

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------------------------------------

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**Đề tài: XÂY DỰNG ỨNG DỤNG CHATBOT PHỤC VỤ TUYỂN SINH TẠI HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

| **Giảng viên hướng dẫn:** | ThS. Vũ Hoài Thư |
| --- | --- |
| **Sinh viên thực hiện:** | Nguyễn Văn Đức |
| **Mã sinh viên:** | B20DCCN199 |
| **Lớp:** | D20CNPM04 |
| **Niên khóa:** | 2020-2025 |
| **Hệ đào tạo:** | Đại học chính quy |

**Hà Nội 2024**



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------------------------------

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

**Đề tài: XÂY DỰNG ỨNG DỤNG CHATBOT PHỤC VỤ TUYỂN SINH TẠI HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

| **Giảng viên hướng dẫn:** | ThS. Vũ Hoài Thư |
| --- | --- |
| **Sinh viên:** | Nguyễn Văn Đức |
| **Mã sinh viên:** | B20DCCN199 |
| **Lớp:** | D20CNPM04 |
| **Niên khóa:** | 2020-2025 |
| **Hệ đào tạo:** | Đại học chính quy |

**Hà Nội 2024**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

…………………………………………………………………………………………

**Điểm: (Bằng chữ: )**

Hà Nội, ngày tháng năm 20…

**Giảng viên hướng dẫn**

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

…………………………………………………………………………………………

**Điểm: (Bằng chữ: )**

Hà Nội, ngày tháng năm 20…

**Giảng viên phản biện**

**LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành nhất đến quý thầy cô của Học Viện Công Nghệ Bưu Chính Viễn Thông nói chung và các thầy cô trong Bộ môn Công nghệ phần mềm cũng như khoa Công nghệ thông tin nói riêng đã tận tình giảng dạy và truyền đạt những kiến thức chuyên môn nền tảng để làm cơ sở lý luận cho đồ án này.

Em xin bày tỏ lòng biết ơn sâu sắc đến cô ThS. Vũ Hoài Thư đã dìu dắt và hướng dẫn em trong suốt quá trình làm đồ án. Nhờ sự chỉ bảo và định hướng của cô, em đã tự tin hơn trong việc nghiên cứu vấn đề mới và tìm ra giải pháp tối ưu.

Em xin chân thành cảm ơn Ban Giám hiệu Học viện Công Nghệ Bưu Chính Viễn Thông đã tạo điều kiện thuận lợi để em được học tập và làm đồ án một cách thuận lợi.

Mặc dù đã cố gắng rất nhiều, nhưng chắc chắn trong quá trình học tập cũng như làm đồ án không khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và đóng góp của quý thầy cô và các bạn để đồ án hoàn thiện hơn nữa.

Em xin chân thành cảm ơn!

**Hà Nội, tháng 12 năm 2024**

**Sinh viên**

**Nguyễn Văn Đức**

**MỤC LỤC**

[**LỜI MỞ ĐẦU 7**](#_heading=h.4d34og8)

[**CHƯƠNG 1: TÌM HIỂU BÀI TOÁN ĐIỂM DANH SỬ DỤNG NHẬN DIỆN KHUÔN MẶT 1**](#_heading=h.17dp8vu)

[**1.1.**](#_heading=h.3rdcrjn) **Tổng quan về bài toán 1**

[***1.1.1.***](#_heading=h.26in1rg) ***Bài toán điểm danh bằng khuôn mặt 1***

[***1.1.2.***](#_heading=h.lnxbz9) ***Những khó khăn của việc điểm danh bằng khuôn mặt 1***

[**1.2.**](#_heading=h.35nkun2) **Cơ sở lý thuyết và tổng quan về mạng nơ-ron tích chập (Convolution Neural Network) trong Deep Learning 2**

[***1.2.1.***](#_heading=h.1ksv4uv) ***Mạng nơ-ron nhân tạo (Neural Network) 2***

[***1.2.2.***](#_heading=h.1y810tw) ***Mạng nơ-ron tích chập (Convolution Neural Network) 6***

[**a.**](#_heading=h.2xcytpi) **Kiến trúc cơ bản của mạng nơ-ron tích chập 7**

[**b.**](#_heading=h.2p2csry) **Một số kiến trúc mạng nơ-ron tích chập 12**

[**CHƯƠNG 2: KHẢO SÁT CÁC CÔNG NGHỆ XỬ LÝ ẢNH, HỌC MÁY TRONG HỆ THỐNG NHẬN DIỆN GƯƠNG MẶT 17**](#_heading=h.41mghml)

[***Tổng quan hệ thống nhận diện khuôn mặt 17***](#_heading=h.2grqrue)

[**2.1.**](#_heading=h.3fwokq0) **Bộ phát hiện khuôn mặt (Face Detector) 18**

[***❖***](#_heading=h.4f1mdlm) ***Detect dựa trên các hand-craft feature (đặc trưng cứng) 18***

[***❖***](#_heading=h.19c6y18) ***Detect dựa trên các feature được trích chọn từ mạng CNN 19***

[***Một số mô hình phát hiện khuôn mặt dựa trên mạng CNN. 19***](#_heading=h.28h4qwu)

[**2.2.**](#_heading=h.2lwamvv) **Bộ căn chỉnh khuôn mặt (Face Alignment) 24**

[**2.3.**](#_heading=h.4k668n3) **Bộ chống giả mạo khuôn mặt (Face Anti-spoofing) 25**

[***Một số phương pháp chính cho bài toán chống tấn công giả mạo khuôn mặt: 26***](#_heading=h.1egqt2p)

[***Một số nghiên cứu mạng CNN cho bài toán chống giả mạo khuôn mặt: 27***](#_heading=h.3ygebqi)

[**2.4.**](#_heading=h.3cqmetx) **Bộ trích xuất đặc trưng khuôn mặt (Face Feature Extraction) 29**

[***Một số Loss Function: 30***](#_heading=h.2r0uhxc)

[***❖***](#_heading=h.3q5sasy) ***ArcFace (Additive Angular Margin Loss) 31***

[**2.5.**](#_heading=h.kgcv8k) **Bộ nhận diện khuôn mặt 32**

[***Một số phương pháp tính khoảng cách: 32***](#_heading=h.34g0dwd)

[***❖***](#_heading=h.43ky6rz) ***Cosine Similarity 33***

[***❖***](#_heading=h.xvir7l) ***Hamming Distance 33***

[**CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 35**](#_heading=h.1x0gk37)

[**3.1.**](#_heading=h.4h042r0) **Phân tích hệ thống 35**

[***3.1.1.***](#_heading=h.2w5ecyt) ***Mô tả các chức năng chính của hệ thống: 35***

[***3.1.2.***](#_heading=h.1baon6m) ***Use Case tổng quan của hệ thống: 36***

[***3.1.3.***](#_heading=h.2afmg28) ***Đặc tả các Use Case của hệ thống: 36***

[**3.2.**](#_heading=h.j8sehv) **Thiết kế hệ thống 50**

[***3.2.1.***](#_heading=h.338fx5o) ***Biểu đồ lớp của hệ thống 50***

[***3.2.2.***](#_heading=h.42ddq1a) ***Biểu đồ CSDL 50***

[***3.2.3.***](#_heading=h.wnyagw) ***Thiết kế vật lý của hệ thống 52***

[**3.3.**](#_heading=h.2uxtw84) **Tổng quan chức năng điểm danh nằng khuôn mặt 53**

[**CHƯƠNG 4: THỬ NGHIỆM CÀI ĐẶT VÀ XÂY DỰNG ỨNG DỤNG 57**](#_heading=h.3ls5o66)

[**4.1.**](#_heading=h.20xfydz) **Cài đặt Server 57**

[**4.2.**](#_heading=h.4kx3h1s) **Cài đặt Client 57**

[**4.3.**](#_heading=h.302dr9l) **Thiết kế giao diện 58**

[***4.3.1.***](#_heading=h.1f7o1he) ***Giao diện đăng nhập 58***

[***4.3.2.***](#_heading=h.2eclud0) ***Giao diện danh sách lớp học 59***

[***4.3.3.***](#_heading=h.3dhjn8m) ***Giao diện điểm danh 60***

[***4.3.4.***](#_heading=h.4cmhg48) ***Giao diện kết quả điểm danh 61***

[***4.2.5.***](#_heading=h.16x20ju) ***Giao diện đăng nhập của Admin 62***

[***4.2.6.***](#_heading=h.261ztfg) ***Giao diện thêm mới tài khoản 62***

[***4.2.7.***](#_heading=h.356xmb2) ***Giao diện thêm ảnh vào tài khoản sinh viên 63***

[***4.2.8.***](#_heading=h.44bvf6o) ***Giao diện sửa tài khoản 63***

[***4.2.9.***](#_heading=h.ymfzma) ***Giao diện xóa tài khoản 64***

[**KẾT LUẬN 65**](#_heading=h.1xrdshw)

[**1.**](#_heading=h.4hr1b5p) **Kết quả đạt được 65**

[**2.**](#_heading=h.2wwbldi) **Hướng phát triển 65**

[**TÀI LIỆU THAM KHẢO 66**](#_heading=h.3w19e94)

[**Tiếng Anh 66**](#_heading=h.2b6jogx)

[**Danh mục các website tham khảo 66**](#_heading=h.qbtyoq)

**DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT**

| **STT** | **Từ viết tắt** | **Từ chuẩn** | **Diễn giải** |
| --- | --- | --- | --- |
| 1 | AI | Artificial Intelligence | Trí tuệ nhân tạo |
| 2 | NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |

**DANH MỤC CÁC HÌNH VẼ**

**DANH MỤC CÁC BẢNG**

# LỜI MỞ ĐẦU

Trong thời đại công nghệ số hiện nay, việc áp dụng các giải pháp tự động hóa vào các hoạt động thường ngày, đặc biệt trong lĩnh vực giáo dục, ngày càng trở nên phổ biến. Một trong những ứng dụng nổi bật là chatbot, công cụ hỗ trợ giao tiếp tự động, mang lại sự thuận tiện và hiệu quả trong việc giải đáp thông tin. Chatbot không chỉ giúp giảm tải công việc cho nhân viên mà còn nâng cao trải nghiệm người dùng nhờ khả năng phản hồi nhanh chóng và chính xác.

Đồ án này nhằm phát triển một chatbot hỗ trợ quá trình tư vấn và giải đáp thông tin tuyển sinh cho thí sinh một cách nhanh chóng và hiệu quả. Chatbot này sẽ cung cấp các thông tin liên quan đến kỳ thi tuyển sinh, điều kiện xét tuyển, chương trình đào tạo, và các thủ tục đăng ký một cách tự động.

Mục tiêu chính của đồ án là xây dựng một chatbot có khả năng hiểu ngôn ngữ tự nhiên, đáp ứng nhu cầu của thí sinh trong việc tra cứu thông tin tuyển sinh, qua đó góp phần cải thiện quy trình tuyển sinh của nhà trường. Đồng thời, đồ án cũng sẽ đề cập đến các kỹ thuật và công nghệ hiện đại được áp dụng trong quá trình xây dựng chatbot.

Trong đồ án em sẽ tập trung trình bày một số nội dung chính như sau:

* **Chương 1: Tổng quan các hệ thống chatbot:**

Chương này đã trình bày và giới thiệu những kiến thức tổng quan nhất về hệ thống chatbot, phân tích các ưu nhược điểm của các mô hình chatbot hiện nay từ đó định hướng xây dựng mô hình chatbot mà đồ án nghiên cứu và xây dựng.

* **Chương 2: Cơ sở lý thuyết:**
* **Chương 3: Xây dựng chatbot tuyển sinh:**
* **Chương 4: Tổng kết:**

Tổng kết bài toán, tóm tắt những kết quả đã đạt được và còn chưa đạt được. Từ đó đề xuất mục tiêu hướng tới cũng như hướng nghiên cứu, phát triển tiếp theo.

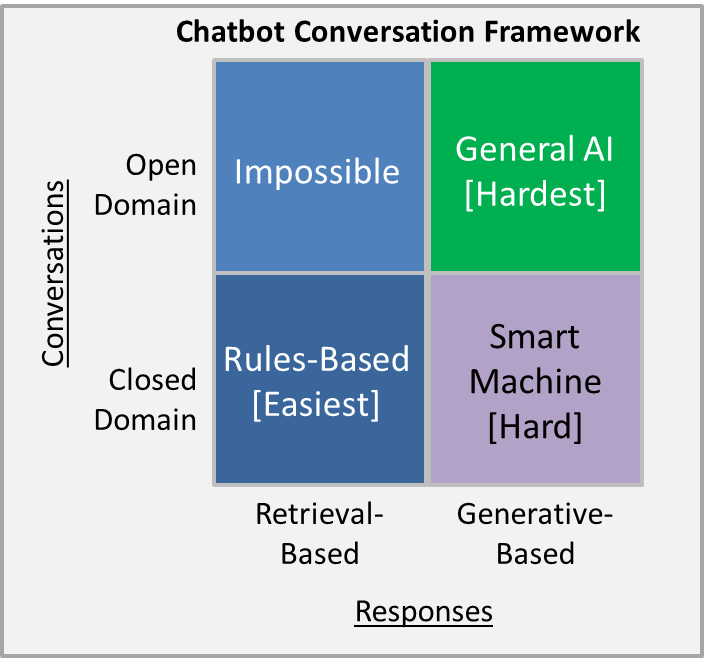
# CHƯƠNG 1: TỔNG QUAN CÁC HỆ THỐNG CHATBOT

## Giới thiệu

Chatbot là một ứng dụng hoặc chương trình máy tính được tạo ra với mục đích tương tác với con người thông qua các cuộc trò chuyện. Sử dụng công nghệ Al và xử lý ngôn ngữ tự nhiên NLP, chatbot có khả năng hiểu và phản hồi lại các câu hỏi từ người dùng một cách tự động. Bằng cách này, chatbot giúp tạo ra trải nghiệm gần như giao tiếp với con người, giải quyết các vấn đề cơ bản và cung cấp thông tin trong nhiều lĩnh vực khác nhau, từ dịch vụ khách hàng đến giáo dục và giải trí... Sự phổ biến của chatbot đã tăng lên trong thời gian gần đây, với việc tích hợp chúng vào các nền tảng trò chuyện trực tuyến, ứng dụng di động và trang web, mang lại lợi ích to lớn cho cả người dùng và doanh nghiệp thông qua việc tiết kiệm thời gian và công sức.

Chatbot có thể được chia thành hai loại chính: Open Domain (miền mở), Close Domain (miền đóng) dựa trên chủ đề và tính linh hoạt trong việc xử lý thông tin.

* **Open Domain** (Miền mở): Chatbot này có khả năng trò chuyện về nhiều chủ đề khác nhau và không giới hạn trong một lĩnh vực cụ thể, nó dựa vào các mô hình học máy và xử lý ngôn ngữ tự nhiên (NLP) để hiểu ý nghĩa của câu hỏi và đưa ra phản hồi dựa trên ngữ cảnh. Nó tương tự như cuộc hội thoại trên mạng xã hội khi các bạn chat với nhau, bạn có thể nói chuyện về một bộ phim và chuyển sang chủ đề nhạc, rồi lại quay sang chủ đề bóng đá. Cũng vì do số lượng chủ đề là rất lớn, hệ thống cần phải trả về lượng response tương ứng nên đây trở thành một bài toán khá khó. Các trợ lý ảo như Siri, Google Assistant hoặc ChatGPT đều là ví dụ điển hình của miền mở.
* **Close Domain** (Miền đóng): Chatbot này được thiết kế để hoạt động trong một phạm vi cụ thể và chỉ trả lời các câu hỏi liên quan đến lĩnh vực hoặc chủ đề nhất định, ví dụ như: thời tiết, y tế, giáo dục, mua sắm, tâm lý, du lịch, ngân hàng, …nó thường dựa vào các quy tắc và kịch bản đã được lập trình sẵn, giúp xử lý những yêu cầu cụ thể. Khi nhận được câu hỏi từ người dùng, chatbot sẽ tìm từ khóa và dựa vào kịch bản để đưa ra câu trả lời thích hợp. Hệ thống hỗ trợ kỹ thuật hay các dịch vụ chăm sóc khách hàng là các ứng dụng thuộc miền đóng.



Hình: …

Hình ảnh trên với bốn phần được phân loại theo kiểu hội thoại (Miền mở với miền đóng) và kiểu phản hồi (Dựa trên truy xuất với dựa trên sinh). Cụ thể:

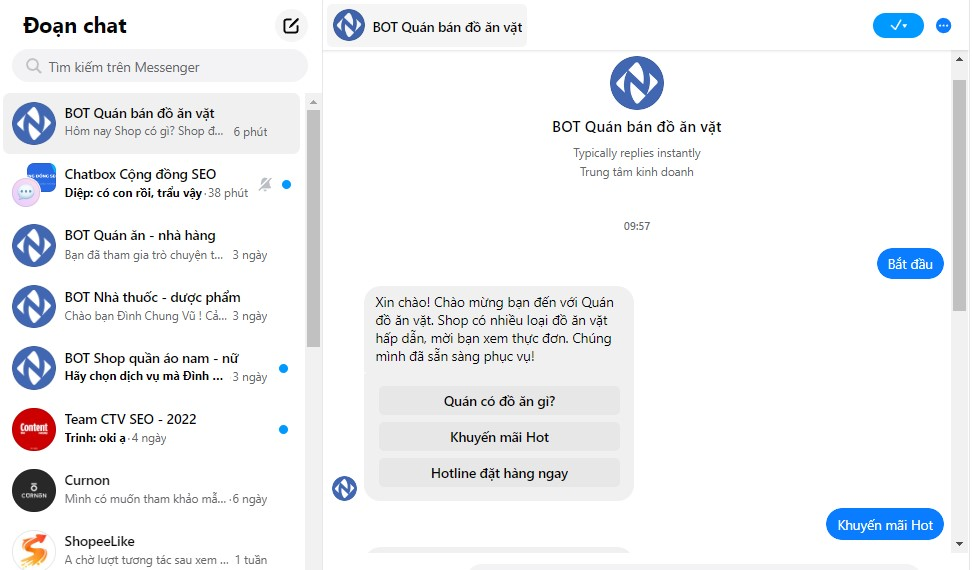
* Closed Domain, Retrieval-Based: Nó là loại chatbot dễ xây dựng nhất. Những chatbot này hoạt động trong một miền rất giới hạn hoặc được định nghĩa trước và truy xuất các phản hồi đã được lập trình sẵn dựa trên đầu vào. Vì chúng hoạt động trong môi trường đóng với các quy tắc cố định, chúng dễ xây dựng nhất. Ví dụ, một chatbot FAQ cơ bản trả lời bằng các câu trả lời từ một cơ sở dữ liệu cố định sẽ thuộc loại này.
* Closed Domain, Generative-Based: Nó là loại chatbot thông minh, khó xây dựng hơn so với loại trên. Những chatbot này tạo ra các phản hồi dựa trên đầu vào nhưng vẫn trong một miền đóng. Chúng yêu cầu các thuật toán phức tạp hơn để tạo ra các phản hồi tự nhiên thay vì chỉ truy xuất các câu trả lời đã được lập trình sẵn. Ví dụ là các chatbot nâng cao hơn sử dụng trong dịch vụ khách hàng.
* Open Domain, Retrieval-Based: Chatbot này được kỳ vọng xử lý các cuộc hội thoại trong miền mở (bất kỳ chủ đề nào) nhưng bị giới hạn bởi các phản hồi dựa trên truy xuất. Vì không thể có câu trả lời được định sẵn cho mọi câu hỏi có thể có trong một miền mở, nên điều này được coi là "không thể thực hiện."
* Open Domain, Generative-Based: Chatbot này khó xây dựng nhất. Những chatbot này hướng tới xử lý các cuộc hội thoại trong miền mở và tạo ra các phản hồi phù hợp cho bất kỳ đầu vào nào. Đây được coi là khó nhất vì nó đòi hỏi trí tuệ nhân tạo tiên tiến với mức độ hiểu biết và lý luận gần giống con người. .

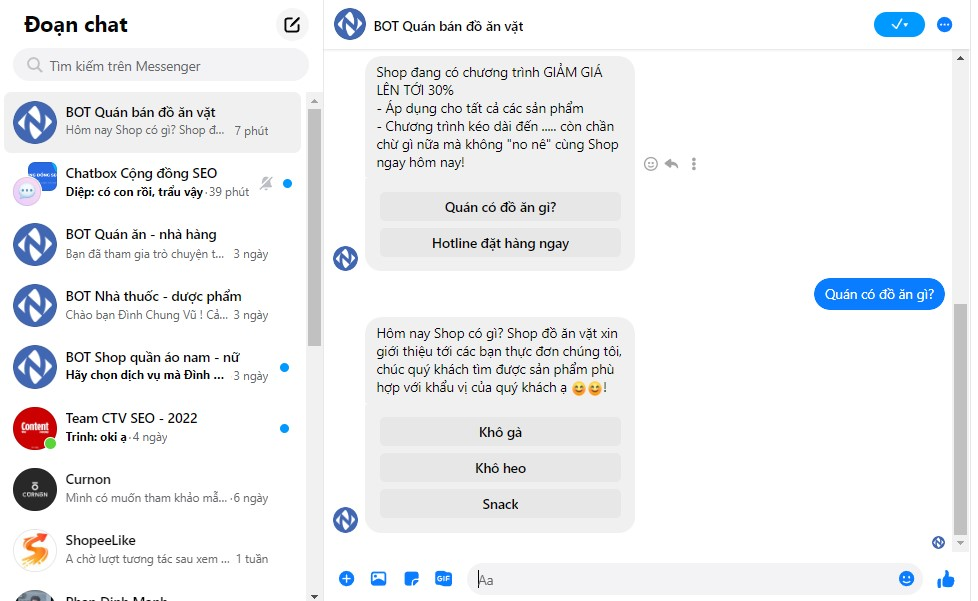
Mỗi cách tiếp cận bài toán đều có hướng giải quyết khác nhau dẫn tới việc sử dụng các kỹ thuật khác nhau. Trong đồ án này, em tập trung xây dựng chatbot thuộc loại miền đóng, cụ thể là bài toán xây dựng chatbot phục vụ việc tuyển sinh tại học viện.

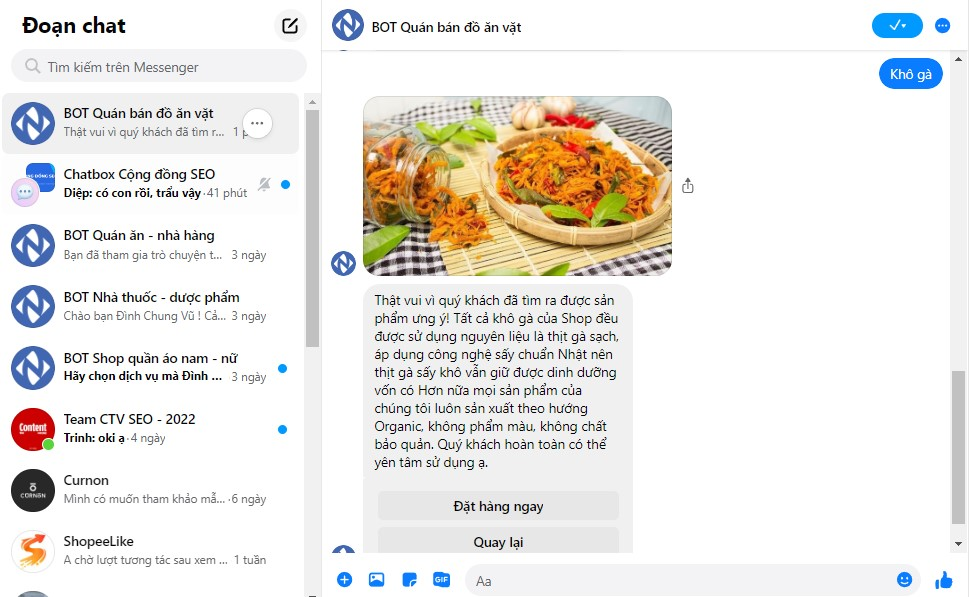
## Các mô hình chatbot tiêu biểu hỗ trợ tiếng Việt hiện nay

### Chatbot dựa trên kịch bản (Script-Based Chatbots)

Chatbot dựa trên kịch bản là một loại chatbot hoạt động dựa trên các kịch bản hoặc quy trình đã được lập trình sẵn để tương tác với người dùng. Loại chatbot này hoạt động theo một chuỗi các bước được xác định trước, điều này giúp nó cung cấp phản hồi và hướng dẫn người dùng theo một trình tự nhất định. Dưới đây là một vài ví dụ về chatbot này.







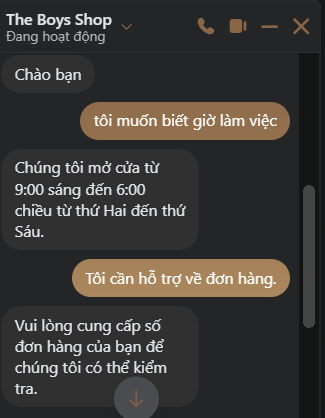
*Hình : Ví dụ về chatbot dựa trên kịch bản.*

Ở ví dụ trên chatbot đã xây dựng sẵn các button tương ứng với yêu cầu của người dùng, chỉ cần chọn click vào đúng nút theo yêu cầu của mình thì chatbot sẽ phản hồi dựa trên yêu cầu đó.

Ưu điểm của chatbot này là xây dựng rất dễ dàng, độ chính xác cao vì người dùng đưa ra yêu cầu dựa trên kịch bản có trước. Tuy nhiên người dùng rất bị động trước những câu hỏi không có trong kịch bản, mà phụ thuộc vào các button chatbot cung cấp sẵn.

### Chatbot dựa trên quy tắc (Rule-Based Chatbots)

Một trong những chatbot thuộc loại này là chatbot nhận dạng từ khóa. Nó là loại chatbot hoạt động dựa trên việc phân tích và nhận diện các từ khóa trong các câu hỏi hoặc lệnh của người dùng. Loại chatbot này sử dụng kỹ thuật đơn giản để phân tích văn bản và đưa ra phản hồi phù hợp dựa trên các từ khóa đã được xác định trước. Dưới đây là ví dụ về chatbot này.



*Hình : Ví dụ về chatbot dựa trên kịch bản.*

Trong đoạn hội thoại trên, khi câu nói của người dùng xuất hiện từ khóa “giờ làm việc” thì chatbot sẽ hiểu và đưa câu trả lời liên quan đến từ khóa đấy.

Ưu điểm của chatbot này là dễ triển khai, cung cấp phản hồi nhanh chóng cho các câu hỏi có cấu trúc rõ ràng và từ khóa dễ nhận diện, chủ động hơn trong việc đưa ra yêu cầu mà không cần phải chọn các button điều hướng như chatbot dựa trên kịch bản, làm cho cuộc trò chuyện diễn ra tự nhiên hơn. Tuy nhiên ngoài những ưu điểm đấy thì chatbot này vẫn có các nhược điểm như khi người dùng sử dụng các từ đồng nghĩa với từ khóa thì chatbot không thể phát hiện được để trả lời phù hợp, không xử lý được các câu hỏi phức tạp cũng như không nắm bắt được ngữ cảnh cuộc trò chuyện.

Hiện nay, loại chatbot dựa trên kịch bản kết hợp chatbot dựa trên quy tắc đang rất phổ biến trong lĩnh vực dịch vụ như bán hàng, đặt hàng, hỗ trợ khách hàng…

### Chatbot dựa trên nền tảng (Platform-Based Chatbots)

Chatbot dựa trên nền tảng là loại chatbot được phát triển và triển khai thông qua các nền tảng chatbot chuyên biệt như Dialogflow của Google, Microsoft Bot Framework, IBM Watson, hoặc Rasa. Các nền tảng này cung cấp các công cụ và giao diện lập trình ứng dụng (API) để nhà phát triển có thể nhanh chóng xây dựng, huấn luyện và quản lý chatbot mà không cần phải xây dựng từ đầu. Các nền tảng này hỗ trợ tích hợp các công nghệ tiên tiến như xử lý ngôn ngữ tự nhiên (NLP) và học máy, giúp chatbot có khả năng hiểu và tương tác với người dùng hiệu quả hơn.

Một trong những điểm mạnh của chatbot dựa trên nền tảng là tính tiện lợi và linh hoạt. Các nền tảng này thường đi kèm với giao diện thân thiện, giúp cho cả những nhà phát triển không chuyên về AI vẫn có thể dễ dàng thiết lập chatbot theo yêu cầu. Các công cụ tích hợp cho phép phát triển chatbot nhanh chóng với khả năng triển khai trên nhiều kênh giao tiếp khác nhau như Facebook Messenger, Slack, WhatsApp, hoặc website. Điều này giúp doanh nghiệp tiết kiệm được thời gian phát triển và tận dụng những tính năng có sẵn của nền tảng để đưa chatbot vào hoạt động nhanh chóng.

Tuy nhiên, nhược điểm của nó là sự phụ thuộc vào khả năng của nền tảng mà được phát triển. Điều này có nghĩa là nếu nền tảng không hỗ trợ một số tính năng hoặc yêu cầu đặc thù, việc tùy chỉnh chatbot có thể bị hạn chế. Ngoài ra, các nền tảng thường cung cấp nhiều tính năng cao cấp nhưng đi kèm với chi phí sử dụng cao, đặc biệt khi cần mở rộng quy mô hoặc yêu cầu khả năng xử lý cao.

Với các lợi thế về tốc độ phát triển và khả năng tích hợp, chatbot dựa trên nền tảng là lựa chọn lý tưởng cho nhiều tổ chức muốn triển khai chatbot nhanh chóng và dễ dàng. Tuy nhiên, việc lựa chọn nền tảng phù hợp sẽ phụ thuộc vào yêu cầu cụ thể của ứng dụng và khả năng tài chính của doanh nghiệp.

### Chatbot dựa trên mô hình ngôn ngữ lớn (Large Language Model Chatbots)

Chatbot dựa trên mô hình ngôn ngữ lớn là loại chatbot sử dụng các mô hình học sâu với khả năng xử lý ngôn ngữ tự nhiên (NLP) mạnh mẽ để tương tác với người dùng. Các mô hình như GPT (Generative Pre-trained Transformer), BERT (Bidirectional Encoder Representations from Transformers), T5 (Text-To-Text Transfer Transformer)... có khả năng hiểu và tạo ra ngôn ngữ một cách tự nhiên và chính xác. Nhờ vào việc được huấn luyện trên khối lượng dữ liệu khổng lồ, những mô hình này có thể nắm bắt ngữ nghĩa, ngữ cảnh, và các sắc thái của ngôn ngữ, từ đó tạo ra các phản hồi hợp lý và mạch lạc trong cuộc trò chuyện.

Một trong những ưu điểm nổi bật của các chatbot dựa trên mô hình ngôn ngữ lớn là khả năng hiểu ngữ nghĩa sâu rộng và tạo ra các phản hồi tự nhiên. Điều này cho phép chatbot có thể xử lý các câu hỏi phức tạp, duy trì mạch trò chuyện một cách liên tục, và đáp ứng linh hoạt với các yêu cầu đa dạng của người dùng. Việc sử dụng các mô hình ngôn ngữ lớn cũng giúp giảm thiểu sự cần thiết phải xây dựng và duy trì các quy tắc và kịch bản phức tạp, vì mô hình đã được huấn luyện để hiểu và phản ứng dựa trên ngữ cảnh của cuộc trò chuyện.

Tuy nhiên, việc triển khai chatbot dựa trên mô hình ngôn ngữ lớn cũng đi kèm với một số thách thức. Một trong những vấn đề chính là yêu cầu về tài nguyên tính toán và bộ nhớ lớn, do các mô hình này thường có kích thước rất lớn và đòi hỏi sức mạnh xử lý cao để hoạt động hiệu quả. Điều này có thể dẫn đến chi phí cao cho việc vận hành và duy trì hệ thống. Hơn nữa, mặc dù các mô hình ngôn ngữ lớn có khả năng tạo ra phản hồi chính xác và tự nhiên, nhưng chúng vẫn có thể gặp phải các vấn đề như sinh ra thông tin sai lệch hoặc không liên quan nếu không được huấn luyện và tinh chỉnh đúng cách.

Mặc dù vậy, các chatbot dựa trên mô hình ngôn ngữ lớn đã chứng tỏ được hiệu quả trong việc nâng cao trải nghiệm người dùng và hỗ trợ các ứng dụng tương tác phức tạp. Chúng cung cấp một giải pháp mạnh mẽ cho việc tạo ra các cuộc trò chuyện tự nhiên và phong phú, đồng thời mở ra cơ hội mới cho việc phát triển các ứng dụng tương tác thông minh.

### Mô hình chatbot mà đồ án nghiên cứu - ***Chatbot dựa trên*** truy xuất ***(***Retrieval***-Based Chatbots)***

Chatbot dựa trên truy xuất là một loại chatbot hoạt động bằng cách tìm kiếm và trả lời các câu hỏi hoặc yêu cầu của người dùng dựa trên một tập hợp các câu hỏi và câu trả lời đã được lưu trữ trước đó. Phương pháp này chủ yếu dựa vào cơ sở dữ liệu hoặc kho lưu trữ thông tin chứa các câu hỏi và câu trả lời mẫu, cùng với các công cụ truy vấn để tìm ra câu trả lời phù hợp nhất cho câu hỏi của người dùng. Khi người dùng gửi một câu hỏi, chatbot sẽ so sánh câu hỏi đó với các câu hỏi trong cơ sở dữ liệu và chọn ra câu trả lời tương ứng mà nó cho là phù hợp nhất.

Một trong những ưu điểm chính của các chatbot dựa trên truy xuất là sự dễ dàng trong việc triển khai và quản lý. Do chúng hoạt động dựa trên các câu trả lời đã được chuẩn bị sẵn, việc phát triển và duy trì các chatbot này không đòi hỏi phải xây dựng các mô hình học sâu phức tạp. Điều này cũng giúp giảm thiểu nguy cơ phát sinh lỗi trong phản hồi, vì các câu trả lời đã được kiểm chứng trước và được lưu trữ một cách có tổ chức.

Tuy nhiên, hạn chế của các chatbot dựa trên truy xuất là chúng có thể gặp khó khăn khi phải xử lý các câu hỏi hoặc yêu cầu không nằm trong cơ sở dữ liệu đã được lập trình sẵn. Khi người dùng đặt câu hỏi ngoài phạm vi các câu hỏi đã chuẩn bị, chatbot có thể không cung cấp được phản hồi thỏa đáng hoặc có thể trả lời sai. Điều này có thể dẫn đến trải nghiệm người dùng không tốt và hạn chế khả năng của chatbot trong việc đáp ứng các yêu cầu đa dạng và phức tạp.

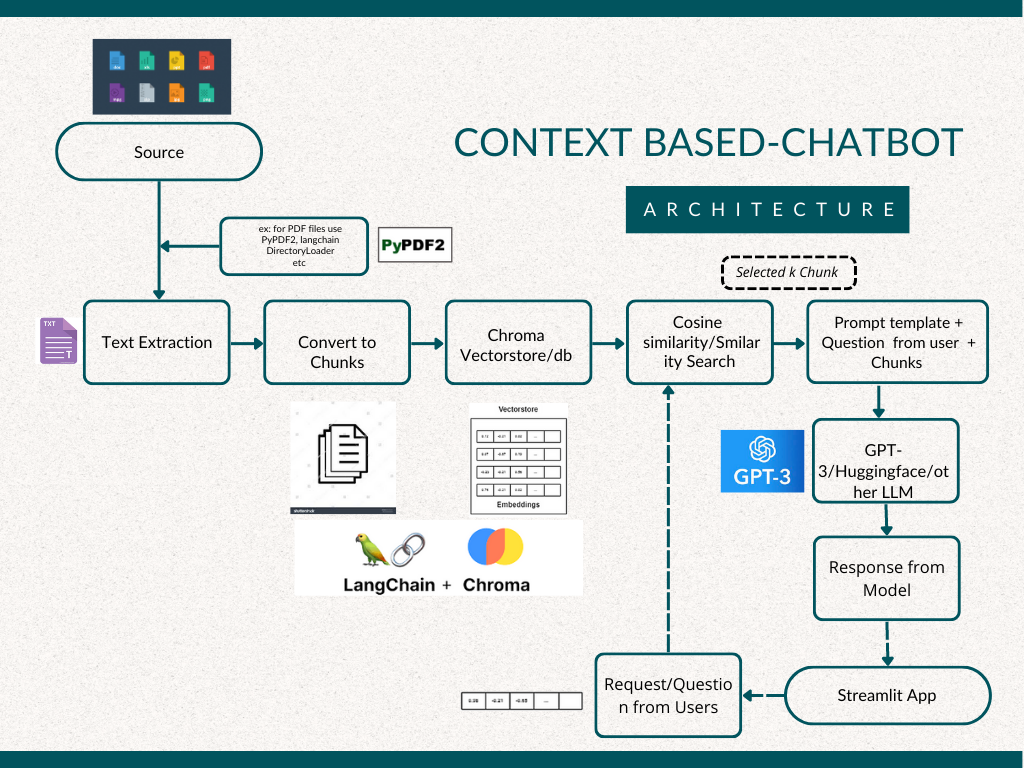
### Chatbot dựa trên RAG (Retrieval-Augmented Generation)

Chatbot dựa trên RAG (Retrieval-Augmented Generation) là loại chatbot kết hợp giữa khả năng truy xuất thông tin và khả năng tạo sinh văn bản. Thay vì chỉ dựa vào một mô hình ngôn ngữ lớn để tạo ra câu trả lời, RAG kết hợp việc truy xuất thông tin từ một cơ sở dữ liệu (retrieval) và khả năng tạo ngôn ngữ của các mô hình AI (generation). Điều này cho phép chatbot đưa ra các phản hồi chính xác hơn, dựa trên kiến thức cập nhật và ngữ cảnh của người dùng.

Ưu điểm nổi bật của chatbot dựa trên RAG là khả năng cung cấp câu trả lời dựa trên các nguồn dữ liệu cập nhật và đa dạng. Thay vì phải dựa hoàn toàn vào kiến thức giới hạn của mô hình ngôn ngữ lớn (LLM), chatbot có thể tìm kiếm và sử dụng thông tin từ các tài liệu, trang web, hoặc cơ sở dữ liệu mới nhất để phản hồi người dùng. Điều này giúp cải thiện tính chính xác và độ tin cậy của câu trả lời, đặc biệt là trong các tình huống đòi hỏi kiến thức chuyên sâu hoặc dữ liệu mới.

Tuy nhiên, hạn chế của chatbot RAG là phức tạp hơn trong việc phát triển và triển khai. Hệ thống cần có sự kết hợp giữa công nghệ truy xuất thông tin hiệu quả và mô hình ngôn ngữ mạnh mẽ, đồng thời phải xử lý tốt vấn đề tối ưu hóa để đảm bảo tốc độ phản hồi nhanh. Mặc dù vậy, với sự phát triển của công nghệ xử lý ngôn ngữ tự nhiên, chatbot RAG ngày càng trở thành giải pháp hiệu quả cho nhiều ứng dụng thực tế, từ hỗ trợ khách hàng đến cung cấp thông tin chuyên ngành.

## Kiến trúc của hệ thống chatbot



*Hình : Kiến trúc hệ thống RAG chatbot.*

* *Source(Nguồn dữ liệu):*

Đây có thể là bất kỳ nguồn tài liệu hoặc dữ liệu văn bản nào mà chatbot sẽ sử dụng để cung cấp các phản hồi dựa trên ngữ cảnh. Các nguồn này có thể bao gồm các file PDF, DOCX hoặc các tài liệu văn bản khác.

* Text Extraction(Trích xuất văn bản):

Sử dụng các công cụ như PyPDF2, PyMuPDF, langchain.DocumentLoaders hoặc các phương pháp tương tự được sử dụng để trích xuất văn bản từ các tài liệu ở định dạng như PDF. Quá trình này là cần thiết để lấy dữ liệu thô và đưa vào hệ thống chatbot.

* Convert to Chunks(Chuyển thành các đoạn nhỏ):

Văn bản đã được trích xuất sẽ được chia thành các phần nhỏ có ý nghĩa (chunks). Những đoạn văn bản này là đơn vị thông tin dễ lưu trữ và tìm kiếm trong cơ sở dữ liệu của chatbot.

* Chroma Vectorstore(Cơ sở dữ liệu Vector Chroma):

Sau khi phân chia thành các đoạn nhỏ, hệ thống sử dụng Chroma làm cơ sở dữ liệu vector để lưu trữ các đoạn nhỏ kèm theo các biểu diễn dưới dạng embedding. LangChain hỗ trợ quá trình này bằng cách kết nối giữa văn bản được trích xuất, các đoạn nhỏ và cơ sở dữ liệu. Embeddings biểu diễn ý nghĩa của văn bản dưới dạng số, giúp việc tìm kiếm tương đồng trở nên hiệu quả.

* Cosine similarity(Độ tương đồng Cosine):

Khi người dùng đặt câu hỏi, hệ thống thực hiện tìm kiếm tương đồng bằng cosine giữa câu hỏi của người dùng và các đoạn văn bản đã lưu trong cơ sở dữ liệu vector. Điều này giúp tìm ra những đoạn thông tin liên quan nhất đến câu hỏi của người dùng.

* Prompt Template + Question from User + Chunks (Mẫu Prompt + câu hỏi từ người dùng + đoạn văn bản):

Các đoạn văn bản được chọn dựa trên sự tương đồng sẽ được kết hợp với câu hỏi của người dùng theo một mẫu prompt. Điều này tạo ra một truy vấn có cấu trúc để đưa vào mô hình ngôn ngữ lớn (LLM) để tạo ra phản hồi phù hợp.

* GPT-3/Huggingface:

Truy vấn có cấu trúc (từ bước trước) sẽ được đưa vào một mô hình ngôn ngữ lớn (như GPT-3 hoặc các mô hình từ Hugging Face) để tạo ra phản hồi dựa trên ngữ cảnh từ các đoạn văn bản đã được chọn và câu hỏi của người dùng.

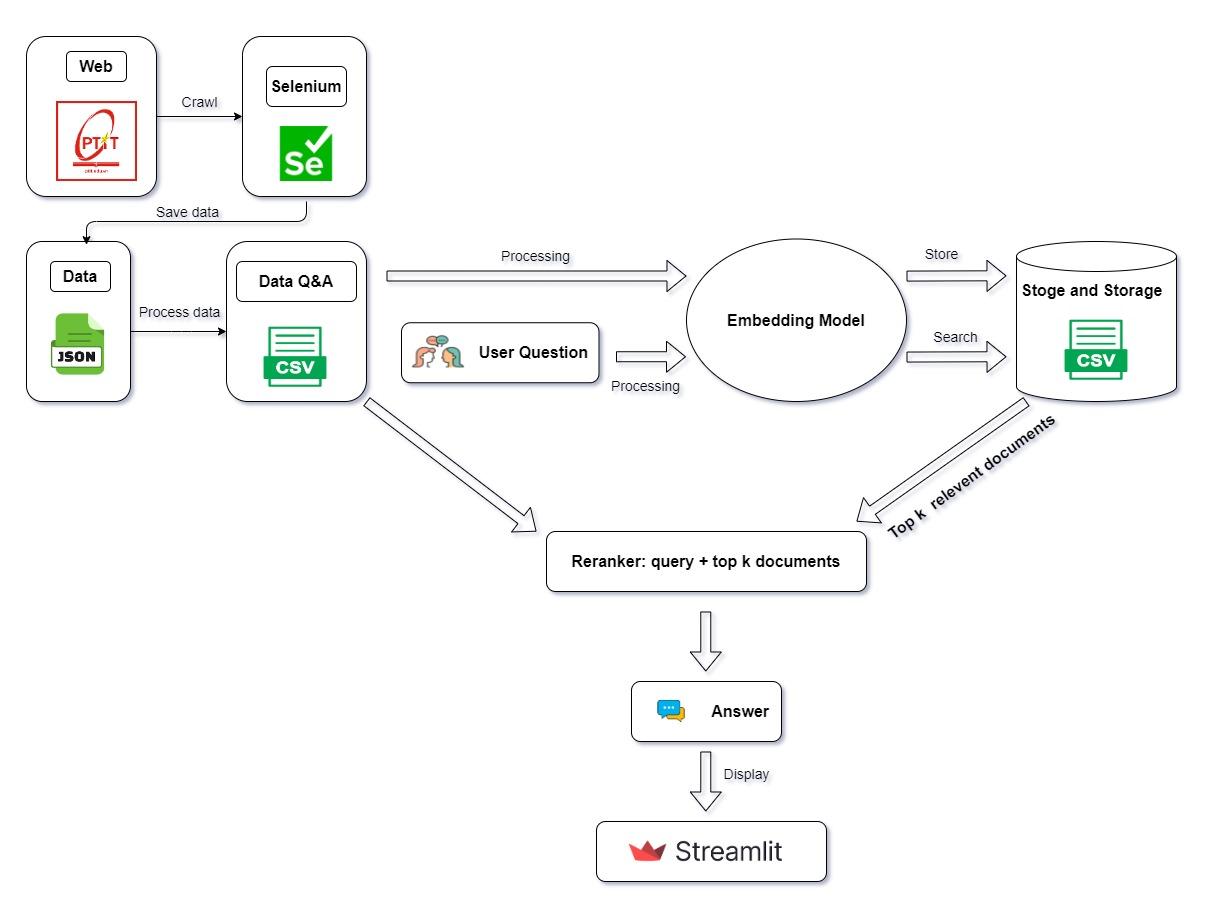
* Response from Model(Phản hồi từ model):

Mô hình LLM sẽ trả về phản hồi được tạo ra, phản hồi này có ngữ cảnh dựa trên các đoạn văn bản đã được truy xuất từ cơ sở dữ liệu.

* Streamlit App(Ứng dụng Streamlit):

Giao diện của chatbot được tạo bằng Streamlit, một framework để xây dựng ứng dụng web. Ứng dụng này tương tác với người dùng, nhận câu hỏi của họ và hiển thị các phản hồi do LLM tạo ra.

Kiến trúc trên cho phép chatbot phản hồi chính xác hơn với các câu trả lời dựa trên ngữ cảnh, bằng cách truy xuất thông tin liên quan từ cơ sở dữ liệu tài liệu trước khi tạo ra phản hồi thông qua các mô hình ngôn ngữ lớn, đặc trưng cho phương pháp RAG.



*Hình : Kiến trúc hệ thống Retrieval-Based Chatbot.*

* Web + Selenium:

Dữ liệu từ trang web liên quan đến tuyển sinh PTIT(Học viện Công nghệ Bưu chính Viễn thông) được thu thập thông qua quá trình crawl dữ liệu sử dụng Selenium, một công cụ tự động hóa trình duyệt.

* Data(Dữ liệu):

Sau khi crawl data, dữ liệu được lưu lại dưới dạng json để thuận lợi cho việc xử lý.

* Data Q&A:

Dữ liệu sẽ được xử lý để chuyển đổi thành các cặp câu hỏi và câu trả lời (Q&A) và được lưu trữ trong file CSV cho mục đích truy xuất sau này. Các câu hỏi trong tập Q&A cũng được xử lý và được nhúng vào mô hình embedding.

* User Question(Câu hỏi người dùng):

Khi người dùng nhập câu hỏi cho chatbot, câu hỏi sẽ được xử lý và được nhúng vào mô hình embedding.

* Embedding model:

Sử dụng mô hình PhoBERT embedding để chuyển câu hỏi của người dùng và câu hỏi trong tập dữ liệu Q&A đã được xử lý thành các vector. Các vector này là các biểu diễn số của câu hỏi và dữ liệu, giúp hệ thống hiểu được ý nghĩa ngữ nghĩa của chúng.

* Store and Storage:

Dữ liệu sau khi được chuyển thành embedding sẽ được lưu trữ trong file CSV và có thể được truy xuất sau này để so khớp với câu hỏi của người dùng. Khi có một truy vấn từ người dùng, hệ thống sẽ thực hiện tìm kiếm các tài liệu tương đồng nhất trong cơ sở dữ liệu và trả về top k tài liệu liên quan nhất.

* Reranker:

Một mô hình xếp hạng lại kết hợp cả truy vấn của người dùng và top k tài liệu để tìm ra kết quả tốt nhất và phù hợp nhất.

* Answer:

Sau khi reranker xác định được tài liệu phù hợp nhất, câu trả lời cuối cùng sẽ được tạo ra và gửi lại cho người dùng.

* Ứng dụng Streamlit:

Streamlit là giao diện mà người dùng sẽ tương tác với chatbot. Các câu hỏi được nhập vào đây và câu trả lời sẽ được hiển thị lại cho người dùng thông qua giao diện này.

Trên đây là kiến trúc của Retrieval-Based Chatbot cũng như là kiến trúc của đồ án chatbot mà em nghiên cứu. Kiến trúc này tối ưu hóa việc xử lý câu hỏi người dùng thông qua mô hình PhoBERT embedding. Bằng cách chuyển đổi câu hỏi và dữ liệu thành các vector số hóa, hệ thống có thể hiểu được ngữ nghĩa của chúng và tìm kiếm các tài liệu liên quan nhất thông qua các phép toán so sánh độ tương đồng. Hơn nữa, quy trình reranking cho phép cải thiện độ chính xác của câu trả lời cuối cùng bằng cách kết hợp giữa truy vấn của người dùng và các tài liệu liên quan nhất. Toàn bộ quá trình này được tích hợp trên giao diện Streamlit, giúp người dùng tương tác một cách dễ dàng và tiện lợi. Kiến trúc này không chỉ phù hợp với hệ thống chatbot hỗ trợ tuyển sinh mà còn có tiềm năng ứng dụng trong nhiều lĩnh vực khác, mang lại hiệu quả cao trong việc truy xuất thông tin và tương tác với người dùng.

## Kết luận chương

Chương này đã trình bày và giới thiệu những kiến thức tổng quan nhất về hệ thống chatbot, phân tích các ưu nhược điểm của các mô hình chatbot hiện nay từ đó định hướng xây dựng mô hình chatbot mà đồ án nghiên cứu và xây dựng.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

1. Chatbot và NLP

2. Word embedding

• Count Vector

• TF-IDF

• GloVe

• Word2vec

• PhoBert

3. Phương pháp tính độ tương đồng

• Khoảng cách Jaro

• Kc Euclide

• Hamming Distance

• Hệ số jaccard

• Độ tương đồng cosine

4. Các phương pháp tách từ trong tiếng việt

• So khớp từ dài nhất (Longest Matching)

• So khớp cực đại (Maximum Matching)

• Mô hình ngôn ngữ (underthesea)

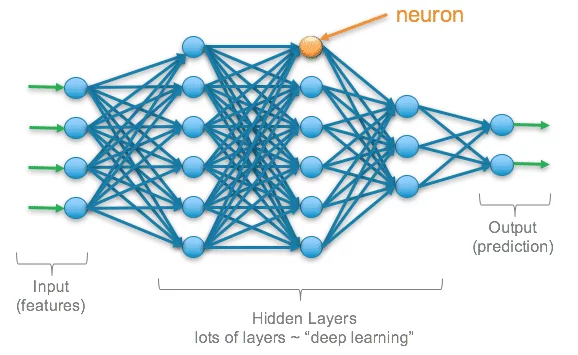
• Phương pháp sử dụng mô hình Markov ẩn

• Phương pháp tách từ sử dụng mô hình xác suất có điều kiện và độ hỗ trợ loạn cực đại

• Phương pháp tách từ sử dụng mô hình Pointwise

## Kiến trúc mạng nơ-ron nhân tạo

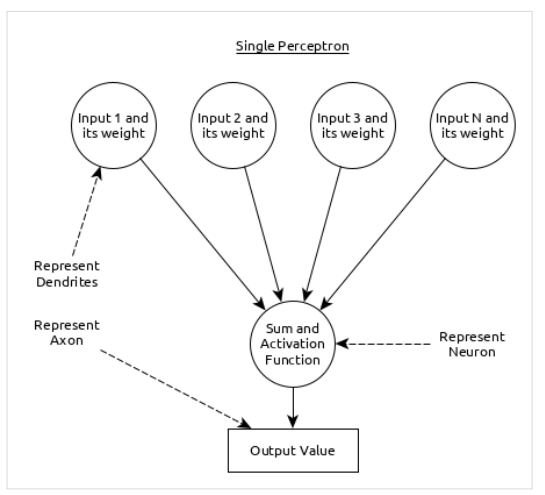
Thuật ngữ "Mạng nơ-ron nhân tạo" bắt nguồn từ mạng nơ-ron sinh học phát triển cấu trúc của não người. Tương tự như não người có các nơ-ron được kết nối với nhau, mạng nơ-ron nhân tạo cũng có các nơ-ron được kết nối với nhau ở nhiều lớp khác nhau của mạng. Các nơ-ron này được gọi là các nút.

**

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

[*https://trituenhantao.io/kien-thuc/xay-dung-mo-hinh-neural-network/*](https://trituenhantao.io/kien-thuc/xay-dung-mo-hinh-neural-network/)

Kiến trúc chung của ANN bao gồm 3 thành phần đó là input layer, hidden layer và output layer. Trong đó, lớp ẩn (hidden layer) gồm các nơ-ron, nhận dữ liệu từ input từ các nơ-ron ở lớp trước đó và chuyển đổi input này cho các lớp xử lý tiếp theo. Quá trình xử lý thông tin trong một nơ-ron như sau.

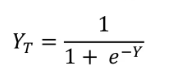
**

*Hình : Quá trình xử lý thông tin trong nơ-ron.*

Mỗi input tương ứng với 1 đặc trưng của dữ liệu. Ví dụ như trong ứng dụng của ngân hàng xem xét có chấp nhận cho khách hàng vay tiền hay không thì mỗi input là 1 đặc trưng của khách hàng như thu nhập, nghề nghiệp, tuổi… Output là kết quả, với bài toán trên, nếu chấp nhận cho khách hàng vay tiền thì output là yes - cho vay hoặc no - không cho vay. Trọng số (weights) là thành phần rất quan trọng trong ANN, nó thể hiện mực độ quan trọng của dữ liệu đầu vào đối với quá trình xử lý thông tin, chuyển đổi dữ liệu từ layer này qua layer khác. Quá trình học của ANN thực chất là quá trình điều chỉnh các trọng số (weights) của các input để có được kết quả mong muốn. Hàm tính tổng (Sum Function) cho phép tính tổng trọng số của tất cả các input và độ lệch bias. Phép tính này được biểu diễn dưới dạng:



Nó xác định tổng trọng số được truyền như một đầu vào cho hàm kích hoạt để tạo ra đầu ra. Các hàm kích hoạt chọn xem nút đó có nên kích hoạt hay không. Chỉ những nút được kích hoạt mới được đưa vào lớp đầu ra. Các nhiều hàm kích hoạt riêng biệt có thể áp dụng riêng cho từng bài toán, một trong những hàm kích hoạt phổ biến trong ANN là sigmoid function.



Kết quả của sigmoid function thuộc khoảng [0,1] nên còn gọi là hàm chuẩn hóa (Normalized Function). Ứng dụng thực tế của mạng nơ-ron rất đa dạng và phong phú, nó được sử dụng trong các bài toán như phân loại hình ảnh, hệ thống giám sát, trợ lý ảo, dịch máy, dự đoán thị trường, phát hiện gian lận, chuẩn đoán bệnh, hệ thống lái xe tự động, phân tích khách hàng và tối ưu quảng cáo…

## Mạng nơ-ron hồi quy RNN

## **Mạng nơ-ron LSTM**

## Word embedding

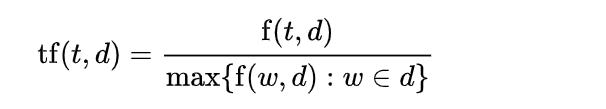
Word embedding là một không gian vector dùng để biểu diễn dữ liệu có khả năng miêu tả được mối liên hệ, sự tương đồng về mặt ngữ nghĩa, văn cảnh(context) của dữ liệu. Không gian này bao gồm nhiều chiều và các từ trong không gian đó mà có cùng văn cảnh hoặc ngữ nghĩa sẽ có vị trí gần nhau. Ví dụ như ta có hai câu : "Hôm nay ăn táo " và "Hôm nay ăn xoài ". Khi ta thực hiện word embedding, "táo" và "xoài" sẽ có vị trí gần nhau trong không gian chúng ta biễu diễn do chúng có vị trị giống nhau trong một câu và đều có ý nghĩa là hoa quả.

### 2.4.2. TF-IDF <https://nguyenvanhieu.vn/tf-idf-la-gi/>

TF-IDF (Term Frequency – Inverse Document Frequency) là 1 kĩ thuật sử dụng trong khai phá dữ liệu văn bản. Trọng số này được sử dụng để đánh giá tầm quan trọng của một từ trong một văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện trong văn bản nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu.

* **TF là gì?**

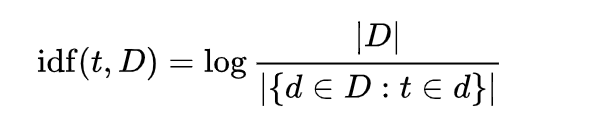
TF: Term Frequency(Tần suất xuất hiện của từ) là số lần từ xuất hiện trong văn bản. Vì các văn bản có thể có độ dài ngắn khác nhau nên một số từ có thể xuất hiện nhiều lần trong một văn bản dài hơn là một văn bản ngắn. Như vậy, term frequency thường được chia cho độ dài văn bản( tổng số từ trong một văn bản).



Trong đó:

* tf(t, d): tần suất xuất hiện của từ t trong văn bản d
* f(t, d): Số lần xuất hiện của từ t trong văn bản d
* max({f(w, d) : w ∈ d}): Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản d
* IDF là gì?

IDF: Inverse Document Frequency(Nghịch đảo tần suất của văn bản), giúp đánh giá tầm quan trọng của một từ. Khi tính toán TF, tất cả các từ được coi như có độ quan trọng bằng nhau. Nhưng một số từ như “is”, “of” và “that” thường xuất hiện rất nhiều lần nhưng độ quan trọng là không cao. Như thế chúng ta cần giảm độ quan trọng của những từ này xuống.



Trong đó:

* idf(t, D): giá trị idf của từ t trong tập văn bản
* |D|: Tổng số văn bản trong tập D
* |{d ∈ D : t ∈ d}|: thể hiện số văn bản trong tập D có chứa từ t.

Cơ số logarit trong công thức này không thay đổi giá trị của 1 từ mà chỉ thu hẹp khoảng giá trị của từ đó. Vì thay đổi cơ số sẽ dẫn đến việc giá trị của các từ thay đổi bởi một số nhất định và tỷ lệ giữa các trọng lượng với nhau sẽ không thay đổi. (nói cách khác, thay đổi cơ số sẽ không ảnh hưởng đến tỷ lệ giữa các giá trị IDF). Tuy nhiên việc thay đổi khoảng giá trị sẽ giúp tỷ lệ giữa IDF và TF tương đồng để dùng cho công thức TF-IDF như bên dưới.

Giá trị TF-IDF:



Những từ có giá trị TF-IDF cao là những từ xuất hiện nhiều trong văn bản này, và xuất hiện ít trong các văn bản khác. Việc này giúp lọc ra những từ phổ biến và giữ lại những từ có giá trị cao (từ khoá của văn bản đó).

IDF có ứng dụng quan trọng trong việc tìm kiếm. Ví dụ, khi người dùng gửi một truy vấn, hệ thống cần biết từ nào là từ người dùng quan tâm nhất. Chẳng hạn: truy vấn của người dùng là "làm thế nào để sửa máy ủi". Sau khi tách từ, chúng ta sẽ có tập các từ: làm, thế nào, để, sửa, máy ủi. Trong các từ này, "máy ủi" sẽ có IDF cao nhất. Hệ thống sẽ lấy ra tất cả các văn bản có chứa từ máy ủi và sau đó mới thực hiện việc đánh giá và so sánh dựa trên toàn bộ câu truy vấn.

### 2.4.3. Word2Vec

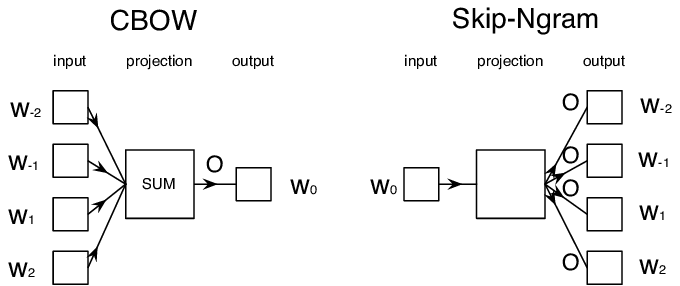
Word2vec là một kỹ thuật trong xử lý ngôn ngữ tự nhiên (NLP) để thu thập các biểu diễn vectơ của từ. Các vectơ này nắm bắt thông tin về nghĩa của từ dựa trên các từ xung quanh. Thuật toán word2vec ước tính các biểu diễn này bằng cách mô hình hóa văn bản trong một ngữ liệu lớn. Sau khi được đào tạo, một mô hình như vậy có thể phát hiện các từ đồng nghĩa hoặc gợi ý các từ bổ sung cho một câu không trọn vẹn. Word2vec được phát triển bởi Tomáš Mikolov và các đồng nghiệp tại Google và được xuất bản vào năm 2013

.

*Hình : Mô hình từ nhúng.*

Có hai kiến ​​trúc Word2Vec chính được sử dụng để biểu diễn phân tán các từ:

* Continuous bag-of-words (CBOW) – Thứ tự của các từ ngữ cảnh không ảnh hưởng đến dự đoán (giả định bag-of-words) (theo Aaron (Ari) Bornstein trên Towards Data Science). Trong kiến ​​trúc continuous skip-gram, mô hình sử dụng từ hiện tại để dự đoán phạm vi xung quanh của các từ ngữ cảnh.
* Continuous skip-gram chú trọng các từ ngữ cảnh ở phạm vi gần. Mỗi vectơ ngữ cảnh được cân nhắc và so sánh độc lập với CBOW.



*Hình : Mô hình CBOW và Skip-Ngram.*

**CBOW:** Dự đoán từ ngữ hiện tại dựa trên ngữ cảnh của các từ trước đó.

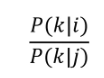
* Cho các từ ngữ cảnh
* Đoán xác suất của 1 từ đích

**Skip-Ngram:** Dự đoán từ các từ xung quanh dựa trên từ ngữ hiện tại.

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

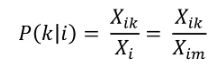
### 2.4.4. GloVe

GloVelà một trong những phương pháp để xây dựng vector từ, nó được xây dựng dựa trên ma trận đồng xảy ra (Co-occurrence Matrix). Nó được phát triển như một dự án nguồn mở tại Stanford[2] và được ra mắt vào năm 2014. Nó được thiết kế như một đối thủ cạnh tranh của word2vec và bài báo gốc đã ghi nhận nhiều cải tiến của GloVe so với word2vec. GloVe dựa trên ý tưởng tính tỉ lệ xác suất.



*Hình : Xác suất từ k xuất hiện trên ngữ cảnh của từ i và j.*

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



*Hình : Công thức tính xác suất xuất hiện từ k trên ngữ cảnh từ i.*

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

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

Ý tưởng của GloVe được tính toán dựa trên sự giống nhau về ngữ nghĩa giữa 2 từ i, j và được xác định thông qua sự tương đồng về ngữ nghĩa giữa các từ k và mỗi từ i,j, các từ k có định nghĩa ngữ nghĩa tốt là những từ khiến giá trị công thức P(k|i)/P(k|j) nằm trong phạm vi [0,1].

Ví dụ: nếu từ i là “melon”, từ j là “pen” và từ k là “fruit” thì công thức P(k|i)/P(k|j) sẽ cho giá trị tiệm cận là 1 vì “fruit” có nghĩa gần với “melon” hơn. Nếu trong trường hợp khác chúng ta thay thế từ k bằng “car” thì giá trị của công thức trên sẽ xấp xỉ bằng 0 vì từ “car” gần như không liên quan gì đến 2 từ “melon” và “pen”.

### 2.4.5. PhoBert

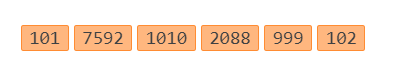
Các mô hình PhoBERT được đào tạo trước là mô hình ngôn ngữ tiên tiến dành cho người Việt Nam (Pho, tức là "Phở", là một món ăn phổ biến ở Việt Nam). Hai phiên bản PhoBERT “base” và “large” là mô hình ngôn ngữ đơn ngữ quy mô lớn đầu tiên được đào tạo trước cho tiếng Việt, nó được dựa trên kiến trúc của BERT “base” và BERT ”large”. Phương pháp đào tạo trước PhoBERT dựa trên RoBERTa giúp tối ưu hóa quy trình đào tạo trước BERT để có hiệu suất mạnh mẽ hơn.

Để biểu diễn dữ liệu đầu vào dạng văn bản, BERT dựa vào 3 loại nhúng riêng biệt: Nhúng mã thông báo (Token Embeddings), Nhúng vị trí (Position Embeddings) và Nhúng loại mã thông báo (Token Type Embeddings).

* **Token Embeddings**

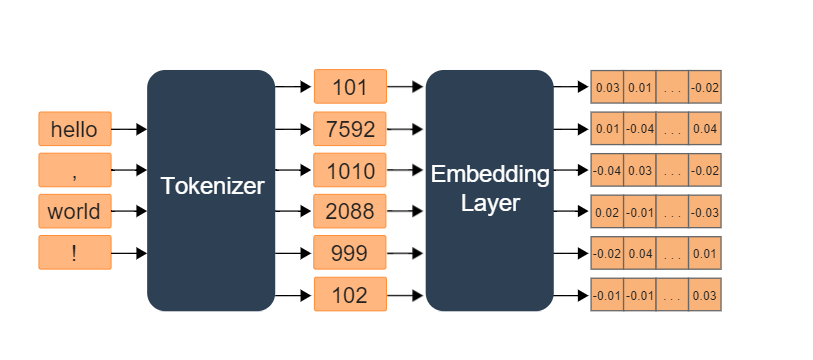
Trước khi một chuỗi văn bản được chuyển đến mô hình BERT, BERT Tokenizer được sử dụng để chuyển đổi dữ liệu đầu vào từ một chuỗi thành danh sách các ID mã thông báo số nguyên, trong đó mỗi ID được ánh xạ trực tiếp đến một từ hoặc một phần của từ trong chuỗi gốc.

Ví dụ: chuỗi "hello, world!" được Tokenizer chuyển đổi thành các ID mã thông báo sau:



*Hình : Chuỗi được mã hóa thành mã thông báo.*

Đối với mỗi ID mã thông báo duy nhất (tức là đối với mỗi trong số 30.522 từ và từ phụ trong từ vựng của Trình mã thông báo BERT), mô hình BERT chứa phần nhúng được đào tạo để thể hiện mã thông báo cụ thể đó. Lớp nhúng trong mô hình chịu trách nhiệm ánh xạ mã thông báo tới các phần nhúng tương ứng của chúng.

*Hình : Hình biểu diễn vector nhúng của chuỗi qua các bước.*

Thuộc tính bert.embeddings của đối tượng mô hình có thể được sử dụng để xem và thực thi lớp nhúng của chính nó. Thuộc tính weight của lớp chứa các phần nhúng mã thông báo (tức là các phần nhúng cho mỗi mã thông báo trong từ vựng của trình mã thông báo BERT).

**>>>** embedding\_layer **=** model**.**bert**.**embeddings

**>>>** print**(**embedding\_layer**.**weight**)**

**<**tf**.**Variable 'tf\_bert\_model/bert/embeddings/word\_embeddings/weight:0' shape**=(30522,** **768)** dtype**=**float32**,** numpy**=**

array**([[-0.01018257,** **-0.06154883,** **-0.02649689,** **...,** **-0.01985357,**

**-0.03720997,** **-0.00975152],**

**[-0.01170495,** **-0.06002603,** **-0.03233192,** **...,** **-0.01681456,**

**-0.04009988,** **-0.0106634** **],**

**[-0.01975381,** **-0.06273633,** **-0.03262176,** **...,** **-0.01650258,**

**-0.04198876,** **-0.00323178],**

**...,**

**[-0.02176224,** **-0.0556396** **,** **-0.01346345,** **...,** **-0.00432698,**

**-0.0151355** **,** **-0.02489496],**

**[-0.04617237,** **-0.05647721,** **-0.00192082,** **...,** **0.01568751,**

**-0.01387033,** **-0.00945213],**

**[** **0.00145601,** **-0.08208051,** **-0.01597912,** **...,** **-0.00811687,**

**-0.04746607,** **0.07527421]],** dtype**=**float32**)>**

Các phần nhúng được trả về dưới dạng ma trận 30522 x 768 hoặc tensor 2 chiều:

* Kích thước đầu tiên của tensor này là kích thước từ vựng của mã thông báo BERT: 30.522
* Số thứ hai là kích thước nhúng, còn được gọi là kích thước ẩn. Đây là số lượng trọng số có thể huấn luyện cho mỗi mã thông báo trong từ vựng. Mô hình BERT ban đầu có kích thước ẩn là 768, nhưng các biến thể khác của BERT đã được huấn luyện với các giá trị nhỏ hơn và lớn hơn của kích thước ẩn.
* **Position Embeddings**

Ngoài các phần nhúng mã thông báo được mô tả cho đến nay, BERT còn dựa vào các phần nhúng vị trí. Trong khi nhúng mã thông báo được sử dụng để thể hiện từng từ hoặc từ phụ có thể được cung cấp cho mô hình, thì nhúng vị trí thể hiện vị trí của từng mã thông báo trong chuỗi đầu vào.

****

*Hình : Hình chuỗi được đánh ký hiệu vị trí position.*

Thuộc tính position\_embeddings của lớp nhúng được sử dụng để truy cập vào Vị trí nhúng:

**>>>** print**(**embedding\_layer**.**position\_embeddings**)**

**<**tf**.**Variable 'tf\_bert\_model/bert/embeddings/position\_embeddings/embeddings:0' shape**=(512,** **768)** dtype**=**float32**,** numpy**=**

array**([[** **1.7505383e-02,** **-2.5631009e-02,** **-3.6641564e-02,** **...,**

**3.3437202e-05,** **6.8312453e-04,** **1.5441139e-02],**

**[** **7.7580423e-03,** **2.2613001e-03,** **-1.9444324e-02,** **...,**

**2.8909724e-02,** **2.9752752e-02,** **-5.3246655e-03],**

**[-1.1287465e-02,** **-1.9644140e-03,** **-1.1572698e-02,** **...,**

**1.4907907e-02,** **1.8740905e-02,** **-7.3139993e-03],**

**...,**

**[** **1.7417932e-02,** **3.4902694e-03,** **-9.5621375e-03,** **...,**

**2.9599327e-03,** **4.3434653e-04,** **-2.6948910e-02],**

**[** **2.1686664e-02,** **-6.0216337e-03,** **1.4735642e-02,** **...,**

**-5.6118402e-03,** **-1.2589799e-02,** **-2.8084971e-02],**

**[** **2.6412839e-03,** **-2.3297865e-02,** **5.4921862e-03,** **...,**

**1.7536936e-02,** **2.7549751e-02,** **-7.7655964e-02]],** dtype**=**float32**)>**

Trong khi có 30.522 Token Embedding khác nhau, chỉ có 512 Position Embedding khác nhau. Điều này là do chuỗi đầu vào lớn nhất được mô hình BERT chấp nhận dài 512 token.

* **Token Type Embeddings**

Kiểu nhúng cuối cùng được BERT sử dụng là nhúng loại mã thông báo, còn được gọi là Nhúng phân đoạn (Segment Embedding) trong BERT paper. Một trong những nhiệm vụ mà BERT ban đầu được đào tạo để giải quyết là Dự đoán câu tiếp theo. Nghĩa là, cho hai câu A và B, BERT được huấn luyện để xác định xem B có tuân theo A một cách logic hay không.

Thuộc tính token\_type\_embeddings của đối tượng lớp nhúng được sử dụng để truy cập vào phần nhúng loại mã thông báo:

**>>>** print**(**embedding\_layer**.**token\_type\_embeddings**)**

**>>>** **<**tf**.**Variable 'tf\_bert\_model/bert/embeddings/token\_type\_embeddings/embeddings:0' shape**=(2,** **768)** dtype**=**float32**,** numpy**=**

array**([[** **0.00043164,** **0.01098826,** **0.00370439,** **...,** **-0.00661185,**

**-0.00336983,** **-0.00864201],**

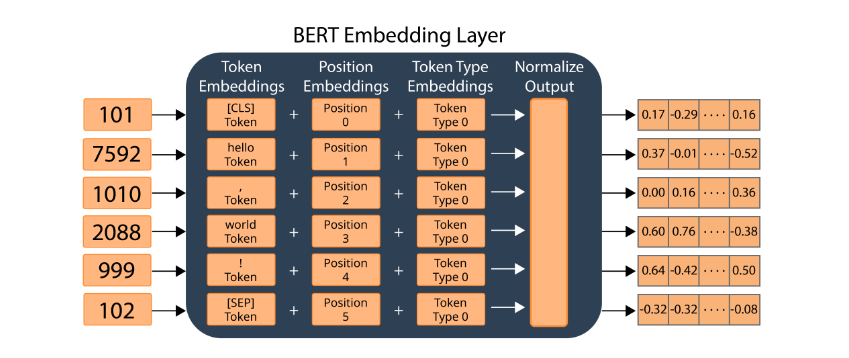
**[** **0.00111319,** **-0.00299169,** **-0.00317028,** **...,** **0.00474542,**

**-0.0052443** **,** **-0.01121742]],** dtype**=**float32**)>**

Chỉ có hai Nhúng loại mã thông báo khác nhau: một được sử dụng để biểu diễn các mã thông báo trong câu A và một được sử dụng để biểu diễn các mã thông báo trong câu B.

**BERT Embedding Layer**

Ba vectơ nhúng được thêm vào cùng nhau biểu thị biểu diễn mã thông báo ban đầu dưới dạng hàm của ba phần thông tin này. Sau khi nhúng, biểu diễn vectơ được chuẩn hóa bằng thao tác LayerNorm, xuất ra vectơ 768 chiều cho mỗi mã thông báo đầu vào. Sau đó, các vectơ biểu diễn được truyền tiếp qua 12 khối mã hóa Transformer và được giải mã trở lại không gian từ vựng 30.000 chiều bằng cách sử dụng lớp biến đổi affine cơ bản. Hình minh họa bên dưới cho thấy cách Lớp nhúng BERT tính toán việc nhúng cho chuỗi "hello, world!"



*Hình : Hình mô tả lớp nhúng của mô hình BERT.*

Ngoài ra, BERT còn được trang bị cơ chế attention, nó có vai trò rất quan trọng trong việc nắm bắt mối quan hệ giữa các từ trong câu mà không phụ thuộc vào khoảng cách của chúng trong chuỗi. Nó còn giúp BERT hiểu được ngữ nghĩa phức tạp cũng như nắm bắt bối cảnh toàn cục (ví dụ như câu "nhà tôi nuôi 20 con gà" và "bạn chơi game này gà quá". Từ "gà" có các vector embeddings khác nhau vì cơ chế attention của BERT giúp nó hiểu được ngữ cảnh của từng câu). Cơ chế này là một phần quan trọng giúp BERT đạt được hiệu quả cao trong việc xử lý ngôn ngữ tự nhiên, từ các tác vụ như phân loại văn bản, trả lời câu hỏi cho đến phân tích ngữ nghĩa.

PhoBERT có kiến trúc tương tự như BERT, nhưng được điều chỉnh đặc biệt để phù hợp với ngôn ngữ tiếng Việt. PhoBERT được huấn luyện trên một tập dữ liệu lớn gồm 20GB văn bản tiếng Việt, bao gồm các bài viết từ Wikipedia tiếng Việt, các bài báo, và nhiều nguồn dữ liệu phong phú khác. Sự khác biệt chính giữa PhoBERT và BERT là PhoBERT tối ưu hóa cho các đặc trưng ngôn ngữ tiếng Việt, giúp mô hình xử lý tốt hơn các tác vụ ngôn ngữ tự nhiên liên quan đến tiếng Việt. Với những lý do trên, em đã sử dụng nhúng các từ bằng PhoBERT model cho đồ án của mình.

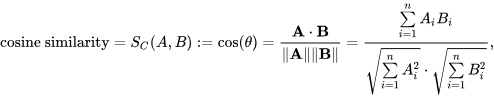
## Các phương pháp tính toán độ tương đồng

### 2.5.1. Độ tương đồng cosine

Độ tương đồng cosine là thước đo độ tương tự giữa hai vectơ khác 0 được áp dụng rộng rãi trong nhiều ứng dụng phân tích dữ liệu và học máy. Nó thực sự đo cosin của góc giữa hai vectơ. Kết quả là, một ý tưởng được đưa ra về khoảng cách hai vectơ bất kể độ lớn của chúng. Nó có thể được sử dụng phổ biến trong các nhiệm vụ phân tích văn bản, chẳng hạn như so sánh sự giống nhau giữa các tài liệu, truy vấn tìm kiếm và thậm chí cả hệ thống đề xuất để có thể khớp với sở thích của người dùng.

**Độ tương đồng cosine là gì?**

Độ tương đồng cosine là một thước đo hữu ích trong việc xác định mức độ giống nhau của các đối tượng dữ liệu bất kể kích thước của chúng. Trong độ tương đồng cosine, các đối tượng dữ liệu trong tập dữ liệu được coi là một vectơ. Công thức tìm độ tương đồng cosin giữa hai vectơ là



Trong đó 𝐴i và 𝐵i lần lượt là các 𝑖 thành phần thứ của vectơ 𝐴 và 𝐵 .

Độ tương đồng có kết quả được dao động từ -1 có nghĩa là hoàn toàn đối lập, đến 1 có nghĩa là hoàn toàn giống nhau, với 0 biểu thị tính trực giao hoặc không tương quan, trong khi các giá trị ở giữa biểu thị độ tương tự hoặc độ khác nhau trung gian.

Trong so khớp văn bản, các vectơ thuộc tính A và B thường là các vectơ tần số thuật ngữ của tài liệu. Độ tương tự cosine có thể được coi là một phương pháp chuẩn hóa độ dài tài liệu trong quá trình so sánh. Trong trường hợp truy xuất thông tin, độ tương tự cosine của hai tài liệu sẽ nằm trong khoảng từ [0,1], vì tần số thuật ngữ không thể âm. Điều này vẫn đúng khi sử dụng trọng số TF-IDF. Góc giữa hai vectơ tần số không được lớn hơn 90°.

Ví dụ, hãy xem xét độ tương đồng giữa 2 vector x và y sử dụng độ tương đồng cosine với vector x có giá trị x = {4,1,2,6} và y = {1,0,0,0}.

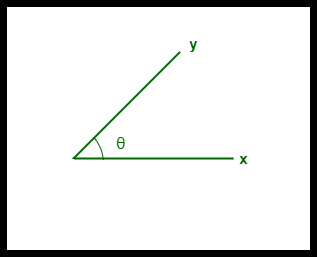
Sc(x,y) = (x \* y) /( ||x|| ||y|| )

= ( 4\*1 + 1\*0 + 2\*0 + 6\*0 )/ (√(42+12+2+262)\*√(12+02+0+202))

= 4 / (7.55\*1) = 0.53

Sự khác nhau giữa 2 vector x và y được tính bởi công thức:

Dc(x,y) = 1 - Sc(x,y) = 1 - 0.53 = 0.47



*Hình : Hình biểu diễn độ tương đồng cosine giữa 2 vector.*

* Độ tương tự cosine giữa hai vectơ được đo bằng góc 'θ'.
* Nếu θ = 0°, vectơ ‘x’ và ‘y’ trùng nhau, do đó chứng tỏ chúng giống nhau.
* Nếu θ = 90°, vectơ ‘x’ và ‘y’ là không giống nhau.

Ưu điểm:

* Độ tương đồng cosin có lợi vì ngay cả khi hai đối tượng dữ liệu tương tự cách xa nhau theo khoảng cách Euclidean do kích thước, chúng vẫn có thể có góc nhỏ hơn giữa chúng. Góc càng nhỏ, độ tương đồng càng cao.
* Khi biểu diễn trên không gian đa chiều, độ tương đồng cosin sẽ nắm bắt được hướng (góc) của các đối tượng dữ liệu chứ không phải độ lớn.

Nhược điểm:

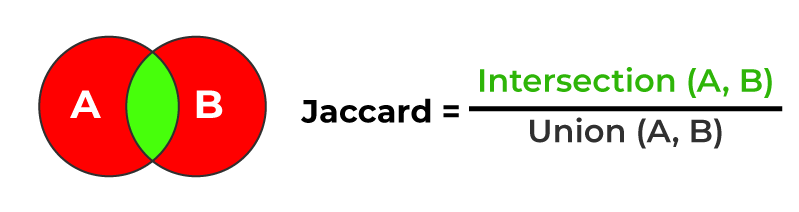
* Nhạy cảm với dữ liệu thưa thớt: Độ tương đồng cosine có thể không hiệu quả khi áp dụng cho dữ liệu thưa thớt trong đó nhiều thành phần của nó bằng không trong các vectơ.
* Không tính đến sự khác biệt tuyệt đối: Độ tương đồng cosin chỉ xem xét góc giữa các vectơ chứ không phải độ lớn của chúng; do đó, điều này có thể bỏ qua sự khác biệt về độ lớn, trong một số bối cảnh nhất định, có thể có liên quan.
* Tính đối xứng: Độ tương tự cosin là đối xứng, nghĩa là nó không thể phân biệt được thứ tự so sánh. Đối với một số tác vụ, điều này có thể không mong muốn vì tính định hướng có thể có liên quan.
* Không áp dụng cho các giá trị âm: Độ tương đồng cosin thường không áp dụng được trong các tập dữ liệu chứa giá trị âm vì có thể thu được kết quả gây hiểu lầm hoặc việc giải thích góc giữa các vectơ trở nên có vấn đề.

### 2.5.2. Độ tương đồng jaccard

**Độ tương đồng jaccard là gì?**

Độ tương đồng Jaccard hay còn gọi là chỉ số Jaccard, là một thống kê để đo lường mức độ tương tự giữa hai bộ dữ liệu. Nó được đo bằng kích thước giao (intersection) của hai tập hợp chia cho kích thước hợp (union) của chúng.

Ví dụ: Cho hai bộ A và B, Độ tương đồng jaccard được tính bằng



*Hình : Hình biểu diễn độ tương đồng jaccard*

Hay công thức:

**

Trong đó:

*  là số lượng phần tử (kích thước) của giao hai tập hợp A và B.
*  là số lượng phần tử (kích thước) của hợp hai tập hợp A và B.

Các giá trị của nó nằm trong khoảng từ 0 đến 1. Trong đó 0 nghĩa là không có sự tương đồng và các giá trị càng gần 1 nghĩa là độ tương tự càng tăng. 1 nghĩa là các tập dữ liệu giống nhau.

Ví dụ, cho 2 tập hợp A và B với các giá trị của A = {1,2,3,6,7} và B = {1,2,5,8,9}

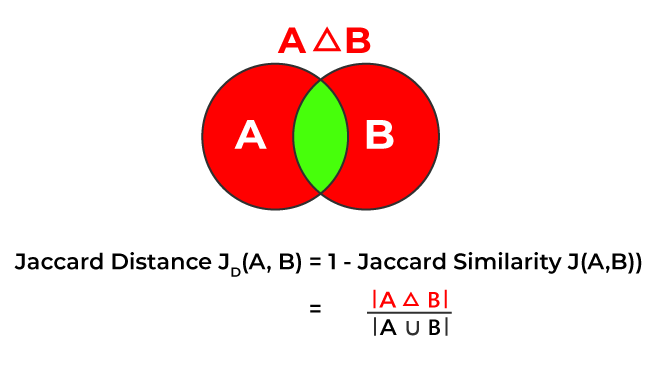
 = {1,2}

 = {1,2,3,5,6,7,8,9}

J(A,B) = 2/8 = 0.25

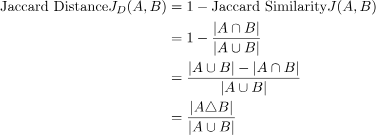
**Khoảng cách Jaccard**

Khoảng cách jaccard là thước đo mức độ khác nhau của hai bộ. Tức là không giống như hệ số jaccard xác định mức độ giống nhau của hai bộ. Khoảng cách jaccard được tính bằng cách lấy 1 trừ hệ số jaccard.



*Hình : Hình biểu diễn khoảng cách jaccard*

Hay công thức:



Trong đó:

*  là số lượng phần tử (kích thước) của giao hai tập hợp A và B.
*  là số lượng phần tử (kích thước) của hợp hai tập hợp A và B.
*  biểu diễn số lượng (kích thước) của hiệu đối xứng của các tập hợp (A) và (B), chứa các phần tử nằm trong một trong hai tập hợp nhưng không nằm trong giao của chúng.

Ví dụ, cho 2 tập hợp A và B với các giá trị của A = {1,2,3,6,7} và B = {1,2,5,8,9}

 = {1,2}

 = {1,2,3,5,6,7,8,9}

J(A,B) = 2/8 = 0.25

JD(A,B) = 1 - J(A,B) = 0.75

Ưu điểm:

* Đơn giản và dễ tính toán: Hệ số jaccard sử dụng các phép toán cơ bản trên tập hợp, giúp dễ dàng hiểu và triển khai, đặc biệt trong việc so sánh các tập hợp từ.

## Kết luận chương

Chương này giới thiệu một số kiến thức nền tảng về mạng nơ-ron nhân tạo, cách thức hoạt động của mạng nơ-ron và một số kỹ thuật được sử dụng trong việc xử lý ngôn ngữ tự nhiên nói riêng hay xây dựng chatbot nói chung.

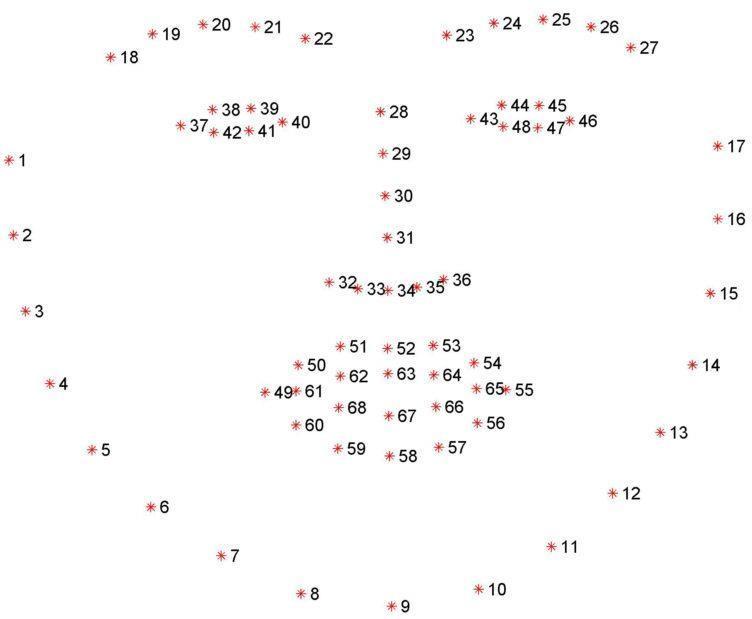
## Bộ căn chỉnh khuôn mặt (Face Alignment)

Sau khi phát hiện và xác định được vị trí khuôn mặt, vùng khuôn mặt được tách ra từ ảnh có thể ở các trạng thái khác nhau, các góc độ khác nhau. Có những khuôn mặt bị chéo và cũng có thể bị lệch do bước detect chưa chính xác trong việc lấy ra khung hình chuẩn của mặt. Vì thế việc căn chỉnh lại khuôn mặt là cần thiết để nâng cao độ chính xác khi nhận diện khuôn mặt.

Để làm được điều này, chúng ta cần xác định được những điểm đặc biệt trên khuôn mặt như mắt, mũi, ... từ đó áp dụng các phương pháp biến đổi hình học như: *Euclidean Transformation, Similarity Transformation, Affine Transformation, Projective Transformation* để khuôn mặt trở về theo quy chuẩn.

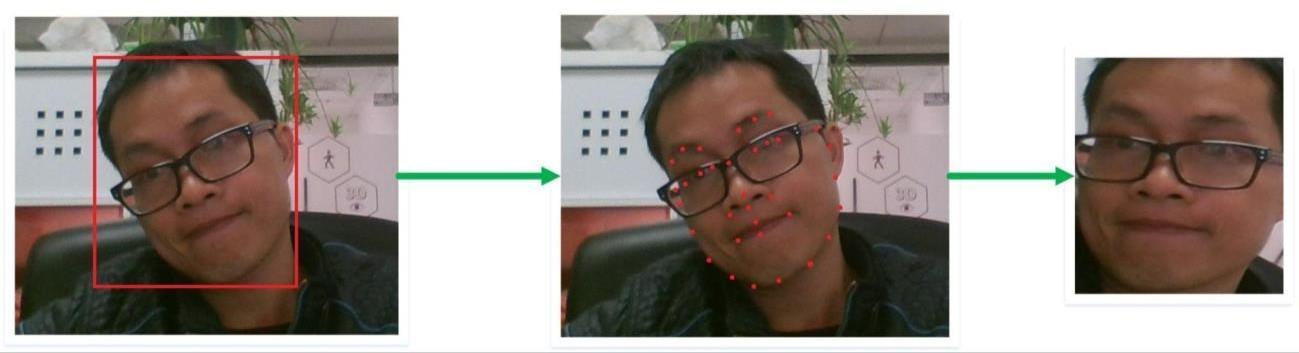


*Hình 2.11. Các phép biến đổi hình học*

Để làm được điều này cần xác định được cấu trúc khuôn mặt. Có rất nhiều kiểu cấu trúc khuôn mặt khác nhau nhưng về cơ bản, chúng ta sẽ xác định các phần miệng, lông mày phải, lông mày trái, mắt phải, mắt trái, mũi, quai hàm. Có rất nhiều thuật toán cho vấn đề này, trong đó có bộ xác định *facial landmark* của thư viện *Dlib* được huấn luyện trên bộ dữ liệu iBUG 300-W. Bộ xác định facial landmark này sẽ xác định 68 điểm chính theo tọa độ (x, y) cấu tạo nên khuôn mặt người.

*Hình 2.12. Ví dụ các điểm facial landmark*

Việc xác định được càng nhiều điểm landmark thì khuôn mặt người càng cụ thể, rõ ràng, chính xác hơn. Từ đó căn chỉnh lại khuôn mặt chuẩn hơn.



*Hình 2.13. Ví dụ alignment face*

## Bộ chống giả mạo khuôn mặt (Face Anti-spoofing)

Phát hiện giả mạo khuôn mặt là một bước quan trọng trong các hệ thống nhận dạng khuôn mặt. Việc hình ảnh hoặc video chứa khuôn mặt của ai đó trên các trang web hay mạng xã hội khiến cho các hệ thống sinh trắc học nhận diện dựa trên khuôn mặt rất dễ bị tấn công khi kẻ tấn công thay vì đưa mặt mình để hệ thống nhận diện thì sử dụng hình ảnh khuôn mặt đã được in trên giấy hoặc xuất hiện trong video quay sẵn của một người có trong hệ thống.



*Hình 2.14. Ví dụ các cách tấn công giả mạo trong các hệ thống bảo mật*

***Một số phương pháp chính cho bài toán chống tấn công giả mạo khuôn mặt:***

* + - *Phương pháp dựa trên các tác động của con người*

Phương pháp này chủ yếu dựa trên những hành động, cử chỉ tự nhiên của người dùng hoặc do hệ thống yêu cầu. Trong đó có 2 phương pháp chính là phát hiện nháy mắt và thử thách - phản hồi.

* + - *Phương pháp phát hiện nháy mắt*

Nháy mắt là hành động tự nhiên của con người, do đó hệ thống sẽ thực hiện quay kiểm tra một chuỗi hình ảnh chứa khuôn mặt có nháy mắt hay không, nếu có nháy mắt thì khuôn mặt là thật và ngược lại, nếu không có nháy mắt thì khuôn mặt là giả mạo.

Phương pháp này dễ triển khai trên các thiết bị, chi phí thực hiện thấp, thuận tiện với người dùng do nháy mắt là hoạt động hoàn toàn tự nhiên của con người, chống tấn công giả mạo bằng hình ảnh in sẵn. Tuy nhiên, phương pháp này có một số hạn chế lớn là không thể phát hiện chống tấn công giả mạo bằng video hoặc sử dụng các sản phẩm 3D.

* + - *Phương pháp thử thách - phản hồi*

Hệ thống sẽ yêu cầu người dùng thực hiện một hoặc một số thao tác, hành động nào đó trên khuôn mặt (thử thách) như: quay mặt sang trái, phải, trên, xuống, nhắm mắt, cười, ..., các hành động này được yêu cầu ngẫu nhiên, không theo thứ tự có sẵn và số lượng để tăng tính bảo mật, sau đó người dùng phải thực hiện (phản hồi) để hệ thống kiểm tra.

Phương pháp thử thách - phản hồi đã khắc phục được nhược điểm lớn nhất của phương pháp phát hiện nháy mắt là người dùng sử dụng video đã quay sẵn. Tuy nhiên, nhược điểm lớn là người dùng có những trải nghiệm không tốt khi phải thực hiện nhiều thao tác mới được nhận diện. Nếu thực hiện sai thao tác nào thì sẽ phải làm lại từ đầu và rất khó cho những người khiếm thị.

* + - *Phương pháp dựa trên các đặc trưng của khuôn mặt*

Phương pháp này chủ yếu dựa trên những đặc trưng của khuôn mặt mà hệ thống trích xuất được sau đó sử dụng các thuật toán phân loại để phân biệt ảnh đầu vào là khuôn mặt thật hay giả mạo. Có 2 phương pháp chính là dùng Local Binary Pattern (LBP) và Support Vector Machine (SVM) và phương pháp dùng mạng nơ- ron tích chập.

* + - * *Phương pháp sử dụng LBP và SVM*

LBP (Local Binary Pattern) là một phương pháp trích rút đặc trưng của ảnh khuôn mặt thành các vector đặc trưng. Các vector đặc trưng sau đó có thể dùng để đưa vào các mô hình học máy để huấn luyện và phân loại. SVM (Support Vector Machine) là một thuật toán học máy nhận đầu vào là các dữ liệu đó để phân chia chúng thành các nhóm riêng biệt.

Phương pháp sử dụng LBP và SVM là phương pháp đơn giản, dễ triển khai trên các thiết bị. Tuy nhiên các đặc trưng mà LBP trích rút được thường bị ảnh hưởng bởi các điều kiện khác nhau về ánh sáng, môi trường, hướng khuôn mặt, nhiễu, ... làm giảm khả năng phân loại.

* + - * *Phương pháp sử dụng mạng nơ-ron tích chập*

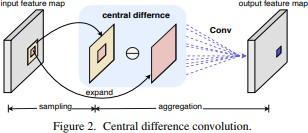
Mạng nơ-ron tích chập như đã trình bày phía trên là một giải pháp tốt nhất cho các bài toán về “Thị giác máy tính” hiện nay. Mạng nơ-ron tích chập chủ yếu sử dụng các lớp tích chập chồng lên nhau, lên bức ảnh khuôn mặt đầu vào nhắm trích xuất các đặc trưng, thông tin trừu tượng tốt hơn về ảnh để phân loại tốt.

Phương pháp sử dụng mạng nơ-ron tích chập phân loại trên ảnh có ưu điểm lớn nhất là dễ triển khai, trích xuất được nhiều đặc trưng của ảnh hơn và không bị ảnh hưởng nhiều bởi các yếu tố như ánh sáng, nhiễu, Tuy nhiên phương pháp này

đòi hỏi nhiều dữ liệu để huấn luyện mạng và đôi khi xảy ra hiện tượng Overfitting (kết quả phân biệt tốt trên bộ dữ liệu huấn luyện nhưng lại không tốt trên các bộ dữ liệu khác).

***Một số nghiên cứu mạng CNN cho bài toán chống giả mạo khuôn mặt:***

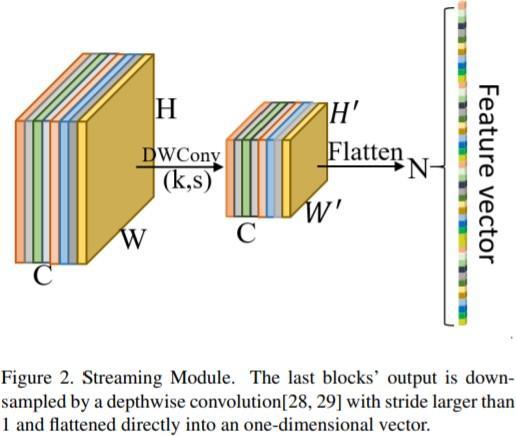
*Nghiên cứu chống giả mạo khuôn mặt sử dụng CDC:*

*Hình 2.15. Cấu trúc Central Different Convolution [5]*

Ý tưởng chính của phương pháp này là sử dụng Central Different Convolution (CDC) thay vì chỉ sử dụng các Convolution bình thường để phát hiện ra những đặc điểm nhất quán trong các bức ảnh giả mạo.

*Nghiên cứu chống giả mạo khuôn mặt FeatherNet:*



*Hình 2.16. Cấu trúc Streaming Module [6]*

Phương pháp này sử dụng các khối Streaming Module, các khối này sử dụng Depthwise Convolution sau đó làm phẳng trực tuyến (tọa độ của phần tử thứ n trong feature vector là tọa độ của phần tử (y, x) trong kênh thứ m.



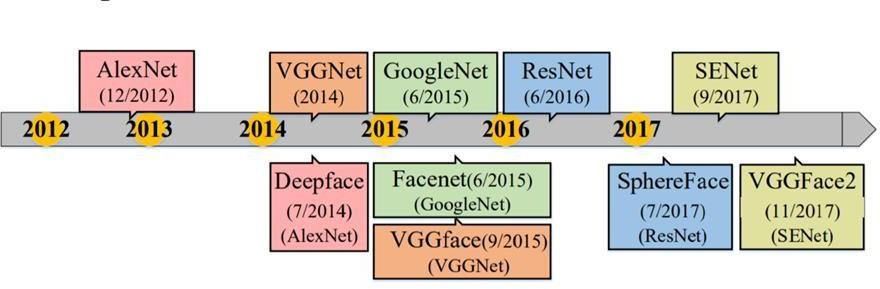
Depthwise Convolution là một loại tích chập trong đó nó chia features map đầu vào thành các group có số channel cố định bằng 1 để thực hiện nhân chập rồi xếp chồng lên nhau để được kết quả đầu ra. Với phương pháp đó, Depthwise Convolution giúp giảm kích thước mô hình và giảm độ phức tạp tính toán.

Làm phẳng trực tuyến giúp khắc phục điểm yếu của Global Average Pooling (GPA - làm giảm kích thước feature map, coi vị trí các điểm ảnh có tầm quan trọng như nhau và điều đó không tốt cho các bài toán về khuôn mặt).

* + - *Phương pháp 3D face shape*

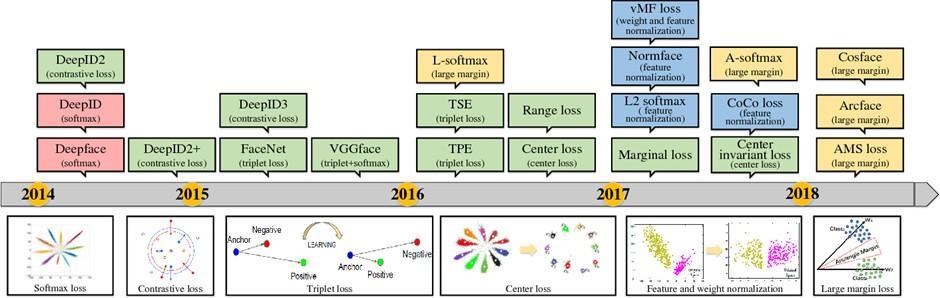
Phương pháp 3D face shape là phương pháp sử dụng các thiết bị cảm biến chiều sâu và cảm biến nhiệt để hỗ trợ phân biệt đầu vào là ảnh thật hay ảnh giả vì cơ thể người sẽ có mức nhiệt cụ thể là khoảng 37℃. Phương pháp này có ưu điểm là tốc độ nhận diện nhanh, chống được các trường hợp giả mạo như dùng ảnh, hoặc video có sẵn, .... Tuy nhiên phương pháp này tốn kém rất nhiều chi phí và không triển khai được trên mobile.

## Bộ trích xuất đặc trưng khuôn mặt (Face Feature Extraction)

Sau khi căn chỉnh khuôn mặt, bước tiếp theo là trích chọn đặc trưng của khuôn mặt phục vụ cho nhận diện khuôn mặt. Trích xuất đặc trưng của khuôn mặt là một bước rất quan trọng trong các công nghệ như face tracking, nhận dạng cảm xúc khuôn mặt hay nhận diện khuôn mặt. Ảnh khuôn mặt sẽ được đi qua một bộ trích chọn đặc trưng mà đầu ra ở đây là một vector biểu thị cho khuôn mặt. Có rất nhiều phương pháp để trích chọn đặc trưng của một bức ảnh. Các phương pháp truyền thống thường sử dụng các bộ trích chọn đặc trưng hand-craft feature (đặc trưng cứng) như: HOG, SIFT, LBP, … Tuy nhiên, các bộ trích chọn đặc trưng sử dụng mạng CNN lại có hiệu quả vượt bậc so với các bộ trích chọn đặc trưng truyền thống.

*Hình 2.17. Các kiến trúc CNN được dùng cho nhận diện khuôn mặt qua các năm [7]*

Cùng với sự phát triển của các kiến trúc mạng là sự phát triển của các hàm Loss Function để tối ưu kết quả cho bài toán. Nhiều loss function có thể kể đến như *Euclidean Distance Based Loss, Angular/Cosine-Margin-Based Loss* và *Softmax Loss* và một số biến thể của nó.



*Hình 2.18. Các Loss Functions phục vụ cho nhận diện khuôn mặt qua các năm [7]*

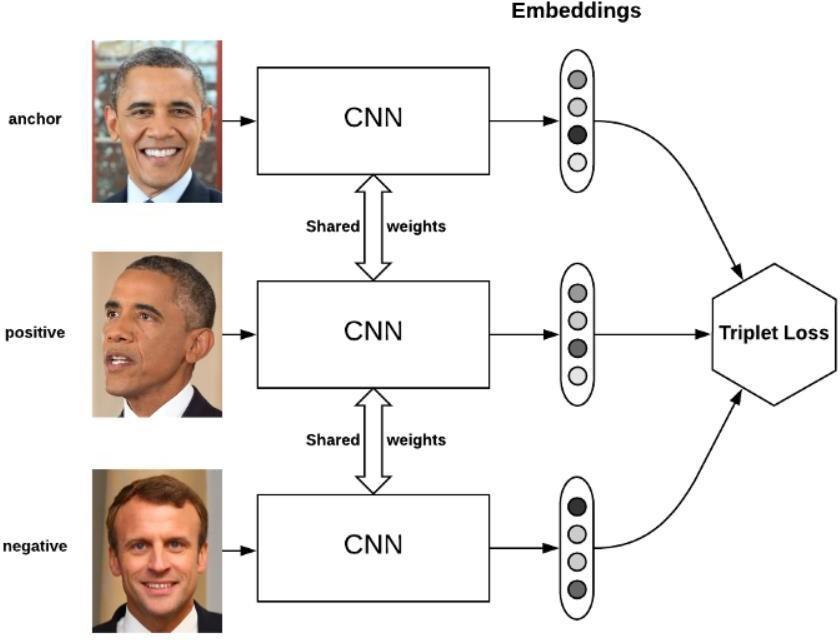
***Một số Loss Function:***

* + - ***Triplet Loss***

Thay vì lấy hai giá trị đầu vào, Triplet Loss đưa ra một công thức mới gồm 3 giá trị đầu vào: Anchor ( 𝑥𝑎 ảnh đầu vào của mạng), Positive ( 𝑥𝑝 ảnh cùng là một

𝑖 𝑖

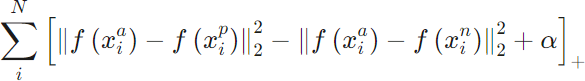
người với Anchor), Negative ( 𝑥𝑛 ảnh khuông cùng là một người với Anchor). Sau đó các ảnh đầu vào sẽ được trích xuất đặc trưng thành các embedding (embedding là một vector với số chiều cố định nhỏ hơn feature vector bình thường, đại diện cho các feature và được dùng trong nhiệm vụ phân loại các đối tượng) rồi thực hiện phân loại [15].



*Hình 2.19. Hàm Triplet Loss [14]*

Mô tả đầu vào của Triplet Loss

Hàm Triplet Loss được tính theo công thức:



𝑓(𝑥𝑎) chính là embedding của 𝑥𝑎. Công thức trên cho ta thấy khoảng cách

𝑖 𝑖

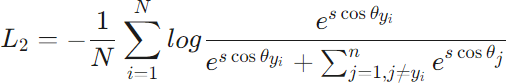
giữa 2 embedding của Anchor và Positive với Anchor và Negative phải nhỏ hơn ít nhất một số 𝛼. Để mạng học được nhiều hơn, điểm Positive được chọn phải nằm xa nhất có thể so với Anchor và điểm Negative phải nằm gần nhất có thể so với Anchor là những trường hợp xấu nhất mà mạng học cần vượt qua [15].

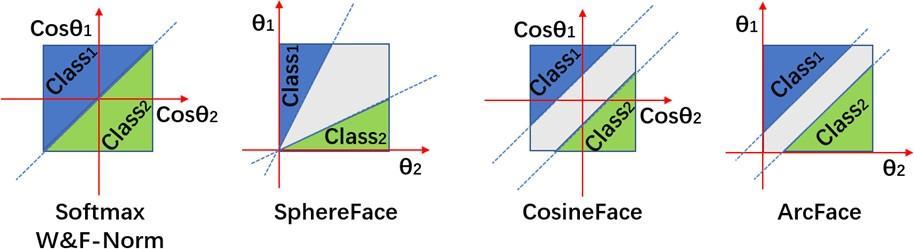
Như vậy khi áp dụng Triple loss vào các mô hình convolutional neural network chúng ta có thể tạo ra các biểu diễn vector tốt nhất cho mỗi một bức ảnh. Những biểu diễn vector này sẽ phân biệt tốt các ảnh Negative rất giống ảnh Positive. Và đồng thời các bức ảnh thuộc cùng một label sẽ trở nên gần nhau hơn trong không gian chiếu euclidean.

Tuy nhiên, khi gặp dataset lớn sẽ có sự bùng nổ về số lượng bộ ba triplet samples dẫn đến sự lặp lại đáng kể các bước.

***ArcFace (Additive Angular Margin Loss)***

ArcFace được cải tiến dựa trên hàm Softmax Loss. Công thức tính ArcFace:





*Hình 2.20. ArcFace và một số hàm Loss khác [4]*

Ý nghĩa của việc dùng hàm mất mát ArcFace là làm cho các mẫu cùng loại co cụm, tăng sự khác biệt giữa các lớp. Việc sử dụng Softmax sẽ giúp chúng ta phân tách các vector đặc trưng tuy nhiên lại tạo nên một sự mơ hồ trên đường ranh giới quyết định, trong khi đó hàm ArcFace lại có sự phân chia rõ rệt giữa các lớp gần nhau. Như vậy hàm ArcFace có thể kết hợp cùng với rất nhiều mô hình CNN để tạo ra các mô hình nhận diện khuôn mặt tốt hơn.

## Bộ nhận diện khuôn mặt

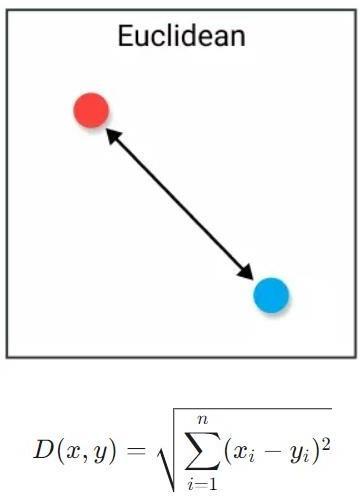
Sau khi trích chọn đặc trưng, mỗi khuôn mặt sẽ có các feature vector. Tùy vào ứng dụng thực tế mà việc nhận diện khuôn mặt được phân loại thành các bài toán khác nhau: *Face Verification* hay *Face Identification*.

* *Face Verification:* bài toán xác minh các ảnh có phải là của chung một người không, ứng dụng nhiều nhất của bài toán này là bảo mật thông tin. Một khuôn mặt sẽ được đăng ký với hệ thống và sau đó so sánh với khuôn mặt đầu vào để mở khóa hệ thống.
* *Face Identification:* bài toán xác định danh tính của khuôn mặt với một tập các khuôn mặt. Trong đó có 2 bài toán con là *Open-set Face Identification* (khuôn mặt nhận diện sẽ được phân vào một trong các khuôn mặt nằm trong tập thư viện) và *Close-set Face Identification* (bài toán xác định khuôn mặt có nằm trong tập thư viện không rồi mới xác định danh tính của khuôn mặt đó).

Ở đây, bài toán sẽ là xác định khuôn mặt *Close-set Face Identification*. Với mỗi khuôn mặt đầu vào cần xác minh, sử dụng một độ đo thích hợp: *Euclidean Distance, Manhattan Distance, Cosine Similarity*, ... để tính độ tương đồng giữa vector đặc trưng của ảnh đầu vào với vector đặc trưng của các ảnh trong cơ sở dữ liệu. Cụ thể là khuôn mặt nhận diện sẽ được phân vào một trong các khuôn mặt nằm trong thư viện khi vượt quá một giá trị ngưỡng cho phép. Trường hợp giá trị đo độ giống nhau không vượt qua ngưỡng thì khuôn mặt đó không có trong tập dữ liệu và không được định danh.

### Một số phương pháp tính khoảng cách:

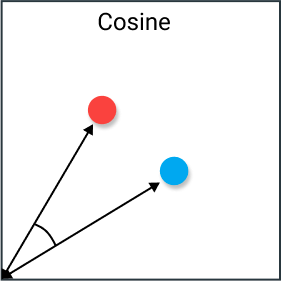
* + - ***Euclidean Distance***

Euclidean Distance còn được biết đến với cái tên 𝐿*2* distance và được tính bằng căn bậc hai tổng bình phương hiệu của các điểm.

*Hình 2.21. Euclidean Distance [16]*

Phương pháp tính này phổ biến và có hiệu quả tốt, đặc biệt với những dữ liệu ít chiều. Tuy nhiên Euclide distance có thể bị ảnh hưởng bởi đơn vị của feature, vì vậy cần phải normalize trước khi tính toán.

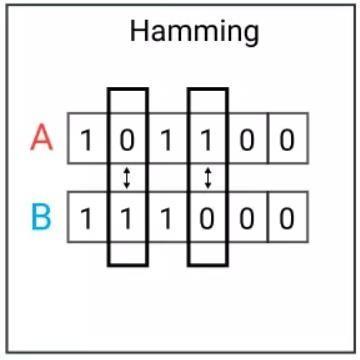
### Cosine Similarity

Cosine Similarity thường được sử dụng để giải quyết vấn đề của Euclidean distance ở không gian nhiều chiều. Ý tưởng đơn giản là tính góc tạo thành giữa 2 vector.

*Hình 2.22. Cosine Similarity [16]*

Phương pháp này thường được sử dụng trong các dữ liệu đa chiều và không quá phụ thuộc vào độ lớn của vector.

### Hamming Distance



*Hình 2.23. Hamming Distance [16]*

Khoảng cách Hamming là số giá trị khác nhau giữa 2 vector. Phương pháp này thường được dùng để so sánh 2 chuỗi binary có độ dài bằng nhau. Phương pháp này cũng có thể so sánh độ tương đồng của vector bằng cách tính số lượng ký tự khác nhau.

Chương 2 đã trình bày kiến trúc chính nhận diện khuôn mặt của hệ thống điểm danh bằng khuôn mặt gồm 5 phần: bộ phát hiện khuôn mặt, bộ căn chỉnh khuôn mặt, bộ chống giả mạo, bộ trích xuất đặc trưng, bộ nhận diện khuôn mặt và khảo sát các phương pháp được sử dụng trong các khâu của hệ thống. Ở phần sau sẽ trình bày chi tiết thiết kế của hệ thống cũng như các phương pháp được lựa chọn thực thi cho từng phần.

# CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

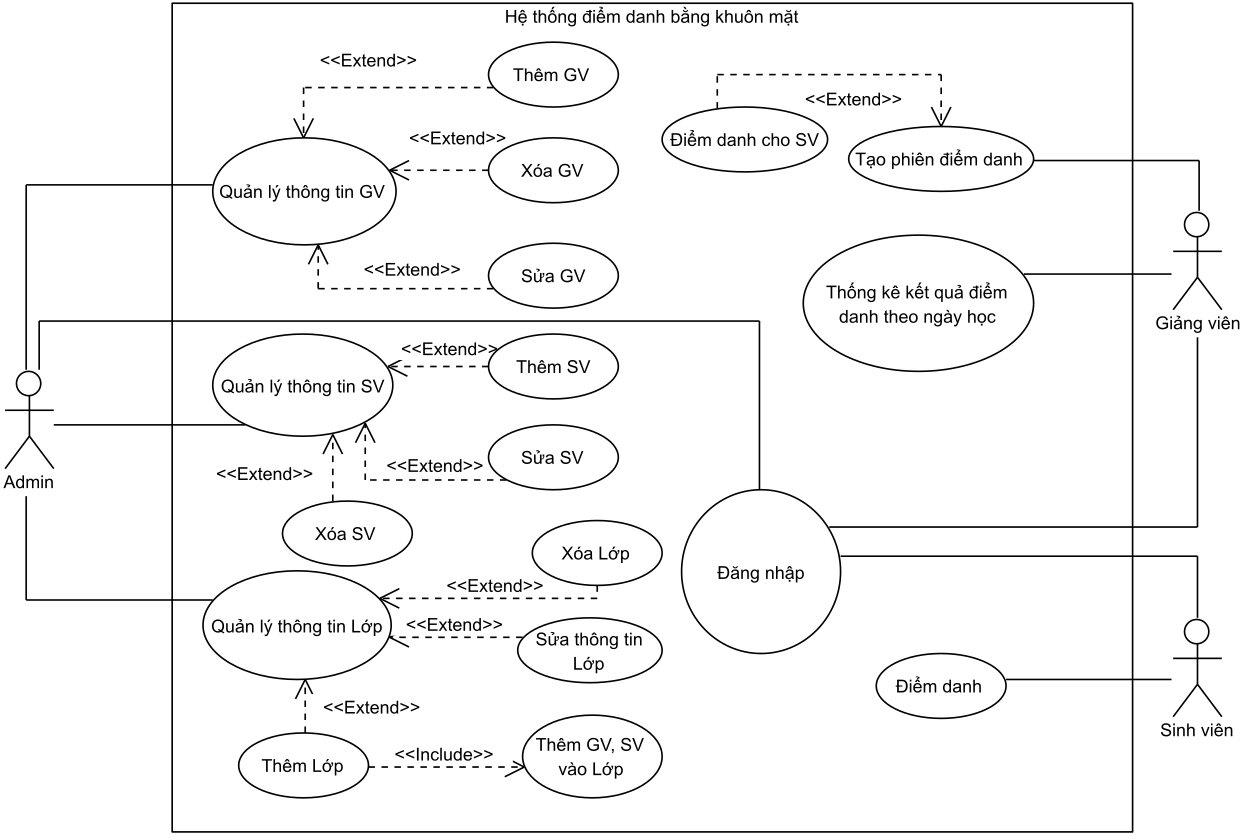
## Phân tích hệ thống

### Mô tả các chức năng chính của hệ thống:

Hệ thống điểm danh bằng khuôn mặt sẽ gồm 3 actor chính: Admin, Giảng viên, Sinh viên. Các chức năng chính của các actor sau khi đăng nhập vào hệ thống:

* Admin:
  + Thêm, sửa, xóa tài khoản giảng viên.
  + Thêm, sửa xóa tài khoản sinh viên.
  + Thêm, sửa, xóa lớp học: thêm sinh viên vào lớp học, gán giảng viên cho lớp học.
* Giảng viên:
  + Tạo phiên điểm danh cho lớp học. Mỗi ngày học của một lớp học có thể không có hoặc có nhiều phiên điểm danh. Mỗi khi điểm danh Giảng viên tạo một phiên mới và gửi thông báo cần phải điểm danh đến mỗi sinh viên trong lớp học.
  + Giảng viên có quyền chỉnh sửa kết quả điểm danh của session khi đang open.
  + Giảng viên có chức năng thống kê kết quả điểm danh của lớp học.
* Sinh viên:
  + Sinh viên có chức năng điểm danh trong session của lớp học.

### Use Case tổng quan của hệ thống:



*Hình 3.1. Biểu đồ Use Case của toàn hệ thống*

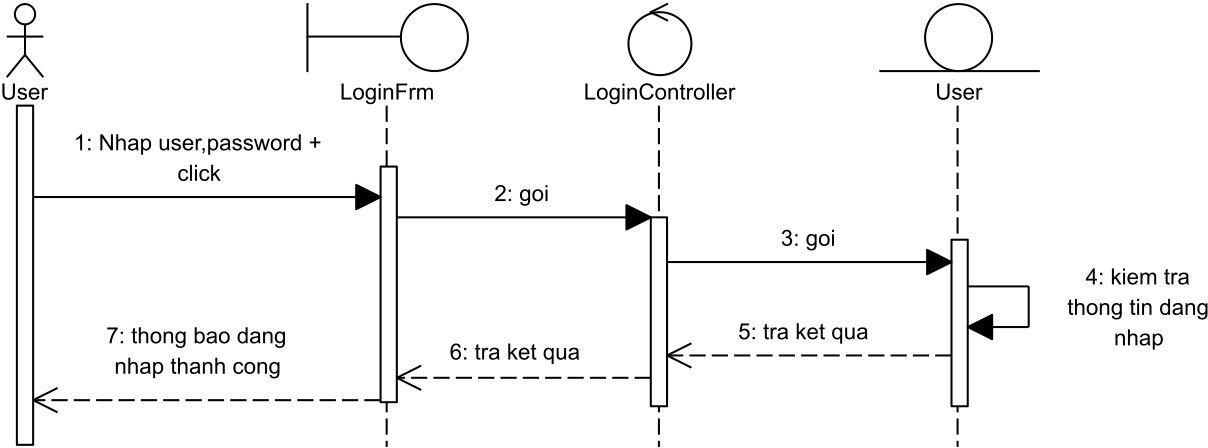
### Đặc tả các Use Case của hệ thống:

Đăng nhập

| **Use Case** | Đăng nhập |
| --- | --- |
| **Actor** | Người dùng |
| **Tiền điều kiện** | Người dùng truy cập vào ứng dụng |
| **Hậu điều kiện** | Người dùng đăng nhập thành công |
| **Chuỗi sự kiện chính** | 1. Người dùng nhập thông tin đăng nhập gồm username (mã người dùng) và password rồi click nút Đăng nhập. 2. Hệ thống thông báo đăng nhập thành công và hiện lên giao diện chính ứng với người dùng. |

|  |  |
| --- | --- |
| **Ngoại lệ** | Người dùng nhập sai username hoặc password. |

*Bảng 3.1. Kịch bản ca sử dụng đăng nhập*

*Sequence diagram*

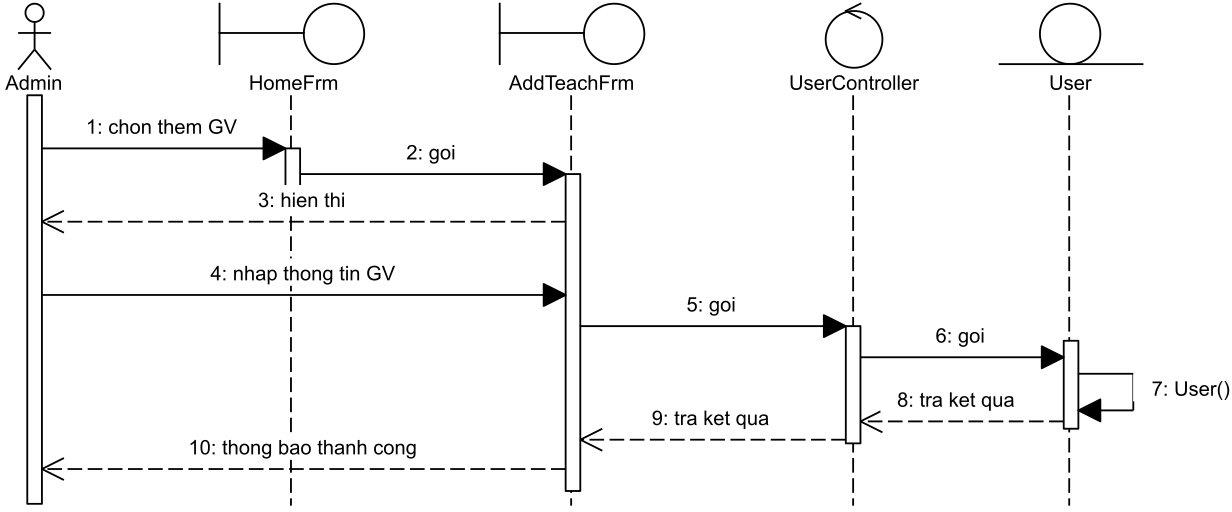
*Hình 3.2. Biểu đồ tuần tự chức năng đăng nhập*

Các Use Case của Admin

* Thêm thông tin GV

| **Use Case** | Thêm thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Thêm GV. 4. Hệ thống hiển thị giao diện Thêm GV. 5. Admin nhập các thông tin của GV và click Thêm GV. 6. Hệ thống lưu GV vào CSDL và thông báo thành công. |
| **Ngoại lệ** |  |

*Bảng 3.2. Kịch bản ca sử dụng thêm thông tin GV*

*Sequence diagram*

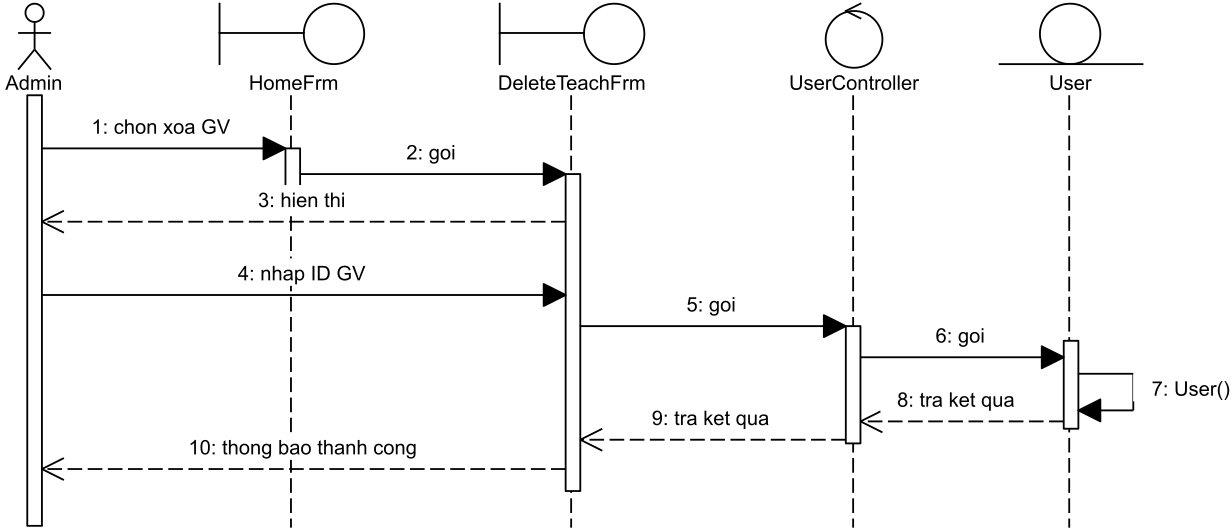
*Hình 3.3. Biểu đồ tuần tự chức năng thêm thông tin GV*

* Xóa Thông tin GV

| **Use Case** | Xóa thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Xóa GV. 4. Hệ thống hiển thị giao diện Xóa GV. 5. Admin nhập vào mã của GV và click Xóa GV. 6. Hệ thống Xóa GV và thông báo thành công. |
| **Ngoại lệ** | Mã GV nhập vào không tồn tại |

*Bảng 3.3. Kịch bản ca sử dụng xóa GV*

*Sequence diagram*

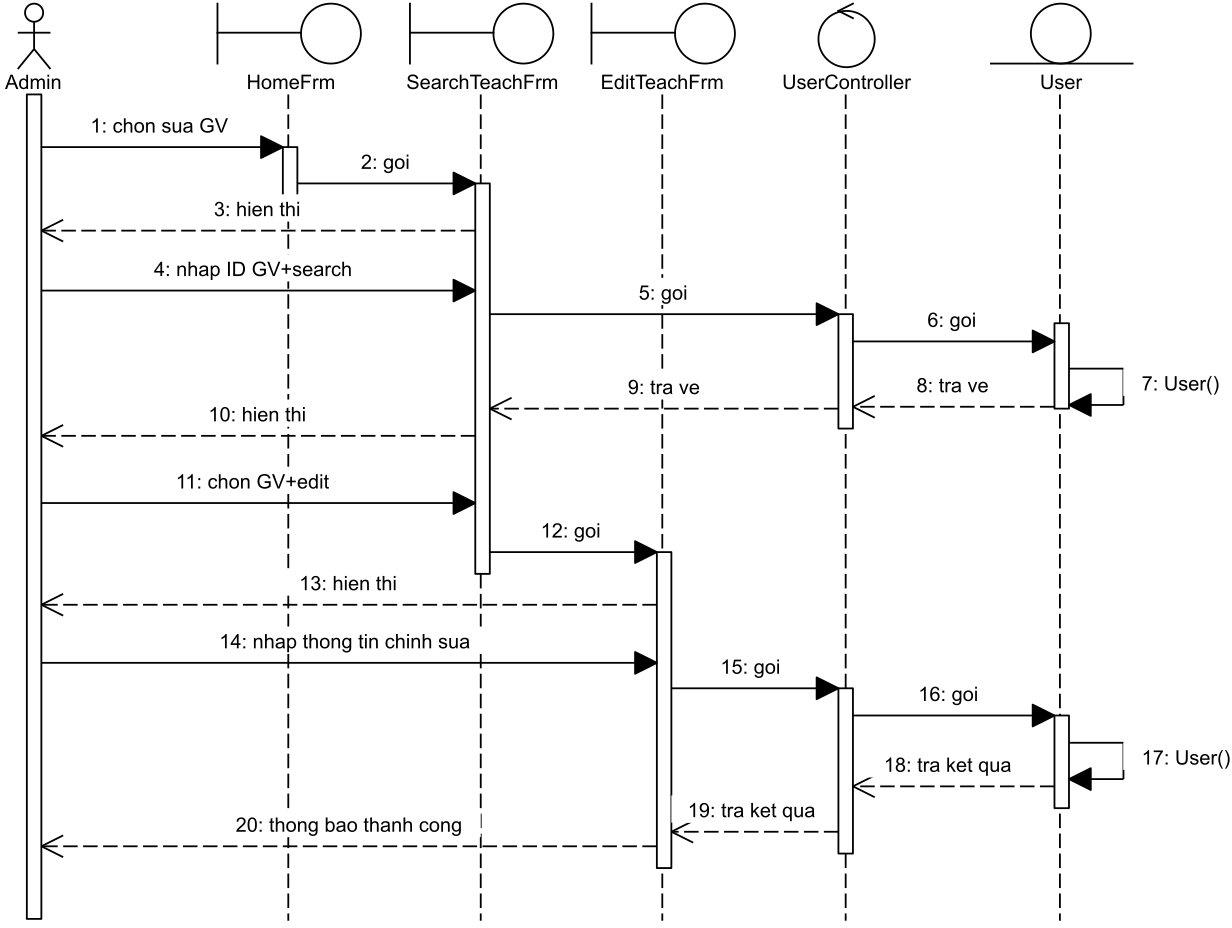


*Hình 3.4. Biểu đồ tuần tự chức năng xóa GV*

* Chỉnh sửa thông tin GV

| **Use Case** | Sửa thông tin GV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin GV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin GV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm GV, Sửa GV, Xóa GV. 3. Admin chọn chức năng Sửa GV. 4. Hệ thống hiển thị giao diện Sửa GV. 5. Admin nhập vào mã của GV và click chọn GV. 6. Hệ thống hiển thị giao diện thông tin của GV. 7. Admin thực hiện sửa thông tin của GV rồi click Sửa. 8. Hệ thống lưu lại thông tin GV vào CSDL và thông báo thành công. |
| **Ngoại lệ** | Mã GV nhập vào không tồn tại |

*Bảng 3.4. Kịch bản ca sử dụng sửa thông tin GV*

*Sequence diagram*

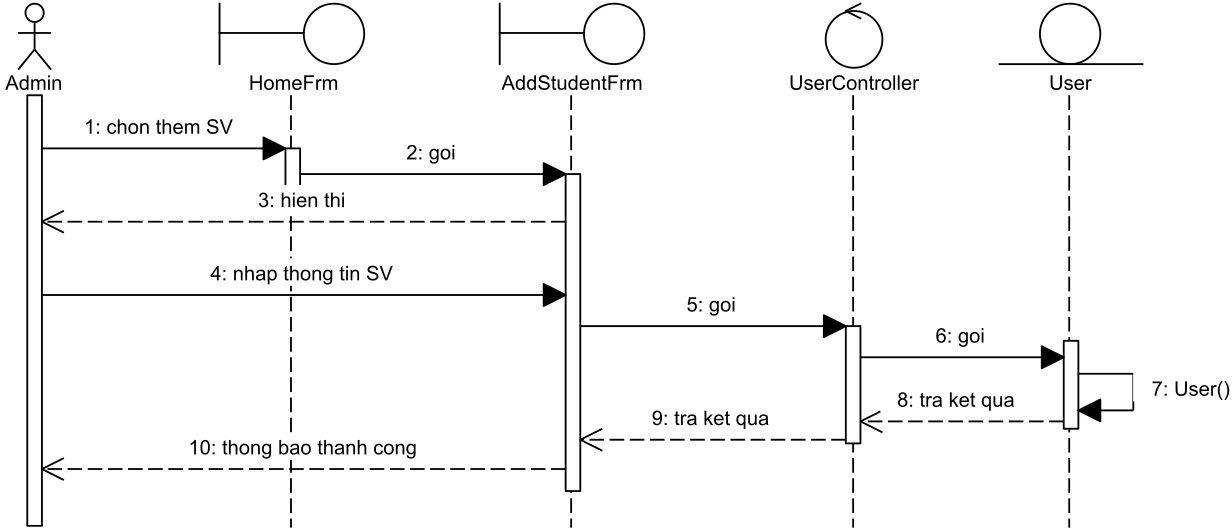
*Hình 3.5. Biểu đồ tuần tự chức năng sửa thông tin GV*

* Thêm thông tin SV

| **Use Case** | Thêm thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Thêm SV. 4. Hệ thống hiển thị giao diện Thêm SV. |

|  | 1. Admin nhập các thông tin của SV và click Thêm SV. 2. Hệ thống lưu SV vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** |  |

*Bảng 3.5. Kịch bản ca sử dụng thêm thông tin SV*

*Sequence diagram*

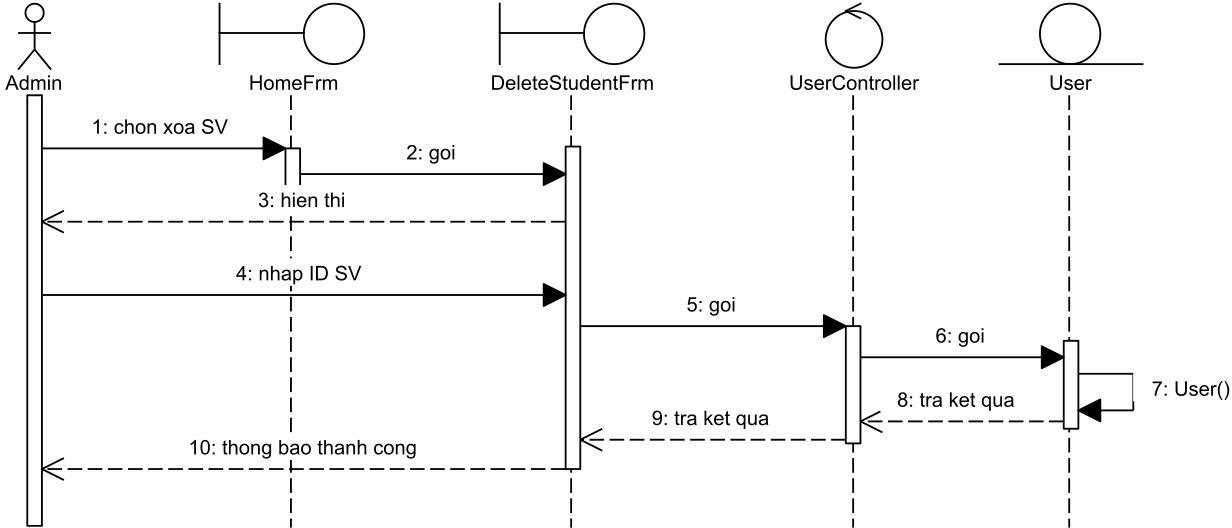
*Hình 3.6. Biểu đồ tuần tự chức năng thêm thông tin SV*

* Xóa Thông tin SV

| **Use Case** | Xóa thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Xóa SV. 4. Hệ thống hiển thị giao diện Xóa SV. 5. Admin nhập vào mã của SV và click Xóa SV. |

|  | 6. Hệ thống Xóa SV và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã SV nhập vào không tồn tại |

*Bảng 3.6. Kịch bản ca sử dụng xóa SV*

*Sequence diagram*

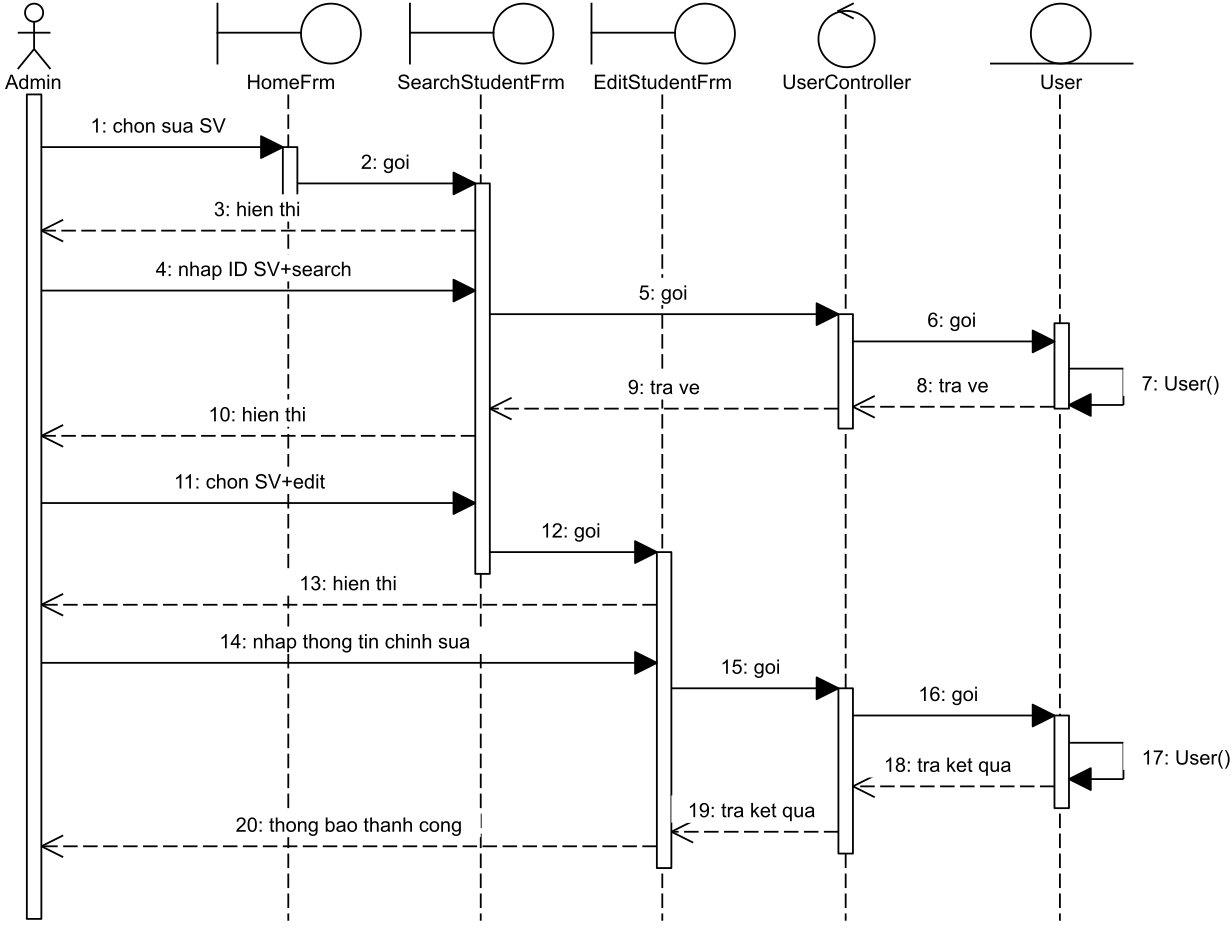
*Hình 3.7. Biểu đồ tuần tự chức năng xóa SV*

* Chỉnh sửa thông tin SV

| **Use Case** | Sửa thông tin SV |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin SV thành công |
| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin SV trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm SV, Sửa SV, Xóa SV. 3. Admin chọn chức năng Sửa SV. 4. Hệ thống hiển thị giao diện Sửa SV. 5. Admin nhập vào mã của SV và click chọn SV. 6. Hệ thống hiển thị giao diện thông tin của SV. |

|  | 1. Admin thực hiện sửa thông tin của SV rồi click Sửa. 2. Hệ thống lưu lại thông tin SV vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã SV nhập vào không tồn tại |

*Bảng 3.7. Kịch bản ca sử dụng sửa thông tin SV*

*Sequence diagram*

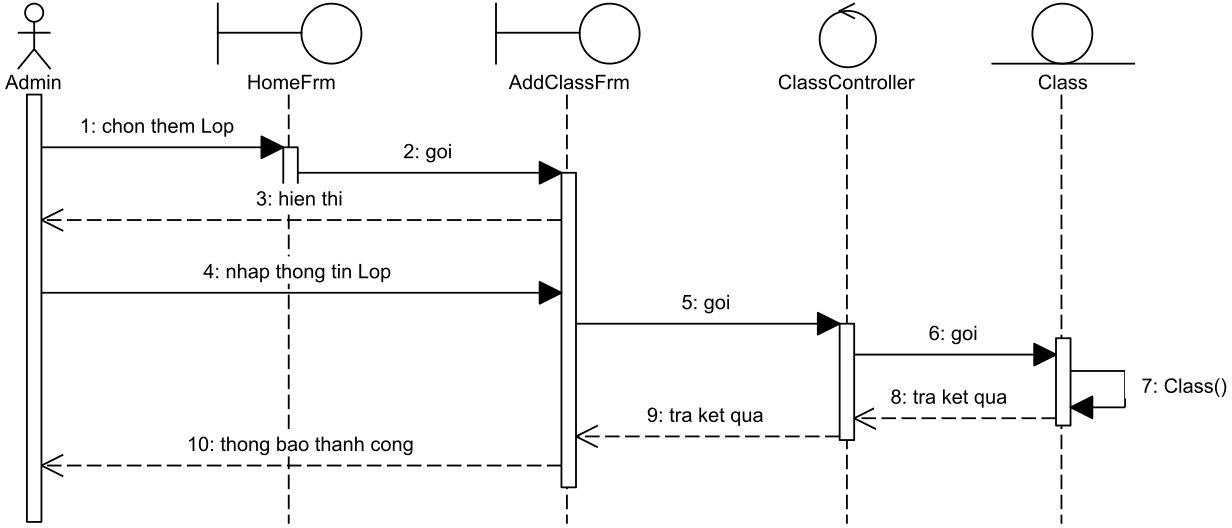
*Hình 3.8. Biểu đồ tuần tự chức năng sửa thông tin SV*

* Thêm thông tin Lớp học

| **Use Case** | Thêm thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Thêm thông tin Lớp thành công |

| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 3. Admin chọn chức năng Thêm Lớp. 4. Hệ thống hiển thị giao diện Thêm Lớp. 5. Admin nhập các thông tin của Lớp gồm thông tin GV, danh sách SV và click Thêm Lớp. 6. Hệ thống lưu Lớp vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** |  |

*Bảng 3.8. Kịch bản ca sử dụng thêm thông tin lớp học Sequence diagram*



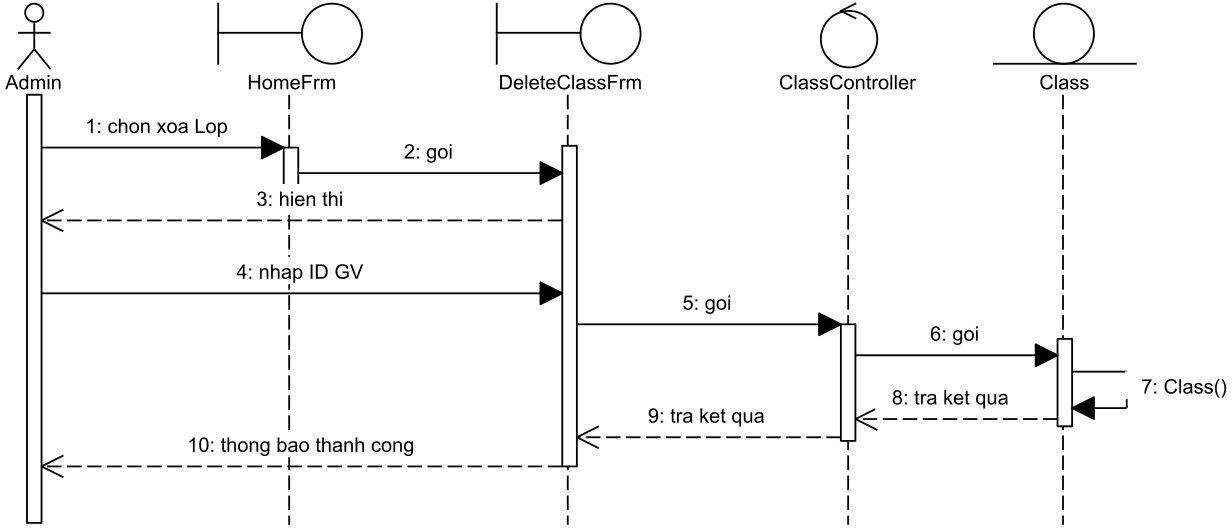
*Hình 3.9. Biểu đồ ca sử dụng thêm thông tin lớp học*

* Xóa Thông tin Lớp

| **Use Case** | Xóa thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Xóa thông tin Lớp thành công |

| **Chuỗi sự kiện chính** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao diện chính của hệ thống. 2. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 3. Admin chọn chức năng Xóa Lớp. 4. Hệ thống hiển thị giao diện Xóa Lớp. 5. Admin nhập vào mã của Lớp và click Xóa Lớp. 6. Hệ thống Xóa Lớp và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã Lớp nhập vào không tồn tại |

*Bảng 3.9. Kịch bản ca sử dụng xóa lớp học*

*Sequence diagram*

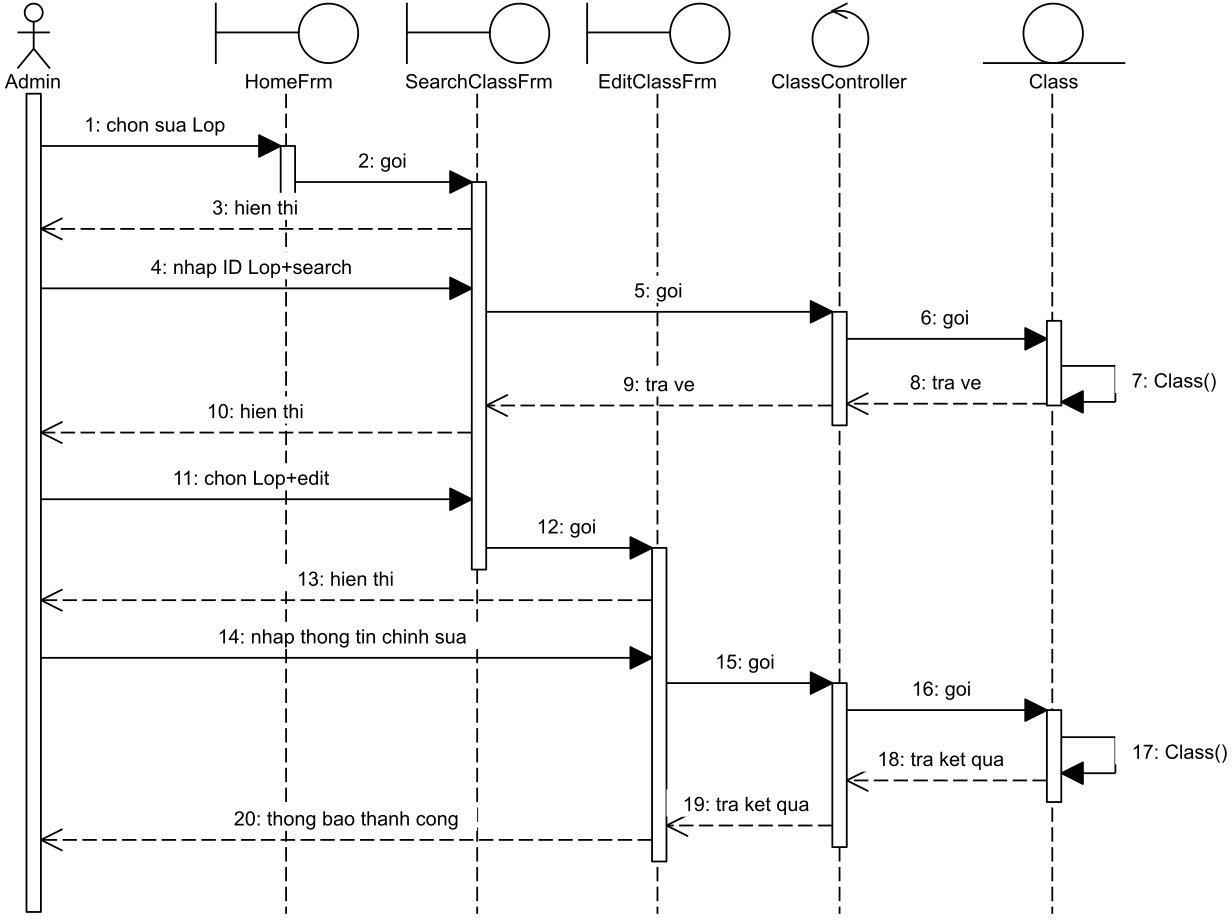
*Hình 3.10. Biểu đồ ca sử dụng xóa lớp học*

* Chỉnh sửa thông tin Lớp

| **Use Case** | Sửa thông tin Lớp |
| --- | --- |
| **Actor** | Admin |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Sửa thông tin Lớp thành công |
| **Chuỗi sự kiện** | 1. Amin chọn chức năng Quản lý thông tin Lớp trên giao |

| **chính** | diện chính của hệ thống.   1. Hệ thống hiển thị các chức năng: Thêm Lớp, Sửa Lớp, Xóa Lớp. 2. Admin chọn chức năng Sửa Lớp. 3. Hệ thống hiển thị giao diện Sửa Lớp. 4. Admin nhập vào mã của Lớp và click chọn Lớp. 5. Hệ thống hiển thị giao diện thông tin của Lớp. 6. Admin thực hiện sửa thông tin của Lớp rồi click Sửa. 7. Hệ thống lưu lại thông tin Lớp vào CSDL và thông báo thành công. |
| --- | --- |
| **Ngoại lệ** | Mã Lớp nhập vào không tồn tại |

*Bảng 3.10. Kịch bản ca sử dụng sửa thông tin lớp học Sequence diagram*



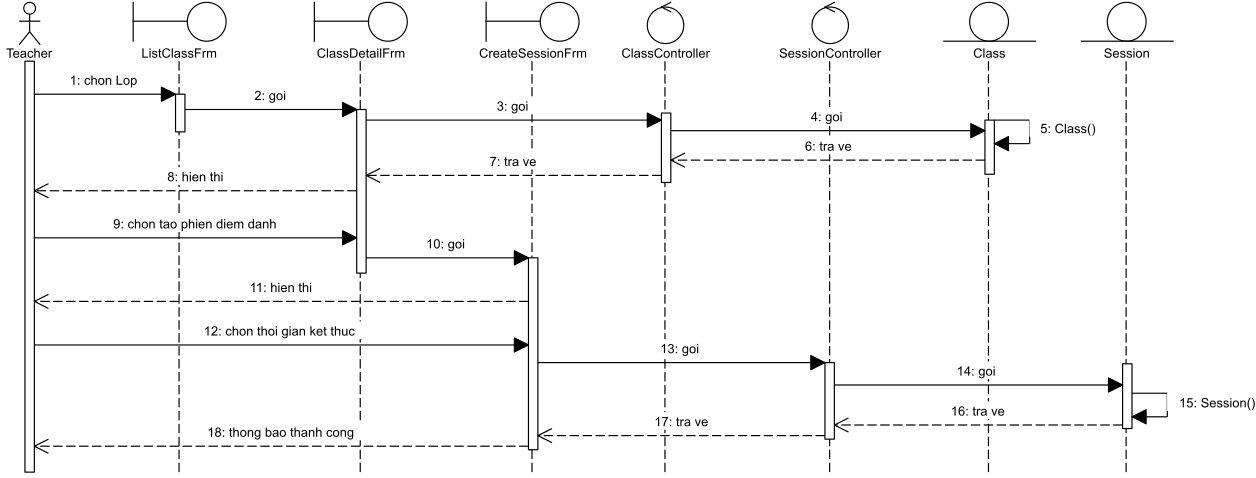
*Hình 3.11. Biểu đồ ca sử dụng sửa thông tin lớp học*

**Các Use Case của Giảng viên**

* Tạo phiên điểm danh

| **Use Case** | Tạo phiên điểm danh |
| --- | --- |
| **Actor** | Giảng viên |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| **Hậu điều kiện** | Tạo phiên điểm danh cho lớp học thành công |
| **Chuỗi sự kiện chính** | 1. Giảng viên chọn lớp học vào click tạo phiên điểm danh. 2. Hệ thống thông báo tạo phiên điểm danh thành công và gửi đến các SV trong lớp học. |
| **Ngoại lệ** | GV điểm danh cho SV không thực hiện điểm danh được trong lớp học. |

*Bảng 3.11. Kịch bản tạo phiên điểm danh*

*Sequence diagram*

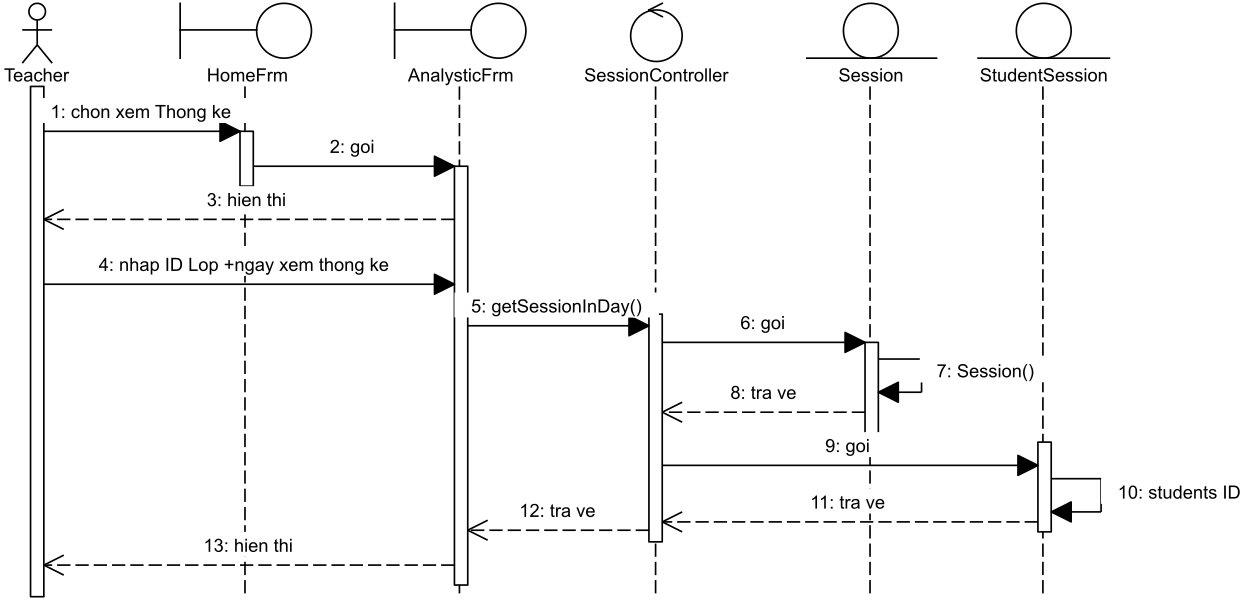
*Hình 3.12. Biểu đồ ca sử dụng tạo phiên điểm danh*

* Thống kê kết quả điểm danh theo ngày học

| **Use Case** | Thống kê kết quả điểm danh theo ngày học |
| --- | --- |
| **Actor** | Giảng viên |

| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |
| --- | --- |
| **Hậu điều kiện** | GV xem được kết quả điểm danh theo ngày học |
| **Chuỗi sự kiện chính** | 1. GV chọn chức năng Xem thống kê điểm danh theo ngày học trên giao diện của hệ thống. 2. Hệ thống hiển thị giao diện nhập thông tin để xem thống kê. 3. GV nhập mã lớp, ngày tháng năm muốn xem kết quả điểm danh và chọn xem thống kê. 4. Hệ thống hiển thị ngày giờ điểm danh của các sinh viên trong lớp trong ngày đã nhập. |
| **Ngoại lệ** |  |

*Bảng 3.12. Kịch bản ca sử dụng thống kê kết quả điểm danh theo ngày học Sequence diagram*



*Hình 3.13. Biểu đồ ca sử dụng thống kê kết quả điểm danh theo ngày học*

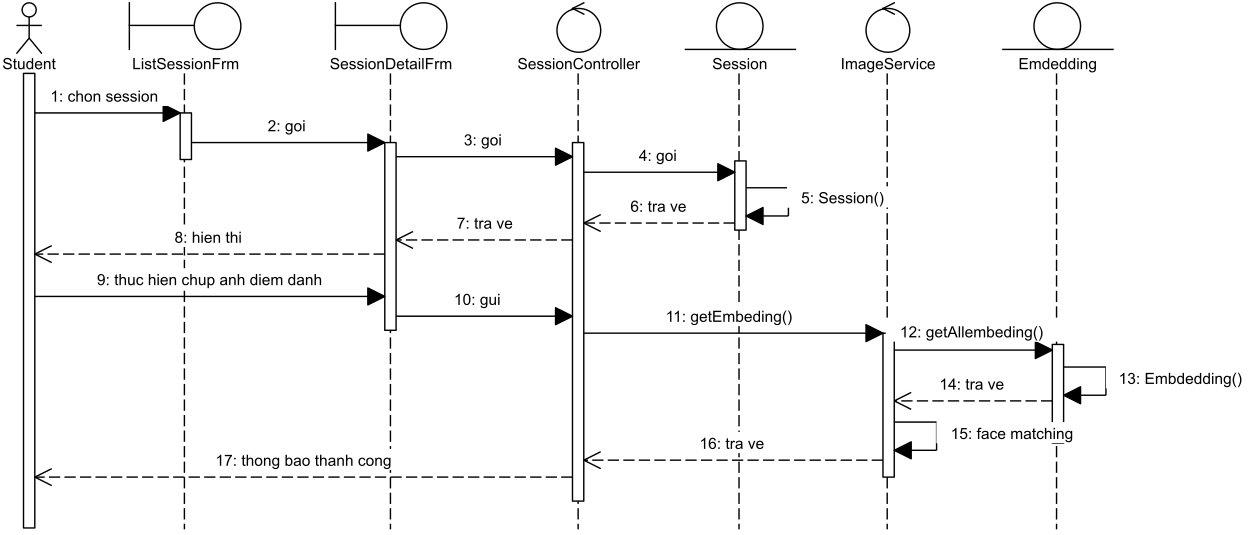
**Các Use Case của Sinh viên**

* Điểm danh

| **Use Case** | Điểm danh |
| --- | --- |
| **Actor** | Sinh viên |
| **Tiền điều kiện** | Đăng nhập thành công vào hệ thống |

| **Hậu điều kiện** | SV thực hiện điểm danh thành công |
| --- | --- |
| **Chuỗi sự kiện chính** | 1. SV click vào thông báo có phiên điểm danh trên giao diện danh sách các lớp học của SV. 2. Hệ thống hiển thị giao diện Camera. 3. SV thực hiện chụp ảnh và gửi lên server. 4. Hệ thống thông báo điểm danh thành công. |
| **Ngoại lệ** | Hệ thống thông báo ảnh giả mạo và SV cần thực hiện chụp lại ảnh. |

*Bảng 3.13. Kịch bản ca sử dụng điểm danh*

*Sequence diagram*

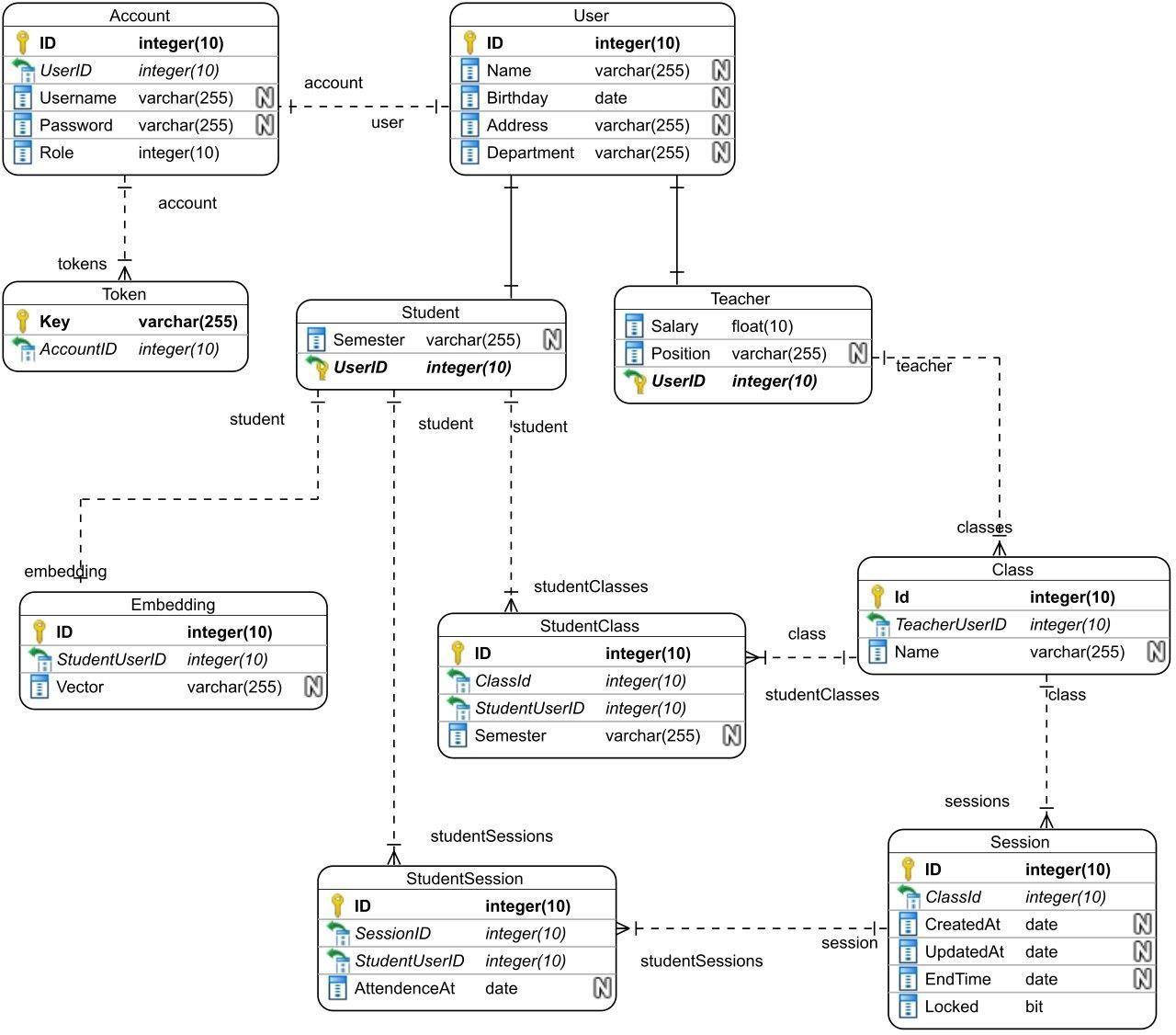
*Hình 3.14. Biểu đồ ca sử dụng điểm danh*

## Thiết kế hệ thống

### Biểu đồ lớp của hệ thống

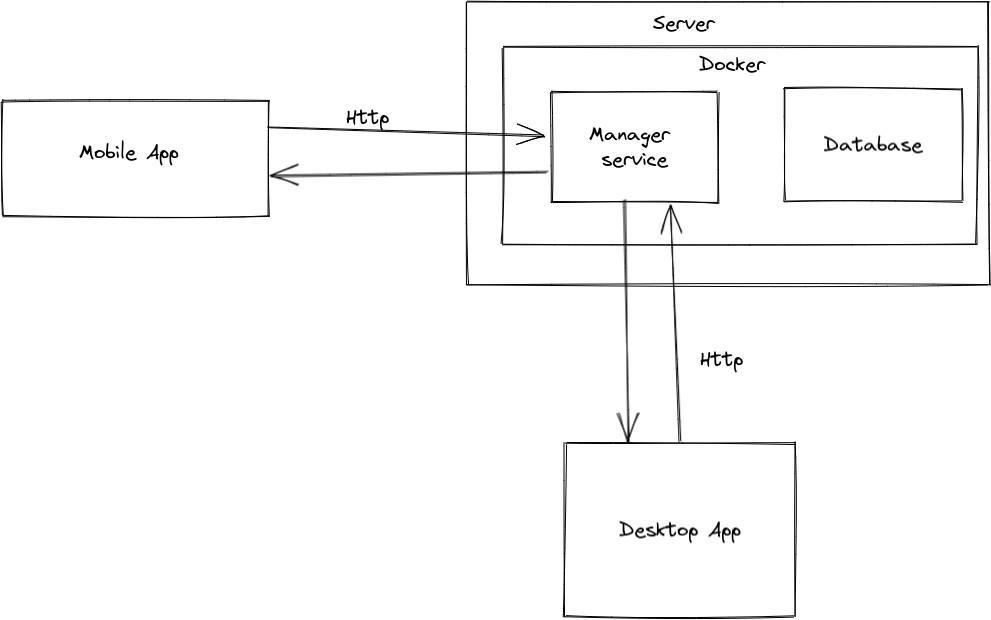
*Hình 3.15. Biểu đồ lớp phân tích của hệ thống*

### Biểu đồ CSDL



*Hình 3.16. Biểu đồ CSDL*

### Thiết kế vật lý của hệ thống



*Hình 3.17. Thiết kế vật lý của hệ thống*

Server

* + *Docker*

Để dễ dàng trong quá trình cài đặt, các dịch vụ của hệ thống được đóng gói lại thành image bằng docker sau đó đẩy lên gitlab container registry.

Docker là một nền tảng cho lập trình viên để develop, deploy và run application với container. Nó cho phép tạo các môi trường độc lập và tách biệt để khởi chạy và phát triển ứng dụng và môi trường này được gọi là container. Khi cần deploy lên bất kỳ server nào chỉ cần run container của Docker thì application sẽ được khởi chạy ngay lập tức.

Lợi ích của việc sử dụng docker

* Khởi chạy nhanh, nhẹ hơn máy ảo.
* Không phụ thuộc nền tảng, không ảnh hưởng đến các phần mềm khác trong hệ thống.
* Dễ dàng thiết lập môi trường làm việc.
  + *Manager service*

Manager service được viết bằng Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép xây dựng các ứng dụng web từ đơn giản tới phức tạp. Flask

có ưu điểm là nhẹ, ít phụ thuộc vào các cập nhật và dễ tìm kiếm các lỗi bảo mật.

* + *Database*

Sử dụng hệ quản trị cơ sở dữ liệu là MySQL.

**Client:**

Về phía client, hệ thống bao gồm ứng dụng di động chạy trên hệ điều hành Android dành cho giảng viên, sinh viên và ứng dụng trên PC. Ứng dụng đóng vai trò là giao diện để người dùng có thể dễ dàng tương tác với phần server của hệ thống. Ứng dụng Android được viết bằng ngôn ngữ lập trình Java. Ứng dụng Desktop được viết bằng ngôn ngữ PythonQT.

## Tổng quan chức năng điểm danh nằng khuôn mặt

Ứng dụng gồm có 5 phần chính: bộ phát hiện khuôn mặt, bộ căn chỉnh khuôn mặt, bộ chống giả mạo, bộ trích xuất đặc trưng và bộ nhận diện khuôn mặt, cụ thể là:

Bộ phát hiện khuôn mặt:

Sử dụng ảnh RGB làm đầu vào và phát hiện khuôn mặt trong ảnh đó dưới dạng các hộp giới hạn (bounding box) sau đó sẽ làm đầu vào cho các khối chức năng tiếp theo. Có rất nhiều phương pháp phát hiện khuôn mặt, dưới đây là một thống kê độ chính xác (độ đo được sử dụng để đánh giá là AP) phát hiện khuôn mặt của một số mô hình trên tập dữ liệu WIDER FACE (dễ, trung bình, khó):

| **Mô hình** | **WIDER FACE (Dễ)** | **WIDER FACE**  **(Trung bình)** | **WIDER FACE**  **(Khó)** |
| --- | --- | --- | --- |
| **MTCNN** | 85.1% | 82.0 % | 60.7% |
| **CMS-RCNN** | 90.2% | 87.4% | 64.3% |
| **SSH** | 92.7% | 91.5% | 84.4% |
| **PyramidBox** | 95.6% | 94.6% | 90.0% |
| **RetinaFace** | 96.3% | 95.6% | 91.4% |

*Bảng 3.14. Độ chính xác các mô hình phát hiện khuôn mặt trên tập WIDER FACE*

*[8]*

Dựa vào kết quả trên, em sử dụng pre-train model Retina Face và thư viện ncnn-android để thực hiện.

**Bộ căn chỉnh khuôn mặt:**

Mô hình phát hiện khuôn mặt Retina Face cho kết quả là bounding box và 5 điểm trên khuôn mặt: mắt trái, mắt phải, mũi, hai điểm trên môi. Từ đó ta dễ dàng thực hiện căn chỉnh khuôn mặt.

**Bộ chống giả mạo:**

Sử dụng các ảnh chỉ chứa khuôn mặt làm đầu vào, đưa qua mô hình đã được huấn luyện để tiến hành phân loại khuôn mặt đầu vào là thật hay giả mạo. Em xây dựng một mô hình có kiến trúc khai thác điểm mạnh của Depthwise Convolution và các khối residual trong ResNet, train trên Colab và để phát hiện chống giả mạo trên android.

*Chuẩn bị dữ liệu:*

Bộ dữ liệu train gồm 48,134 hình ảnh về khuôn mặt thật và khuôn mặt giả mạo. Các khuôn mặt thật được chụp với các góc mặt khác nhau. Các khuôn mặt giả mạo được tạo ra bằng các màn hình hiển thị khác nhau. Cụ thể:

| Tập dữ liệu | Huấn luyện | Kiểm thử |
| --- | --- | --- |
| Số khuôn mặt thật | 11,025 | 2,945 |
| Số khuôn mặt giả mạo | 21,845 | 12,319 |
| Tổng số khuôn mặt | 32,870 | 15,264 |

*Bảng 3.15. Số lượng hình ảnh khuôn mặt trong tập train*

Bộ dữ liệu test được lấy một phần từ bộ Oulu-NPU và SiW. Bộ SiW được tạo ra từ 165 người khác nhau, có tổng số 1,434,748 khuôn mặt trong tập train, 1,220,369 khuôn mặt trong tập test, ảnh được tạo bởi các góc và cử chỉ khuôn mặt khác nhau. Bộ Oulu-NPU gồm 4950 video về khuôn mặt thật và khuôn mặt giả mạo, có tổng số 241,279 khuôn mặt trong tập train, 179,898 khuôn mặt trong tập test, ảnh được tạo dưới các điều kiện môi trường khác nhau, ảnh giả mạo được thu qua các màn hình và máy in khác nhau. Vì bộ dữ liệu gốc rất lớn nên em chỉ sử dụng một phần làm tập test gọi là Oulu-NPU-mini và SiW-mini, cụ thể:

| Tập dữ liệu (ours) | Oulu-NPU-mini | SiW-mini |
| --- | --- | --- |
| Số khuôn mặt thật | 3,260 | 6,667 |
| Số khuôn mặt giả mạo | 5,840 | 7,742 |
| Tổng số | 9,100 | 14,409 |

*Bảng 3.16. Số lượng hình ảnh khuôn mặt trong tập test Thông số đánh giá:*

Trong bài toán chống giả mạo khuôn mặt, em xác định lớp Positive là khuôn mặt giả mạo, lớp Negative là khuôn mặt thật. Do đó False Positive (FP) là lỗi mô hình dự đoán khuôn mặt giả mạo là khuôn mặt thật và False Negative (FN) là lỗi mô hình dự đoán khuôn mặt thật thành khuôn mặt giả mạo. Từ các định nghĩa trên, em đánh giá mô hình dựa trên các độ đo:

* + APCER: tỉ lệ từ chối sai, có nghĩa là tỉ lệ phân loại nhầm ảnh thật thành ảnh giả, người dùng thật truy cập vào hệ thống nhưng lại bị từ chối. Nếu tỉ lệ này càng lớn sẽ dẫn đến việc người dùng khó đăng nhập vào hệ thống.

APCER = FP / (TN + FP)

* + NPCER: tỉ lệ chấp nhận sai, có nghĩa là tỉ lệ phân loại nhầm ảnh giả thành ảnh thật, người dùng truy cập là giả mạo nhưng hệ thống vẫn chấp nhận. Nếu tỉ lệ này lớn thì hệ thống có khả năng chống giả mạo kém. Đây là một mối nguy hiểm cho các hệ thống bảo mật.

NPCER = FN / (TP + FN)

* + ACER: tỉ lệ lỗi trung bình của hệ thống. Tỉ lệ này giúp ta có cái nhìn tổng quan hơn và đánh giá khách quan hơn về các mức độ từ chối và chấp nhận giữa các phương pháp khác nhau.

ACER = (APCER + NPCER) / 2

*Kết quả thực nghiệm:*

| Bộ dữ liệu đánh giá | APCER (%) | NPCER (%) | ACER (%) |
| --- | --- | --- | --- |
| Oulu-NPU-mini | 2.4 | 0.56 | 1.49 |
| SiW-mini | 0.49 | 0.5 | 0.49 |

*Bảng 3.17. Kết quả đánh giá mô hình chống giả mạo*

**Bộ trích xuất đặc trưng**

Sau khi phân loại được khuôn mặt ở bộ chống giả mạo, khuôn mặt thật sẽ được đưa vào bộ trích xuất đặc trưng. Dựa trên tiêu chí cân bằng giữa tốc độ và độ chính xác, em lựa chọn mô hình mạng LResNet100E-IR được xây dựng bởi Jiankang Deng làm mô hình trích chọn đặc trưng cho khuôn mặt. Kiến trúc mạng này có backbone là mạng ResNet100 và sử dụng làm loss ArcFace để tối ưu kết quả. Đầu ra của mạng là một vector đại diện cho khuôn mặt có số chiều là 512.

**Bộ nhận diện khuôn mặt**

Với mỗi hình ảnh khuôn mặt đầu vào sẽ được trích xuất thành một vector đặc trưng, sau đó sử dụng độ đo khoảng cách Euclidean để phân khuôn mặt đầu vào là một trong các khuôn mặt nằm trong CSDL khi vượt qua một giá trị ngưỡng cho phép.

# CHƯƠNG 4: THỬ NGHIỆM CÀI ĐẶT VÀ XÂY DỰNG ỨNG DỤNG

## Cài đặt Server

Hệ thống được thuê trên dịch vụ AWS (Amazon Web Services) với cấu hình như sau.

| CPU | Ram | Bộ nhớ |
| --- | --- | --- |
| 2 Core | 4GB | 32GB SSD |

Phiển bản phầm mềm được sử dụng.

* Python 3.9.9 bulleye
* Mysql 5.7
* Docker 19

Container của chứa backend service và data sử dụng chung 1 network ảo và được export 1 port là 8092 để giúp cho client có thể giao tiếp với backend.

## Cài đặt Client

*Android Application*

Em sử dụng trình biên dịch Android Studio Artic Fox để lập trình app Android Cấu hình thiết bị kiểm thử được mô tả ở các bảng dưới:

| Android verion | API version | SoC | RAM |
| --- | --- | --- | --- |
| 10.0 | API 29 | Snapdragon 660 | 4GB |

*Desktop Application*

Được viết trên QT framework với ngôn ngữ lập trình là Python.

QT là được sử dụng để tạo các chương trình GUI không phụ thuộc nền tảng, các phần mềm được tạo ra bởi QT chạy được trên hầu hết các hệ điều hành PC hiện tại. Các chương trình được tạo bởi QT bao gồm các thành phần như QTCore, QWidget, QtUiTools

… QT cung cấp rất nhiều công cụ mạnh mẽ để tạo các chương trình chạy trên desktop. Môi trường phát triển QT là QT Creator.

## Thiết kế giao diện

### Giao diện đăng nhậpGraphical user interface, application Description automatically generatedGraphical user interface, application Description automatically generated

*Hình 4.1. Giao diện đăng nhập*

Người dùng ở đây là Sinh viên hoặc Giảng viên sẽ nhập mã và mật khẩu rồi chọn ĐĂNG NHẬP trên giao diện.

### Giao diện danh sách lớp họcGraphical user interface, application Description automatically generatedGraphical user interface, application Description automatically generated

*Hình 4.2. Giao diện danh sách lớp học*

Sinh viên xem danh sách các lớp học đang cần được điểm danh và chọn ĐIỂM DANH trên giao diện (hình bên trái).

Giảng viên xem danh sách các lớp học của mình và chọn một lớp để tạo phiên điểm danh (hình bên phải).

### Giao diện điểm danhGraphical user interface, application Description automatically generatedGraphical user interface, website Description automatically generated

*Hình 4.3. Giao diện điểm danh*

Sinh viên thực hiện chụp ảnh điểm danh, khi gửi lên hình ảnh giả mạo (hình bên trái) sẽ không được chấp nhận, chỉ khi chụp hình người thật mới được chấp nhận (hình bên phải).

### Giao diện kết quả điểm danhGraphical user interface Description automatically generated

*Hình 4.4. Giao diện kết quả điểm danh*

Giao diện kết quả điểm danh của Sinh viên ở một lớp, những Sinh viên chưa điểm danh được có thể được Giảng viên chọn điểm danh cho.

### Giao diện đăng nhập của Admin

*Hình 4.5. Giao diện đăng nhập của Admin*

Admin nhập tên đăng nhập và mật khẩu trên và chọn đăng nhập trên giao diện.

### Giao diện thêm mới tài khoản

*Hình 4.6. Giao diện thêm mới tài khoản*

Admin chọn chức năng tạo tài khoản và nhập thông tin cho Sinh viên, Giảng viên.

### Giao diện thêm ảnh vào tài khoản sinh viên

*Hình 4.7. Giao diện thêm ảnh vào tài khoản sinh viên*

Admin thêm ảnh của từng sinh viên vào tài khoản sinh viên để sau này hệ thống kiểm tra và điểm danh cho sinh viên bằng hình ảnh.

### Giao diện sửa tài khoản

*Hình 4.8. Giao diện sửa tài khoản*

Admin chọn sửa tài khoản và thay đổi các thông tin của người dùng.

### Giao diện xóa tài khoản

*Hình 4.9. Giao diện xóa tài khoản*

Admin chọn chức năng xóa tài khoản người dùng trên giao diện.

# KẾT LUẬN

Qua thời gian tìm hiểu, học hỏi và xây dựng đề tài **“Phát triển hệ thống điểm danh trong lớp học sử dụng công nghệ nhận diện khuôn mặt”,** em đã đạt được một số kết quả nhất định:

## Kết quả đạt được

* + Tìm hiểu các cách tiếp cận và phương pháp để giải quyết bài toán.
  + Ứng dụng giải pháp sử dụng mạng nơ-ron tích chập cho bài toán.
  + Nâng cao kỹ năng phân tích, thiết kế hệ thống.
  + Trau dồi thêm kỹ năng lập trình.

## Hướng phát triển

Hiện tại hệ thống chưa đạt kết quả tốt và chưa xử lý hết các trường hợp xảy đến trong quá trình điểm danh. Vì vậy, trong tương lai, em sẽ thay đổi các model để kết quả chính xác hơn, tốc độ xử lý nhanh hơn và có thể triển khai cho thực tế.

**Tiếng Việt**

# TÀI LIỆU THAM KHẢO

1. Nguyễn Thanh Tuấn - Deep Learning Cơ Bản. Copyright 2019.

## Tiếng Anh

1. Shervin Minaee, Ping Luo, Zhe Lin, Kevin Bowyer, “Going Deeper into Face Detection: A Survey”.
2. Meng Yuana, Seyed Yahya Nikoueia, Alem Fitwia, Yu Chena, Yunxi Dong, “Minor Privacy Protection Through Real-time Video Processing at the Edge”.
3. Jiankang Deng, Jia Guo, Niannan Xue, “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”.
4. Zitong Yu, Chenxu Zhao, Zezheng Wang, Yunxiao Qin, Zhuo Su, Xiaobai Li, Feng Zhou, Guoying Zhao, “Searching Central Difference Convolutional Networks for Face Anti-Spoofing”.
5. Peng Zhang, Fuhao Zou1, Zhiwen Wu, Nengli DaiSkarpness Mark, Michael Fu, Juan Zhao, Kai Li, “FeatherNets: Convolutional Neural Networks as Light as Feather for Face Anti-spoofing”.
6. Mei Wang, Weihong Deng, “Deep Face Recognition: A Survey”.
7. Jialiang Zhang, Xiongwei Wu, Jianke Zhu, Steven C.H. Hoi, “Feature Agglomeration Networks for Single Stage Face Detection”.

## Danh mục các website tham khảo

1. “<https://dominhhai.github.io/vi/2018/04/nn-intro/>”.
2. “https://thanhvie.com/tim-hieu-ve-mang-no-ron-tich-chap-convolutional-neural-networks/”.
3. “<https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html>”.
4. “<https://paperswithcode.com/method/inception-v4>”.
5. “<https://viblo.asia/p/detect-object-su-dung-mo-hinh-ssd-WAyK84rnKxX>”.
6. “[https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-](https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-208539308200695) [208539308200695](https://ichi.pro/vi/giai-thich-ve-siamese-net-triplet-loss-va-circle-loss-208539308200695)”.
7. “[https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-](https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN4lD8) [phan-1-Qbq5QDN4lD8](https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN4lD8)”.
8. “<https://viblo.asia/p/distance-measure-trong-machine-learning-ByEZkopYZQ0>”.