Định nghĩa phương thức
            datatype: 'json', // Định nghĩa kiểu dữ liệu
            data:{csrfmiddlewaretoken: '{{ csrf_token }}'}, //Định nghĩa csrf token
            success: function (data){ //Nếu nhận dữ liệu được trả về thành công
                console.log(data.id) // Hiển thị tin nhắn được trả về
                let message = data.id //Gán biến message là giá trị tin nhắn trả về

                // Thực hiện thêm tin nhắn vừa gửi vào bên phải giao diện
                newMessage()

                // Hiển thị tin nhắn được trả về từ phía máy chủ và hiển thị bên trái màn hình (replies)

                $('<li class="replies"><p>' + message + '</p></li>').appendTo($('.messages ul'));
                $('.message-input input').val(null);
                $('.contact.active .preview').html('<span>You: </span>' + message);
                $('.messages').animate({ scrollTop: $(document).height() + 3000 }, "fast");
            },
            error: function (data){
            },
        });
        return false;
    }
});
```

Hình 4.61: Xử lý bắt sự kiện và gửi dữ liệu đến phía máy chủ.

4.5 Tổng kết chương 4

Như vậy là chúng ta đã đi qua chương thứ 4 của bài báo cáo, ở chương này chúng ta đã tìm hiểu sâu hơn về sơ đồ hoạt động của chương trình, tìm hiểu các thư viện được sử dụng và giải thích các mã được sử dụng để xây dựng nên một hệ thống Chatbot trên nền Django và xử lý ngôn ngữ tự nhiên (NLP). Bên cạnh đó, chúng ta cũng đã tìm hiểu và giải thích về các thành phần trong Django như các khái niệm cơ bản, mô hình MVT cho đến từng module thành phần trong nó để hiểu rõ hơn về cách thức hoạt động của nó. Thêm nữa, chúng ta đã thực hiện xây dựng mô hình Seq2Seq dựa trên mạng Neural hồi quy mà ở đây là LSTM giúp cho việc dự đoán các từ tiếp theo trong câu được chính xác hơn. Cuối cùng, chúng ta đã đi qua bước thực thi chương trình để có một cái nhìn tổng quan hơn về chương trình được xây dựng.

CHƯƠNG 5: TỔNG KẾT VÀ ĐÁNH GIÁ

5.1 Tổng kết

Sau khi đi qua bốn chương của bài báo cáo của môn “Nhập môn tính toán đa phương tiện”, chúng em đã hiểu rõ được các khái niệm từ cơ bản đến nâng cao trong lĩnh xử lý ngôn ngữ tự nhiên. Từ các khái niệm cơ bản về ngôn ngữ tự nhiên, lịch sử hình thành và ra đời của việc xử lý ngôn ngữ tự nhiên cho đến các ứng dụng thực tế và lợi ích mà nó mang đến cho con người chúng ta trong cuộc sống hiện nay. Đây là một trong những lĩnh vực có tiềm năng rất lớn và đang tăng trưởng, phát triển xa hơn trong tương lai. Bên cạnh các kiến thức cơ bản về lĩnh vực xử lý ngôn ngữ tự nhiên, chúng em cũng đã có cơ hội trau dồi những kiến thức về lĩnh vực lập trình web mà ở đây là sử dụng Django. Với đề tài “Xây dựng hệ thống trả lời tự động đa chủ đề”, nó đã giúp cho chúng em có thêm nhiều kiến thức hơn trong lĩnh vực xử lý ngôn ngữ tự nhiên và lập trình web nói riêng. Hiểu được các bước của việc xây dựng nên một mô hình, hiểu được các đặc điểm của nó, cách để triển khai chương trình Chatbot bằng framework Django trong Python và nhiều các ứng dụng của xử lý ngôn ngữ tự nhiên trong thực tế.Thêm nữa, chúng em cũng đã hiểu thêm về các kiến thức liên quan đến lĩnh vực học máy, hiểu được sâu hơn về các thư viện học máy như Tensorflow, Keras, ... và các mô hình trong nó. Với đề tài “Xây dựng hệ thống trả lời tự động đa chủ đề” thì chúng em cũng đã thiết kế và xây dựng nên một chương trình có thể giao tiếp và trò chuyện với nó như một thực thể sống. Qua bài báo cáo lần này, chúng em có thể ứng dụng kiến thức đã học và tự xây dựng cho bản thân mình những ứng dụng riêng đặc biệt là trong lĩnh vực trả lời tự động.

5.2 Đánh giá và phân tích

5.2.1 Phân tích

Sau khi thực hiện xong bài báo cáo, chúng em có những phân tích về việc thực hiện báo cáo như sau. Đầu tiên, trong quá trình thực hiện bài báo cáo thì chúng em đã có cơ hội tiếp xúc với nhiều các kiến thức về xử lý ngôn ngữ tự nhiên mà dường như chưa được tiếp xúc trong lĩnh vực công nghệ thông tin. Bên cạnh đó, các kiến thức về việc sử dụng framework Django cũng được chúng em tìm hiểu và hiểu rõ hơn về nó. Từ các khái niệm cơ bản cho đến nâng cao về lĩnh vực xử lý ngôn ngữ tự nhiên. Điều thứ hai, đó chính là việc triển khai chương trình trả lời tự động được gọi là Chatbot. Sau khi tìm hiểu và triển khai, chúng em đã có một cái nhìn tổng quát và khách quan hơn, hiểu

rõ hơn về phương pháp Seq2Seq cũng như là bước xây dựng mô hình và đặc điểm của nó. Trong quá trình triển khai, chúng em cũng được tiếp xúc với nhiều các thư viện khác nhau trong ngôn ngữ Python. Điều này làm cho chúng em có thêm nhiều kiến thức và kinh nghiệm hơn về ngôn ngữ lập trình Python trong xử lý ngôn ngữ tự nhiên. Và còn rất nhiều các kiến thức mà nhóm chúng em đã tìm hiểu và tích lũy được cho riêng bản thân của mỗi người.

5.2.2 Đánh giá

Trong bài tiểu luận lần này, nhóm chúng em đã quyết định xây dựng nên một mô hình chương trình dựa trên ngôn ngữ lập trình Python. Việc xây dựng và tìm hiểu này đã giúp chúng em đã tìm hiểu được rất nhiều vấn đề mới trong lĩnh vực xử lý ngôn ngữ tự nhiên nói chung và trong lĩnh vực lập trình web nói riêng. Sau khi thực hiện xong bài báo cáo của môn “Nhập môn tính toán đa phương tiện”, chúng em đánh giá rằng xử lý ngôn ngữ tự nhiên là một trong các lĩnh vực rất có ích đối với con người và nó đang là xu hướng phát triển hiện nay.Thêm nữa, việc tìm hiểu và xây dựng chương trình trả lời tự động giúp chúng em định hình được cách thức hoạt động và các bước của phương pháp Seq2Seq.

Và cuối cùng, chúng em xin kết luận lại phần đánh giá về kết quả thực hiện đề tài của báo cáo đó là chúng em đã có thêm rất nhiều kiến thức và có cơ hội tiếp xúc trực tiếp với các mô hình trong lĩnh vực xử lý ngôn ngữ tự nhiên. Đây là một trong những cơ hội hiếm hoi để chúng em có thể tiếp xúc, hiểu được và lập trình ra được các chương trình trả lời tự động mang lại cho lợi ích và có thể ứng dụng rất cao trong thực tế. Với tư cách là những sinh viên thực hiện bài báo cáo của môn “Xử lý ngôn ngữ tự nhiên” thì chúng em xin gửi lời cảm ơn chân thành đến thầy Vũ Đình Hồng và thầy Lê Anh Cường đã tạo cơ hội cho chúng em tìm hiểu sâu hơn về lĩnh vực xử lý ngôn ngữ tự nhiên nói chung và trong cả lĩnh vực lập trình web nói riêng. Các thầy đã giúp chúng em có thêm nhiều các kiến thức nền tảng cơ bản về hai lĩnh vực trên. Ngoài ra, các thầy cũng đã tận tâm giúp đỡ chúng em trong suốt quá trình học bằng cách đưa ra các bài thực hành rất sát với đề tài. Với những hạn chế còn thiếu sót về mặt kiến thức, chúng em rất mong nhận được sự đánh giá và nhận xét của các thầy về tiểu luận lần này của nhóm. Để chúng em có thể cung cấp kiến thức cho bản thân mình và cũng như là sẽ phục vụ tốt hơn trong công việc tương lai của chúng em. Chúng em xin chân thành cảm ơn.

TÀI LIỆU THAM KHẢO

Tài liệu tham khảo chính

- [1] Abdullahi, S. S., Yiming, S., Abdullahi, A., & Aliyu, U. (2019, December). Open domain chatbot based on attentive end-to-end Seq2Seq mechanism. In *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence* (pp. 339-344).
- [2] Cahn, J. (2017). CHATBOT: Architecture, design, & development. *University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science.*
- [3] Kang, Y., Cai, Z., Tan, C. W., Huang, Q., & Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, 7(2), 139-172.
- [4] Lokman, A. S., & Ameedeen, M. A. (2018, November). Modern chatbot systems: A technical review. In *Proceedings of the future technologies conference* (pp. 1012-1023). Springer, Cham.
- [5] Qiu, M., Li, F. L., Wang, S., Gao, X., Chen, Y., Zhao, W., ... & Chu, W. (2017, July). Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (pp. 498-503).

Tài liệu tham khảo phụ

- [6] <https://timviec365.vn/blog/ngon-ngu-tu-nhien-la-gi-new15440.html>
- [7] <https://gialaicpc.com.vn/cac-ung-dung-xu-ly-ngon-ngu-tu-nhien-nlp-hang-dau-trong-kinh-doanh/>
- [8] <https://trituenhantao.io/kien-thuc/ung-dung-cua-nlp-7-thanh-tu-noi-bat/>
- [9] https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_th%C3%B4ng_tin
- [10] <https://viblo.asia/p/entropy-cross-entropy-va-kl-divergence-naQZRMRXKvx>
- [11] <https://viblo.asia/p/introduction-of-natural-language-processing-GrLZD9xElk0>
- [12] https://www.csie.ntu.edu.tw/~yvchen/s105-icb/doc/170502_NLG.pdf

- [13] https://www.researchgate.net/figure/A-second-generation-Artificial-Neural-Network-ANN-with-a-sigmoidal-activation_fig4_262493920
- [14] <https://dominhhai.github.io/vi/2017/10/what-is-rnn/>
- [15] <https://dominhhai.github.io/vi/2017/10/what-is-lstm/>
- [16] <https://viblo.asia/p/gioi-thieu-bai-toan-tom-tat-van-ban-su-dung-mo-hinh-lstm-based-seq2seq-va-co-che-attention-6J3ZgWeRZmB>
- [17] <https://ironhackvietnam.edu.vn/django-la-gi/>
- [18] <https://viblo.asia/p/python-co-ban-voi-django-framework-Ljy5VxGkZra>
- [19] <https://www.miai.vn/thu-vien-mi-ai/>
- [20] <https://www.pluralsight.com/guides/nmt:-encoder-and-decoder-with-keras>
- [21] <https://ichi.pro/vi/xay-dung-mot-chatbot-dich-ngon-ngu-bang-python-tung-buoc-4428266027672>
- [22] <https://ichi.pro/vi/chatbot-dua-tren-tao-ra-57910830423224>
- [23] <https://bootsnipp.com/snippets/exR5v>
- [24] <https://codepen.io/lordgamer2354/pen/NEroLg>
- [25] <https://jsfiddle.net/ivanov11/dghm5cu7/>
- [26] <https://codepen.io/andstudio/pen/jeDty>
- [27] <https://codepen.io/janmez/pen/LJOdar>

BẢNG PHÂN CÔNG CÔNG VIỆC

Công việc	Phân công	Đánh giá
Tìm kiếm tài liệu liên quan đến đề tài	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Thực hiện chương 1	Nguyễn Đồng Huy Nguyễn Minh Thành	Hoàn thành tốt
Thực hiện chương 2	Nguyễn Đình Luân Nguyễn Minh Thành	Hoàn thành tốt
Thực hiện chương 3	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Thực hiện chương 4	Nguyễn Minh Thành	Hoàn thành tốt
Thực hiện chương 5	Nguyễn Minh Thành	Hoàn thành tốt
Xây dựng và đóng góp ý kiến cho đề tài	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Tìm kiếm hình ảnh cho đề tài (báo cáo, ứng dụng)	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Xây dựng mô hình Seq2Seq cho Chatbot	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Xây dựng giao diện cho chương trình với Django	Nguyễn Minh Thành	Hoàn thành tốt
Tổng hợp chương trình	Nguyễn Minh Thành	Hoàn thành tốt
Kiểm tra định dạng báo cáo	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Kiểm tra lỗi chính tả của báo cáo	Nguyễn Minh Thành Nguyễn Đồng Huy Nguyễn Đình Luân	Hoàn thành tốt
Thực hiện thuyết trình	Nguyễn Minh Thành	Hoàn thành tốt

Tổng hợp báo cáo và chịu trách nhiệm báo cáo	Nguyễn Minh Thành	Hoàn thành tốt
---	-------------------	----------------